

Localized Operator Learning with Adaptive Partition-of-Unity Mixture-of-Expert Networks

Madison Cooley

*Scientific Computing Institute
University of Utah
Salt Lake City, UT*

mcooley@sci.utah.edu

Ramansh Sharma

*Kahlert School of Computing
University of Utah
Salt Lake City, UT*

ramansh@cs.utah.edu

Shandian Zhe

*Kahlert School of Computing
University of Utah
Salt Lake City, UT*

shandian.zhe@utah.edu

Robert M. Kirby

*Scientific Computing Institute
University of Utah
Salt Lake City, UT*

mike.kirby@utah.edu

Varun Shankar

*Kahlert School of Computing
University of Utah
Salt Lake City, UT*

shankar@cs.utah.edu

Reviewed on OpenReview: <https://openreview.net/forum?id=ccEY1fZ5gA>

Abstract

Operator learning methods such as DeepONets and FNOs often struggle with PDE families featuring sharp interfaces, heterogeneous coefficients, and localized multiscale structures. We introduce a partition-of-unity (POU) mixture-of-experts framework for localized operator learning, in which geometry-aware gating networks produce smooth spatial partitions which blend the contributions of local expert networks. Our main contribution is HiRefPOU, a residual-style hierarchical POU architecture for DeepONets that organizes localized representations through nested parent-child partitions while preserving global continuity. We also show that the same POU principle can be incorporated into Fourier Neural Operators to introduce spatial adaptivity without modifying the underlying spectral layers. On heterogeneous Darcy and reaction–diffusion benchmarks, HiRefPOU achieves substantially lower error than global DeepONet and static POU-MoE baselines, while the broader operator-learning experiments show that the benefits of localization depend on the PDE structure and the chosen neural-operator backbone. The learned partitions are interpretable and align with interfaces and regions of rapid solution variation. These results show that explicit geometric localization can improve both accuracy and interpretability in neural operator learning.

1 Introduction

The emergence of machine learning (ML)-based partial differential equation (PDE) solvers has enabled the efficient and data-driven modeling of complex physical systems. Among these are neural operator frameworks, such as Deep Operator Networks (DeepONets) (Lu et al., 2021) and Fourier Neural Operators (FNOs) (Li et al., 2020), which learn mappings between function spaces and capture the input-output behavior of entire PDE families. These models have achieved notable success across a range of domains, including fluid dynamics, porous media flow, and materials science, where scalable operator-learning approaches are increasingly central to practical surrogate modeling in scientific computing.

Neural operators, while designed to learn mappings between families of PDEs, can encounter limitations when approximating solutions with localized, multi-scale, or heterogeneous structures. In practice, many neural operator architectures rely on globally supported components, such as dense neural networks defined across the full problem domain or spectral expansions, which tend to under-represent high-frequency or spatially localized features. While such global formulations preserve smoothness and continuity, they limit the network’s ability to allocate representational capacity to regions of higher solution complexity, often resulting in an insufficient representation of fine-scale structure, leading to reduced accuracy in these areas.

A natural remedy is to introduce locality through domain decomposition or mixture-of-experts formulations. In the physics-informed setting, methods such as APINNs (Hu, 2023), XPINNs (Jagtap & Karniadakis, 2020), and POU-PINNs (Rodriguez et al., 2024) decompose the domain into subregions, while recent ensemble DeepONet models (Sharma & Shankar, 2025) introduce partition-of-unity (POU) mixtures on the operator-learning side. These approaches demonstrate the value of localization, but they are typically fixed, single-level, or only weakly structured geometrically, and therefore do not provide learned nested hierarchical partitioning for neural operator learning.

On the operator-learning side, ensemble and mixture-based formulations have emerged for introducing locality while preserving a global input–output structure. In particular, recent ensemble DeepONet architectures (Sharma & Shankar, 2025) decompose the trunk network (i.e., the evaluation-coordinate encoder) into a mixture of expert components satisfying a partition-of-unity (POU) constraint, and demonstrate that combining multiple operator experts improves expressivity relative to both single global models and comparably parameterized baselines. These results provide strong motivation for incorporating geometrically structured ensembles into neural operator models. However, these approaches remain limited to flat (single-level) decompositions and do not support learned nested parent–child spatial partitioning.

POU constructions provide a natural mechanism for blending multiple localized models through smooth, overlapping weighting functions by introducing learnable partition functions that modulate the contributions of local subnetworks (Lee et al., 2021). Existing neural POU approaches have primarily been explored in the context of single-function approximation and have seen more limited development in operator-learning settings. Moreover, these methods typically rely on flat partitioning schemes and do not jointly incorporate adaptive partitioning, geometric awareness, and expressive neural parameterizations within a hierarchical framework.

A complementary line of work introduces hierarchical refinement through residual network architectures, which improve accuracy in PDE problems with sharp gradients by incrementally correcting coarse global approximations. Hybrid Residual PINNs (HyResPINNs) (Cooley et al., 2025) exemplify this approach by refining a global solution using localized residual components. While effective, such residual architectures rely on predefined combinations of neural and RBF-based components, leading to fixed, radially structured regions of influence and limiting the ability to form flexible, data-adaptive partitions or hierarchical mixture-of-experts decompositions. Taken together, these observations motivate the extension of hierarchical decomposition principles from solution learning to operator learning.

We address this limitation with a hierarchical POU framework for operator learning that combines smooth learnable partitions, localized experts, and additive multi-level spatial representations. The resulting architecture preserves global continuity while allowing spatially varying contributions from localized experts.

Although the proposed framework is also relevant to physics-informed settings, the main contributions of this paper are in localized neural operator learning. Our architectural developments, mathematical formulation, and numerical studies focus on DeepONet- and FNO-type models, while PINN-related discussions are included primarily for motivation and conceptual contrast.

Specifically, this paper makes the following contributions:

- We propose HiRefPOU, a hierarchical partition-of-unity framework for localized neural operator learning.
- We introduce a hierarchical POU architecture and study one-, two-, and three-level instantiations, with the two-level formulation serving as the primary architectural case and the three-level variant used to probe the effect of an additional nested partition level.
- We embed the proposed POU mechanism in DeepONet and FNO architectures and evaluate it on a range of operator-learning benchmarks, with a targeted PINN case study included in the appendix.
- Across these experiments, we show that learned POU localization improves accuracy relative to global and fixed-partition DeepONet baselines on several heterogeneous PDE benchmarks. Comparisons among the 1L, 2L, and 3L variants further show that the benefit of organizing these localized representations hierarchically is problem-dependent and does not increase monotonically with depth.

Novelty scope. Partition-of-unity approximation, mixture-of-experts weighting, learned domain decomposition, additive global-local representations, hierarchical spatial partitioning, POD augmentation, and DeepONet/FNO backbones are established ideas and are not individually introduced in this work. Accordingly, we position HiRefPOU as a localized neural-operator architecture rather than as a fundamentally new operator-learning paradigm. The contribution of HiRefPOU is their specific integration for localized neural operator learning: a geometry-aware hybrid gate learns smooth coordinate-dependent partitions; child weights are constrained through parent POU weights; the resulting global and level-wise local trunk components are combined additively; and the same geometric POU principle is studied across DeepONet- and FNO-type backbones. The empirical contribution is a systematic evaluation of when this localization bias is useful and when additional hierarchy or POU augmentation provides limited benefit.

The remainder of this paper is organized as follows. Section 2 reviews related work in physics-informed and operator-learning architectures. Section 3 introduces the relevant background related to PINNs, operator learning, and hierarchical POU models and their mathematical formulation. Section 4 presents the hierarchical POU architecture and geometry-aware gating formulation. Section 5 describes its instantiation in DeepONet and FNO architectures and analyzes computational complexity. Section 6 outlines the numerical experiments, baseline methods and relevant implementation details, while Section 7 discusses results and Section 8 concludes.

2 Related Works

This section reviews prior work most relevant to the proposed hierarchical partition-of-unity (POU) framework: physics-informed neural networks, neural operators, domain decomposition and POU methods, mixture-of-experts architectures, and hierarchical neural representations. Our emphasis is on localized operator learning, which is the primary methodological and empirical focus of this paper.

2.1 Physics-Informed Neural Networks (PINNs)

In the context of PDE solving, PINNs represent the most direct attempt to approximate physical laws through global neural surrogates (Raissi et al., 2017a;b) and approximate the solutions of PDEs by embedding physical laws directly into the neural network’s loss function. They offer a flexible, mesh-free framework in which expressivity arises from the architecture, activation functions, and optimization process (Hornik et al., 1989; Cybenko, 1989). While universal approximation theorems establish that sufficiently large networks can represent any continuous function, realizing this theoretical expressivity in practice remains

challenging (DeVore et al., 2020; Adcock & Dexter, 2021; Marwah et al., 2021). In particular, standard PINNs employ globally coupled representations, where parameters are updated according to a loss aggregated over the entire domain. This implicit, global adaptivity contrasts sharply with the localized refinement mechanisms used in classical numerical methods and limits the network’s ability to capture sharp gradients, discontinuities, or multiscale behaviors (Wang et al., 2022).

2.2 Neural Operator Learning

Neural operators learn mappings between function spaces and provide data-driven surrogates for PDE families. Two prominent examples are DeepONets (Lu et al., 2021), which combine branch and trunk networks to approximate nonlinear operators, and Fourier Neural Operators (Li et al., 2020), which learn operators through spectral convolutions. Despite their success, standard neural operators remain largely global representations. DeepONets rely on global trunk bases, while FNOs rely on globally supported Fourier modes; both can struggle with localized, heterogeneous, or multiscale structure. Recent extensions introduce multiresolution kernels, adaptive discretizations, or localized spectral filters, but these approaches often remain computationally expensive or lack clear geometric interpretability. This motivates architectures that combine global operator structure with adaptive spatial localization.

2.3 Domain Decomposition and Partition of Unity Methods

Recent extensions of PINNs mitigate optimization and scalability challenges through domain decomposition, which divides the computational domain into smaller subregions for localized approximation and can support parallelization. Methods such as XPINNs (Jagtap & Karniadakis, 2020) and DT-PINNs (Sharma & Shankar, 2022) still rely largely on fixed or manually designed partitions, while soft decomposition approaches use adaptive weighting schemes to reduce hand-crafted partitioning (Barbara et al., 2024; Hu, 2023). These results highlight the value of learnable localization, but most existing approaches remain non-hierarchical and use standard MLP-based gating.

Partition-of-unity (POU) methods provide a classical way to build globally continuous approximations from overlapping local components. In finite element and meshfree settings, the POU constraint enables local adaptivity while preserving global consistency, making it a natural foundation for machine-learning-based PDE solvers. Unlike xPINN-style methods, which train separate models on disjoint subdomains and enforce consistency through interface penalties, our POU formulation constructs a single global approximation as a weighted sum of local experts. The weights are learned jointly with the solution, remain continuous, and sum to one over the domain, so continuity is enforced implicitly while the learned partitions can align with spatial structure that recurs across the training distribution.

2.4 Mixture-of-Experts and Adaptive Gating Architectures

Mixture-of-experts (MoE) models provide an adaptive counterpart to domain decomposition by representing a global approximation as a weighted combination of specialized subnetworks. In scientific machine learning, this allows different experts to focus on distinct spatial regions, physical regimes, or modes of variation while remaining coupled through a shared training objective. Related ideas have appeared in both PINN and operator-learning settings, where learnable gating is used to decompose complex mappings into simpler local components (Hu, 2023; Wang et al., 2025; Sharma & Shankar, 2025). In particular, the ensemble DeepONet framework of Sharma & Shankar (2025) demonstrates that partitioning the trunk representation into localized experts can improve expressivity and accuracy relative to purely global operator networks, providing direct motivation for localized operator learning.

Despite this promise, standard MoE formulations remain limited for PDE and operator-learning problems. Most gating networks are implemented as generic neural modules with little explicit geometric structure, making the learned partitions difficult to interpret and not necessarily aligned with meaningful physical subregions (Lee et al., 2021). Moreover, gating is often performed in latent feature space rather than directly in the spatial coordinates, which can hinder spatial coherence and reduce control over how experts divide the domain. Even structured alternatives such as RBF-based gating typically impose purely radial supports,

which can be too restrictive for anisotropic or irregular localized behavior. These limitations suggest that adaptive specialization alone is not sufficient: for PDE and operator-learning problems, the gating mechanism should also be geometry-aware and spatially interpretable.

The closest operator-learning predecessor is the POU-MoE DeepONet of Sharma & Shankar (2025), which also constructs a localized trunk by blending independent local trunk networks through POU weights. Their reported formulation uses a flat collection of overlapping patches with prescribed centers and radii and normalized compactly supported kernel weights. Compact support permits local expert evaluation because each coordinate lies in only a subset of patches. HiRefPOU retains the localized-trunk interpretation but differs in four principal ways. First, its partition parameters are learned jointly through a hybrid center-distance and neural gate rather than specified as fixed kernel patches. Second, its learned weights may deform beyond purely radial compact supports. Third, it supports nested parent-child POU weights and additive level-wise trunk contributions. Fourth, the current HiRefPOU implementation evaluates all experts densely, rather than exploiting compact support for sparse local execution. Thus, the principal extension is not the introduction of a POU-MoE trunk itself, but the learned geometric gating and nested organization studied within neural-operator architectures.

2.5 Multi-Fidelity and Hierarchical Neural Architectures

Hierarchical and multi-fidelity architectures address a complementary limitation of global neural PDE models: many problems exhibit structure across multiple spatial or temporal scales and therefore benefit from coarse-to-fine refinement. This principle is classical in numerical analysis, where multigrid, wavelet, and adaptive finite element methods improve approximation quality by combining coarse global structure with targeted fine-scale correction (Gerhard et al., 2022; Restrepo & Leaf, 1995; Mitchell, 1991). Analogous ideas have emerged in scientific machine learning through multi-level PINNs, multi-fidelity surrogates, and hierarchical neural operators, where progressively refined subnetworks or feature maps improve accuracy and stability (Howard et al., 2023).

Among the most closely related ideas are residual hierarchical architectures, which refine a coarse approximation through additional localized correction terms. For example, HyResPINNs (Cooley et al., 2025) combine global and local components in a residual hierarchy to improve approximation of PDE solutions with sharp gradients and heterogeneous structure. This provides strong evidence that hierarchical refinement can be beneficial in scientific machine learning. However, such approaches are primarily designed for solution learning rather than operator learning, and their local regions of influence are typically induced by predefined architectural choices rather than by an adaptive partition-of-unity decomposition learned jointly from data.

More broadly, most existing hierarchical neural architectures operate over fixed scales or latent feature hierarchies rather than over learned geometric partitions of the physical domain. As a result, they provide multiscale refinement without explicitly learning where localized specialization should occur. This is the main distinction we emphasize in the present work. Our framework combines additive level-wise representations with geometry-aware partition-of-unity gating, yielding a model in which localized representations are organized through learned nested spatial partitions rather than through fixed scales or latent feature hierarchies. Viewed this way, the proposed method bridges two strands of prior work: localized expert decompositions for operator learning (Sharma & Shankar, 2025) and residual hierarchical refinement in scientific machine learning (Cooley et al., 2025).

Summary. Existing approaches typically provide either adaptive expert specialization without strong geometric structure, or hierarchical refinement without learned spatial partitioning. In contrast, the proposed hierarchical POU framework combines geometry-aware, learnable partitions with additive level-wise representations organized through nested spatial partitions in a single operator-learning architecture. This yields a model that is globally continuous, spatially interpretable, and adaptively localized across nested spatial regions, linking localized expert decompositions in operator learning (Sharma & Shankar, 2025) with residual hierarchical architectures in scientific machine learning (Cooley et al., 2025). From this perspective, HiRefPOU is best viewed as a localized neural-operator architecture that introduces a particular spatial inductive bias into established operator-learning backbones.

Table 1: Comparison of HiRefPOU with closely related localization and domain-decomposition approaches. Execution refers to the implementation described in the cited work.

Method	Primary setting	Partition / localization	Organization	Operator learning	Expert / local execution
POU _{net} Lee et al. (2021)	Function regression	Learned POU partition	Flat	No	Dense weighted local basis
XPINN Jagtap & Karniadakis (2020)	PINN solution learning	Prescribed subdomains	Flat	No	Subdomain-local networks
APINN Hu (2023)	PINN solution learning	Learned soft gate	Flat	No	Dense weighted subnetworks
HyResPINN Cooley et al. (2025)	PINN solution learning	Neural/RBF representation	Residual blocks	No	Dense
POU-MoE DeepONet Sharma & Shankar (2025)	Operator learning	Prescribed compact-support POU patches	Flat	Yes	Sparse local patches
GeoFNO Li et al. (2023)	Operator learning	Learned geometric deformation	Global	Yes	Dense
LocalFNO Liu-Schiaffini et al. (2024)	Operator learning	Localized integral / differential kernels	Layer-level locality	Yes	Dense local kernels
HiRefPOU (ours)	Operator learning	Learned hybrid geometric POU	Flat or nested parent-child	Yes	Dense all-expert blending

3 Preliminaries

3.1 Problem Setup

We consider PDE solution approximation and, more generally, operator learning between function spaces. Let $\mathcal{D} = [0, T] \times \Omega \subset \mathbb{R}^{1+d}$, where $\Omega \subset \mathbb{R}^d$ is a bounded spatial domain with boundary $\partial\Omega$. A general time-dependent PDE takes the form

$$u_t + \mathcal{F}(u, \nabla u, \dots) = f, \quad (1)$$

subject to initial and boundary conditions $u(0, \mathbf{x}) = g(\mathbf{x})$ for $\mathbf{x} \in \Omega$ and $\mathcal{B}[u(t, \mathbf{x})] = 0$ for $t \in [0, T]$ and $\mathbf{x} \in \partial\Omega$.

Operator learning seeks to approximate a mapping $\mathcal{G} : \mathcal{U} \rightarrow \mathcal{V}$ from admissible inputs $a \in \mathcal{U}$ to solution functions $s \in \mathcal{V}$. We write the governing problem abstractly as

$$\mathcal{N}(a, s) = 0, \quad \mathcal{B}(a, s) = 0,$$

and assume that each admissible a defines a unique solution $s = s(a)$, inducing the solution operator $\mathcal{G} : a \mapsto s$.

Given paired data $\{a^{(i)}, s^{(i)}\}_{i=1}^N$, the goal is to learn $\mathcal{G}_\theta \approx \mathcal{G}$ so that unseen inputs can be mapped to their corresponding solutions without solving the PDE from scratch. DeepONets and FNOs implement this idea through different parameterizations of the input-function representation and output evaluation map.

3.2 Physics-Informed Neural Networks (PINNs)

The general formulation of PINNs (Raissi, 2018) aims to approximate the unknown solution $u(t, \mathbf{x})$ using a parameterized model $u_\theta(t, \mathbf{x})$, such that $u_\theta(t, \mathbf{x}) \approx u(t, \mathbf{x})$ and θ denotes the set of all trainable parameters of the network. The parameters θ are optimized to minimize the composite loss function:

$$L_{\text{pin}}(\theta) = \lambda_{ic} L_{ic} + \lambda_{bc} L_{bc} + \lambda_r L_r, \quad (2)$$

where L_{ic} , L_{bc} , and L_r represent the loss components associated with the initial conditions, boundary conditions, and the residual of the PDE, respectively; and λ_{ic} , λ_{bc} , and λ_r are corresponding weighting coefficients that balance the contribution of each loss term. These terms can be set as static weights or adapted during training using various techniques such as NTK weighting (Wang et al., 2021b) or gradient annealing (Wang et al., 2021a), among others.

Each loss term in Equation (2) is computed as the mean squared error of the initial condition, boundary, and PDE residuals. Specifically, each loss term is defined as:

$$L_{ic} = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} |u_{\theta}(0, \mathbf{x}_{ic}^i) - g(\mathbf{x}_{ic}^i)|^2, \quad (3)$$

$$L_{bc} = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} |\mathcal{B}[u_{\theta}](t_{bc}^i, \mathbf{x}_{bc}^i)|^2, \quad (4)$$

$$L_r = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \frac{\partial u_{\theta}}{\partial t}(t_r^i, \mathbf{x}_r^i) + \mathcal{F}[u_{\theta}](t_r^i, \mathbf{x}_r^i) - f(t_r^i, \mathbf{x}_r^i) \right|^2. \quad (5)$$

The training data points $\{\mathbf{x}_{ic}^i\}_{i=1}^{N_{ic}}$, $\{t_{bc}^i, \mathbf{x}_{bc}^i\}_{i=1}^{N_{bc}}$ and $\{t_r^i, \mathbf{x}_r^i\}_{i=1}^{N_r}$ can be drawn from a fixed set or resampled randomly at each iteration of a gradient descent algorithm.

3.3 Deep Operator Networks (DeepONets)

Deep Operator Networks (DeepONets) are designed to approximate nonlinear operators $\mathcal{G} : \mathcal{U} \rightarrow \mathcal{V}$ by decomposing the mapping into two subnetworks: the branch network and the trunk network. The branch network B_{θ} takes as input a finite-dimensional representation of the function $a \in \mathcal{U}$, for example

$$\mathbf{a} = (a(x_1), \dots, a(x_m))$$

evaluated at fixed sensor locations $\{x_i\}_{i=1}^m$, and produces a latent coefficient vector

$$\mathbf{b}(a) = (b_1(\mathbf{a}), b_2(\mathbf{a}), \dots, b_q(\mathbf{a}))^{\top} \in \mathbb{R}^q.$$

The trunk network T_{ϕ} takes as input the evaluation coordinate $y \in \mathbb{R}^p$ and outputs a latent basis vector,

$$\mathbf{t}(y) = (t_1(y), t_2(y), \dots, t_q(y))^{\top} \in \mathbb{R}^q.$$

The output of the DeepONet for a given a and y is obtained by merging the branch and trunk outputs through an inner product:

$$G_{\theta}(a)(y) = \sum_{k=1}^q b_k(a(x_1), \dots, a(x_m)) t_k(y) + c,$$

where $c \in \mathbb{R}$ is an optional bias term. For vector-valued outputs of dimension n , the latent dimension q is partitioned as $0 = q_0 < q_1 < \dots < q_n = q$, and each component is computed as

$$G_{\theta}^{(i)}(a)(y) = \sum_{k=q_{i-1}+1}^{q_i} b_k(a(x_1), \dots, a(x_m)) t_k(y) + c, \quad i = 1, \dots, n.$$

This formulation enables DeepONets to flexibly approximate both scalar- and vector-valued operators.

To quantify the discrepancy between the DeepONet prediction and the true PDE solution, we define the pointwise mean-squared error for a given input a as

$$\mathcal{L}(a, \theta) = \frac{1}{P} \sum_{j=1}^P |G_{\theta}(a)(y_j) - s(y_j)|^2,$$

where $\{y_j\}_{j=1}^P$ are evaluation locations in the domain of $G(a)$, and $s(y_j)$ denotes the corresponding PDE solution values obtained from the reference model. The total training objective is then obtained by averaging over all N samples in the dataset:

$$\mathcal{L}_{ol}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(a^{(i)}, \theta). \quad (6)$$

3.4 Fourier Neural Operators

Fourier Neural Operators (FNOs) are another class of operator networks. The central building block of an FNO is the Fourier layer, which replaces traditional local convolutions with multiplications in the frequency domain. Given an intermediate feature representation $v(x)$ defined on a discretized domain, the Fourier layer computes

$$v'(x) = \mathcal{F}^{-1}(R_\theta(\hat{v}(\xi)))(x) + Wv(x), \quad (7)$$

where $\hat{v}(\xi) = \mathcal{F}[v](\xi)$ is the Fourier transform of v , R_θ denotes a learnable spectral multiplier acting on a truncated set of modes, and W is a trainable pointwise linear transformation. The inverse Fourier transform $\mathcal{F}^{-1}$ maps the modified representation back to the spatial domain. An FNO consists of a sequence of Fourier layers interleaved with pointwise nonlinearities. The network is typically structured as

$$v^{(0)} = P(a), \quad v^{(\ell+1)} = \sigma\left(\mathcal{F}^{-1}(R_\theta(\hat{v}^{(\ell)})) + Wv^{(\ell)}\right),$$

where P is a lifting operator that maps the input a into a higher-dimensional latent space, and σ is a nonlinearity such as tanh or ReLU. A final projection operator Q maps the latent representation back to the output space:

$$\mathcal{G}_\theta(a) = Q(v^{(L)}).$$

By operating directly in the Fourier domain, FNOs efficiently capture long-range dependencies and global interactions, which are challenging for purely local architectures. However, standard FNOs rely on globally supported Fourier modes and lack explicit mechanisms for localized or adaptive refinement, which can degrade performance near sharp gradients or localized features.

Unified perspective. Despite their different training settings, PINNs, DeepONets, and FNOs share a common limitation: they rely primarily on global parameterizations. This can hinder accuracy when the target solution or operator contains fine-scale or spatially localized structure. Partition-of-unity models address this issue by blending local components while preserving global continuity. The approach developed here extends that idea by learning geometry-aware partitions organized through nested parent-child relationships and combining their spatially weighted expert contributions additively across levels.

3.5 Partition-of-Unity and Mixture-of-Experts Formulations

Partition-of-unity (POU) methods provide a general mathematical framework for constructing globally continuous approximations from a collection of local approximants. Originally developed in the context of finite element and mesh-free methods, the POU principle ensures global smoothness by blending overlapping local basis functions through nonnegative weighting functions that sum to one over the domain. Formally, let $\{\phi_j(\mathbf{x})\}_{j=1}^{N_p}$ denote a set of partition functions satisfying

$$\phi_j(\mathbf{x}) \geq 0, \quad \text{and} \quad \sum_{j=1}^{N_p} \phi_j(\mathbf{x}) = 1, \quad \forall \mathbf{x} \in \Omega. \quad (8)$$

Each ϕ_j is supported primarily on a subregion $\Omega_j \subset \Omega$, with overlapping supports across neighboring partitions.

Mixture-of-Experts (MoE) models provide a closely related perspective by representing a global function as a weighted combination of local approximants. Given a collection of expert functions $\{f_j(\mathbf{x})\}_{j=1}^{N_c}$, the global approximation is expressed as

$$u(\mathbf{x}) = \sum_{j=1}^{N_c} \phi_j(\mathbf{x}) f_j(\mathbf{x}), \quad (9)$$

where the weighting functions ϕ_j act as a gating mechanism. When the weights satisfy the partition-of-unity conditions (8), the resulting model inherits the stability and smoothness properties of classical POU formulations.

In this context, the POU framework can be interpreted as a structured MoE model in which the gating functions are explicitly constrained to be nonnegative and to sum to one. The experts $\{f_j\}$ may be linear models, radial basis functions, multilayer perceptrons, or other specialized subnetworks. When the experts are nonlinear subnetworks, the model becomes a locally adaptive ensemble in which each expert specializes in a region of the input space, while smooth blending across regions is guaranteed by the partition-of-unity property.

In practice, the gating functions $\{\phi_j(\mathbf{x})\}$ may be parameterized through normalized radial basis functions, distance-based kernels, or other geometry-aware constructions. In this work, we define the gating functions using learnable, spatially structured mechanisms, which combine RBF activations with box initialization-based partitions—which are described in detail in Section 4. This design preserves the interpretability and continuity guarantees of classical POU methods while enabling data-driven, adaptive partitioning within a neural network framework.

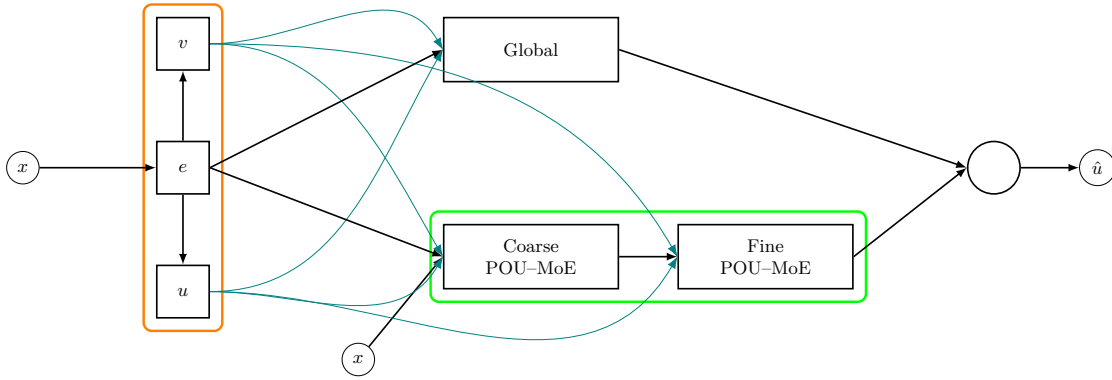


Figure 1: Residual-style hierarchical architecture used in HiRefPOU. The input x is embedded and passed through a global backbone to obtain $u_g(x)$. A coarse Mixture-of-Experts (MoE) layer predicts $u_c(x)$. A second MoE layer predicts the fine-level contributions $u_f(x)$ through nested parent-child partitioning. The final output is the sum of all three components, $u(x) = u_g(x) + u_c(x) + u_f(x)$.

4 Hierarchical Partition-of-Unity Models for Adaptive Operator Learning

We introduce HiRefPOU, a hierarchical POU architecture that combines a global neural approximator with multi-level Partition-of-Unity Mixture-of-Experts (POU-MoE) subnetworks. Figure 1 shows our full network architecture diagram. Because several terms used below have distinct architectural meanings, Table 2 summarizes the terminology used throughout this work. The hierarchy is defined through the organization of the partition functions: each child partition is modulated by and locally normalized within a parent partition. This construction permits additional spatial subdivision within the regions emphasized by a parent gate while preserving smooth global aggregation. The level outputs are combined through additive residual-style updates. However, all components are learned jointly from the final training objective, and the architecture does not require any level to approximate a detached residual left by the preceding levels. Thus, HiRefPOU should be interpreted as a nested spatial representation rather than as a strict spectral or orthogonal multiresolution decomposition.

Throughout this work, the term ‘hierarchical’ refers to the nested parent-child organization of the POU weights, and residual refers to the additive composition of the level-wise outputs. The construction guarantees nested spatial weighting, smooth aggregation, and a globally continuous representation. It does not explicitly impose frequency separation, orthogonality, or stage-wise residual fitting between levels. Accordingly, the terms ‘coarse’ and ‘fine’ identify levels of the spatial partition hierarchy rather than predetermined frequency bands or uniquely identifiable components of the target operator.

Table 2: Terminology used throughout this work. These definitions describe the specific architectural meaning of each term in HiRefPOU and are not intended as general definitions for all hierarchical or mixture-of-experts models.

Term	Meaning in this work
Adaptive	The parameters defining the POU gates, including centers, scales, neural gate parameters, and mixing coefficients, are learned jointly from the training objective rather than prescribed a priori. In operator-learning experiments, the gates depend only on evaluation coordinates and are shared across input samples; they are therefore not instance-conditioned.
Hierarchical	The POU weights are organized through nested parent–child relationships. A child partition is spatially weighted by its parent partition, producing a nested POU structure across levels.
Residual-style	Successive POU levels contribute additively to the model representation. The term describes the additive architecture and does not imply stage-wise training against detached residual targets.
Coarse / fine level	Labels identifying relative levels of the spatial partition hierarchy. They do not denote prescribed low- and high-frequency components or an orthogonal multiresolution decomposition.
Localization	Spatially varying POU weights assign different relative contributions to local expert representations across the evaluation domain.
Expert specialization	An expert may contribute more strongly in regions where its learned POU weight is large. Specialization is an empirical property of the learned representation rather than an imposed functional decomposition.
Gating	When used in this work, spatial gating refers to soft weighting and blending of expert outputs. The current implementation evaluates all experts densely and does not perform sparse conditional routing or top- k expert selection.

4.1 Residual-Style Hierarchical POU Representation

Formally, given an input $x \in \mathbb{R}^d$, the two-level HiRefPOU model produces the representation

$$u_\theta(x) = u_g(x; \theta_g) + u_c(x; \theta_c) + u_f(x; \theta_f), \quad (10)$$

where u_g denotes the global component, u_c denotes the coarse-level POU component, and u_f denotes the fine-level POU component. The parameter sets θ_g , θ_c , and θ_f collectively contain the trainable parameters associated with the corresponding global network, local experts, and gating networks.

We refer to this construction as residual-style because the POU levels are combined additively with the global component. This terminology describes the architecture and does not imply stage-wise training against residual targets. In particular, all components are optimized jointly from the final training objective, and no frequency separation, orthogonality, or level-wise identifiability is imposed.

The term hierarchical instead refers to the organization of the POU weights. Fine-level partitions are nested within coarse-level partitions through the parent–child construction introduced in Section 4.6. Accordingly, the labels coarse and fine refer to relative levels of the spatial partition hierarchy rather than prescribed low- and high-frequency components of the target function.

4.2 Global Approximator Network.

The global approximator ($u_g(x; \theta_g)$) is implemented as a modified residual network that incorporates shared conditioning features to stabilize optimization and provides a globally supported coordinate representation (Wang et al., 2021a; 2024; Rodriguez et al., 2024). Let $e(x) \in \mathbb{R}^m$ denote the shared embedding of the input coordinate. Starting from $g^{(0)}(x) = e(x)$, the global network applies a sequence of L gated bottleneck blocks of the form

$$g^{(\ell+1)}(x) = \mathcal{B}\left(g^{(\ell)}(x), h_u(x), h_v(x)\right), \quad \ell = 0, \dots, L-1, \quad (11)$$

where $h_u(x)$ and $h_v(x)$ are shared conditioning vectors derived from the embedding. Each bottleneck block consists of a stack of fully connected layers interleaved with nonlinear activations, followed by element-wise gating using the conditioning features. Specifically, intermediate activations are modulated according to

$$z \mapsto z \odot h_u(x) + (1 - z) \odot h_v(x), \quad (12)$$

which enables the network to interpolate between two learned feature representations and improves stability across heterogeneous regions of the domain (Wang et al., 2021a).

To promote stable training while allowing the transformed representation to be introduced gradually during optimization, each block is equipped with a learnable residual connection. Let x_{in} denote the block input and x_{out} its transformed output. The final block output is given by

$$\mathcal{B}(x_{\text{in}}, h_u, h_v) = \alpha x_{\text{out}} + (1 - \alpha) x_{\text{in}}, \quad (13)$$

where α is a scalar parameter initialized near zero and learned during training. This design allows the network to begin close to the identity map and gradually increase expressivity as optimization proceeds (Cooley et al., 2025; Wang et al., 2024). A final linear projection produces the global prediction

$$u_g(x) = W_g g^{(L)}(x). \quad (14)$$

The resulting global component provides a smooth, globally supported baseline representation that is combined additively with the localized POU components.

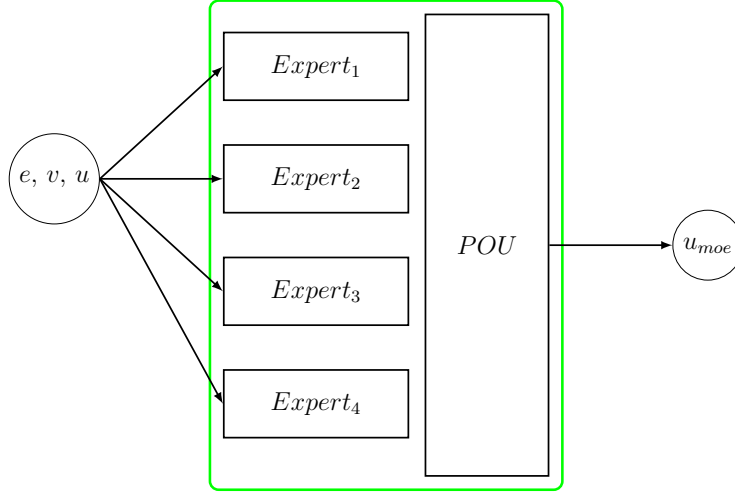


Figure 2: A POU-Mixture-of-Experts (MoE) layer predicts $u_p(x)$.

4.3 Geometry-Aware Gating Networks for Adaptive Domain Decomposition

A central contribution of this work is a geometry-aware gating network that combines center-based distance partitions with box-initialized residual gating. This design generalizes classical radial basis function (RBF) partitions while retaining the flexibility of learned, anisotropic domain decompositions. Each partition is associated with a learnable center point $c_j \in \mathbb{R}^d$, which acts as an explicit geometric anchor, allowing for easier partition initialization, regularization, and interpretation. We then compute the Euclidean distances as,

$$d_j^2(x) = \|x - c_j\|^2, \quad (15)$$

which serve as intrinsic geometric features that define the proximity of each point to the available partitions. These distances help define the center-based gating logits,

$$z_j^{\text{cntr}}(x) = -\frac{d_j^2(x)}{2\alpha_j^2}, \quad (16)$$

where $\alpha_j > 0$ is a learnable, per-center scaling parameter that controls the sharpness of the partition. This formulation preserves the interpretability and smoothness of RBF-like partitions while allowing the effective support of each region to be learned and updated during training.

To further increase the flexibility of the isotropic radial supports, we combine the center-based gating with a box-initialized residual gating network. This network consists of a shallow multilayer perceptron with residual connections, initialized such that its decision boundaries form approximately axis-aligned, non-overlapping hyperrectangular regions across the domain. During training, these initial box-like partitions are progressively deformed through learned nonlinear transformations, enabling the gating to capture anisotropic and nonconvex structures.

Let $z^{\text{box}}(x)$ denote the logits produced by the box-based gating network and $z^{\text{cntr}}(x)$ those produced by the center-distance gate. The final gating logits are obtained through a learnable convex combination

$$z(x) = w z^{\text{box}}(x) + (1 - w) z^{\text{cntr}}(x), \quad (17)$$

where $w = \text{sigmoid}(m)$ is a scalar mixing coefficient obtained by applying a sigmoid to a trainable parameter m such that $w \in (0, 1)$. This learned interpolation allows the model to transition smoothly between geometry-dominated and neural gating regimes.

Finally, a softmax is applied to the combined logits to obtain smooth POU weights,

$$\phi_j(x) = \frac{\exp(z_j(x))}{\sum_k \exp(z_k(x))}, \quad (18)$$

which satisfy $\sum_j \phi_j(x) = 1$ for all x . These weights define continuous domain partitions that are jointly optimized with the expert networks during training.

Throughout this work, the term ‘adaptive’ refers to the learned model parameters associated with the gating networks, including the learned center points c_j , scaling parameters α_j , gate networks, mixing parameters associated with the POU weighting networks. These parameters are learned jointly with the other model parameters rather than being prescribed, or fixed prior to training. In the operator learning settings, the gating network does not receive the input functions or their branch-network representation, coefficient fields, or any other sample-specific information. Consequently, the learned partitions are shared across input functions, and represent dataset-level geometric adaptation, rather than instance-wise adaptation. This construction is well-suited for spatial structures that occur across consistent locations across the data distribution, but it cannot reposition the partitions to follow sample-dependent moving interfaces or features. This would be an interesting area for future development.

Further, we emphasize that the gating construction above is written in the coordinate representation provided to the model. In Euclidean domains, this corresponds directly to the physical coordinates. For the three-dimensional reaction–diffusion benchmark considered here, the domain is the ball, not a manifold-valued surface. Thus, the gating network is applied directly to the Euclidean coordinates used by the operator model, and we do not introduce any intrinsic surface parameterization, geodesic distance, or manifold-specific gating constraints. Any locality is therefore learned in this ambient coordinate representation.

4.4 Partition-of-Unity Mixture-of-Experts Ensembles

Each POU level (u_c and u_f in Equation (10)) is implemented as a MOE ensemble composed of independently parameterized residual multilayer perceptrons. Let $\{\phi_j(x)\}_{j=1}^{N_c}$ denote the partition weights produced by the gating network at a given level. The corresponding expert ensemble produces local expert outputs $\{f_j(x)\}_{j=1}^{N_c}$, which are combined through a weighted sum.

Similar to the global approximator network (u_g), each expert f_j is implemented as a residual modified MLP conditioned on the shared features ($h_u(x), h_v(x)$). Given an input feature vector x_0 , the expert applies a sequence of fully connected layers,

$$x_{k+1} = \sigma(W_k x_k), \quad k = 0, \dots, D - 1, \quad (19)$$

where σ denotes a nonlinear activation function. As in the global network, each intermediate activation is modulated through the shared conditioning,

$$x_k \mapsto x_k \odot h_u(x) + (1 - x_k) \odot h_v(x), \quad (20)$$

which enables coordinated feature sharing across experts while preserving independent parameters.

To improve optimization and prevent overfitting in early training, each expert incorporates a learnable residual connection. Let x_{in} denote the expert input and x_{out} the transformed output. The residual update is given by

$$\tilde{x} = \alpha x_{\text{out}} + (1 - \alpha) x_{\text{in}}, \quad (21)$$

where α is a learnable scalar initialized near zero. A final linear projection maps $\tilde{x}$ to a scalar expert output.

All experts at a given POU level contain independent parameters and are evaluated at every query coordinate. Each expert evaluation is executed in parallel due to batch vectorization. The ensemble output is then formed as,

$$u_{\text{POU}}(x) = \sum_{j=1}^{N_c} \phi_j(x) f_j(x), \quad (22)$$

which guarantees smooth blending across partitions and preserves global continuity through the POU constraint. This design enables each expert to specialize to localized features in regions with large POU weights while the residual structure and shared conditioning promote coherent behavior across the ensemble and global network. While parallelization of experts improves hardware utilization compare to serial execution, the arithmetic cost increases with additional experts. This implementation level parallelism does not constitute sparse routing, which is a direction for future work.

4.5 Geometric Regularization for Learned Partitions

To encourage well-distributed domain partitions, we regularize the gating network and its learnable centers using lightweight geometric losses. These regularizers are applied jointly with the full model, and do not require access to solution data.

Let $X = \{x_i\}_{i=1}^N \subset \Omega$ denote a set of sample points and $C = \{c_j\}_{j=1}^{N_p}$ the corresponding set of learnable partition centers. For $\Omega \subset \mathbb{R}^d$, we define a generic coverage regularizer by comparing a geometric measure of the spread of the centers to the measure of the domain. In one dimension, this reduces to the length of the interval spanned by the centers; in two dimensions, to the area of their convex hull; and in three dimensions, to the volume of their convex hull. Denoting this measure generically by $M(C)$ and the corresponding domain measure by M_Ω , we write

$$L_{\text{cov}} = \lambda_{\text{cov}} \frac{|M(C) - M_\Omega|}{M_\Omega + \varepsilon}, \quad (23)$$

which discourages collapsed or poorly distributed center configurations.

For example, when $\Omega = [0, 1]$, we may take

$$M(C) = \max_j c_j - \min_j c_j, \quad M_\Omega = 1. \quad (24)$$

When $\Omega = [0, 1]^2$, $M(C)$ is the area of the convex hull of the centers, recovering the two-dimensional form used in this work. When $\Omega = [0, 1]^3$, $M(C)$ is the volume of the convex hull of the centers.

We also apply a soft containment penalty to discourage centers from drifting outside a prescribed bounding box containing the computational domain. Let

$$\mathcal{B} = \prod_{k=1}^d [a_k, b_k]$$

denote an axis-aligned bounding box satisfying $\Omega \subseteq \mathcal{B}$. We define the containment regularizer as

$$L_{\text{cont}} = \lambda_{\text{cont}} \sum_{j=1}^{N_p} \sum_{k=1}^d [\max(0, a_k - c_{j,k}) + \max(0, c_{j,k} - b_k)], \quad (25)$$

where $c_{j,k}$ denotes the k th coordinate of center $\mathbf{c}_j$. This penalty is zero when a center lies within the prescribed bounding box and increases linearly when it moves outside the admissible coordinate range.

For rectangular domains, the bounding box coincides with the computational domain. For nonrectangular domains, such as the three-dimensional unit ball considered in this work, we use the corresponding axis-aligned enclosing box. Thus, the containment term constrains the centers to remain within the coordinate extent of the problem domain, but does not impose the exact geometry of a nonrectangular boundary.

4.6 Hierarchical Partition-of-Unity Refinement

While a single-level POU model provides one collection of spatially weighted local experts, the hierarchical extension organizes additional POU levels through explicit parent-child relationships. The purpose of the hierarchy is to permit further spatial subdivision within the support of a parent partition; it does not prescribe a frequency decomposition of the represented function.

Let $\{\phi_j^c(x)\}_{j=1}^{N_c}$ denote the coarse-level POU weights, produced by a gating network as described in the previous section. These weights satisfy

$$\sum_{j=1}^{N_c} \phi_j^c(x) = 1, \quad \forall x \in \Omega.$$

The coarse-level POU contribution in Equation (10) is defined as

$$u_c(x) = \sum_{j=1}^{N_c} \phi_j^c(x) f_j^c(x), \quad (26)$$

where $f_j^c(x)$ denotes the expert associated with the j -th coarse partition.

To introduce a second level of nested spatial partitioning, each coarse partition $\phi_j^c(x)$ is then subdivided by a second gating network that produces child weights $\{\psi_{j,k}(x)\}_{k=1}^{N_f}$, locally normalized within each parent partition. The resulting fine-level POU weights are defined by

$$\phi_{j,k}^f(x) = \phi_j^c(x) \psi_{j,k}(x), \quad \sum_{k=1}^{N_f} \psi_{j,k}(x) = 1,$$

which guarantees the nested POU property $\sum_{j=1}^{N_c} \sum_{k=1}^{N_f} \phi_{j,k}^f(x) = 1$. The resulting parent-child construction is the sense in which the model is hierarchical: each fine-level weight inherits the support and spatial weighting of its parent through the multiplicative factor $\phi_j^c(x)$. The terms “coarse” and “fine” therefore refer to positions in this spatial partition tree.

In practice, this constraint is satisfied exactly by defining each fine-level weight as the product of its parent coarse weight and a locally normalized softmax over the children of that parent. Each fine partition $\phi_{j,k}^f(x)$ is paired with its own local expert $f_{j,k}^f(x)$, yielding the fine-level POU contribution from Equation (10) as,

$$u_f(x) = \sum_{j=1}^{N_c} \sum_{k=1}^{N_f} \phi_{j,k}^f(x) f_{j,k}^f(x). \quad (27)$$

This structure mirrors the additive skip-connection paradigm of residual networks, with the key distinction that the additional contributions are spatially weighted through nested POU partitions rather than added

as unstructured features. The coarse and fine levels provide first- and second-level localized contributions, respectively. Although we focus on a two-level hierarchy in this work, the construction extends naturally to deeper partition trees by recursively subdividing partitions and adding corresponding POU level contributions. The full hierarchical POU model is trained end-to-end by minimizing the appropriate physics-informed or data-driven loss for the target application.

5 Neural Operator Instantiations of the POU Framework

The Partition-of-Unity (POU) framework extends naturally to neural operator architectures.

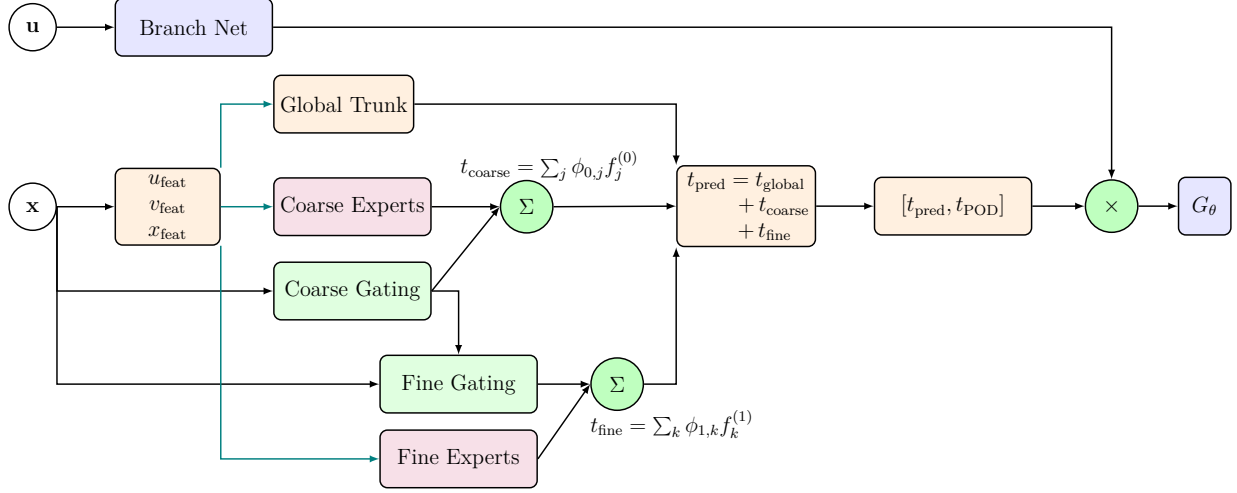


Figure 3: Residual-style hierarchical architecture used in HiRefPOU-DeepONet. The input x is embedded and passed through a global backbone to obtain $t_g(x)$. A coarse Mixture-of-Experts (MoE) layer predicts $t_c(x)$. A second MoE layer further produces fine-level contributions $t_f(x)$ through nested parent-child partitioning. The final output is the sum of all three components, $t(x) = t_g(x) + t_c(x) + t_f(x)$.

5.1 Hierarchical POU Deep Operator Networks (HiRefPOU-DeepONets)

We extend the POU framework to operator learning by introducing HiRefPOU-DeepONets, which incorporate spatially localized expert structure into the trunk representation of a Deep Operator Network (DeepONet) (Lu et al., 2021). As reviewed in Section 3, a standard DeepONet represents an operator through an inner product between (i) a branch network encoding an input function and (ii) a trunk network encoding evaluation coordinates.

The proposed construction mirrors the residual-style additive POU representation used in HiRefPOU for PDE approximation, but applies it instead to the coordinate-dependent component of the operator representation. In our formulation, the branch network remains global and maps sampled observations of the input function a to a coefficient vector $b(a) \in \mathbb{R}^{p+p_{\text{pod}}}$. The trunk representation is augmented with a POU mixture-of-experts defined over the evaluation coordinate y .

Specifically, we define a global trunk feature $t_g(y) \in \mathbb{R}^p$ together with a collection of N_c coarse local trunk experts $\{t_j(y) \in \mathbb{R}^p\}_{j=1}^{N_c}$. These experts are blended using smooth partition-of-unity weights $\{\phi_j(y)\}_{j=1}^{N_c}$ satisfying $\sum_{j=1}^{N_c} \phi_j(y) = 1$, yielding the POU-enhanced trunk feature

$$t_c(y) = t_g(y) + \sum_{j=1}^{N_c} \phi_j(y) t_j(y), \quad t_c(y) \in \mathbb{R}^p. \quad (28)$$

To incorporate problem-specific low-rank structure, we concatenate $t_{\text{pod}}(y)$ with a fixed POD basis vector $t_c(y) \in \mathbb{R}^{p_{\text{pod}}}$, forming the final trunk feature

$$t(y) = [t_c(y), t_{\text{pod}}(y)] \in \mathbb{R}^{p+p_{\text{pod}}}. \quad (29)$$

The operator prediction is then obtained via the standard DeepONet contraction

$$\hat{G}(a)(y) = \langle b(a), t(y) \rangle + b, \quad (30)$$

where $b \in \mathbb{R}$ is a learned bias. This formulation preserves the global expressivity of DeepONets while enabling spatially localized specialization through the POU-weighted trunk experts.

We further introduce a hierarchical two-level POU structure on the trunk side to organize localized operator representations through nested parent-child spatial partitions. At the coarse level, a gating network produces weights $\{\phi_j^c(y)\}_{j=1}^{N_c}$ satisfying $\sum_{j=1}^{N_c} \phi_j^c(y) = 1$, which blend a set of coarse trunk experts. Each coarse partition is then subdivided by a second gating network that produces child weights $\{\psi_{j,k}(y)\}_{k=1}^{N_f}$, locally normalized within each parent partition. The resulting fine-level POU weights are defined by

$$\phi_{j,k}^f(y) = \phi_j^c(y) \psi_{j,k}(y), \quad \sum_{k=1}^{N_f} \psi_{j,k}(y) = 1,$$

which guarantees the nested POU property $\sum_{j=1}^{N_c} \sum_{k=1}^{N_f} \phi_{j,k}^f(y) = 1$. The full hierarchical POU trunk representation is therefore given by

$$t_{\text{pou}}(y) = t_g(y) + \sum_{j=1}^{N_c} \phi_j^c(y) t_j^c(y) + \sum_{j=1}^{N_c} \sum_{k=1}^{N_f} \phi_{j,k}^f(y) t_{j,k}^f(y), \quad (31)$$

where t_j^c and $t_{j,k}^f$ denote coarse- and fine-level trunk experts, respectively. This hierarchical construction provides nested spatial localization of the operator representation while preserving global consistency through the POU structure.

5.2 Partition-of-Unity Fourier Neural Operators (POU-FNO)

In their standard form, FNOs act globally on the entire spatial domain, which can limit their ability to capture strongly localized features. We extend the partition-of-unity (POU) framework to FNOs by introducing a geometry-aware gating mechanism that blends multiple operator components using smooth, spatially localized weights. Importantly, the POU-FNO reuses the same geometry-aware gating network introduced in Section 4.3, with no changes to its functional form or training procedure. The resulting architecture combines global spectral operator learning with spatially localized POU weighting, enabling spatial adaptivity and localized feature learning. We emphasize that the POU-FNO is intended as a simple architectural extension illustrating the generality of the POU principle, rather than as a primary hierarchical contribution of this work.

For each spatial coordinate y_ℓ , a shared gating network produces logits $\ell(y_\ell) \in \mathbb{R}^{N_p}$, which are converted into partition weights as described in Section 4.3. The gate depends only on spatial coordinates and is shared across the batch, ensuring geometry-consistent partitions. Because the POU weights are applied pointwise in physical space, this construction preserves the global FFT structure of the FNO and avoids introducing artificial subdomain boundaries.

A single 1D FNO layer updates hidden features $h(y) \in \mathbb{R}^C$ via

$$\tilde{h}(y) = \mathcal{F}^{-1} \left(\sum_{|k| \leq K} \hat{W}_k \mathcal{F}[h]_k \right) + W_1 h(y), \quad (32)$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the forward and inverse discrete Fourier transforms, $\hat{W}_k \in \mathbb{C}^{C \times C}$ are learnable complex-valued spectral coefficients applied to the lowest K modes, and $W_1 \in \mathbb{R}^{C \times C}$ is a learned pointwise

linear transformation. Equation (32) is followed by a nonlinearity such as ReLU or GELU, and stacked for D layers to form an FNO block, which may be either shared across partitions or instantiated per partition depending on the architectural variant.

Given N_p expert predictions $\{\hat{v}_p(y)\}_{p=1}^{N_p}$, the final output is formed by a POU-weighted sum,

$$\hat{v}(y_\ell) = \sum_{p=1}^{N_p} \phi_p(y_\ell) \hat{v}_p(y_\ell), \quad (33)$$

which ensures smooth transitions between partitions. For $N_p = 1$, the model reduces exactly to a standard global FNO.

We consider two variations of this POU-FNO construction. In the first (POU-ExpertFNO), each partition is associated with an independent local FNO, resulting in a fully localized spectral mixture. In the second (POU-SharedFNO), which we use for most experiments, a single global FNO backbone is shared across all partitions, and locality is introduced through partition-specific expert heads applied to the shared spectral features. Both variants satisfy the POU aggregation in (33) and differ only in the degree to which spectral parameters are shared. In both variants, the partition-specific outputs are evaluated densely and subsequently combined using the POU weights. The gates therefore provide spatially varying output weighting rather than conditional selection of FNO experts or heads.

5.3 Computational Complexity of HiRefPOU-DeepONets

We summarize the computational complexity of the HiRefPOU-DeepONet architectures in terms of the branch, trunk, expert, and gating components. Let B denote the batch size and N_y the number of evaluation coordinates per input function. Let p denote the learned trunk latent dimension and p_{pod} the number of appended POD modes. We assume that the global trunk network and each local trunk expert have comparable hidden width w_t and depth L_t , while the gating networks have hidden width w_g and depth L_g . We denote the cost of evaluating the branch network on a batch of B input functions by $C_{\text{branch}}(B)$.

Because the trunk and gating networks depend only on the evaluation coordinate y , their cost depends on whether all batch elements share the same evaluation grid. We define

$$\eta_y = \begin{cases} N_y, & \text{if all batch elements share the same evaluation coordinates,} \\ BN_y, & \text{if evaluation coordinates differ across batch elements.} \end{cases}$$

The final DeepONet contraction is evaluated for every sample-coordinate pair and therefore scales as

$$\mathcal{O}(BN_y(p + p_{\text{pod}})).$$

The fixed POD basis does not introduce trainable parameters and contributes only through this final concatenation and contraction.

For HiRefPOU-1L with M local trunk experts, the trunk representation is

$$t_{\text{pou}}(y) = t_g(y) + \sum_{j=1}^M \phi_j(y) t_j(y).$$

Under the current dense implementation, all experts are evaluated at all query coordinates. The dominant forward-pass cost is therefore

$$C_{1L} = \mathcal{O}(C_{\text{branch}}(B) + \eta_y(1 + M)L_t w_t^2 + C_\phi(M) + BN_y(p + p_{\text{pod}})),$$

where $C_\phi(M)$ is the cost of evaluating the gate with M outputs. For a dense MLP gate,

$$C_\phi(M) = \mathcal{O}(\eta_y(L_g w_g^2 + w_g M + M d)),$$

where the Md term accounts for center-distance features in d spatial dimensions.

For HiRefPOU-2L, let M_c denote the number of coarse experts and M_f the total number of fine experts across all coarse parents. The hierarchical trunk representation contains $1 + M_c + M_f$ densely evaluated trunk networks, so the dominant forward-pass cost is

$$C_{2L} = \mathcal{O}(C_{\text{branch}}(B) + \eta_y(1 + M_c + M_f)L_t w_t^2 + C_{\phi_c}(M_c) + C_{\psi}(M_f) + BN_y(p + p_{\text{pod}})).$$

If each coarse parent has K_f children, then $M_f = M_c K_f$, recovering the balanced-tree scaling.

Ignoring lower-order input and output projections, the corresponding trainable parameter counts scale as

$$P_{1L} = \mathcal{O}(P_{\text{branch}} + (1 + M)L_t w_t^2 + L_g w_g^2 + w_g M),$$

and

$$P_{2L} = \mathcal{O}(P_{\text{branch}} + (1 + M_c + M_f)L_t w_t^2 + 2L_g w_g^2 + w_g M_c + w_g M_f).$$

More generally, for a hierarchy with L_h POU refinement levels and M_ℓ experts at level ℓ , the dense expert-evaluation cost scales linearly with the total number of experts:

$$C_{\text{trunk}} = \mathcal{O}\left(\eta_y \left(1 + \sum_{\ell=1}^{L_h} M_\ell\right) L_t w_t^2\right).$$

Thus, the practical cost of the current implementation grows with the total number of trunk experts across levels. Since all experts are evaluated densely at all query coordinates, the current implementation does not obtain computational savings from sparse routing. An ideal routed implementation could reduce the constant factor by evaluating only the active experts associated with each coordinate, but this is not used in the experiments reported here.

6 Benchmark Problems and Implementation Details

6.1 Baseline Comparison Methods

To evaluate the proposed HiRefPOU framework, we primarily benchmark against established neural operator (NO) architectures. Our focus is on operator learning problems, where the objective is to approximate mappings between infinite-dimensional function spaces rather than individual PDE solutions. Accordingly, the majority of our numerical experiments compare HiRefPOU to representative operator-learning baselines under matched training protocols and, where possible, comparable DeepONet-family partition or parameter budgets.

Neural Operator Baselines. We compare against the following operator-learning methods:

- **Vanilla DeepONet:** the standard DeepONet architecture with a single global trunk network and no explicit spatial localization.
- **(P+1)-Vanilla DeepONet:** an overparameterized baseline consisting of $P + 1$ independent global trunk networks, where P matches the number of partitions used in POU-based models. This provides a parameter-enlarged global baseline for assessing whether increased global model capacity alone reproduces the gains obtained from spatial localization. Exact parameter counts and remaining capacity differences are reported in Table 18
- **(POD):** a DeepONet augmented with a fixed Proper Orthogonal Decomposition (POD) basis in the trunk, providing a global low-rank representation of the solution space.
- **(Vanilla, POU)** (Sharma & Shankar, 2025): a DeepONet with a partition-of-unity trunk using fixed (non-adaptive) spatial partitions, isolating the effect of localization without adaptive gating.

- **(POD, POU)** (Sharma & Shankar, 2025): a localized DeepONet combining global POD modes with fixed POU-based spatial experts.
- **(Vanilla, POD, POU)** (Sharma & Shankar, 2025): a hybrid DeepONet that includes global vanilla, global POD, and localized POU expert components, testing whether additional global capacity improves upon POD–POU alone.
- **Geo-Fourier Neural Operator (GeoFNO)** (Li et al., 2023): which extends the Fourier Neural Operator to irregular geometries and performs global spectral convolution in Fourier space. GeoFNO is particularly well-suited for learning solution operators of PDEs posed on complex domains.
- **LocalFNO** (Liu-Schiaffini et al., 2024): a Fourier neural operator augmented with localized integral and/or differential kernel layers. Unlike standard FNOs, which rely entirely on global spectral convolutions, this architecture introduces explicit local receptive fields. We include LocalFNO as a strong locality-aware spectral baseline for comparison with our POU-based localized operator models.

These models provide strong baselines for assessing predictive accuracy and measured computational cost in operator-learning settings.

Motivating Physics-Informed Case Study. In addition to operator-learning benchmarks, we include a limited set of physics-informed neural network (PINN) experiments as a motivating case study. Specifically, we consider a two-dimensional Darcy flow problem with discontinuous coefficients, which highlights known limitations of globally smooth neural approximations when applied to interface-dominated PDEs. These experiments are not intended as an exhaustive comparison with PINN or domain-decomposition variants such as XPINNs, but rather to illustrate why localized representations are advantageous for problems with sharp material interfaces. For completeness, we compare against representative PINN-based methods in this setting, including: the standard PINN formulation (Raissi, 2018), Fourier-feature PINNs (Wang et al., 2021b), and PirateNets (Wang et al., 2024). Additional PINN experiments and implementation details are provided in Appendix B.

6.2 Training Setup and Evaluation Metrics

To assess predictive accuracy, we report the relative ℓ_2 error, which quantifies the discrepancy between the network prediction and the ground truth solution. Specifically, the error is defined as

$$\ell_2 = \frac{\sqrt{\sum_{i=1}^N |u(x_i, t_i) - \hat{u}(x_i, t_i)|^2}}{\sqrt{\sum_{i=1}^N |u(x_i, t_i)|^2}}, \quad (34)$$

where u denotes the exact solution and $\hat{u}$ corresponds to the approximation produced by the neural network. This metric provides a relative measure of the error by normalizing with respect to the magnitude of the reference solution. For the operator-learning benchmarks, we report relative ℓ_2 and, in the detailed tables, ℓ_∞ errors together with training and inference times. For the physics-informed Darcy case study, residual and boundary losses are used during training, while the reported metrics focus on solution and flux errors.

Timing measurements were collected after model compilation and warm-up, with device execution synchronized before recording wall-clock time. For the full-batch DeepONet experiments, the reported training time corresponds to a single optimizer update evaluated over the entire training set. For the FNO experiments, the reported training time corresponds to one complete pass through the training set (i.e., one training epoch) and therefore includes multiple optimizer updates. In both cases, the timed training region includes the forward model evaluation, loss evaluation, reverse-mode differentiation, and parameter update, while excluding periodic metric evaluation and checkpoint writing. The reported inference time denotes the wall-clock time required to evaluate the complete test set.

Unless otherwise specified, benchmark-specific optimization settings are reported in Appendix Table 15. Each DeepONet-family model is trained for 150,000 gradient updates without early stopping, and the reported

results use the final training checkpoint. Results are reported as the mean and standard deviation across five independent runs. The fixed train/test datasets listed in Table 17 were held constant across all compared methods, and no separate validation split was used. The test set was not used to select or tune the primary model configurations reported in the main benchmark comparisons. The additional gate, partition-count, residual-scaling, and child-allocation ablations reported in Appendices C and D were conducted post hoc, after the primary model configurations had been fixed, and are included only as retrospective sensitivity analyses.

For the Vanilla DeepONet, $(P + 1)$ -Vanilla DeepONet, POD-POU, and related DeepONet baselines, we followed the architectures used in the corresponding prior publication. The HiRefPOU-1L and HiRefPOU-2L architectures were selected to approximately match the trainable parameter counts of these competing DeepONet models where possible, although exact matching was not always feasible because of differences in model structure and partition configuration. GeoFNO and LocalFNO were reimplemented for the present study, with architectures selected to provide parameter counts comparable to the other models where feasible. The experiments were not designed as an exhaustive equal-budget hyperparameter or architecture search; rather, baseline configurations were inherited from prior work when available, while the proposed HiRefPOU and reimplemented FNO configurations were selected with approximate parameter comparability as a primary consideration. Accordingly, we interpret the comparisons as applying to the reported model configurations rather than as the outcome of an equal-budget neural architecture search.

Training was conducted on an NVIDIA H100 GPU under JAX with single-precision arithmetic. All code was implemented in Python 3.10.18 using JAX 0.6.2, with DeepMind Haiku 0.0.14 for neural network modules and Optax 0.2.5 for optimization¹. Complete architecture, training, dataset, and solver specifications are provided in Appendix A.

6.3 Training Objectives

We next summarize the training objectives used for the motivating PINN example in Section 7.1 and the operator-learning experiments considered throughout the remainder of this section. For the Darcy flow experiment, the baseline PINN models are trained using the standard physics-informed objective defined in Eq. (2). For the HiRefPOU variants, this objective is augmented with the geometric POU regularizers introduced in Section 4.5:

$$L = L_{\text{PINN}} + L_{\text{cov}} + L_{\text{cont}}, \quad (35)$$

where L_{cov} and L_{cont} are defined in Eqs. (23) and (25), respectively. The weighting coefficients λ_{cov} and λ_{cont} are included directly in the definitions of these regularization terms.

The operator-learning models are trained using the data-driven objective introduced in Eq (6). The HiRefPOU variants are similarly combined with the geometric POU regularization described in Section 4.5 yielding the complete training objective:

$$L = L_{\text{op}} + L_{\text{cov}} + L_{\text{cont}}. \quad (36)$$

The hierarchical POU construction does not introduce separate loss terms for the individual refinement levels. In particular, the coarse- and fine-level contributions are not trained against independently defined residual targets. Instead, the loss is evaluated on the final model prediction, and the global network, local experts, and gating parameters at all hierarchy levels are optimized jointly through the corresponding objective above. Thus, the term residual-style refers to the additive composition of the architecture described in Section 4, rather than to a separate hierarchical residual loss. No staged training, parameter freezing, detached residual targets, or stop-gradient operations are used. All global-network, local-expert, and gating parameters are optimized simultaneously through the complete objective, and every hierarchy level contributes to the final prediction and receives gradients from the corresponding prediction loss throughout training.

6.4 HiRefPOU Model Variants

For each problem, we compare the following variants of the proposed hierarchical POU architecture, which differ in the number and organization of learned POU refinement levels. In each variant, the gate parameters

¹The code, configuration files, and scripts used to reproduce the reported results will be released publicly upon publication.

are optimized jointly with the remaining model parameters. In the operator-learning examples, the resulting coordinate-dependent partitions are shared across samples.

- **HiRefPOU-1L**: a single-level POU model ($u(x) = u_g(x) + u_0(x)$) in which the coarse gating network is co-trained jointly with the global network and local experts.
- **HiRefPOU-2L**: the full hierarchical POU model ($u(x) = u_g(x) + u_0(x) + u_1(x)$) in which the coarse and fine gates are co-trained along all other model parameters throughout PDE/operator training.
- **HiRefPOU-3L**: a three-level POU model ($u(x) = u_g(x) + u_0(x) + u_1(x) + u_2(x)$) that tests whether an additional level of nested spatial partitioning provides additional accuracy beyond the 2L hierarchy.

Further, to demonstrate the generality of the proposed POU framework, we additionally embed the POU mechanism within neural operator architectures as described in Section 5.2. These extensions are not hierarchical, but instead aim to illustrate how POU-based localization can improve operator learning in FNO models. Specifically, we consider the following POU-FNO realizations:

- **POU-SharedFNO**: a FNO with a shared global spectral backbone and partition-specific expert heads.
- **POU-ExpertFNO**: a more expressive but costly variant in which each partition is associated with an independent local FNO, yielding a fully localized spectral mixture.

7 Numerical Results

7.1 Motivating PINN Example: Darcy Flow with Discontinuous Coefficients

We begin with a two-dimensional Darcy flow problem with discontinuous coefficients, included as a motivating example to illustrate the limitations of globally smooth PINNs on interface-dominated PDEs. Darcy flow serves as a canonical elliptic benchmark in heterogeneous media, where accurate flux recovery across material interfaces is essential. The Darcy experiment is included as a motivating physics-informed case study, while the remaining experiments evaluate the proposed localization mechanisms in the operator-learning setting.

We consider the steady Darcy problem

$$-\nabla \cdot (\mu(x) \nabla \phi(x)) = f(x), \quad x \in \Omega = [0, 1]^2, \quad (37)$$

with mixed Dirichlet–Neumann boundary conditions. The permeability field $\mu(x)$ is piecewise constant and discontinuous, inducing sharp material interfaces.

Five-strip benchmark. We employ the standard five-strip manufactured-solution test (Trask et al., 2017; Nakshatrala et al., 2006), in which the domain is partitioned into horizontal layers with distinct permeability values. The exact solution is chosen such that the pressure and normal flux remain continuous across interfaces, while tangential flux components exhibit discontinuities.

Results. Figure 14 and Table 3 compare solution and flux predictions for representative PINN-based methods and the proposed HiRefPOU. Both HiRefPOU variants substantially reduce the solution and flux errors relative to PirateNets and APINNs, demonstrating the benefit of localized, partition-based representations for PDEs with discontinuous coefficients. However, the additional hierarchical level added in the HiRefPOU-2L variant does not provide a consistent advantage over the -1L variant across the reported metrics. Further technical details, interface conditions, and additional PINN results are provided in Appendix B.

7.2 2D Lid-Driven Cavity Flow

Problem Setup. We next consider the two-dimensional lid-driven cavity flow problem, a classical benchmark for incompressible fluid dynamics in a confined domain. The flow is governed by the incompressible Navier–Stokes

Table 3: Solution and flux errors for the 2D rough Darcy flow benchmark using HiRefPOU-1L/-2L, APINN, and PirateNet. For each column, the best value is shown in **bold**, and the second-best value is underlined. Rows corresponding to the proposed adaptive POU-based models are highlighted.

Model	(N_c, N_f)	Rel. ℓ_2 u Error	Rel. ℓ_2 flux-x Error	Abs. flux-y Error
PirateNet	$(-, -)$	$5.1 \times 10^{-5} \pm 2.9 \times 10^{-5}$	$2.8 \times 10^{-5} \pm 9.0 \times 10^{-6}$	$8.0 \times 10^{-2} \pm 5.5 \times 10^{-2}$
APINN	$(2, -)$	$3.7 \times 10^{-5} \pm 1.8 \times 10^{-5}$	$2.7 \times 10^{-5} \pm 1.4 \times 10^{-5}$	$4.1 \times 10^{-2} \pm 1.9 \times 10^{-2}$
HiRefPOU-1L	$(2, -)$	$1.5 \times 10^{-7} \pm 4.6 \times 10^{-8}$	<u>$1.6 \times 10^{-7} \pm 2.9 \times 10^{-8}$</u>	$1.3 \times 10^{-4} \pm 1.2 \times 10^{-5}$
HiRefPOU-2L	$(2, 3)$	<u>$1.8 \times 10^{-7} \pm 3.8 \times 10^{-7}$</u>	$1.2 \times 10^{-7} \pm 1.9 \times 10^{-7}$	<u>$1.4 \times 10^{-4} \pm 2.2 \times 10^{-2}$</u>

equations

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla p + \nu \Delta \mathbf{u}, \quad \nabla \cdot \mathbf{u} = 0, \quad \mathbf{x} \in \Omega, \quad t \in T, \quad (38)$$

$$\mathbf{u} = \mathbf{u}_b, \quad (39)$$

where $\mathbf{u} = (u, v)$ is the velocity field, p is the pressure, and ν denotes the kinematic viscosity. The Dirichlet boundary condition $\mathbf{u} = \mathbf{u}_b$ enforces the prescribed lid motion.

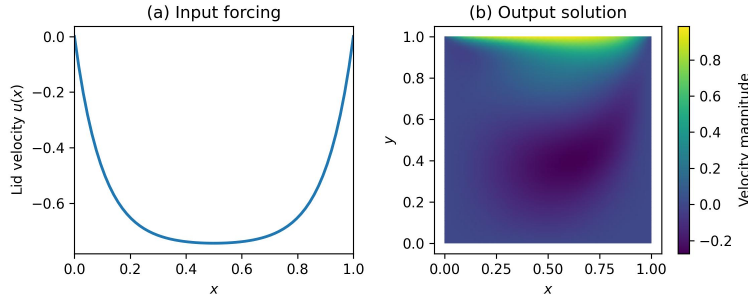


Figure 4: Example input–output pair from the lid-driven cavity flow experiment. (a) One-dimensional lid velocity profile $u(x)$ prescribed along the top boundary of the cavity and used as the model input. (b) Corresponding two-dimensional cavity flow solution for a representative test instance.

We learn an operator mapping from the one-dimensional lid velocity profile prescribed along the top boundary to the resulting steady-state two-dimensional velocity field:

$$\mathcal{G} : u_b(x) \mapsto \mathbf{u}(x, y).$$

An example input–output pair is shown in Figure 4. We consider the steady-state regime on the unit square domain $\Omega = [0, 1]^2$, following the dataset specification of (Lu et al., 2022). The lid velocity profile is defined as

$$u_b(x) = U \left(1 - \frac{\cosh(r(x - \frac{1}{2}))}{\cosh(\frac{r}{2})} \right), \quad v_b = 0, \quad (40)$$

with $r = 10$. All remaining boundaries are stationary. Steady-state solutions are generated using a lattice Boltzmann method (LBM) solver. The dataset consists of 100 training and 10 test input–output pairs, following (Lu et al., 2022; Sharma & Shankar, 2025).

Results. Quantitative results for the lid-driven cavity flow problem are reported in Tables 4 and 5. Among DeepONet-based models, the HiRefPOU variants substantially reduce both relative ℓ_2 and ℓ_∞ errors compared with the vanilla and static-partition baselines. HiRefPOU-1L achieves the lowest relative ℓ_2 and ℓ_∞ errors among the DeepONet-family models. Among the HiRefPOU variants, however, increasing hierarchy depth does not improve the aggregate error metrics. HiRefPOU-1L achieves the lowest mean relative ℓ_2 and ℓ_∞ errors, while the 2L model remains comparable and the 3L model is less accurate. The differences between

Table 4: Testing relative l_2 errors for the **Burgers**, **cavity flow**, **Full-Field Burgers**, **Kuramoto–Sivashinsky**, **2D reaction–diffusion**, and **3D reaction–diffusion** problems. RD stands for reaction–diffusion. For each problem, the best result across all models is shown in **bold**, and the second-best result is underlined. Rows corresponding to the proposed adaptive PoU-based models are highlighted.

Model		Burgers	Cavity flow
DeepONets	Vanilla	$2.24 \times 10^{-1} \pm 3.0 \times 10^{-3}$	$4.86 \times 10^{-2} \pm 6.2 \times 10^{-3}$
	(P+1)-Vanilla	$2.25 \times 10^{-1} \pm 6.5 \times 10^{-3}$	$2.05 \times 10^{-2} \pm 3.3 \times 10^{-3}$
	(POD, PoU)	$1.95 \times 10^{-1} \pm 1.4 \times 10^{-3}$	$2.28 \times 10^{-3} \pm 1.5 \times 10^{-4}$
	(Vanilla, PoU)	$2.43 \times 10^{-1} \pm 1.1 \times 10^{-2}$	$1.73 \times 10^{-2} \pm 4.7 \times 10^{-3}$
	(Vanilla, POD, PoU)	$1.92 \times 10^{-1} \pm 2.4 \times 10^{-3}$	$2.16 \times 10^{-3} \pm 1.7 \times 10^{-4}$
	HiRefPOU-1L (POD)	$1.75 \times 10^{-1} \pm 1.1 \times 10^{-3}$	$1.55 \times 10^{-3} \pm 1.3 \times 10^{-4}$
	HiRefPOU-2L (POD)	$1.74 \times 10^{-1} \pm 1.2 \times 10^{-3}$	<u>$1.69 \times 10^{-3} \pm 1.2 \times 10^{-4}$</u>
	HiRefPOU-3L (POD)	$1.74 \times 10^{-1} \pm 2.1 \times 10^{-3}$	<u>$1.79 \times 10^{-3} \pm 9.6 \times 10^{-5}$</u>
FNOs	GeoFNO	$3.56 \times 10^{-1} \pm 3.0 \times 10^{-2}$	$5.56 \times 10^{-2} \pm 1.2 \times 10^{-2}$
	LocalFNO	$1.16 \times 10^{-1} \pm 4.4 \times 10^{-2}$	$1.38 \times 10^{-2} \pm 1.2 \times 10^{-3}$
	POU-SharedFNO	$1.74 \times 10^{-1} \pm 3.7 \times 10^{-2}$	$4.15 \times 10^{-2} \pm 1.5 \times 10^{-2}$
	POU-ExpertFNO	<u>$1.24 \times 10^{-1} \pm 1.3 \times 10^{-2}$</u>	$8.07 \times 10^{-3} \pm 4.4 \times 10^{-5}$
Model		Full-Field Burgers	Kuramoto–Sivashinsky
DeepONets	Vanilla	$1.02 \times 10^{-1} \pm 2.0 \times 10^{-3}$	$1.70 \pm 1.3 \times 10^{-1}$
	(P+1)-Vanilla	$1.27 \times 10^{-1} \pm 2.8 \times 10^{-3}$	$1.68 \pm 1.3 \times 10^{-1}$
	(POD, PoU)	$8.84 \times 10^{-2} \pm 2.3 \times 10^{-3}$	$1.59 \pm 2.2 \times 10^{-2}$
	(Vanilla, PoU)	$1.01 \times 10^{-1} \pm 1.6 \times 10^{-3}$	$1.58 \pm 1.2 \times 10^{-2}$
	(Vanilla, POD, PoU)	$9.06 \times 10^{-2} \pm 1.0 \times 10^{-3}$	$1.50 \pm 1.4 \times 10^{-2}$
	HiRefPOU-1L (POD)	$8.76 \times 10^{-2} \pm 1.5 \times 10^{-3}$	$1.25 \pm 6.3 \times 10^{-2}$
	HiRefPOU-2L (POD)	$8.62 \times 10^{-2} \pm 2.2 \times 10^{-3}$	$1.33 \pm 2.3 \times 10^{-2}$
	HiRefPOU-3L (POD)	$8.68 \times 10^{-2} \pm 1.1 \times 10^{-3}$	$1.39 \pm 2.2 \times 10^{-2}$
FNOs	GeoFNO	$4.21 \times 10^{-2} \pm 3.5 \times 10^{-3}$	$9.03 \times 10^{-1} \pm 4.5 \times 10^{-3}$
	LocalFNO	$3.17 \times 10^{-2} \pm 4.3 \times 10^{-3}$	$7.85 \times 10^{-1} \pm 9.9 \times 10^{-3}$
	POU-SharedFNO	$2.49 \times 10^{-2} \pm 3.4 \times 10^{-3}$	$7.62 \times 10^{-1} \pm 1.5 \times 10^{-2}$
	POU-ExpertFNO	<u>$2.73 \times 10^{-2} \pm 2.3 \times 10^{-3}$</u>	<u>$7.80 \times 10^{-1} \pm 1.9 \times 10^{-2}$</u>
Model		2D RD	3D RD
DeepONets	Vanilla	$1.61 \times 10^{-3} \pm 4.4 \times 10^{-4}$	$1.20 \times 10^{-3} \pm 3.6 \times 10^{-4}$
	(P+1)-Vanilla	$5.77 \times 10^{-4} \pm 2.8 \times 10^{-4}$	$1.30 \times 10^{-1} \pm 1.4 \times 10^{-1}$
	(POD, PoU)	$4.74 \times 10^{-4} \pm 9.8 \times 10^{-5}$	$4.46 \times 10^{-4} \pm 2.0 \times 10^{-4}$
	(Vanilla, PoU)	$7.81 \times 10^{-4} \pm 2.1 \times 10^{-4}$	$5.57 \times 10^{-4} \pm 3.1 \times 10^{-4}$
	(Vanilla, POD, PoU)	$7.74 \times 10^{-4} \pm 1.4 \times 10^{-4}$	$4.22 \times 10^{-4} \pm 1.6 \times 10^{-4}$
	HiRefPOU-1L (POD)	$4.48 \times 10^{-4} \pm 2.4 \times 10^{-4}$	$3.68 \times 10^{-4} \pm 2.2 \times 10^{-4}$
	HiRefPOU-2L (POD)	<u>$3.68 \times 10^{-4} \pm 1.3 \times 10^{-4}$</u>	<u>$3.12 \times 10^{-4} \pm 1.6 \times 10^{-4}$</u>
	HiRefPOU-3L (POD)	$3.56 \times 10^{-4} \pm 6.2 \times 10^{-5}$	$2.55 \times 10^{-4} \pm 3.0 \times 10^{-5}$
FNOs	GeoFNO	$4.84 \times 10^{-2} \pm 1.1 \times 10^{-1}$	$8.56 \times 10^{-1} \pm 5.2 \times 10^{-1}$
	LocalFNO	$6.31 \times 10^{-3} \pm 4.9 \times 10^{-6}$	$9.30 \times 10^{-3} \pm 2.3 \times 10^{-5}$
	POU-SharedFNO	$7.47 \times 10^{-3} \pm 2.1 \times 10^{-4}$	1.02 ± 1.2
	POU-ExpertFNO	$4.70 \times 10^{-3} \pm 1.1 \times 10^{-4}$	$9.98 \times 10^{-1} \pm 4.1 \times 10^{-2}$

1L and 2L are also modest relative to the reported variability across seeds. Thus, the cavity experiment supports adaptive localization, but does not provide evidence that a deeper hierarchy is necessary for this operator. The FNO-based results are mixed: LocalFNO and POU-ExpertFNO are competitive in relative ℓ_2 , but the HiRefPOU-DeepONet variants achieve the best errors.

Figure 5 provides a complementary spatial comparison between the 1L and 2L models. Although their aggregate errors are similar, the error-difference field shows that the 2L model reduces error in some localized regions while increasing it in others. Figure 6 likewise shows that the hierarchical construction subdivides the learned coarse regions into finer child partitions. These results illustrate the additional spatial organization provided by the hierarchy, but do not imply that this organization produces uniformly lower prediction error.

Finally, Table 5 reports training and inference times. The adaptive HiRefPOU models incur additional cost relative to the simplest vanilla DeepONet variants, reflecting the overhead of gating and expert evaluation. However, this overhead is moderate within the DeepONet family and accompanies substantial improvements in both relative ℓ_2 and ℓ_∞ error. Thus, the cavity results support an accuracy–cost tradeoff: adaptive partitioning improves predictive accuracy and interpretability, while requiring additional computation relative to the simplest baselines.

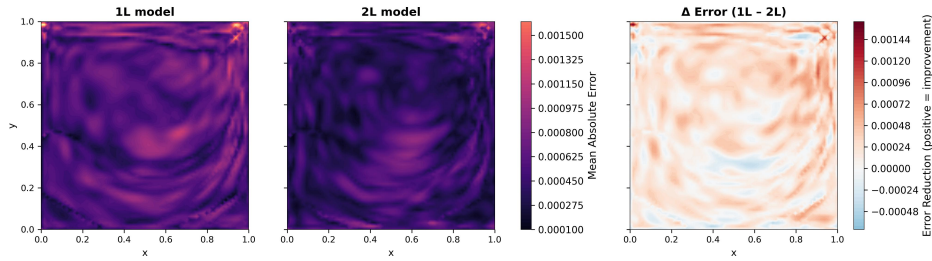


Figure 5: Comparison of single-level (1L) and hierarchical two-level (2L) HiRefPOU DeepONet models for the cavity flow problem. Both models employ six total partitions initialized at identical center locations. The left and center panels show the mean absolute error (MAE) fields for the 1L (e_{1L}) and 2L (e_{2L}) architectures, respectively. The rightmost panel displays the local error difference $\delta e = e_{1L} - e_{2L}$, where positive values indicate regions in which the hierarchical model improves upon the single-level model.

Table 5: Testing ℓ_∞ errors, training times, and inference times for the Cavity Flow problem. For the error and inference-time columns, the best value is shown in bold and the second-best value is underlined. Training times are reported for computational context but are not ranked across model families because the reported unit differs between full-batch DeepONet and minibatch FNO models. Ties are bolded when reported values are equal to the displayed precision. Rows corresponding to proposed adaptive POU-based models are highlighted.

	Model	ℓ_∞ Error	Mean Training Time [†] (s)	Inference Time (s)
DeepONets	Vanilla	$1.67 \times 10^{-1} \pm 2.7 \times 10^{-2}$	$1.01 \times 10^{-3} \pm 5.8 \times 10^{-4}$	$9.54 \times 10^{-4} \pm 6.0 \times 10^{-4}$
	(P+1)-Vanilla	$7.02 \times 10^{-2} \pm 2.7 \times 10^{-2}$	$1.01 \times 10^{-3} \pm 7.4 \times 10^{-5}$	$1.46 \times 10^{-4} \pm 1.3 \times 10^{-5}$
	(Vanilla, POU)	$6.33 \times 10^{-2} \pm 2.6 \times 10^{-2}$	$5.44 \times 10^{-3} \pm 3.8 \times 10^{-3}$	$5.26 \times 10^{-3} \pm 3.8 \times 10^{-3}$
	(POD, POU)	$4.81 \times 10^{-3} \pm 3.7 \times 10^{-4}$	$3.33 \times 10^{-3} \pm 5.3 \times 10^{-5}$	$3.20 \times 10^{-3} \pm 7.1 \times 10^{-6}$
	(Vanilla, POD, POU)	$6.95 \times 10^{-3} \pm 2.0 \times 10^{-3}$	$4.42 \times 10^{-3} \pm 6.6 \times 10^{-5}$	$4.29 \times 10^{-3} \pm 1.1 \times 10^{-4}$
	HiRefPOU-1L (POD)	$3.03 \times 10^{-3} \pm 5.2 \times 10^{-4}$	$3.60 \times 10^{-3} \pm 7.4 \times 10^{-4}$	$4.74 \times 10^{-4} \pm 1.1 \times 10^{-4}$
	HiRefPOU-2L (POD)	$3.61 \times 10^{-3} \pm 4.6 \times 10^{-4}$	$4.05 \times 10^{-3} \pm 1.1 \times 10^{-3}$	$4.68 \times 10^{-4} \pm 8.5 \times 10^{-5}$
	HiRefPOU-3L (POD)	$4.85 \times 10^{-3} \pm 5.6 \times 10^{-4}$	$4.97 \times 10^{-3} \pm 7.7 \times 10^{-7}$	$4.57 \times 10^{-4} \pm 1.9 \times 10^{-5}$
FNOs	GeoFNO	$9.63 \times 10^{-2} \pm 2.0 \times 10^{-2}$	$1.92 \times 10^{-1} \pm 6.3 \times 10^{-2}$	$1.78 \times 10^{-1} \pm 1.4 \times 10^{-1}$
	LocalFNO	$1.98 \times 10^{-2} \pm 1.1 \times 10^{-3}$	$4.01 \times 10^{-2} \pm 1.2 \times 10^{-3}$	$2.37 \times 10^1 \pm 6.12 \times 10^{-1}$
	POU-SharedFNO (GeoFNO)	$9.71 \times 10^{-2} \pm 4.1 \times 10^{-2}$	$5.42 \times 10^{-2} \pm 9.0 \times 10^{-4}$	$1.88 \times 10^{-2} \pm 2.2 \times 10^{-4}$
	POU-ExpertFNO (GeoFNO)	$1.24 \times 10^{-2} \pm 8.0 \times 10^{-4}$	$1.05 \times 10^{-1} \pm 2.8 \times 10^{-5}$	$2.88 \times 10^{-2} \pm 1.1 \times 10^{-4}$

[†]For full-batch DeepONet-family models, training time denotes one optimizer update over the complete training set. For minibatch FNO-family models, it denotes one complete training epoch.

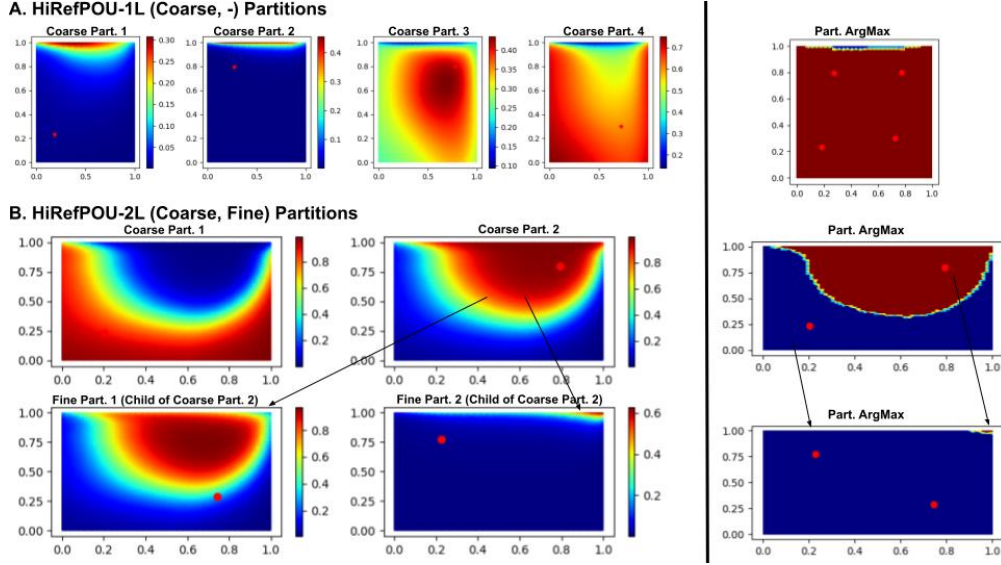


Figure 6: Learned partition-of-unity (POU) gating functions for the cavity flow problem using the single-level (1L) and hierarchical two-level (2L) HiRefPOU-DeepONet models. (A) Single-level (1L) model: spatially varying coarse gating functions associated with individual experts, together with the induced hard partition obtained via the pointwise ArgMax. The ArgMax maps are shown only to visualize the induced partition structure; model predictions use the continuous POU weights and dense evaluation of experts. (B) Hierarchical two-level (2L) model: learned coarse-level gating functions (top row) and their corresponding fine-level child partitions (bottom row). Rightmost panels show the induced hard partitions at each level. Colors indicate gating weight magnitude, with warmer colors corresponding to larger POU gating weights. The hierarchical model subdivides the learned coarse partitions into localized fine-level child regions while preserving the nested parent-child structure.

7.3 2D Reaction-Diffusion Problem

Problem Setup. We next consider a two-dimensional reaction-diffusion equation describing the spatiotemporal concentration $c(y, t)$ of a chemical species undergoing binding and unbinding reactions. The governing partial differential equation is

$$\frac{\partial c}{\partial t} = k_{\text{on}}(R - c) c_{\text{amb}} - k_{\text{off}} c + \nu \Delta c, \quad y \in \Omega, \quad t \in T, \quad (41)$$

with homogeneous Neumann boundary conditions

$$\nu \frac{\partial c}{\partial n} = 0, \quad y \in \partial\Omega. \quad (42)$$

Here, k_{on} and k_{off} denote reaction on- and off-rates, $\nu = 0.1$ is the diffusion coefficient, and $R = 2$ controls the available reactant concentration. The ambient concentration

$$c_{\text{amb}}(y, t) = [1 + \cos(2\pi y_1) \cos(2\pi y_2)] e^{-\pi t}$$

introduces spatial and temporal variability into the reaction term.

Both reaction coefficients are discontinuous in the horizontal direction y_1 :

$$k_{\text{on}}(y_1) = \begin{cases} 2, & y_1 \leq 1, \\ 0, & \text{otherwise,} \end{cases} \quad k_{\text{off}}(y_1) = \begin{cases} 0.2, & y_1 \leq 1, \\ 0, & \text{otherwise.} \end{cases} \quad (43)$$

This discontinuity induces a sharp transition across the interface $y_1 = 1$, producing localized gradients that are challenging for global operator-learning models. Our objective is to learn the operator

$$\mathcal{G} : c(y, 0) \mapsto c(y, T), \quad T = 0.5,$$

mapping the initial concentration field to the solution at the final time. The problem is defined on $\Omega = [0, 2]^2$ with initial condition $c(y, 0) \sim \mathcal{U}(0, 1)$. An example input–output pair is shown in Figure 7. Reference solutions are taken from Sharma & Shankar (2025), computed using an RBF–FD discretization (Shankar & Fogelson, 2018; Shankar et al., 2021). The dataset consists of 2207 spatial collocation points, with 1000 training and 200 test input–output pairs.

Results. We next compare the 1L, 2L, and 3L HiRefPOU DeepONet models on the 2D reaction–diffusion benchmark. Quantitative results are reported in Tables 4 and 6. All three HiRefPOU variants improve upon the vanilla and static-partition DeepONet baselines in both relative ℓ_2 and ℓ_∞ error, again demonstrating the benefit of learned spatial localization in the presence of the fixed coefficient discontinuity at $y_1 = 1$. The 1L and 2L models use the same total number of partitions and comparable parameter counts, so their performance differences isolate the effect of organizing the same partition budget hierarchically. The 3L model introduces an additional nested partition level and a larger finest-level partition count, so its comparison with the shallower variants reflects both hierarchy depth and increased model capacity.

Within the HiRefPOU family, the mean errors exhibit a favorable trend with additional hierarchy: HiRefPOU-3L has the lowest mean relative ℓ_2 and ℓ_∞ errors, followed by 2L and then 1L. However, these differences are modest relative to the reported variability across random seeds, and the corresponding mean ($\pm$) standard-deviation ranges overlap. We therefore interpret these results as evidence that hierarchical organization can be beneficial for this interface-dominated problem, rather than as evidence that increasing hierarchy depth is uniformly superior.

Figure 8 provides additional spatial context for the 1L–2L comparison. The error-difference field indicates regions near the coefficient interface in which the hierarchical model reduces the prediction error, together with regions where the single-level model remains competitive. Figure 9 shows the learned partitions, where the second level subdivides the coarse regions into more localized child partitions (shown in the bottom row of the figure). Taken together, the quantitative and spatial results suggest that the hierarchy can provide useful additional localization around persistent heterogeneous structure, while the magnitude of the resulting accuracy improvement remains problem- and initialization-dependent.

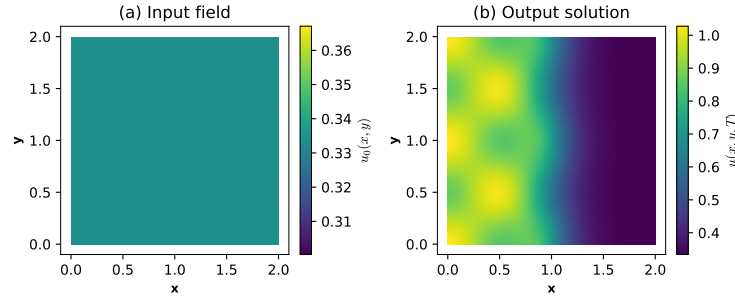


Figure 7: Example input–output pair for the two-dimensional reaction-diffusion problem. (a) Spatial input field $c(y, 0)$ provided as input to the operator-learning model. (b) Corresponding solution field $c(y, T)$ at the final time.

7.4 3D Reaction-Diffusion with Variable Diffusion Coefficient

Problem Setup. We next consider a three-dimensional reaction-diffusion system with a spatially varying diffusion coefficient, posed on a curved domain. Let $c(y, t)$ denote the concentration field with $y \in \mathbb{R}^3$. The governing equation is

$$\frac{\partial c}{\partial t} = k_{\text{on}}(R - c)c_{\text{amb}} - k_{\text{off}}c + \nabla \cdot (K(y)\nabla c), \quad y \in \Omega, t \in T, \quad (44)$$

with homogeneous Neumann boundary conditions

$$K(y)\frac{\partial c}{\partial n} = 0, \quad y \in \partial\Omega. \quad (45)$$

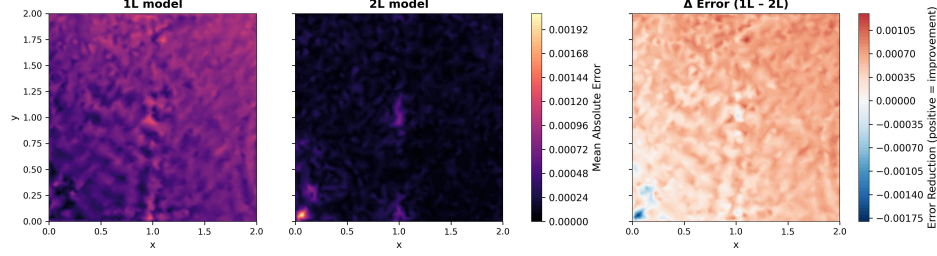


Figure 8: Comparison of single-level (1L) and hierarchical two-level (2L) HiRefPOUDeepONet models. Both models employ six total partitions with identical initial center locations. The left and center panels show the mean absolute error (MAE) fields for the 1L (e_{1L}) and 2L (e_{2L}) architectures, respectively. The rightmost panel displays the local error difference $\delta e = e_{1L} - e_{2L}$, where positive values indicate regions in which the hierarchical model improves upon the single-level model.

Table 6: Testing ℓ_∞ errors, training times, and inference times for the 2D Reaction-Diffusion problem. For the error and inference-time columns, the best value is shown in bold and the second-best value is underlined. Training times are reported for computational context but are not ranked across model families because the reported unit differs between full-batch DeepONet and minibatch FNO models. Rows corresponding to proposed adaptive PoU-based models are highlighted.

	Model	ℓ_∞ Error	Mean Training Time [†] (s)	Inference Time (s)
DeepONets	Vanilla	$5.65 \times 10^{-3} \pm 1.3 \times 10^{-3}$	$3.66 \times 10^{-4} \pm 4.3 \times 10^{-6}$	$8.74 \times 10^{-5} \pm 4.7 \times 10^{-6}$
	(P+1)-Vanilla	$1.81 \times 10^{-3} \pm 4.8 \times 10^{-4}$	$9.88 \times 10^{-4} \pm 3.2 \times 10^{-5}$	<u>$1.41 \times 10^{-4} \pm 6.1 \times 10^{-6}$</u>
	(Vanilla, POU)	$2.45 \times 10^{-3} \pm 5.5 \times 10^{-4}$	$2.84 \times 10^{-3} \pm 8.7 \times 10^{-4}$	$2.52 \times 10^{-3} \pm 1.2 \times 10^{-3}$
	(POD, POU)	$2.15 \times 10^{-3} \pm 6.4 \times 10^{-4}$	$6.05 \times 10^{-3} \pm 2.7 \times 10^{-3}$	$5.75 \times 10^{-3} \pm 3.1 \times 10^{-3}$
	(Vanilla, POD, POU)	$2.46 \times 10^{-3} \pm 2.2 \times 10^{-4}$	$7.38 \times 10^{-3} \pm 3.5 \times 10^{-4}$	$7.34 \times 10^{-3} \pm 3.2 \times 10^{-5}$
	HiRefPOU-1L (POD)	$1.22 \times 10^{-3} \pm 3.1 \times 10^{-4}$	$3.42 \times 10^{-3} \pm 9.6 \times 10^{-4}$	$3.76 \times 10^{-4} \pm 3.9 \times 10^{-5}$
	HiRefPOU-2L (POD)	<u>$1.20 \times 10^{-3} \pm 2.5 \times 10^{-4}$</u>	$4.72 \times 10^{-3} \pm 1.0 \times 10^{-3}$	$5.62 \times 10^{-4} \pm 1.3 \times 10^{-4}$
	HiRefPOU-3L (POD)	$1.14 \times 10^{-3} \pm 2.2 \times 10^{-4}$	$4.65 \times 10^{-3} \pm 4.0 \times 10^{-5}$	$5.73 \times 10^{-4} \pm 2.9 \times 10^{-5}$
FNOs	GeoFNO	$1.57 \times 10^{-1} \pm 3.3 \times 10^{-1}$	$5.30 \times 10^{-1} \pm 1.3 \times 10^{-1}$	$5.44 \times 10^{-2} \pm 5.3 \times 10^{-2}$
	LocalFNO	$2.27 \times 10^{-2} \pm 6.8 \times 10^{-6}$	$6.94 \times 10^{-2} \pm 1.2 \times 10^{-3}$	$7.16 \times 10^{-2} \pm 2.7 \times 10^{-3}$
	PoU-SharedFNO (GeoFNO)	$2.50 \times 10^{-2} \pm 3.5 \times 10^{-3}$	$5.35 \times 10^{-2} \pm 9.4 \times 10^{-5}$	$1.89 \times 10^{-2} \pm 9.2 \times 10^{-5}$
	PoU-ExpertFNO (GeoFNO)	$7.00 \times 10^{-2} \pm 1.5 \times 10^{-3}$	$7.51 \times 10^{-1} \pm 1.2 \times 10^{-4}$	$4.19 \times 10^{-2} \pm 1.1 \times 10^{-5}$

[†]For full-batch DeepONet-family models, training time denotes one optimizer update over the complete training set. For minibatch FNO-family models, it denotes one complete training epoch.

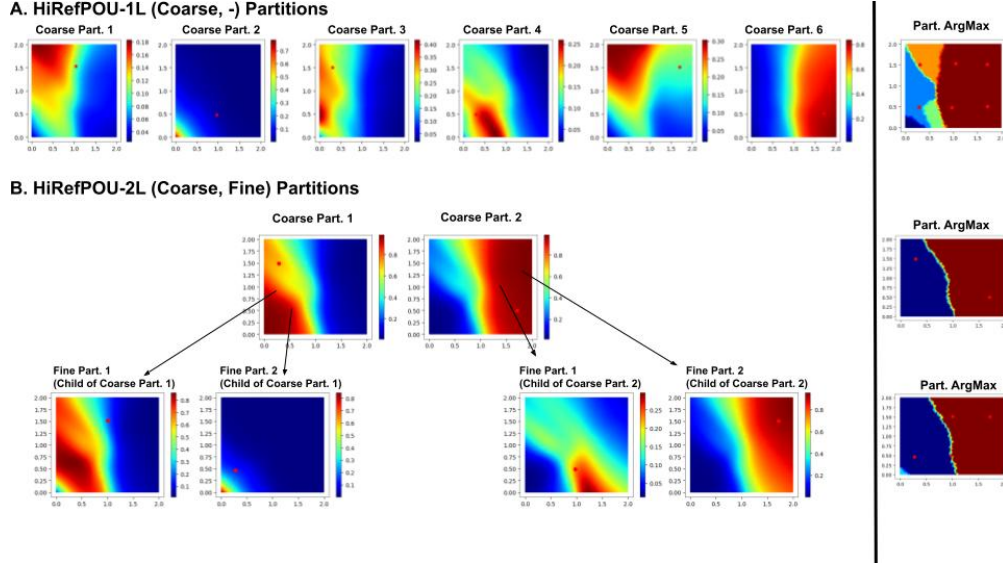


Figure 9: Comparison of single-level (1L) and hierarchical two-level (2L) HiRefPOU DeepONet learned partitions. Each panel shows the learned POU gating functions corresponding to different experts within the network. The top row illustrates the partitions produced by the single-level (1L) model, while the middle and bottom rows correspond to the hierarchical two-level (2L) model, where coarse partitions are further subdivided into locally refined subregions. The hierarchical model learns localized second-level child partitions near the coefficient discontinuity while preserving coherent coarse-level partitions across the domain.

The computational domain is the unit ball

$$\Omega = \{y \in \mathbb{R}^3 : \|y\|_2 \leq 1\},$$

with boundary $\partial\Omega = \mathbb{S}^2$, and the time interval is $T = [0, 0.5]$. An example input–output pair is shown in Figure 10. The reaction coefficients k_{on} and k_{off} follow the same configuration as in the two-dimensional case, taking nonzero values in the half-space $y_1 \leq 0$ and vanishing elsewhere. The ambient source term is defined as

$$c_{\text{amb}}(y, t) = [1 + \cos(2\pi y_1) \cos(2\pi y_2) \sin(2\pi y_3)] e^{-\pi t}.$$

The spatially varying diffusion coefficient $K(y)$ is designed to introduce sharp gradients,

$$K(y_1) = B + \frac{C}{\tanh(A)} \left[(A - 3) \tanh(8y_1 - 5) - (A - 15) \tanh(8y_1 + 5) + A \tanh(A) \right], \quad (46)$$

with $A = 9$, $B = 0.0215$, and $C = 0.005$. This configuration yields strong spatial heterogeneity in the diffusion operator. Our objective is to learn the solution operator

$$\mathcal{G} : c(y, 0) \mapsto c(y, 0.5).$$

Reference solutions are taken from Sharma & Shankar (2025) and computed using a high-order RBF-FD solver (Shankar & Fogelson, 2018; Shankar et al., 2021). The dataset consists of 1000 training and 200 test samples, with each function evaluated at 4325 spatial collocation points.

Results. Quantitative results for the three-dimensional reaction–diffusion problem are reported in Tables 4 and 7. The HiRefPOU variants achieve lower errors than the vanilla and static-partition DeepONet baselines, indicating that learned spatial localization remains effective in this three-dimensional heterogeneous setting. The mean results also favor the hierarchical variants: HiRefPOU-3L achieves the lowest relative ℓ_2 error, while HiRefPOU-2L achieves the lowest mean ℓ_∞ error.

These results provide the strongest indication among the tested problems that additional hierarchical organization can be useful when the operator contains persistent spatial heterogeneity at multiple spatial

scales. Nevertheless, the differences among the 1L, 2L, and 3L variants remain comparable to the reported run-to-run variability, particularly for the 1L and 2L models. We therefore do not interpret the ordering of the mean errors as evidence of monotonic improvement with hierarchy depth. Rather, the results suggest that additional levels can provide useful representational flexibility for this problem, with the preferred depth depending on the error metric and training realization.

Figure 11 further illustrates the geometric effect of the hierarchy. The single-level model produces smooth learned partitions over the ball, whereas the hierarchical model subdivides these regions through nested child partitions. The learned subdivision is concentrated in regions associated with strong spatial variation in the diffusion coefficient, providing qualitative evidence that the additional partition levels can organize localized model capacity around persistent heterogeneous structure.

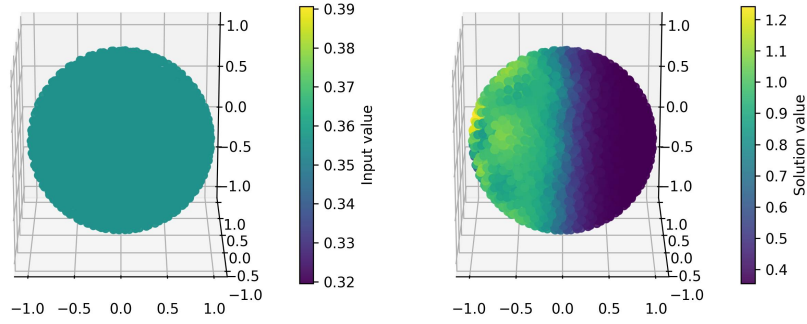


Figure 10: Example input-output pair for the three-dimensional reaction-diffusion problem. (Left) Spatial input field provided as input to the operator-learning model. (Right) Corresponding solution field at the final time.

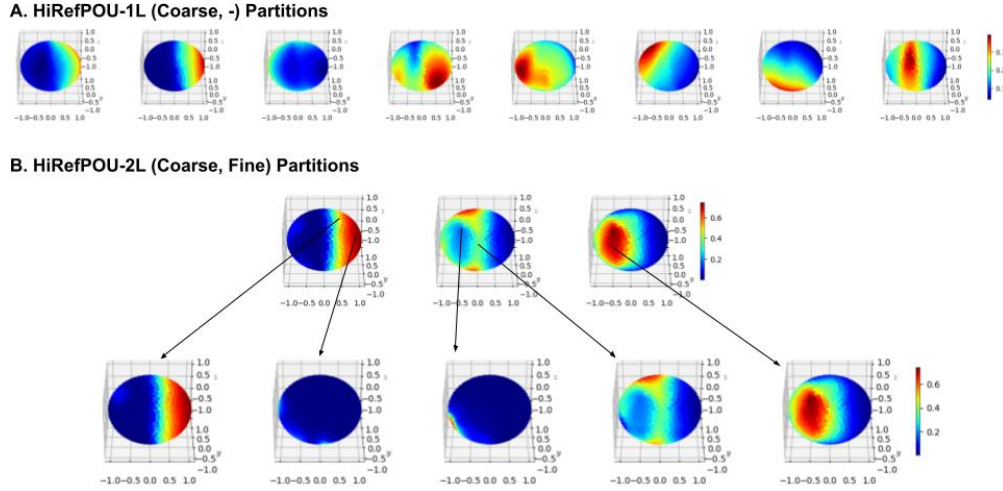


Figure 11: Comparison of single-level (1L) and hierarchical two-level (2L) HiRefPOU DeepONet learned partitions. Each panel shows the learned POU gating functions corresponding to different experts within the network. The top row illustrates the partitions produced by the single-level (1L) model, while the middle and bottom rows correspond to the hierarchical two-level (2L) model, where coarse partitions are further subdivided into locally refined subregions. The hierarchical model learns localized second-level child partitions aligned with regions of strong diffusion variation while preserving coherent coarse-level partitions across the ball domain.

Table 7: Testing ℓ_∞ errors, training times, and inference times for the 3D Reaction-Diffusion problem. For the error and inference-time columns, the best value is shown in bold and the second-best value is underlined. Training times are reported for computational context but are not ranked across model families because the reported unit differs between full-batch DeepONet and minibatch FNO models. Rows corresponding to proposed adaptive PoU-based models are highlighted.

	Model	ℓ_∞ Error	Mean Training Time [†] (s)	Inference Time (s)
DeepONets	Vanilla	$3.81 \times 10^{-3} \pm 1.5 \times 10^{-3}$	$4.61 \times 10^{-4} \pm 1.0 \times 10^{-4}$	$4.50 \times 10^{-4} \pm 1.2 \times 10^{-4}$
	(P+1)-Vanilla	$9.77 \times 10^{-2} \pm 1.1 \times 10^{-1}$	$1.69 \times 10^{-3} \pm 5.7 \times 10^{-4}$	$1.64 \times 10^{-3} \pm 5.9 \times 10^{-4}$
	(Vanilla, PoU)	$1.13 \times 10^{-3} \pm 5.1 \times 10^{-4}$	$1.40 \times 10^{-2} \pm 6.7 \times 10^{-3}$	$1.39 \times 10^{-2} \pm 6.7 \times 10^{-3}$
	(POD, PoU)	$1.83 \times 10^{-3} \pm 5.2 \times 10^{-4}$	$1.11 \times 10^{-2} \pm 9.7 \times 10^{-5}$	$1.09 \times 10^{-2} \pm 4.8 \times 10^{-5}$
	(Vanilla, POD, PoU)	$1.36 \times 10^{-3} \pm 5.0 \times 10^{-4}$	$1.13 \times 10^{-2} \pm 1.9 \times 10^{-4}$	$1.11 \times 10^{-2} \pm 1.1 \times 10^{-4}$
	HiRefPOU-1L (POD)	$6.87 \times 10^{-4} \pm 4.1 \times 10^{-4}$	$2.63 \times 10^{-3} \pm 1.0 \times 10^{-4}$	$3.54 \times 10^{-4} \pm 2.7 \times 10^{-5}$
	HiRefPOU-2L (POD)	$4.75 \times 10^{-4} \pm 1.9 \times 10^{-4}$	$2.86 \times 10^{-3} \pm 1.3 \times 10^{-4}$	<u>$4.38 \times 10^{-4} \pm 3.3 \times 10^{-5}$</u>
	HiRefPOU-3L (POD)	<u>$4.95 \times 10^{-4} \pm 5.2 \times 10^{-5}$</u>	$7.13 \times 10^{-3} \pm 5.7 \times 10^{-5}$	$6.16 \times 10^{-4} \pm 8.7 \times 10^{-6}$
FNOs	GeoFNO	$1.80 \pm 7.3 \times 10^{-1}$	$5.61 \pm 8.9 \times 10^{-3}$	$2.52 \times 10^{-1} \pm 2.5 \times 10^{-2}$
	LocalFNO	$3.20 \times 10^{-2} \pm 1.1 \times 10^{-3}$	$1.51 \times 10^{-1} \pm 8.4 \times 10^{-3}$	$8.36 \times 10^{-2} \pm 2.5 \times 10^{-3}$
	PoU-SharedFNO (GeoFNO)	$1.09 \pm 5.6 \times 10^{-2}$	$7.35 \pm 9.4 \times 10^{-2}$	$4.89 \times 10^{-1} \pm 9.2 \times 10^{-2}$
	PoU-ExpertFNO (GeoFNO)	$7.09 \times 10^{-1} \pm 1.6 \times 10^{-2}$	$1.05 \times 10^1 \pm 5.2 \times 10^{-1}$	$1.09 \pm 1.7 \times 10^{-1}$

[†]For full-batch DeepONet-family models, training time denotes one optimizer update over the complete training set. For minibatch FNO-family models, it denotes one complete training epoch.

7.5 1D Burgers' Equation

We next consider two one-dimensional Burgers benchmarks. The first is the standard periodic Gaussian-random-field benchmark used widely in operator-learning studies. The second is a modified multiscale periodic benchmark, constructed to contain more explicit mixed coarse- and fine-scale spatial structure. The purpose of the second benchmark is to more directly probe the regimes in which hierarchical adaptive partitioning is beneficial.

Problem Setup. We next consider the one-dimensional Burgers' equation:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad x \in (0, 1), \quad t \in (0, 1] \quad (47)$$

with periodic boundary conditions and viscosity $\nu = 0.001$. Examples of final-time solutions are shown in Figure 13.

In this problem, we learn the operator mapping from the initial condition $u(x, 0) = u_0(x)$ to the solution $u(x, t)$ at $t = 1$,

$$\mathcal{G} : u_0(x) \mapsto u(x, 1). \quad (48)$$

Similar to (Lu et al., 2022), we generate each dataset such that each initial condition is generated according to a random distribution:

$$\mu = \mathcal{N}(0, 625(-\Delta + 25I)^{-2}). \quad (49)$$

The initial condition u_0 is modeled as a sample from a Gaussian random field with mean zero and covariance operator

$$C = \sigma^2 (-\Delta + \tau^2 I)^{-\gamma}, \quad (50)$$

where $\sigma^2 = 25$, $\tau = 5$, and $\gamma = 2.5$. The field is defined on the periodic domain $(0, 1)$, and realizations are generated spectrally by drawing independent standard normal coefficients $\xi_k, \eta_k \sim \mathcal{N}(0, 1)$ and scaling by the eigenvalues of the covariance operator:

$$\lambda_k = \sqrt{2} |\sigma| ((2\pi k)^2 + \tau^2)^{-\gamma/2}.$$

The resulting Fourier coefficients are combined using the **Chebfun** package in MATLAB to construct smooth periodic realizations of the form

$$u_0(x) = \sum_{k=1}^K \alpha_k \cos(2\pi kx) + \beta_k \sin(2\pi kx),$$

where K denotes the number of retained Fourier modes and

$$\alpha_k = \lambda_k \xi_k, \quad \beta_k = \lambda_k \eta_k, \quad \xi_k, \eta_k \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1).$$

To generate the dataset, each realization of u_0 is evolved forward in time according to the Burgers' equation using a spectral solver implemented in **Chebfun**. This method employs adaptive Fourier expansions in space and high-order time-stepping schemes to ensure spectral accuracy. We discretize the spatial domain uniformly with $2^{13} = 8192$ collocation points and evolve the system over 500 uniform time steps up to $t = 1$. For each realization, we record both the initial condition $u_0(x)$ and the final-time solution $u(x, 1)$. This is a common benchmark problem for operator learning methods (Lu et al., 2022).

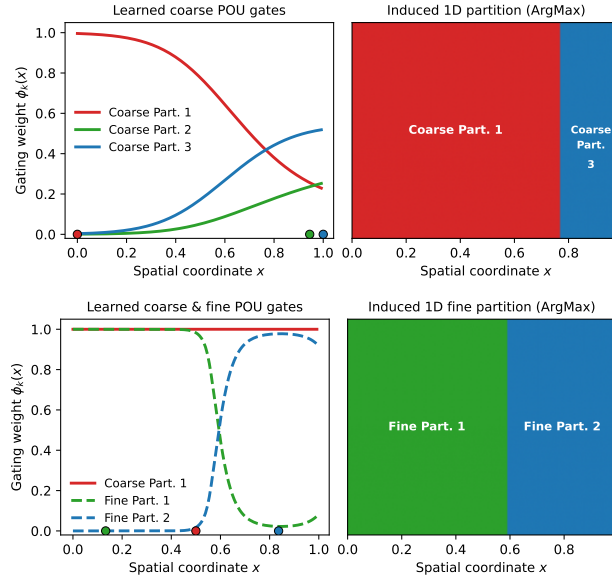


Figure 12: (*1D Burgers Problem.*) Learned partition-of-unity (POU) gates for HiRefPOU-DeepONet models. (Top left) Continuous coarse-level gating functions $\{\phi_j(x)\}_{j=1}^3$ learned by the HiRefPOU-1L model. Colored markers indicate the learned center locations associated with each partition. (Top right) Induced hard partition obtained via $\arg \max_j \phi_j(x)$, illustrating the discrete domain decomposition implied by the learned soft POU representation. (Bottom left) Continuous coarse- and fine-level gating functions $\{\phi_j^c(x)\}_{j=1}^1$ and $\{\phi_k^f(x)\}_{k=1}^2$ learned by the HiRefPOU-2L model. (Bottom right) Induced hard fine-level partition obtained via $\arg \max_k \phi_k^f(x)$.

Results. Table 4 reports testing relative ℓ_2 errors, and Table 8 reports testing ℓ_∞ errors, training times, and inference times for all evaluated operator-learning models on the 1D Burgers benchmark. Within the DeepONet family, the HiRefPOU variants improve upon the vanilla and static-partition baselines, showing that learned spatial localization is useful for this problem. In contrast, there is little evidence that increasing hierarchy depth provides an additional benefit: the 1L, 2L, and 3L models obtain nearly identical relative ℓ_2 errors, and their ℓ_∞ differences are small relative to the reported variability. These results suggest that much of the useful localization for the standard Burgers benchmark can already be captured by a single learned POU level. Additional nested subdivision therefore introduces model complexity without a clear corresponding improvement in predictive accuracy. This observation is consistent with the partition-count ablations in Appendix C, which show that relatively few learned partitions are sufficient for this benchmark.

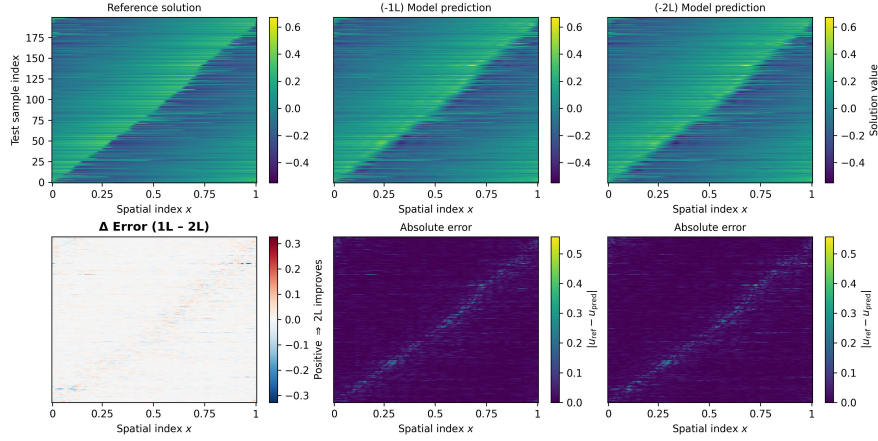


Figure 13: (*1D Burgers Problem.*) Operator learning predictions at final time $t = 1$. (Top row) Reference solution $u(x, 1)$, HiRefPOU-1L prediction, and HiRefPOU-2L prediction, evaluated over the full test set. Each heatmap displays solution values as a function of spatial coordinate $x \in [0, 1]$ (horizontal axis) and test sample index (vertical axis). (Bottom row) Point-wise absolute error fields $|u_{\text{ref}} - u_{\text{pred}}|$ for the 1L and 2L models, along with their difference. Positive values indicate locations where the 2L model achieves lower error than the 1L model.

Table 8: Testing ℓ_∞ errors, training times, and inference times for the 1D Burgers problem. For the error and inference-time columns, the best value is shown in bold and the second-best value is underlined. Training times are reported for computational context but are not ranked across model families because the reported unit differs between full-batch DeepONet and minibatch FNO models. For POU-FNO variants, the FNO backbone used by the model is indicated in parentheses. Rows corresponding to proposed adaptive PoU-based models are highlighted.

	Model	ℓ_∞ Error	Mean Training Time [†] (s)	Inference Time (s)
DeepONets	Vanilla	$5.17 \times 10^{-1} \pm 4.9 \times 10^{-3}$	$2.61 \times 10^{-4} \pm 5.6 \times 10^{-6}$	$5.56 \times 10^{-5} \pm 5.8 \times 10^{-7}$
	(P+1)-Vanilla	$5.35 \times 10^{-1} \pm 1.1 \times 10^{-2}$	$5.59 \times 10^{-4} \pm 1.7 \times 10^{-5}$	$1.30 \times 10^{-4} \pm 2.1 \times 10^{-5}$
	(Vanilla, POU)	$5.65 \times 10^{-1} \pm 2.1 \times 10^{-2}$	$6.60 \times 10^{-4} \pm 2.5 \times 10^{-5}$	$9.46 \times 10^{-5} \pm 6.5 \times 10^{-6}$
	(POD, POU)	$4.79 \times 10^{-1} \pm 5.2 \times 10^{-3}$	$5.36 \times 10^{-4} \pm 1.8 \times 10^{-5}$	<u>$9.22 \times 10^{-5} \pm 2.6 \times 10^{-6}$</u>
	(Vanilla, POD, POU)	$4.83 \times 10^{-1} \pm 6.7 \times 10^{-3}$	$6.60 \times 10^{-4} \pm 2.2 \times 10^{-5}$	<u>$9.27 \times 10^{-5} \pm 2.8 \times 10^{-6}$</u>
	HiRefPOU-1L (POD)	$4.57 \times 10^{-1} \pm 4.6 \times 10^{-3}$	$1.36 \times 10^{-3} \pm 7.8 \times 10^{-5}$	$1.55 \times 10^{-4} \pm 8.0 \times 10^{-6}$
	HiRefPOU-2L (POD)	$4.56 \times 10^{-1} \pm 5.3 \times 10^{-3}$	$2.46 \times 10^{-3} \pm 5.4 \times 10^{-4}$	$2.29 \times 10^{-4} \pm 3.4 \times 10^{-5}$
	HiRefPOU-3L (POD)	$4.56 \times 10^{-1} \pm 7.9 \times 10^{-3}$	$1.62 \times 10^{-3} \pm 8.6 \times 10^{-5}$	$2.66 \times 10^{-4} \pm 1.3 \times 10^{-5}$
FNOs	GeoFNO	$7.98 \times 10^{-1} \pm 4.2 \times 10^{-2}$	$2.39 \times 10^{-2} \pm 2.7 \times 10^{-2}$	$2.22 \times 10^{-3} \pm 7.2 \times 10^{-4}$
	LocalFNO	$2.77 \times 10^{-1} \pm 1.0 \times 10^{-1}$	$8.07 \times 10^{-2} \pm 2.5 \times 10^{-3}$	$7.30 \times 10^{-2} \pm 4.3 \times 10^{-4}$
	PoU-SharedFNO (GeoFNO)	$6.19 \times 10^{-1} \pm 1.7 \times 10^{-1}$	$4.19 \times 10^{-3} \pm 1.5 \times 10^{-4}$	$2.07 \times 10^{-3} \pm 1.3 \times 10^{-5}$
	PoU-ExpertFNO (GeoFNO)	$4.30 \times 10^{-1} \pm 9.3 \times 10^{-2}$	$1.09 \times 10^{-2} \pm 3.6 \times 10^{-5}$	$4.78 \times 10^{-3} \pm 2.2 \times 10^{-5}$
	PoU-SharedFNO (LocalFNO)	$4.01 \times 10^{-1} \pm 8.9 \times 10^{-2}$	$5.96 \times 10^{-2} \pm 1.1 \times 10^{-3}$	$3.70 \times 10^{-3} \pm 1.1 \times 10^{-4}$
	PoU-ExpertFNO (LocalFNO)	<u>$3.07 \times 10^{-1} \pm 3.2 \times 10^{-2}$</u>	$1.01 \times 10^{-1} \pm 6.8 \times 10^{-3}$	$5.38 \times 10^{-3} \pm 9.8 \times 10^{-5}$

[†]For full-batch DeepONet-family models, training time denotes one optimizer update over the complete training set. For minibatch FNO-family models, it denotes one complete training epoch.

The strongest overall results are obtained by LocalFNO. The regular periodic grid and translation-structured dynamics of this benchmark are particularly well matched to its local spectral inductive bias. The Burgers results therefore illustrate two complementary conclusions: learned POU localization improves the DeepONet representation, but neither deeper HiRefPOU hierarchy nor an additional POU decomposition on top of an already locality-aware Fourier backbone is uniformly advantageous.

7.6 Full-Field Burgers Benchmark

To more directly probe the value of deeper hierarchical partitioning, we introduce a synthetic multiscale Burgers benchmark on the periodic domain $\Omega = [-1, 1]$. Unlike the standard Burgers benchmark, which maps an initial condition to a single final-time solution profile, here the learning task is to map a one-dimensional initial condition to the full solution field over space-time. The dataset is constructed so that each realization contains nested coarse-, medium-, and fine-scale localized structure, thereby providing a setting in which additional hierarchy in the model can be meaningfully tested.

Problem Setup. We consider the viscous Burgers equation

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad x \in [-1, 1], \quad t \in [0, 1], \quad (51)$$

with periodic boundary conditions and viscosity $\nu = 10^{-3}$. For each sample, the operator-learning objective is to map the initial condition $u_0(x)$ to the full solution field $u(x, t)$.

Each initial condition is generated as the superposition of a smooth low-frequency periodic function and a nested three-level hierarchy of localized oscillatory packets,

$$u_0(x) = u_{\text{bg}}(x) + \sum_{\ell=1}^3 b_\ell B(x; c_\ell, w_\ell) \sin(2\pi k_\ell x + \phi_\ell), \quad (52)$$

where the background term is given by

$$u_{\text{bg}}(x) = a_1 \sin(2\pi x + p_1) + a_2 \cos(4\pi x + p_2) + a_3 \sin(6\pi x + p_3), \quad (53)$$

and the periodic envelope function is

$$B(x; c, w) = \exp\left(-\frac{\sin^2(\pi(x - c))}{w^2}\right). \quad (54)$$

Here $a_1, a_2, a_3, b_1, b_2, b_3$, and the phases $p_1, p_2, p_3, \phi_1, \phi_2, \phi_3$ are randomly sampled. The packet centers are chosen hierarchically, with c_2 sampled near c_1 and c_3 sampled near c_2 , so that the medium-scale feature is spatially nested inside the broad-scale feature and the fine-scale feature is nested inside the medium-scale feature. The widths and frequencies satisfy

$$w_1 > w_2 > w_3, \quad k_1 < k_2 < k_3, \quad (55)$$

so that the three localized packets represent broad, medium, and fine oscillatory structure, respectively.

This construction yields a family of periodic initial conditions with nested multiscale organization. The broad packet determines a coarse active region, the medium packet refines that region, and the fine packet introduces highly localized oscillatory detail inside it. Because the locations, widths, amplitudes, and phases are randomized from sample to sample, the dataset does not reduce to a fixed template, while still consistently exhibiting hierarchical localized structure. We use this benchmark to assess whether deeper adaptive partitioning provides measurable benefits over shallower adaptive models and non-adaptive models.

Results. Table 4 reports the relative ℓ_2 errors, while Table 9 reports full-field ℓ_∞ errors and timing results. Within the DeepONet family, the HiRefPOU variants are competitive with or improve upon the vanilla and static-partition baselines, indicating that learned localization remains useful when predicting the complete space-time solution field.

Despite the explicitly nested structure used to construct this benchmark, the differences among hierarchy depths are small. HiRefPOU-2L and HiRefPOU-3L obtain slightly lower mean errors than HiRefPOU-1L on some metrics, but these differences are small relative to the reported variability across seeds. Thus, this experiment does not establish a clear advantage from increasing hierarchy depth. Instead, it shows that the hierarchical architecture can represent the nested localized structure without degrading accuracy substantially, while a single adaptive level remains highly competitive.

The strongest overall results are obtained by the POU-FNO variants with a LocalFNO backbone. This indicates that, for a periodic and grid-aligned space-time problem, local Fourier processing provides a particularly strong inductive bias. The results therefore reinforce that the benefit of hierarchy and the appropriate localization mechanism depend on both the structure of the target operator and the underlying neural-operator backbone.

Table 9: Testing ℓ_∞ errors, training times, and inference times for the Full-Field Burgers problem. For the error and inference-time columns, the best value is shown in bold and the second-best value is underlined. Training times are reported for computational context but are not ranked across model families because the reported unit differs between full-batch DeepONet and minibatch FNO models. For POU-FNO variants, the FNO backbone used by the model is indicated in parentheses. Rows corresponding to proposed adaptive PoU-based models are highlighted.

	Model	Full-Field ℓ_∞ Error	Mean Training Time [†] (s)	Inference Time (s)
DeepONets	Vanilla	$3.02 \times 10^{-1} \pm 5.7 \times 10^{-3}$	$2.79 \times 10^{-4} \pm 7.9 \times 10^{-6}$	$1.31 \times 10^{-4} \pm 1.1 \times 10^{-4}$
	(P+1)-Vanilla	$2.90 \times 10^{-1} \pm 5.0 \times 10^{-3}$	<u>$1.08 \times 10^{-3} \pm 2.6 \times 10^{-5}$</u>	$9.11 \times 10^{-4} \pm 3.8 \times 10^{-4}$
	(Vanilla, POU)	$2.72 \times 10^{-1} \pm 3.4 \times 10^{-3}$	$1.47 \times 10^{-2} \pm 1.6 \times 10^{-5}$	$1.47 \times 10^{-2} \pm 1.6 \times 10^{-5}$
	(POD, POU)	$2.69 \times 10^{-1} \pm 4.2 \times 10^{-3}$	$1.45 \times 10^{-2} \pm 7.4 \times 10^{-5}$	$1.46 \times 10^{-2} \pm 3.2 \times 10^{-5}$
	(Vanilla, POD, POU)	$2.84 \times 10^{-1} \pm 4.2 \times 10^{-3}$	$4.87 \times 10^{-2} \pm 4.2 \times 10^{-3}$	$4.90 \times 10^{-2} \pm 1.2 \times 10^{-3}$
	HiRefPOU-1L (POD)	$2.72 \times 10^{-1} \pm 6.7 \times 10^{-3}$	$1.50 \times 10^{-3} \pm 2.6 \times 10^{-5}$	<u>$3.57 \times 10^{-4} \pm 6.0 \times 10^{-5}$</u>
	HiRefPOU-2L (POD)	$2.69 \times 10^{-1} \pm 5.1 \times 10^{-3}$	$3.59 \times 10^{-3} \pm 6.0 \times 10^{-4}$	$5.50 \times 10^{-4} \pm 7.1 \times 10^{-5}$
	HiRefPOU-3L (POD)	$2.66 \times 10^{-1} \pm 5.5 \times 10^{-3}$	$1.49 \times 10^{-2} \pm 1.6 \times 10^{-3}$	$2.12 \times 10^{-3} \pm 3.6 \times 10^{-4}$
FNOs	GeoFNO	$8.26 \times 10^{-2} \pm 8.0 \times 10^{-3}$	$3.97 \pm 1.6 \times 10^{-3}$	$4.36 \times 10^{-1} \pm 1.6 \times 10^{-4}$
	LocalFNO	$6.47 \times 10^{-2} \pm 7.4 \times 10^{-3}$	$1.46 \times 10^{-1} \pm 1.3 \times 10^{-4}$	$7.33 \times 10^{-2} \pm 4.7 \times 10^{-4}$
	POU-SharedFNO (LocalFNO)	$5.28 \times 10^{-2} \pm 5.0 \times 10^{-3}$	$1.59 \times 10^{-1} \pm 1.6 \times 10^{-3}$	$7.59 \times 10^{-2} \pm 1.1 \times 10^{-4}$
	POU-ExpertFNO (LocalFNO)	<u>$5.82 \times 10^{-2} \pm 6.8 \times 10^{-3}$</u>	$2.38 \times 10^{-1} \pm 1.6 \times 10^{-3}$	$7.94 \times 10^{-2} \pm 1.2 \times 10^{-4}$

[†]For full-batch DeepONet-family models, training time denotes one optimizer update over the complete training set. For minibatch FNO-family models, it denotes one complete training epoch.

7.7 Hierarchical Multiscale Kuramoto–Sivashinsky Benchmark

To complement the Burgers experiments with a more dynamically unstable benchmark, we consider the one-dimensional Kuramoto–Sivashinsky (KS) equation on a periodic domain. The KS equation exhibits mixed-scale spatiotemporal behavior, including instability growth, nonlinear interaction, and localized fine-scale structure, making it a natural testbed for hierarchical adaptive operator learning. In contrast to the standard Burgers benchmark, where the output is often a single final-time solution profile, here the learning task is to map an initial condition to the full solution field over space-time.

Problem Setup. We study the periodic KS equation

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} = 0, \quad x \in [0, L], \quad t \in [0, T], \quad (56)$$

with periodic boundary conditions. For each sample, the operator-learning task is to map a one-dimensional initial condition $u_0(x)$ to the full solution field $u(x, t)$.

We construct the initial conditions from a randomized three-level hierarchy of localized oscillatory structure superimposed on a smooth low-frequency periodic background. Specifically, each initial condition is of the form

$$u_0(x) = u_{\text{bg}}(x) + \sum_{\ell=1}^3 b_\ell B(\xi; c_\ell, w_\ell) \sin(2\pi k_\ell \xi + \phi_\ell), \quad \xi = x/L, \quad (57)$$

where the background component is given by

$$u_{\text{bg}}(x) = a_1 \sin(2\pi\xi + p_1) + a_2 \cos(4\pi\xi + p_2) + a_3 \sin(6\pi\xi + p_3), \quad (58)$$

and the periodic bump envelope is

$$B(\xi; c, w) = \exp\left(-\frac{\sin^2(\pi(\xi - c))}{w^2}\right). \quad (59)$$

The three packet centers are sampled hierarchically, with c_2 chosen near c_1 and c_3 chosen near c_2 , so that the medium-scale packet is spatially nested inside the broad-scale packet and the fine-scale packet is nested inside the medium-scale packet. The widths and frequencies are sampled so that

$$w_1 > w_2 > w_3, \quad k_1 < k_2 < k_3, \quad (60)$$

thereby yielding broad, medium, and fine localized oscillatory features, respectively. The amplitudes, phases, and relative signs are randomized across samples.

This benchmark is intended to test whether deeper adaptive partitioning provides measurable gains when the target operator exhibits genuinely nested structure across both space and time. A shallow partition hierarchy may capture shared spatial localization patterns in the solution field, while additional levels provide further nested spatial subdivision within regions emphasized by the parent gates.

Results. Table 4 reports the testing relative ℓ_2 errors, while Table 10 reports full-field ℓ_∞ errors, training times, and inference times for the KS benchmark. Within the DeepONet family, all HiRefPOU variants improve over the vanilla and static-partition baselines, demonstrating that learned spatial localization is beneficial for this dynamically unstable space–time prediction problem.

In contrast to the reaction–diffusion benchmarks, additional hierarchy does not improve the HiRefPOU results. HiRefPOU-1L achieves the lowest mean relative ℓ_2 and ℓ_∞ errors among the DeepONet variants, with performance degrading as additional levels are introduced. This provides a clear example in which a shallower adaptive representation is preferable despite the presence of multiscale dynamics. One possible interpretation is that the spatial organization learned by a single POU level already captures the useful dataset-level localization, while additional coordinate-based subdivision does not correspond to the dominant variation of this strongly evolving solution operator.

The FNO-based models achieve the lowest errors overall, with the LocalFNO-based variants particularly strong. Together with the standard and full-field Burgers results, this reinforces that deeper geometric hierarchy should not be expected to improve performance simply because a problem contains multiscale dynamics; the usefulness of additional partition levels depends on whether those dynamics admit persistent spatial organization that can be exploited by the coordinate-dependent hierarchy.

7.8 Discussion

The results reveal a problem-dependent relationship between adaptive spatial partitioning, hierarchical refinement, and the choice of operator-learning backbone. Across the DeepONet-family models, the proposed HiRefPOU architectures consistently improve over the corresponding vanilla and static-partition DeepONet baselines. This trend is especially clear on the cavity flow and reaction–diffusion benchmarks, where the learned partitions provide a mechanism for assigning greater weight to local experts near coefficient interfaces, localized features, or regions of strong spatial heterogeneity.

The comparison with FNO-based models is more nuanced. On the one-dimensional periodic Burgers and Kuramoto–Sivashinsky benchmarks, LocalFNO and PoU-FNO variants are highly competitive and sometimes achieve the lowest errors overall. This is not unexpected: these benchmarks are defined on regular periodic grids, and their solution fields contain coherent space–time structures that are well matched to Fourier-based convolutional representations. In such settings, LocalFNO provides a strong inductive bias by combining local spatial processing with spectral information, and the advantage of adaptive DeepONet partitioning is reduced. At the same time, the PoU-FNO results show that the proposed partition-of-unity mechanism

Table 10: Testing ℓ_∞ errors, training times, and inference times for the KS problem. For the error and inference-time columns, the best value is shown in bold and the second-best value is underlined. Training times are reported for computational context but are not ranked across model families because the reported unit differs between full-batch DeepONet and minibatch FNO models. For POU-FNO variants, the FNO backbone used by the model is indicated in parentheses. Rows corresponding to proposed adaptive PoU-based models are highlighted.

	Model	Full-Field ℓ_∞ Error	Mean Training Time [†] (s)	Inference Time (s)
DeepONets	(Vanilla)	$2.45 \pm 3.2 \times 10^{-2}$	$7.49 \times 10^{-4} \pm 7.8 \times 10^{-5}$	$7.58 \times 10^{-4} \pm 2.3 \times 10^{-4}$
	(P+1)-Vanilla	$2.09 \pm 3.2 \times 10^{-2}$	$2.22 \times 10^{-3} \pm 6.6 \times 10^{-5}$	$2.26 \times 10^{-3} \pm 2.3 \times 10^{-5}$
	(Vanilla, POU)	$1.96 \pm 3.5 \times 10^{-2}$	$2.09 \times 10^{-2} \pm 2.9 \times 10^{-4}$	$2.10 \times 10^{-2} \pm 1.7 \times 10^{-4}$
	(POD, POU)	$1.87 \pm 2.5 \times 10^{-2}$	$1.31 \times 10^{-2} \pm 4.6 \times 10^{-5}$	$9.91 \times 10^{-3} \pm 2.9 \times 10^{-3}$
	(Vanilla, POD, POU)	$1.91 \pm 2.4 \times 10^{-2}$	$2.03 \times 10^{-2} \pm 4.6 \times 10^{-5}$	$2.06 \times 10^{-2} \pm 2.3 \times 10^{-5}$
	HiRefPOU-1L (POD)	$1.50 \pm 1.4 \times 10^{-2}$	$1.56 \times 10^{-3} \pm 1.4 \times 10^{-5}$	$4.13 \times 10^{-4} \pm 2.7 \times 10^{-5}$
	HiRefPOU-2L (POD)	$1.69 \pm 1.4 \times 10^{-2}$	$4.57 \times 10^{-3} \pm 1.3 \times 10^{-6}$	$3.38 \times 10^{-4} \pm 1.3 \times 10^{-6}$
	HiRefPOU-3L (POD)	$1.80 \pm 4.7 \times 10^{-4}$	$1.07 \times 10^{-2} \pm 1.1 \times 10^{-5}$	$6.84 \times 10^{-4} \pm 3.9 \times 10^{-6}$
FNOs	GeoFNO	$9.09 \times 10^{-1} \pm 1.5 \times 10^{-2}$	$8.65 \times 10^{-1} \pm 1.3 \times 10^{-2}$	$1.51 \times 10^{-1} \pm 3.4 \times 10^{-4}$
	LocalFNO	$8.37 \times 10^{-1} \pm 1.3 \times 10^{-2}$	$3.08 \times 10^{-1} \pm 1.1 \times 10^{-2}$	$2.21 \times 10^{-2} \pm 3.8 \times 10^{-4}$
	POU-SharedFNO (LocalFNO)	$8.23 \times 10^{-1} \pm 1.1 \times 10^{-2}$	$3.22 \times 10^{-1} \pm 7.5 \times 10^{-3}$	$2.24 \times 10^{-2} \pm 4.5 \times 10^{-4}$
	POU-ExpertFNO (LocalFNO)	<u>$8.36 \times 10^{-1} \pm 2.4 \times 10^{-2}$</u>	$4.43 \times 10^{-1} \pm 1.6 \times 10^{-2}$	$2.85 \times 10^{-2} \pm 1.5 \times 10^{-3}$

[†]For full-batch DeepONet-family models, training time denotes one optimizer update over the complete training set. For minibatch FNO-family models, it denotes one complete training epoch.

can also improve Fourier-based backbones in some regimes, most notably on the Full-Field Burgers and KS benchmarks.

For the PoU-FNO variants, the backbone is selected to match the geometry and data representation of each benchmark. For the one-dimensional periodic Burgers and KS problems, we use LocalFNO as the backbone because these problems are naturally represented on regular space–time grids and local Fourier structure is a strong modeling prior. For cavity flow and the reaction–diffusion problems, we use GeoFNO as the backbone because these datasets involve geometry-dependent or point-cloud-based spatial discretizations for which GeoFNO is the more appropriate Fourier baseline. Thus, the parenthetical labels in the PoU-SharedFNO and PoU-ExpertFNO rows identify the underlying FNO architecture being augmented by the PoU mechanism. This design choice allows the PoU extension to be evaluated on top of a suitable FNO backbone rather than against a mismatched baseline.

The reaction–diffusion results also illustrate the setting in which the HiRefPOU-DeepONet architecture is most effective. In the 2D and 3D reaction–diffusion problems, the solution operators are shaped by discontinuous reaction coefficients, spatially varying diffusion, and localized gradients. Here, the hierarchical DeepONet variants achieve the best relative ℓ_2 errors, while the FNO-based models are substantially less accurate. This suggests that adaptive spatial partitioning is particularly beneficial when the solution operator contains localized heterogeneity that is not naturally aligned with a global spectral representation.

Finally, the hierarchy does not uniformly improve performance simply by adding levels. On some benchmarks, the single-level or two-level variant is sufficient, while the three-level model gives the best results on others. This behavior is consistent with the ablation studies: the benefit of additional hierarchy depends on whether the underlying problem contains persistent localized structure that can benefit from additional nested spatial subdivision. Thus, the main conclusion is not that deeper hierarchies are always better, but that hierarchical POU organization provides a flexible mechanism for allocating local capacity through nested spatial partitions when the structure of the PDE solution operator benefits from such localization. For new problems, we therefore recommend using HiRefPOU-1L as the default starting point and introducing additional hierarchy only when performance on held-out data justifies the added model capacity and dense evaluation cost.

7.8.1 Limitations

The POU gates used in this work depend only on the evaluation coordinates and therefore do not adapt to individual input instances. Instead, the learned partitions capture dataset-level geometric structure that recurs at consistent spatial locations across the training distribution. Because the gates are shared across

input samples, they cannot move or reshape their partition boundaries in response to sample-dependent features. For benchmarks such as Burgers and Kuramoto–Sivashinsky, where localized structures may shift across samples, any improvement should therefore be interpreted as a benefit of a shared coordinate-dependent localization representation rather than as evidence that the gates track the features of each individual input. Conditioning the gating network on the input function or its branch-network representation is a natural extension that could enable instance-dependent partitioning in such settings.

7.8.2 Broader Impacts

The methods developed in this work are intended as surrogate modeling tools for scientific computing and should not be interpreted as certified numerical solvers. As with other learned operator models, predictive accuracy may degrade outside the training distribution or in regimes that are insufficiently represented by the available data. Consequently, deployment in safety-critical or otherwise consequential applications should include independent validation against trusted numerical or experimental references, appropriate uncertainty or error assessment, and problem-specific verification of physically relevant quantities. The present study focuses on methodological development and benchmark PDE problems and does not involve human participants, sensitive data, or automated decision-making about individuals.

8 Conclusion

This work introduced HiRefPOU, a hierarchical partition-of-unity (POU) framework for learning PDE solution operators with localized and multiscale structure. The method combines adaptive POU gating with residual-style additive level-wise representations, yielding a structured mixture-of-experts architecture in which a global representation is combined with one or more nested levels of spatially weighted local expert representations. Unlike purely global operator models, the proposed hierarchy provides spatially localized expressivity while preserving a smooth global prediction through the differentiable POU formulation. We view HiRefPOU as a localized neural-operator architecture built from established POU, mixture-of-experts, global–local representation, and hierarchical-partitioning ideas rather than as a fundamentally new operator-learning paradigm. Its architectural contribution is the particular integration of learned hybrid geometric gating, nested parent–child POU weights, and additive level-wise operator representations, together with their study in DeepONet- and FNO-type models.

Across a diverse set of benchmark problems, HiRefPOU consistently improves over the corresponding vanilla and static-partition DeepONet baselines. The gains are most pronounced on problems with localized heterogeneity, sharp coefficient variation, or spatially concentrated error structures, such as cavity flow and the reaction–diffusion benchmarks. The learned partitions capture dataset-shared spatial structure associated with problem-specific features and provide an interpretable mechanism for allocating representational capacity to regions where the solution operator is more difficult to approximate.

The comparison with FNO-based models shows that the best-performing backbone depends on the structure of the problem. For periodic, grid-aligned space–time benchmarks such as Burgers and Kuramoto–Sivashinsky, LocalFNO and POU-FNO variants are highly competitive and sometimes achieve the lowest errors overall. For heterogeneous reaction–diffusion problems, the HiRefPOU-DeepONet variants achieve the strongest accuracy, suggesting that adaptive spatial partitioning is especially valuable when the operator contains localized features that are not naturally captured by global spectral representations. These results indicate that the proposed POU mechanism is not limited to a single neural-operator family: it can improve DeepONet-style models directly and can also be used as a localization module for Fourier-based backbones.

Overall, the results support a nuanced conclusion. Hierarchical POU organization is not uniformly superior to every Fourier-based architecture on every benchmark, nor does increasing the number of levels always improve accuracy. Rather, the proposed framework provides a flexible and interpretable way to introduce adaptive locality into neural operator learning. When the PDE solution operator contains localized, heterogeneous, or multiscale structure, this localization can substantially improve accuracy without relying solely on increased parameter count. These findings suggest that structured, geometry-aware localization is an important ingredient for building reliable operator-learning models for complex scientific computing problems.

A Full Experimental Details

Table 11: HiRefPOU hierarchy configurations used for each benchmark. N_ℓ denotes the number of experts at level ℓ . Parent–child allocations list the number of children assigned to each parent at successive levels. The notation $[2]_{\times k}$ denotes a length- k list in which every parent has two children.

Benchmark	Variant	Experts by level	Parent–child allocation	Leaf partitions
Cavity	1L	$N_1 = 4$	–	4
	2L	$(N_1, N_2) = (2, 2)$	$L_1 \rightarrow L_2 : [0, 2]$	3
	3L	$(N_1, N_2, N_3) = (2, 2, 4)$	$L_1 \rightarrow L_2 : [0, 2], \quad L_2 \rightarrow L_3 : [2, 2]$	4
2D RD	1L	$N_1 = 6$	–	6
	2L	$(N_1, N_2) = (2, 4)$	$L_1 \rightarrow L_2 : [2, 2]$	4
	3L	$(N_1, N_2, N_3) = (2, 4, 8)$	$L_1 \rightarrow L_2 : [2, 2], \quad L_2 \rightarrow L_3 : [2]_{\times 4}$	8
3D RD	1L	$N_1 = 8$	–	8
	2L	$(N_1, N_2) = (3, 5)$	$L_1 \rightarrow L_2 : [2, 2, 1]$	5
	3L	$(N_1, N_2, N_3) = (3, 5, 10)$	$L_1 \rightarrow L_2 : [2, 2, 1], \quad L_2 \rightarrow L_3 : [2]_{\times 5}$	10
1D Burgers	1L	$N_1 = 3$	–	3
	2L	$(N_1, N_2) = (1, 2)$	$L_1 \rightarrow L_2 : [2]$	2
	3L	$(N_1, N_2, N_3) = (2, 4, 8)$	$L_1 \rightarrow L_2 : [2, 2], \quad L_2 \rightarrow L_3 : [2]_{\times 4}$	8
Full-field Burgers	1L	$N_1 = 6$	–	6
	2L	$(N_1, N_2) = (3, 6)$	$L_1 \rightarrow L_2 : [2]_{\times 3}$	6
	3L	$(N_1, N_2, N_3) = (6, 12, 24)$	$L_1 \rightarrow L_2 : [2]_{\times 6}, \quad L_2 \rightarrow L_3 : [2]_{\times 12}$	24
Kuramoto–Sivashinsky	1L	$N_1 = 6$	–	6
	2L	$(N_1, N_2) = (3, 6)$	$L_1 \rightarrow L_2 : [2]_{\times 3}$	6
	3L	$(N_1, N_2, N_3) = (6, 12, 24)$	$L_1 \rightarrow L_2 : [2]_{\times 6}, \quad L_2 \rightarrow L_3 : [2]_{\times 12}$	24

For an output snapshot matrix $X \in \mathbb{R}^{N_{\text{train}} \times n}$, we compute the pointwise training mean μ_j and standard deviation σ_j and form

$$\tilde{X}_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j + 10^{-9}}.$$

The POD modes are obtained from the eigendecomposition of $\tilde{X}^\top \tilde{X} / (N_{\text{train}} - 1)$. The training mean is retained as the first basis vector, followed by the leading covariance eigenvectors, with the total number of retained vectors equal to the rank reported in Table 14. No test outputs are used to compute the mean, standard deviation, or basis vectors. For cavity flow, this procedure is applied independently to the two velocity components. The Kuramoto–Sivashinsky outputs additionally undergo the scalar training-set normalization used by the dataset preprocessing pipeline before the pointwise POD standardization.

Sampling distributions used to generate the Burgers and Kuramoto–Sivashinsky datasets. For 1D Burgers, initial conditions were sampled from the periodic Gaussian random field $\mathcal{N}(0, C)$ with $C = \sigma^2(-\Delta + \tau^2 I)^{-\gamma}$, $\gamma = 2.5$, $\tau = 5$, and $\sigma = 25$. The construction retained $K = 4096$ positive Fourier modes, with independent standard-normal sine and cosine coefficients scaled by $\sqrt{2}\sigma((2\pi k)^2 + \tau^2)^{-\gamma/2}$.

For Full-field Burgers, the global-background amplitudes were sampled as $a_1 \sim \mathcal{U}(0.05, 0.12)$, $a_2 \sim \mathcal{U}(0.02, 0.07)$, and $a_3 \sim \mathcal{U}(0.01, 0.04)$, with independent phases $p_i \sim \mathcal{U}(0, 2\pi)$. The three localized-packet amplitudes were $b_1 \sim \mathcal{U}(0.08, 0.14)$, $b_2 = s_2 \tilde{b}_2$ with $s_2 \in \{-1, 1\}$ equiprobable and $\tilde{b}_2 \sim \mathcal{U}(0.04, 0.09)$, and $b_3 = s_3 \tilde{b}_3$ with $\tilde{b}_3 \sim \mathcal{U}(0.02, 0.06)$. Their phases were independent $\phi_i \sim \mathcal{U}(0, 2\pi)$. The broad center was $c_1 \sim \mathcal{U}(0, 1)$, while $c_2 = (c_1 + \epsilon_2) \bmod 1$ and $c_3 = (c_2 + \epsilon_3) \bmod 1$, where $\epsilon_2 \sim \mathcal{N}(0, 0.03^2)$ and $\epsilon_3 \sim \mathcal{N}(0, 0.01^2)$. Widths were sampled from $w_1 \sim \mathcal{U}(0.14, 0.22)$, $w_2 \sim \mathcal{U}(0.045, 0.070)$, and $w_3 \sim \mathcal{U}(0.010, 0.025)$. Frequencies were discrete uniform on $\{1, 2, 3\}$, $\{4, \dots, 8\}$, and $\{10, \dots, 20\}$, respectively.

Table 12: Network architecture specifications used for the HiRefPOU experiments. Depth denotes the number of hidden trainable layers and excludes the final output projection. For convolutional branch networks, the hidden-width entry lists the output channels of the convolutional layers followed by the width of the dense hidden layer. The branch output contains the p learned DeepONet coefficients and the r_{POD} POD coefficients. Where architectures differ among hierarchy variants, the corresponding 1L, 2L, and 3L values are listed explicitly. Hierarchy configurations are reported in Table 11. The POU center locations are initialized using the same prescribed center configurations as the corresponding prior-work benchmarks where applicable; for the benchmarks introduced in this work, the initial center locations are prescribed as part of the benchmark-specific model configuration. In all cases, the center locations remain trainable and are optimized jointly with the remaining model parameters during training. Unless otherwise specified, neural-network weights use the default Haiku initialization; the POU centers use the prescribed initial configurations described above and remain trainable thereafter.

Benchmark	Network component	Input dim.	Hidden width(s)	Depth	Output dim.	Activation
Cavity flow	Branch CNN	$65 \times 65 \times 2$	64, 128, 128	3	106	tanh
	Global trunk	2	128	3	100	tanh
	Local trunk expert	2	128	3	100	tanh
	POU neural gate	2	128	2	N_{part}	tanh
2D reaction–diffusion	Branch CNN	$20 \times 20 \times 1$	64, 128, 128	3	120	ReLU
	Global trunk	2	128	3	100	ReLU
	Local trunk expert	2	128	3	100	ReLU
	POU neural gate	2	128	2	N_{part}	tanh
3D reaction–diffusion	Branch MLP	4325	128	3	120	ReLU
	Global trunk	3	128	3	100	ReLU
	Local trunk expert	3	128	3	100	ReLU
	POU neural gate	3	64	2	N_{part}	tanh
Standard Burgers	Branch MLP	128	128	3	120	ReLU
	Global trunk	1	128	2	100	ReLU
	Local trunk expert	1	128	2	100	ReLU
	POU neural gate	1	16	1	N_{part}	tanh
Full-field Burgers	Branch MLP	128	128	3	120	ReLU
	Global trunk	2	128	3	100	ReLU
	Local trunk expert	2	128	3	100	ReLU
	POU neural gate	2	128	3	N_{part}	tanh
Kuramoto–Sivashinsky	Branch MLP	128	128	3	120	ReLU
	Global trunk	2	128	3	100	ReLU
	Local trunk expert	2	128	3	100	ReLU
	POU neural gate	2	128	3	N_{part}	tanh

For the cavity and 2D reaction–diffusion branch CNNs, the convolutional layers use 64 and 128 output channels, respectively, with kernel size 5, stride 2, and valid padding, followed by flattening and a 128-unit dense hidden layer. The cavity branch output dimension is $100 + 6 = 106$; all other branch outputs have dimension $100 + 20 = 120$.

Table 13: Architecture specifications for all Fourier neural operator (FNO)-based models used in the experiments. Fourier modes are reported per spatial or space–time dimension. GeoFNO and the POU–FNO variants use five spectral layers. For LocalFNO, the number of local neural-operator layers is reported. Input-channel counts include coordinate channels for GeoFNO and POU–FNO, but not for LocalFNO, which constructs its coordinate representation internally. GeoFNO and GeoFNO-based POU-FNO variants use five spectral layers; LocalFNO and LocalFNO-based POU variants use the number of local operator layers reported below.

Benchmark	Model	Input channels	FNO width	Fourier modes	FNO layers	Output channels	Local / expert structure	POU / gate configuration
Cavity flow	GeoFNO	4	64	(20, 20)	5	2	Single global GeoFNO	–
	LocalFNO	2	50	(20, 20)	3	2	3 local operator layers	–
	POU-SharedFNO	4	64	(20, 20)	5	2	Shared GeoFNO backbone; expert heads (64, 64)	4 partitions; gate width 128, depth 2
	POU-ExpertFNO	4	32	(18, 18)	5	2	4 independent GeoFNO experts; expert heads (64, 64)	4 partitions; gate width 128, depth 2
2D reaction–diffusion	GeoFNO	3	32	(12, 12)	5	1	Single global GeoFNO	–
	LocalFNO	1	30	(12, 12)	2	1	2 local operator layers	–
	POU-SharedFNO	3	32	(10, 10)	5	1	Shared GeoFNO backbone; expert heads (32, 32)	6 partitions; gate width 128, depth 2
	POU-ExpertFNO	3	24	(6, 6)	5	1	4 independent GeoFNO experts; expert heads (32, 32)	4 partitions; gate width 128, depth 2
3D reaction–diffusion	GeoFNO	4	16	(7, 7, 7)	5	1	Single global GeoFNO	–
	LocalFNO	1	12	(8, 8, 8)	2	1	2 local operator layers	–
	POU-SharedFNO	4	16	(6, 6, 6)	5	1	Shared GeoFNO backbone; expert heads (32, 32)	4 partitions; gate width 128, depth 2
	POU-ExpertFNO	4	16	(5, 5, 5)	5	1	2 independent GeoFNO experts; expert heads (32, 32)	2 partitions; gate width 128, depth 2
1D Burgers	GeoFNO	129	50	12	5	1	Single global GeoFNO	–
	LocalFNO	1	32	24	6	1	6 local operator layers	–
	POU-SharedFNO	129	50	12	5	1	Shared GeoFNO backbone; expert heads (32, 32)	3 partitions; gate width 16, depth 1
	POU-ExpertFNO	129	16	6	5	1	3 independent GeoFNO experts; expert heads (32, 32)	3 partitions; gate width 16, depth 1
Full-field Burgers	GeoFNO	3	12	(18, 18)	5	1	Single global GeoFNO	–
	LocalFNO	1	12	(18, 18)	2	1	2 local operator layers	–
	POU-SharedFNO	1	12	(18, 18)	2	1	Shared LocalFNO backbone; expert heads (12, 12)	4 partitions; gate width 12, depth 1
	POU-ExpertFNO	1	12	(8, 8)	2	1	2 independent LocalFNO experts	4 partitions; gate width 12, depth 1
Kuramoto–Sivashinsky	GeoFNO	3	12	(18, 18)	5	1	Single global GeoFNO	–
	LocalFNO	1	12	(18, 18)	2	1	2 local operator layers	–
	POU-SharedFNO	1	12	(18, 18)	2	1	Shared LocalFNO backbone; expert heads (12, 12)	4 partitions; gate width 12, depth 1
	POU-ExpertFNO	1	12	(8, 8)	2	1	2 independent LocalFNO experts	4 partitions; gate width 12, depth 1

Table 14: POD basis construction used in the DeepONet experiments. All POD statistics and basis vectors are computed exclusively from the training outputs. For cavity flow, separate rank-3 bases are constructed for the two velocity components, giving a total POD rank of 6. The POD ranks were fixed a priori and were not selected using test-set performance.

Benchmark	POD rank	Basis samples	Centering / normalization
Cavity	6 (3 + 3)	100 training outputs	Pointwise training mean/std; separate basis per component
2D reaction–diffusion	20	1,000 training outputs	Pointwise training mean/std
3D reaction–diffusion	20	1,000 training outputs	Pointwise training mean/std
1D Burgers	20	1,000 training outputs	Pointwise training mean/std
Full-Field Burgers	20	1,000 training outputs	Pointwise training mean/std
Kuramoto–Sivashinsky	20	1,000 training outputs	Global training mean/std, followed by pointwise training mean/std

Table 15: Training and optimization settings for DeepONet-based experiments. Adam and AdamW use the Optax default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Cavity models use Adam. All other rows use AdamW with weight decay 10^{-4} . The inverse-time schedule is $\eta_t = \eta_0/(1 + \gamma t)$; the implementation uses `decay_steps=1` without staircase decay. Each HiRefPOU model uses $\lambda_{cov} = \lambda_{cont} = 10^{-4}$ and are linearly annealed to zero over the 150,000 gradient updates.

Benchmark	Initial LR	LR schedule	Batch size	Gradient updates
Cavity	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	Full batch (100)	150,000
2D reaction-diffusion	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	Full batch (1,000)	150,000
3D reaction-diffusion	6×10^{-3}	Inverse-time decay ($\gamma = 10^{-4}$)	Full batch (1,000)	150,000
Standard Burgers	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	Full batch (1,000)	150,000
Full-field Burgers	3×10^{-4}	Inverse-time decay ($\gamma = 5 \times 10^{-5}$)	Full batch (1,000)	150,000
Kuramoto-Sivashinsky	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	Full batch (1,000)	150,000

Table 16: Standardized training and optimization settings for the FNO-based experiments. All FNO models use mini-batches of 32 input functions and minimize the mean squared output-field error. AdamW uses the Optax default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, with weight decay 10^{-4} . The inverse-time schedule is $\eta_t = \eta_0/(1 + \gamma t)$, using `decay_steps=1` without staircase decay. The POU-FNO models jointly optimize the backbone or experts, output heads, and POU gate. No containment or coverage penalties are applied: $\lambda_{cont} = \lambda_{cov} = 0$. The number of epochs is selected to give approximately 150,000 gradient updates, computed as $N_{\text{epoch}} \lceil N_{\text{train}}/B \rceil$.

Benchmark	Initial LR	LR schedule	Batch size	Training epochs	Gradient updates
Cavity	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	32	37,500	150,000
2D reaction-diffusion	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	32	4,688	150,016
3D reaction-diffusion	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	32	4,688	150,016
Standard Burgers	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	32	4,688	150,016
Full-field Burgers	3×10^{-4}	Inverse-time decay ($\gamma = 5 \times 10^{-5}$)	32	4,688	150,016
Kuramoto-Sivashinsky	3×10^{-4}	Inverse-time decay ($\gamma = 10^{-4}$)	32	4,688	150,016

Table 17: Dataset provenance, train/test splits, and reference-solution details for each benchmark. Reference-grid resolutions are distinguished from the subsampled grids supplied to the networks.

Benchmark	Operator / task	Domain / network grid	N_{train}	N_{test}	Dataset source	Reference solver	Preprocessing
Cavity flow	$u_b(x) \mapsto (u, v)(x, y)$	$[0, 1]^2$; 65 $\times$ 65 grid; 65 lid-velocity sensors	100	10	External benchmark; Lu et al. / Sharma-Shankar	LBM	Separate train/test files; no coordinate or output scaling
2D reaction-diffusion	$c(\cdot, 0) \mapsto c(\cdot, 0.5)$	$[0, 2]^2$; 2,207 scattered points	1,000	200	Sharma-Shankar	High-order RBF-FD	Fixed contiguous split; no coordinate or output scaling
3D reaction-diffusion	$c(\cdot, 0) \mapsto c(\cdot, 0.5)$	Unit ball in $\mathbb{R}^3$; 4,325 scattered points	1,000	200	Sharma-Shankar	High-order RBF-FD	Fixed contiguous split; no coordinate or output scaling
Standard Burgers	$u_0 \mapsto u(\cdot, 1)$	$[0, 1]$; periodic; 8,192 reference points; 128 network points	1,000	200	Generated in this work	Chebfun spectral solver; $\nu = 10^{-3}$, 500 time points	Spatial stride 64; no coordinate or output scaling
Full-field Burgers	$u_0 \mapsto u(x, t)$	$[-1, 1] \times [0, 1]$; periodic; 256 $\times$ 101 stored; 128 $\times$ 51 network grid	1,000	200	Generated in this work	Chebfun spectral solver; $\nu = 10^{-3}$	Spatial/temporal stride 2; no coordinate or output scaling
Kuramoto-Sivashinsky	$u_0 \mapsto u(x, t)$	$[0, 20] \times [0, 30]$; periodic; 128 $\times$ 101 stored; 64 $\times$ 51 network grid	1,000	200	Generated in this work	Fourier ETDRK4; $\Delta t = 10^{-3}$; burn-in time 23	Spatial/temporal stride 2; coordinates mapped to $[0, 1]^2$; training-statistic input/output z-score

For Kuramoto–Sivashinsky, the pre-burn-in initial condition was $u_0(x) = \sum_{k=1}^{12} [a_k \cos(2\pi kx/L) + b_k \sin(2\pi kx/L)]$, where $a_k, b_k \stackrel{\text{iid}}{\sim} \mathcal{N}(0, k^{-4})$. Each profile was rescaled to have maximum absolute amplitude two and evolved for a burn-in time of 23 before recording the subsequent trajectory.

Table 18: Trainable parameter counts for the principal model variants and benchmarks. Counts include all trainable network, expert, gating, and projection parameters.

Model	Cavity	2D RD	3D RD	1D Burgers	Full-field Burgers	Kuramoto– Sivashinsky
Vanilla	3,488,893	331,465	646,089	108,617	108,745	108,745
$(P + 1)$ -Vanilla	3,766,125	686,713	1,120,777	285,857	463,993	458,617
(POD, POU)	3,618,491	565,585	973,721	203,557	342,865	334,673
(Vanilla, POU)	3,687,025	622,213	1,030,477	260,057	399,493	391,301
(Vanilla, POD, POU)	3,728,199	624,793	1,033,057	262,637	402,073	393,881
HiRefPOU-1L (POD)	3,780,529	760,143	1,174,823	266,356	454,863	454,863
HiRefPOU-2L (POD)	3,813,940	793,554	1,183,400	266,645	791,563	791,563
HiRefPOU-3L (POD)	4,097,623	1,327,511	1,893,067	772,339	2,053,841	2,053,841
GeoFNO	3,390,854	394,757	419,719	109,741	109,077	109,077
LocalFNO	3,438,652	337,111	377,485	109,089	109,357	109,357
POU-SharedFNO	3,701,599	340,164	324,112	111,322	109,846	109,846
POU-ExpertFNO	3,088,169	429,733	426,777	196,355	115,125	115,125

B Additional PINN Experiments: Darcy Flow with Discontinuous Coefficients

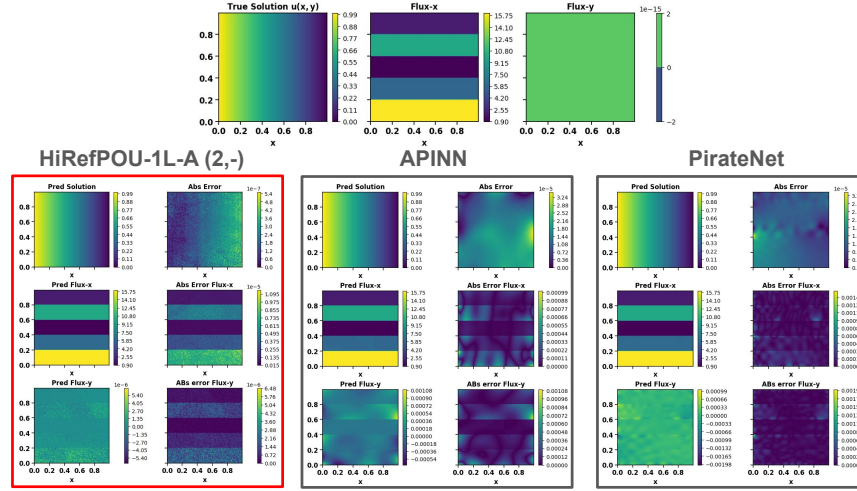


Figure 14: (Top) the true solution and fluxes. (Bottom) Predicted solutions, fluxes, and absolute errors for the Darcy flow benchmark using HiRefPOU-1L, APINN, and PirateNet. Left bottom figures show the predictions while the right bottom figures show the absolute errors.

This appendix provides additional technical details for the Darcy flow experiment presented in Section 7.1. These results are included for completeness and are not intended as an exhaustive comparison with domain-decomposition PINN methods such as XPINNs. All Darcy models were trained for 100,000 Adam updates with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, an initial learning rate of 10^{-3} , and cosine learning-rate decay over the complete training run. At each update, 4,096 residual collocation points were sampled without replacement from a fixed pool of 30,760 interior points. The complete set of 288 Neumann-boundary points was used in the boundary loss. The base PINN objective contained the PDE-residual, Neumann-boundary, and uniqueness-pin terms, initialized with unit weights.

Table 19: Architecture specifications for the Darcy PINN experiments. Width and depth refer to the primary network for PirateNet and to the expert networks for the POU-based models. Gate specifications are reported as hidden width/depth. The HiRefPOU-2L entry lists coarse and fine experts, respectively. Parameter counts include all trainable network, embedding, expert, and gating parameters.

Model	Main / expert width	Main / expert depth	Gate width/depth	POU experts by level	Trainable parameters
APINN	32	4	32/3	2	11,940
PirateNet	32	3	—	—	11,684
HiRefPOU-1L	32	3	16/1	2	11,935
HiRefPOU-2L	32	3	16/1	2 + 3	22,162

All models use the tanh activation. PirateNet uses a trainable 32-dimensional Fourier embedding. APINN and HiRefPOU-1L use two POU components. HiRefPOU-2L uses two coarse experts and three fine experts, with parent-child allocation (1, 2).

Problem formulation and interface conditions We consider the steady Darcy problem

$$-\nabla \cdot (\mu(x) \nabla \phi(x)) = f(x), \quad x \in \Omega = [0, 1]^2, \quad (61)$$

with permeability field $\mu(x)$ that is piecewise constant across subdomains. The flux is defined as $\mathbf{q} = -\mu \nabla \phi$.

At material interfaces Γ_{ij} between subdomains Ω_i and Ω_j , the governing equations impose the physical continuity conditions

$$\phi_i(x) = \phi_j(x), \quad (62)$$

$$\mathbf{n} \cdot \mu_i \nabla \phi_i(x) = \mathbf{n} \cdot \mu_j \nabla \phi_j(x), \quad x \in \Gamma_{ij}, \quad (63)$$

while tangential flux components may be discontinuous.

Five-strip manufactured solution The domain is partitioned into five horizontal layers

$$\Omega_i = \{(x, y) \in [0, 1]^2 : 0.2(i-1) \leq y \leq 0.2i\}, \quad i = 1, \dots, 5,$$

with permeability values

$$\mu|_{\Omega_1} = 16, \mu|_{\Omega_2} = 6, \mu|_{\Omega_3} = 1, \mu|_{\Omega_4} = 10, \mu|_{\Omega_5} = 2.$$

The manufactured solution $\phi_{\text{ex}}(x, y) = 1 - x$ yields exact satisfaction of the interface conditions.

Limitations of globally smooth PINNs A central difficulty in applying standard PINNs to interface problems arises from the use of automatic differentiation over globally smooth neural networks. When $\mu(x)$ is discontinuous, the product $\mu(x)\nabla\phi_\theta(x)$ is not differentiable across interfaces. As a result, the PDE residual computed via automatic differentiation is inconsistent near interfaces, leading to spurious residual terms and degraded accuracy (Yao et al., 2023). Because the neural approximation $\phi_\theta(x)$ is globally smooth, it cannot represent solutions whose gradients are discontinuous, which explains the oscillatory behavior and reduced flux accuracy observed for standard PINN formulations.

Furthest Point Sampling for Center Point Initialization. We initialize the centers and associated radii using the farthest point sampling (FPS) and greedy covering strategy, a technique often used in meshfree approximation and RBF-based partitioning. Given a set of data points $X = \{x_1, \dots, x_N\}$ and a desired number of centers N_c , the algorithm proceeds as follows. Starting from a randomly chosen initial point (usefully one on the domain boundary) $c_1 \in X$, each subsequent center c_i is selected as the data point in X that maximizes its minimum distance to all previously chosen centers $\{c_1, \dots, c_{i-1}\}$. This greedy criterion ensures that each new center lies in the least represented region of the domain, resulting in a near-uniform covering with respect to the underlying data geometry.

An appealing property of the FPS initialization is that it produces an ordered set of centers that progressively improves geometric coverage of the domain. Early-selected centers occupy widely separated regions, while later-selected centers fill regions that remain far from the previously selected centers. In the present framework, this ordering provides a convenient geometry-aware initialization for the POU centers and should not be interpreted as imposing a functional coarse-to-fine decomposition among the corresponding expert outputs. The hierarchy of the HiRefPOU model is instead defined explicitly through the nested parent-child POU weights described in Section 4. After initialization, the center locations and associated gating parameters are optimized jointly with the remaining model parameters. Thus, FPS provides well-distributed initial spatial anchors without imposing frequency separation, stage-wise residual correction, or a multiresolution decomposition of the learned representation. Finally, this initialization strategy is domain-agnostic and dimension-independent, since it only requires access to the input point cloud X .

C Additional Burgers Results

This section reports additional ablations for the standard 1D Burgers benchmark, focusing on the gate parameterization and the sensitivity of the 1L HiRefPOU-DeepONet model to the number of adaptive partitions.

Table 20: *Standard 1D Burgers benchmark: gate ablation for the HiRefPOU-1L DeepONet with $M = 3$ adaptive partitions. We compare RBF, box, and hybrid gate formulations. Reported values are mean $\pm$ standard deviation over random seeds. Epoch time denotes the average wall-clock time per training epoch.*

Gate	Test rel. ℓ_2	Std.	Test rel. ℓ_∞	Std.	Epoch time
RBF gate	0.177134	0.004654	0.457744	0.009059	0.000542
Box gate	0.176758	0.004868	0.457380	0.007711	0.000606
Hybrid gate	0.172893	0.001394	0.454045	0.005357	0.000629

Gate ablation. Table 20 compares three gate parameterizations for the HiRefPOU-1L DeepONet on the standard 1D Burgers benchmark with $M = 3$ adaptive partitions. Among the three variants, the hybrid gate achieves the lowest mean test relative ℓ_2 and ℓ_∞ errors, while also exhibiting the smallest standard deviation across seeds. The box and RBF gates perform similarly to one another, but both are slightly less accurate and less stable than the hybrid formulation. Although the hybrid gate incurs a modest increase in per-epoch cost, the overall trend suggests that combining center-based and box-based structure yields a more robust gate than either component alone on this benchmark.

Table 21: *Standard 1D Burgers benchmark: sensitivity of the HiRefPOU-1L DeepONet to the number of adaptive partitions M when using the hybrid gate. Reported values are mean $\pm$ standard deviation over random seeds. Epoch time denotes the average wall-clock time per training epoch.*

Configuration	Test rel. ℓ_2	Std.	Test rel. ℓ_∞	Std.	Epoch time
Hybrid gate, $M = 2$	0.172622	0.002183	0.452593	0.002469	0.000547
Hybrid gate, $M = 3$	0.176576	0.001287	0.459151	0.006000	0.000629
Hybrid gate, $M = 4$	0.174892	0.002286	0.458236	0.001768	0.000692
Hybrid gate, $M = 6$	0.173398	0.002347	0.453612	0.006004	0.000833

Partition-count sensitivity. Table 21 studies the sensitivity of the HiRefPOU-1L DeepONet to the number of adaptive partitions when using the hybrid gate. The results do not show a monotone improvement as M increases. Instead, relatively small partition counts already perform well, with $M = 2$ giving the best mean test relative ℓ_2 and ℓ_∞ errors in this study, and $M = 6$ achieving comparable accuracy at higher computational cost. This suggests that for the standard 1D Burgers benchmark, the main benefit comes from introducing adaptive localization rather than from aggressively increasing the number of partitions.

D Additional Cavity Ablations

This section reports additional ablations for the cavity benchmark, including the sensitivity of the 1L model to the partition count and residual strength, the effect of gate parameterization, and the effect of child allocation in the 2L hierarchy.

Table 22: *Cavity benchmark: partition-count and residual-strength ablation for the HiRefPOU-1L model. The parameter α denotes the residual scaling used in the POU correction term. Reported values are mean $\pm$ standard deviation over random seeds.*

Gate	M	α	Final test rel. ℓ_2	Std.
hybrid	6	1	1.298×10^{-3}	1.858×10^{-4}
hybrid	6	1.25	1.385×10^{-3}	8.321×10^{-5}
hybrid	6	0.75	1.418×10^{-3}	1.589×10^{-4}
hybrid	2	1.25	1.996×10^{-3}	3.662×10^{-4}
hybrid	2	1	1.868×10^{-3}	7.134×10^{-5}
hybrid	2	0.75	1.984×10^{-3}	1.467×10^{-4}

Partition-count and residual-strength ablation. Table 22 examines the effect of the partition count M and the residual scaling parameter α for the HiRefPOU-1L model on the cavity benchmark. Across both values of M , the choice $\alpha = 1$ gives the best mean test relative ℓ_2 error, while either increasing or decreasing the residual strength leads to a mild degradation in performance. The impact of the partition count is more pronounced: increasing from $M = 2$ to $M = 6$ substantially improves accuracy for all three α values. These results suggest that the cavity problem benefits more strongly from increased spatial specialization than the Burgers benchmark, while also indicating that the unscaled residual contribution ($\alpha = 1$) provides the most reliable 1L setting among the values tested here.

Table 23: *Cavity benchmark: gate-type ablation for the HiRefPOU-1L model. Reported values are mean $\pm$ standard deviation over random seeds.*

Gate	Final test rel. ℓ_2	Std.
box	1.559×10^{-3}	3.561×10^{-4}
hybrid	1.536×10^{-3}	1.588×10^{-4}
rbf	1.821×10^{-3}	2.109×10^{-4}

Gate-type ablation. Table 23 compares the box, hybrid, and RBF gate parameterizations for the HiRefPOU-1L model on the cavity benchmark. The hybrid gate achieves the best mean test relative ℓ_2 error, though its advantage over the box gate is modest. In contrast, the RBF gate performs noticeably worse than both the box and hybrid variants. Taken together with the Burgers results, this supports the use of the hybrid gate as a consistent default choice: it is never worse in these experiments and is generally either the best or tied for best, while also offering lower variability than the purely RBF formulation.

Child-allocation ablation. Table 24 studies how the distribution of fine-level children across the coarse parents affects performance in the HiRefPOU-2L cavity model. The results show that performance is sensitive to how refinement capacity is allocated. Among the tested configurations, the 0_2 allocation yields the best results, while the 2_0 allocation is consistently the weakest. The balanced 2_2 configuration performs

Table 24: *Cavity benchmark: child-allocation ablation for the HiRefPOU-2L model. Here, M_c and M_f denote the numbers of coarse and fine experts, respectively, while α_c and α_f denote the corresponding residual strengths. The allocation label indicates how fine-level children are distributed across the coarse-level parents. Reported values are mean $\pm$ standard deviation over random seeds.*

Allocation	M_c	M_f	α_c	α_f	Final test rel. ℓ_2	Std.
0_2	2	2	1	1.25	1.955×10^{-3}	1.827×10^{-4}
0_2	2	2	1	0.75	1.920×10^{-3}	1.812×10^{-4}
2_0	2	2	1	1	2.013×10^{-3}	3.089×10^{-4}
2_2	2	4	1	1	1.944×10^{-3}	1.421×10^{-4}
2_0	2	2	1	0.75	2.012×10^{-3}	1.736×10^{-4}
2_0	2	2	1	1.25	2.123×10^{-3}	2.716×10^{-4}
0_2	2	2	1	1	1.934×10^{-3}	1.780×10^{-4}
2_2	2	4	1	1.25	1.966×10^{-3}	1.728×10^{-4}
2_2	2	4	1	0.75	1.971×10^{-3}	1.695×10^{-4}

competitively, but does not outperform the best asymmetric allocation in this experiment. This indicates that, for the cavity benchmark, refinement is most effective when it is concentrated in the parent region associated with the more challenging local structure rather than distributed uniformly by default.

References

- Ben Adcock and Nick Dexter. The gap between theory and practice in function approximation with deep neural networks, 2021. URL <https://arxiv.org/abs/2001.07523>.
- Idriss Barbara, Tawfik Masrour, and Mohammed Hadda. Adaptive partition of unity networks (APUNet): A localized deep learning method for solving PDEs. *Evolving Systems*, 15(4):1137–1158, August 2024. ISSN 1868-6478, 1868-6486. doi:10.1007/s12530-023-09544-7.
- Madison Cooley, Mike Kirby, Shandian Zhe, and Varun Shankar. HyresPINNs: A hybrid residual physics-informed neural network architecture designed to balance expressiveness and trainability. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL <https://openreview.net/forum?id=et9WkjkqAw>.
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation, 2020. URL <https://arxiv.org/abs/2012.14501>.
- Nils Gerhard, Siegfried Müller, and Aleksey Sikstel. A wavelet-free approach for multiresolution-based grid adaptation for conservation laws. *Communications on Applied Mathematics and Computation*, 4(1): 108–142, 2022.
- Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi:10.1016/0893-6080(89)90020-8.
- Amanda A. Howard, Mauro Perego, George Em Karniadakis, and Panos Stinis. Multifidelity deep operator networks for data-driven and physics-informed problems. *Journal of Computational Physics*, 493:112462, November 2023. ISSN 00219991. doi:10.1016/j.jcp.2023.112462.
- Zheyuan Hu. Augmented Physics-Informed Neural Networks (APINNs): A gating network-based soft domain decomposition methodology. *Engineering Applications of Artificial Intelligence*, 2023.

- Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.
- Kookjin Lee, Nathaniel A Trask, Ravi G Patel, Mamikon A Gulian, and Eric C Cyr. Partition of unity networks: Deep hp-approximation. *arXiv preprint arXiv:2101.11256*, 2021.
- Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.
- Miguel Liu-Schiaffini, Julius Berner, Boris Boney, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In *International Conference on Machine Learning*, pp. 32576–32594. PMLR, 2024.
- 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, March 2021. ISSN 2522-5839. doi:10.1038/s42256-021-00302-5.
- Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.
- Tanya Marwah, Zachary C. Lipton, and Andrej Risteski. Parametric complexity bounds for approximating pdes with neural networks, 2021. URL <https://arxiv.org/abs/2103.02138>.
- William F. Mitchell. Adaptive refinement for arbitrary finite-element spaces with hierarchical bases. *Journal of Computational and Applied Mathematics*, 36(1):65–78, 1991. ISSN 0377-0427. doi:[https://doi.org/10.1016/0377-0427\(91\)90226-A](https://doi.org/10.1016/0377-0427(91)90226-A). URL <https://www.sciencedirect.com/science/article/pii/037704279190226A>. Special Issue on Adaptive Methods.
- KB Nakshatrala, DZ Turner, KD Hjelmstad, and Arif Masud. A stabilized mixed finite element method for darcy flow based on a multiscale decomposition of the solution. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):4036–4049, 2006.
- Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.
- Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017a.
- Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017b.
- Juan Mario Restrepo and Gary K Leaf. Wavelet-galerkin discretization of hyperbolic equations. *Journal of Computational Physics*, 122(1):118–128, 1995.
- Arturo Rodriguez, Ashesh Chattopadhyay, Piyush Kumar, Luis F Rodriguez, and Vinod Kumar. Partition of Unity Physics-Informed Neural Networks (POU-PINNs): An Unsupervised Framework for Physics-Informed Domain Decomposition and Mixtures of Experts. *arXiv preprint arXiv:2412.06842*, 2024.
- Varun Shankar and Aaron L. Fogelson. Hyperviscosity-based stabilization for radial basis function-finite difference (rbf-fd) discretizations of advection–diffusion equations. *J. Comput. Phys.*, 372:616 – 639, 2018. ISSN 0021-9991. doi:<https://doi.org/10.1016/j.jcp.2018.06.036>. URL <http://www.sciencedirect.com/science/article/pii/S0021999118304145>.

- Varun Shankar, Grady B. Wright, and Aaron L. Fogelson. An efficient high-order meshless method for advection-diffusion equations on time-varying irregular domains. Journal of Computational Physics, 445:110633, 2021. ISSN 0021-9991. doi:<https://doi.org/10.1016/j.jcp.2021.110633>. URL <https://www.sciencedirect.com/science/article/pii/S0021999121005283>.
- Ramansh Sharma and Varun Shankar. Accelerated training of physics-informed neural networks (PINNs) using meshless discretizations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), Advances in Neural Information Processing Systems, 2022. URL <https://openreview.net/forum?id=NYpU9BR0Dos>.
- Ramansh Sharma and Varun Shankar. Ensemble and mixture-of-experts deepoanets for operator learning. Transactions on Machine Learning Research, 2025. URL <https://openreview.net/forum?id=MGdydNfWzQ>.
- Nathaniel Trask, Mauro Perego, and Pavel Bochev. A high-order staggered meshless method for elliptic problems. SIAM Journal on Scientific Computing, 39(2):A479–A502, 2017. doi:10.1137/16M1055992.
- Hong Wang, Haiyang Xin, Jie Wang, Xuanze Yang, Fei Zha, Yan Jiang, et al. Mixture-of-experts operator transformer for large-scale pde pre-training. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.
- Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing, 43(5):A3055–A3081, 2021a.
- Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 384:113938, 2021b.
- Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. Journal of Computational Physics, 449:110768, 2022.
- Sifan Wang, Bowen Li, Yuhao Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning with residual adaptive networks. arXiv preprint arXiv:2402.00326, 2024.
- Yanzhong Yao, Jiawei Guo, and Tongxiang Gu. A deep learning method for multi-material diffusion problems based on physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 417:116395, December 2023. ISSN 00457825. doi:10.1016/j.cma.2023.116395.