GENERALIZED NEURAL OPERATOR FOR PARAMETRIC AND BOUNDARY-VALUE PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many physical systems are governed by Partial Differential Equations (PDEs), where dynamics are dictated by underlying physical parameters and boundary conditions. Modeling real-world phenomena requires a unified solver capable of generalizing across an entire family of PDE instances, yielding instant solutions for various parameters or boundary constraints without necessitating architectural redesign or retraining. However, existing approaches face a fundamental trilemma, failing to simultaneously achieve computational efficiency, mathematical well-posedness, and generalization. Traditional numerical solvers guarantee well-posedness but are computationally expensive and strictly instance-specific. Physics-Informed Neural Networks (PINNs) offer fast inference and physical fidelity, yet they remain instance-specific and require costly retraining for every new condition. Purely data-driven Neural Operators generalize across instances but treat physics implicitly. By attempting to infer parameters and boundaries from historical frames, they lead to ill-posed formulations. Recent large-scale foundational operators inherit this implicitness while scaling to billions of parameters, largely negating their computational advantage over optimized numerical baselines. To resolve this trilemma, we propose a *Generalized Neural Operator* that restores mathematical rigor through explicit physical conditioning. By formalizing the classical conditions for well-posedness within neural operators, our framework demonstrates the theoretical benefits of explicitly conditioning on PDE parameters and boundary conditions. To implement this synthesis without compromising computational speed, we introduce three novel architectural components: a parameter-gated mixture of kernels for efficient parameter generalization, a generalized boundary transfer operator that projects arbitrary boundary constraints into a unified latent Dirichlet representation, and a specialized training objective to ensure stability. Extensive experiments demonstrate that our theoretically grounded approach achieves superior generalization across heterogeneous physical regimes while maintaining inference efficiency comparable to conventional numerical baselines.

1 INTRODUCTION

Partial Differential Equations (PDEs) constitute the standard mathematical framework for modeling physical phenomena. To fully characterize a specific physical system, these equations are formulated as parametric boundary value problems, in which the combination of equation parameters and boundary conditions fully defines the underlying physics. For example, the heat equation serves as a canonical model for understanding diffusion processes, describing how a quantity such as temperature evolves over time to reach equilibrium. $\frac{\partial u(\boldsymbol{x},t)}{\partial t} = \alpha\nabla^2 u(\boldsymbol{x},t)$, where $u(\boldsymbol{x},t)$ represents the temperature at spatial position $\boldsymbol{x}$ and time t. Central to this physical model is the thermal diffusivity parameter, α, which dictates the intrinsic rate at which heat propagates through a specific medium. A high diffusivity characterizes materials like copper that rapidly conduct thermal energy, while low diffusivity defines insulators that retard this flow. The physical behavior of the system is also constrained by its interactions with the environment, defined mathematically as boundary conditions. Dirichlet condition, $u(\boldsymbol{x},t) = f(\boldsymbol{x},t)$ for $\boldsymbol{x} \in \partial\Omega$, is a scenario where the boundary temperature is strictly controlled, such as an object submerged in a thermal bath. Neumann condition, $\frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x},t) = g(\boldsymbol{x},t)$ for $\boldsymbol{x} \in \partial\Omega$, with $\mathbf{n}$ denoting the outward unit normal vector, represents the

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

control of heat flux at the boundary, where a homogeneous value, $g(\boldsymbol{x}, t) = 0$, implies a perfectly insulated surface that allows no energy escape. Periodic conditions simulate a ring-like or infinite geometry where heat leaving one side re-enters the other. Because real-world applications span an enormous diversity of materials and environmental interactions, PDE solvers must generalize across an entire family of PDE instances, accommodating various parameters and boundary conditions without necessitating architectural redesign or retraining.

	Well-posed	Efficiency	Generalization
Numerical solvers	✓	×	×
PINNs	✓	✓	×
Neural operators	×	✓	✓
Foundation models	×	×	✓
Ours	✓	✓	✓

Table 1: **The trilemma of existing PDE solvers.** **Well-posed**: the solver conditions on a complete specification of the problem so that the target solution is uniquely determined. **Efficiency**: optimal accuracy–efficiency trade-off. **Generalization**: generalizing across PDE parameters and boundary conditions without retraining or architectural changes.

However, existing approaches face a fundamental trilemma, failing to simultaneously achieve computational efficiency, mathematical well-posedness, and generalization. Traditional numerical methods prioritize mathematical rigor by computing state evolutions with strict, explicit adherence to governing equations. Yet, they are computationally expensive and strictly instance-specific. For example, a seemingly simple shift from Dirichlet to periodic boundary conditions demands completely replacing a finite difference scheme with a fundamentally different algorithm, such as a pseudo-spectral method. Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) maintain this rigor and offer rapid inference by embedding explicit equations into the loss function. Nevertheless, they remain instance-specific, requiring costly optimization from scratch for every new parameter or boundary constraint. Purely data-driven Neural Operators (Kovachki et al., 2023) generalize across instances and demonstrate superior accuracy-efficiency tradeoffs by learning infinite-dimensional mappings from large-scale datasets (Wang et al., 2025a). However, they treat physics implicitly, omitting parameters and boundary conditions from their inputs and attempting to deduce them solely from historical frames. Mathematically, this constitutes an ill-posed inverse problem. By failing to formally condition on the environment, these models are prone to dataset biases, inherently fail to capture time-dependent boundary dynamics, and produce averaged, non-physical predictions.

Current attempts to build generalized PDE models struggle precisely because they fail to resolve this trilemma. In pursuit of generalization, they lean heavily into the data-driven operator paradigm but abandon the explicit physical grounding that makes numerical solvers robust. Furthermore, these existing foundational models are scaled to massive sizes, which ironically destroys their computational efficiency. Given that recent studies (Wang et al., 2025a; McGreivy & Hakim, 2024) show even lightweight operators offer only marginal speedups over highly optimized numerical solvers, the practical utility of deploying massive, multi-billion-parameter models to guess implicit physics is highly questionable. We present the comparison in Table 1.

To resolve this trilemma and definitively bridge the gap between computational efficiency, mathematical well-posedness, and generalization, we propose a *Generalized Neural Operator*. We formalize the well-established classical conditions for well-posedness within the context of neural operators, demonstrating that restoring mathematical rigor through explicit physical inputs is the key to true generalization. To achieve this synthesis without sacrificing efficiency, our architecture introduces a parameter-gated mixture of kernels, dynamically routing computation based on explicit physical regimes rather than activating a monolithic network. This ensures physics-agnostic deployment while maintaining runtime efficiency strictly comparable to traditional numerical solvers. Furthermore, we propose a generalized boundary transfer operator that projects arbitrary, explicitly provided boundary conditions into a unified latent Dirichlet representation, allowing the core network to maintain rigor while seamlessly generalizing across boundary configurations. Finally, we introduce a specialized training objective to ensure robust performance across diverse physical regimes.

Our contributions are as follows: **(i) Paradigm Synthesis & Problem Identification:** We identify the fundamental struggle in current PDE solvers and propose a framework that resolves this trilemma. **(ii) Theoretical Formalization:** We formalize the well-posedness of foundational neural solvers, demonstrating theoretically why the structural inclusion of PDE parameters and boundary conditions

is beneficial. **(iii) Practical Solution:** We introduce a parameter-gated mixture of kernels, a generalized boundary transfer operator, and a robust optimization objective to achieve generalization across diverse physical regimes. **(iv) Experimental Validation:** We validate the superior generalization performance of our proposed model through extensive empirical benchmarks.

2 Related Work

Early research on neural operators primarily focuses on learning PDEs with single parameters and fixed boundary conditions. This body of work encompasses Eulerian neural simulators (Li et al., 2021; 2023b; Pfaff et al., 2021; Brandstetter et al., 2022; Rahman et al., 2023; Wu et al., 2024; Luo et al., 2025; Li et al., 2023a; Tran et al., 2023; Wu et al., 2023), which operate on regular grids or irregular meshes, and Lagrangian neural simulators (Sanchez-Gonzalez et al., 2020; Toshev et al., 2024a; Prantl et al., 2022; Toshev et al., 2024b), which discretize the domain into particles.

Recent initiatives have sought to develop foundational models that extend the capabilities of neural operators in several directions. Research such as Alesiani et al. (2022); Takamoto et al. (2023) focuses on generalizing operators across PDE parameters, while Boudec et al. (2025) introduces a physics-guided iterative training algorithm to handle diverse parameters and boundary conditions. Unified frameworks have also emerged, including the work by Alkin et al. (2024) for joint Eulerian and Lagrangian simulations and Chen et al. (2025) for multiscale pretraining across 1D, 2D, and 3D domains. Ye et al. (2024) utilizes computational graphs to represent various PDEs, and Shen et al. (2024) embeds systems into shared spaces to enable generalization across dimensions and resolutions.

Saad et al. (2023) investigates architectural designs for enforcing exact boundary conditions, and Yang et al. (2023); Yang & Osher (2024) employs in-context learning to infer operators dynamically using example pairs as prompts. Furthermore, transformer-based multiphysics solvers (Hao et al., 2024; Wang et al., 2025b; McCabe et al., 2024; Subramanian et al., 2023; Sun et al., 2025; Herde et al., 2024; Zhou et al., 2025) conduct pretraining on large-scale datasets containing numerous PDEs and conditions. Some of these works rely on fine-tuning to generalize well to new physics. We approach the problem from a numerical PDE perspective by learning models that generalize to new parameters and boundary conditions without any instance-specific fine-tuning.

The landscape of scientific machine learning encompasses a diverse array of applications that extend far beyond forward modeling. Significant research has been dedicated to tasks such as inferring dense fluid fields from sparse observations (Li et al., 2025c; Zhong et al., 2023; Mo & Magri, 2024; Yadav et al., 2025; Jing et al., 2024; He et al., 2022) and enhancing the performance of classical numerical schemes with neural networks (List et al., 2022; Sun et al., 2023; Greenfeld et al., 2019; Sappl et al., 2019). Furthermore, neural networks play a crucial role in engineering workflows through aerodynamic shape optimization (Elrefaie et al., 2025; 2024) and inverse design methodologies (Behrmann et al., 2019; Teng & Choromanska, 2019; Kruse et al., 2021). While these contributions fall broadly within the same context of scientific machine learning as our work, they address objectives distinct from the primary focus of this study.

3 Problem Statement

We focus on generalizing within specific PDE families to ensure efficiency compared to conventional numerical solvers. We consider the **Parametric Boundary Value Problem**. Let $\Omega \subseteq \mathbb{R}^d$ be a bounded domain with boundary $\partial\Omega$. We consider a family of first-order in time partial differential equations with parameters $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \mathcal{L}_{\boldsymbol{\theta}}[\boldsymbol{u}] = 0 & (\boldsymbol{x}, t) \in \Omega \times (0, T] \\ \mathcal{B}[\boldsymbol{u}] = 0 & \\ \boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}) & \boldsymbol{x} \in \Omega \end{cases} \tag{1}$$

where $\mathcal{L}_{\boldsymbol{\theta}}$ is the parameterized differential operator, $\boldsymbol{u}_0$ is the initial condition, and $\boldsymbol{u}(\boldsymbol{x}, t)$ is the solution of interest. $\mathcal{B}$ is the boundary operator. This operator specifies the type of boundary constraint enforced and the associated boundary data, such as the specific Dirichlet values or the magnitude of

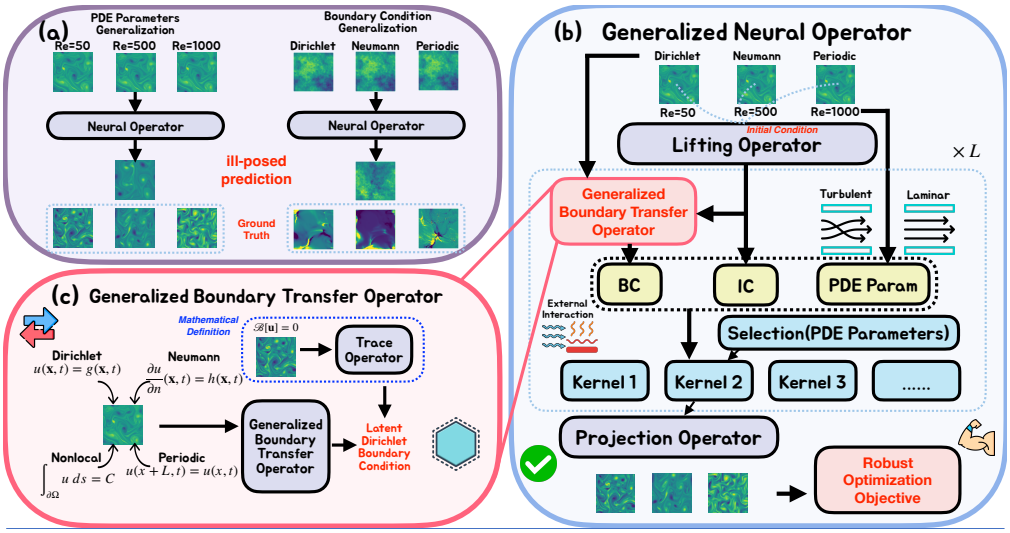


Figure 1: **(a)** Lack of physical inputs renders the problem ill-posed, causing the model to converge toward an averaged solution. **(b)** We propose a *Generalized Neural Operator*, which incorporates a parameter-gated mixture of kernels and a robust optimization training objective to improve generalization. **(c)** To enhance boundary generalization, we introduce a generalized boundary transfer operator that maps arbitrary boundary conditions into a latent Dirichlet boundary state.

the Neumann flux. We adopt the notation where square brackets $[\cdot]$ denote an operator on a function, while parentheses $(\cdot)$ denote the evaluation of a function at specific coordinates.

Let $\mathcal{Y}$ denote the topological vector space representing the solution trajectory. Let $\mathcal{X}_{\text{init}}$ denote the topological vector space representing the initial conditions. We define the solution operator $\mathcal{S}$ with respect to a fixed configuration of the PDE structure. Specifically, for a fixed parameter $\boldsymbol{\theta}$ and boundary $\mathcal{B}$, the operator $\mathcal{S}_{\boldsymbol{\theta},\mathcal{B}} : \mathcal{X}_{\text{init}} \to \mathcal{Y}$ maps the initial condition to the solution trajectory: $\boldsymbol{u} = \mathcal{S}_{\boldsymbol{\theta},\mathcal{B}}[\boldsymbol{u}_0]$, where $\boldsymbol{u}$ satisfies the system in Equation 1.

Assume the structural form of the differential operator $\mathcal{L}$ is fixed. We consider variations in the PDE parameters $\boldsymbol{\theta} \in \Theta$, boundary operator $\mathcal{B} \in \mathbb{B}$, where $\mathbb{B}$ is the set of admissible boundary operators. Let $\mathfrak{S} = \{\mathcal{S}_{\boldsymbol{\theta},\mathcal{B}} \mid \boldsymbol{\theta} \in \Theta, \mathcal{B} \in \mathbb{B}\}$ denote the set of all solution operators arising from admissible combinations of these varying components. Our goal is to learn a **Generalized Neural Operator** $\mathcal{M}_\phi$, parameterized by ϕ, that approximates the family of operators in $\mathfrak{S}$.

4 Method

We first formalize the theoretical conditions required for operator well-posedness, demonstrating that explicit input of PDE parameters and boundary conditions is beneficial. Building on this mathematical foundation, we introduce a parameter-gated mixture of kernels, a generalized boundary transfer operator, and a robust optimization training objective. The overall framework is in Figure 1.

4.1 Theoretical Formalization

Proposition 4.1. *Let $\mathfrak{S} = \{\mathcal{S}_{\boldsymbol{\theta},\mathcal{B}} \mid \boldsymbol{\theta} \in \Theta, \mathcal{B} \in \mathbb{B}\}$ be the family of solution operators defined in Equation 1. Then, a single-valued operator $\mathcal{M}_\phi : \mathcal{X}_{\text{init}} \to \mathcal{Y}$ defined solely on the initial condition space cannot universally represent the family $\mathfrak{S}$. For the Generalized Neural Operator to be well-defined as a function approximating $\mathfrak{S}$, its domain must be augmented to include the parameter space Θ and boundary operator space $\mathbb{B}$.*

Proof. See Appendix Q. □

We present the well-posedness of the neural operator in Proposition 4.1. It reveals a fundamental mathematical limitation in purely implicit foundational models. Since identical initial conditions can

lead to entirely different trajectories under distinct physical configurations, an operator that maps solely from the initial condition space to the solution space lacks sufficient constraints to isolate a single, physically valid outcome.

Remark: Ill-Posedness of Implicit Parameter Inference

While recent works (Yin et al., 2022; Koupaï et al., 2024; Nzoyem et al., 2025; Koupaï et al., 2025) have demonstrated competitive empirical performance when implicitly inferring PDE parameters from historical state trajectories, this approach fundamentally reframes the forward simulation as a hidden inverse problem. Mathematically, deducing governing parameters solely from observed states is inherently ill-posed. Consequently, reliance on implicit inference introduces an irreducible epistemic uncertainty.

Remark: Causal Impossibility of Implicit Boundary Inference

While static boundaries can be inferred from rich historical context, dynamic or time-dependent boundary conditions present a hard causal barrier. Consider a time-varying Dirichlet boundary condition. It is fundamentally impossible for a neural operator to causally deduce the unobserved, future state of this external influence from the system's past sequence. In such scenarios, implicit inference is strictly impossible.

We formulate *Generalized Neural Operator* $\mathcal{M}_\phi$ over an augmented domain that incorporates the PDE configuration. We define $\mathcal{M}_\phi$ as a mapping from the Cartesian product of the initial condition space, PDE parameter space, and boundary operator space to the solution trajectory space. Mathematically, this is expressed as: $\mathcal{M}_\phi : \mathcal{X}_{\text{init}} \times \Theta \times \mathbb{B} \to \mathcal{Y}$. Under this formulation, the neural operator takes the tuple $(\boldsymbol{u}_0, \boldsymbol{\theta}, \mathcal{B})$ as input and predicts the corresponding solution trajectory $\hat{\boldsymbol{u}}$ that approximates the true solution defined by the operator $\mathcal{S}_{\boldsymbol{\theta},\mathcal{B}}[\boldsymbol{u}_0]$. This augmentation ensures that $\mathcal{M}_\phi$ is well-defined, capable of distinguishing between varying physical dynamics and boundary constraints. The PDE parameters are embedded directly from their numerical values, while the boundary conditions are encoded using a combination of their type and the specific boundary data.

Our approach readily generalizes to higher-order temporal PDEs by augmenting the input with additional initial snapshots. For example, in Newtonian dynamics governed by second-order time derivatives, the input would consist of both the initial position and the initial velocity.

4.2 Parameter-Gated Mixture of Kernels

Building upon the neural operator framework (Kovachki et al., 2023), our architecture consists of three components: (1) a local lifting transformation, (2) a sequence of iterative kernel integration layers, and (3) a local projection to the output space: $\mathcal{M}_\phi = (\mathcal{Q} \circ \Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_1 \circ \mathcal{E})$, where L denotes the depth of the network. The lifting operator $\mathcal{E}$ maps the physical inputs to a higher-dimensional latent feature $\boldsymbol{v}^{(0)}$. The operator, Φ_{l+1}, transforms the hidden representation $\boldsymbol{v}^{(l)}$ to $\boldsymbol{v}^{(l+1)}$ via the sum of a local linear operator and a non-local integral kernel. The final representation $\boldsymbol{v}^{(L)}$ is mapped to the target solution $\boldsymbol{u} \in \mathcal{Y}$ via the projection operator $\mathcal{Q}$. We refer readers to Appendix S for our pseudocode.

To generalize across the parameter space Θ, the neural operator must adapt its integration geometry based on $\boldsymbol{\theta}$. In a standard neural operator, the integral kernel acts as a static mechanism, applying the same spatial aggregation rules across all problem instances. We overcome this limitation by replacing the static kernel with a parameter-gated mixture of kernels, allowing the network to dynamically assemble a custom integration rule for each specific PDE instance.

Let $\boldsymbol{W}$ be a learnable weight matrix performing a linear transformation, $\boldsymbol{\kappa}_k$ be K distinct base kernel functions that define the non-local integral operator, g_k be a hard gating function, and σ be an activation function. $\boldsymbol{h}^{(l)}$ is the latent Dirichlet boundary condition created by the generalized boundary transfer operator as introduced in Section 4.3. We use the angle brackets $\langle \cdot, \cdot, \cdot \rangle$ to denote concatenation.

To update the state at any specific target point $\boldsymbol{x}$, the integral simultaneously gathers information from surrounding spatial points (the physical domain) and from the governing conditions (boundaries and

parameters). Physically, $\boldsymbol{\kappa}_k(\boldsymbol{x}, \boldsymbol{x}')$ calculates the influence of the features at source point $\boldsymbol{x}'$ on the target point $\boldsymbol{x}$. We project the PDE parameters into a higher-dimensional feature space, denoted by $\tilde{\boldsymbol{\theta}}$, via a learnable transformation. The update rule for the hidden state $\boldsymbol{v}^{(l)}(\boldsymbol{x})$ at layer l is given by:

$$\boldsymbol{v}^{(l+1)}(\boldsymbol{x}) = \sigma\left(\boldsymbol{W}\langle\tilde{\boldsymbol{\theta}}, \boldsymbol{h}^{(l)}, \boldsymbol{v}^{(l)}(\boldsymbol{x})\rangle + \int_{\Omega\cup\mathcal{X}_{ctx}} \left(\sum_{k=1}^{K} g_k(\boldsymbol{\theta})\boldsymbol{\kappa}_k(\boldsymbol{x}, \boldsymbol{x}')\right) \boldsymbol{\xi}(\boldsymbol{x}')\, d\mu(\boldsymbol{x}') + \boldsymbol{b}(\boldsymbol{x})\right), \quad \forall \boldsymbol{x} \in \Omega \tag{2}$$

The measure $\mu(\boldsymbol{x}')$ is defined as the standard Lebesgue measure for $\boldsymbol{x}' \in \Omega$ and the counting measure for $\boldsymbol{x}' \in \mathcal{X}_{ctx}$. The augmented input function $\boldsymbol{\xi}(\boldsymbol{x}')$ represents the latent state $\boldsymbol{v}^{(l)}(\boldsymbol{x}')$ for $\boldsymbol{x}' \in \Omega$, and the conditioning information (parameter $\boldsymbol{\theta}$ or boundary $\boldsymbol{h}$) for $\boldsymbol{x}' \in \mathcal{X}_{ctx}$. In practice, we adopt self-attention (Vaswani et al., 2023) kernels as this is the best approach reported in Wang et al. (2025a).

We employ the hard-gating strategy to select the most relevant kernel for a given parameter configuration. Let $s_k(\boldsymbol{\theta})$ be a learnable, lightweight score function that measures the affinity between the physical parameter $\boldsymbol{\theta}$ and the kernel κ_k. We compute the selection probabilities $\tilde{g}_k$ and the discrete gates g_k as follows: $\tilde{g}_k(\boldsymbol{\theta}) = \frac{\exp(s_k(\boldsymbol{\theta}))}{\sum_{j=1}^{K}\exp(s_j(\boldsymbol{\theta}))}$, $g_k(\boldsymbol{\theta}) = \begin{cases} 1, & \text{if } k = \arg\max_j \tilde{g}_j(\boldsymbol{\theta}) \\ 0, & \text{otherwise} \end{cases}$

Motivation: Our parameter-gated mixture of kernels is motivated by both experimental observations and physical intuition. We initially tested strong single-kernel baselines that explicitly encode PDE parameters, specifically *CAPE+FNO* and *CAPE+Attn* from Takamoto et al. (2023), as well as *Unisolver* (Zhou et al., 2025). We observed that performance varies significantly depending on the PDE parameters, suggesting that single-kernel models often excel in specific domains while failing in others. This necessitates a mixture of kernels to better adapt to the diverse physical conditions created by varying PDE parameters. We refer readers to the Appendix F for full experimental details.

Physically, variations in PDE parameters often drive the system across distinct behavioral regimes rather than merely scaling the output magnitude. A classic example is the Reynolds number in fluid dynamics. Shifting this parameter transitions the system from laminar flow to turbulence. A single kernel forces a compromise, attempting to learn an average operator that is often too diffusive for turbulence yet too reactive for laminar states. Our parameter-gated approach mirrors this physical reality by treating different parameter spaces as distinct regimes.

Further, as emphasized in Section 1, efficiency is paramount for the practical deployment of foundational PDE solvers. Our mixture of kernels incurs minimal computational overhead during inference, allowing the model to achieve superior accuracy while maintaining a significant speed advantage over traditional numerical solvers.

Discussion with Related Work: Mixture of kernel-based neural operators have been explored in previous works. Deighan et al. (2025) use all kernels where gating depends on the spatial coordinates. Wang et al. (2025b) use more than one kernel and use latent initial conditions as gating. By contrast, we only select one kernel to ensure efficiency and use PDE parameters for explicit domain partition. These innovations, though simple, are crucial for model generalization and practical utility.

4.3 Generalized Boundary Transfer Operator

Mathematical Motivation: The Dirichlet-to-Neumann (DtN) and Neumann-to-Dirichlet (NtD) operators (Arendt et al., 2014; Sauter, 2013; Bossavit, 1991; Quarteroni & Valli, 1999; Oberai et al., 1998; Knockaert & De Zutter, 2008) act as inverse transfer functions for the domain. Let Ω_E be a bounded domain with a smooth boundary $\partial\Omega_E$, governed by a uniformly elliptic linear operator $\mathcal{L}_E$. The Dirichlet-to-Neumann (DtN) Operator, Λ, determines the necessary Neumann flux to sustain a given Dirichlet boundary state. Formally, $\Lambda[\boldsymbol{\varphi}] := \partial_{\boldsymbol{n}}\boldsymbol{u}_E|_{\partial\Omega_E}$, where $\boldsymbol{\varphi}$ is the Dirichlet boundary value. The Neumann-to-Dirichlet (NtD) Operator, η, operating in reverse, yields the Dirichlet boundary values that result from an applied Neumann flux. $\eta[\boldsymbol{\psi}] := \boldsymbol{u}_E|_{\partial\Omega_E}$, where $\boldsymbol{\psi}$ is a prescribed Neumann boundary flux.

Proposed Method: While foundational theory and the majority of existing literature on these operators focus primarily on time-independent elliptic PDEs with standard Neumann or Dirichlet conditions, we propose a more versatile approach. We propose to learn a generalized boundary transfer operator designed to map arbitrary boundary conditions to a shared latent Dirichlet representation. By

unifying diverse boundary inputs into a single modality, this method allows the neural operator kernel to specialize in one type of boundary condition, thereby reducing complexity and enhancing model generalization. For time-dependent PDEs, strictly mapping a boundary condition to a Dirichlet output without the initial condition is an ill-posed problem. Thus, our transfer operator also incorporates the initial condition for capturing the system's complete boundary dynamics.

Let $\gamma : \mathcal{Y} \to \mathcal{Y}_{\partial\Omega}$ denote a trace operator, which restricts a function defined on the domain Ω to a latent value on the boundary $\partial\Omega$. We define this latent value as the latent Dirichlet value: $\boldsymbol{h} := \gamma[\boldsymbol{u}] := \gamma\left[\mathcal{S}_{\boldsymbol{\theta},\mathcal{B}}(\boldsymbol{u}_0)\right]$, where $\boldsymbol{u}$ is the solution trajectory of the PDE and $\boldsymbol{h}$ is the latent Dirichlet value. The trace operator γ is a theoretical construct utilized solely to formally define the latent Dirichlet value. Because defining this value requires the ground truth trajectory, which is inherently unavailable during inference, we approximate it in practice using a generalized boundary transfer operator, $\mathcal{T}_{\boldsymbol{\lambda}}$, parametrized by $\boldsymbol{\lambda}$. We design $\mathcal{T}_{\boldsymbol{\lambda}}$ to function similarly to the classical DtN and NtD operators that map boundary data from one type to another: $\boldsymbol{h}^{(l)} \approx \mathcal{T}_{\boldsymbol{\lambda}}(\mathcal{B}, \boldsymbol{v}^{(l)})$. This operator maps the boundary configuration encapsulated by $\mathcal{B}$ and the interior dynamics to the equivalent latent Dirichlet data. We use the layer-wise latent state $\boldsymbol{v}^{(\ell)}$ as the input to the transfer operator to resolve the state-dependency of boundary interactions. In non-linear and time-dependent PDEs, the influence of a boundary condition is rarely static and is coupled with the instantaneous state of the field near the boundary. This allows the operator to dynamically adjust the boundary forcing based on the evolving physics of the interior domain. We argue that $\boldsymbol{v}^{(\ell)}$ serves as a more informative input than the initial condition $\boldsymbol{u}_0$ to the transfer operator. This learned boundary data subsequently serves as the input for Equation 2 to solve the PDE. In our implementation, $\mathcal{T}_{\boldsymbol{\lambda}}$ is parameterized as a lightweight, single-head self-attention, whose input consists of boundary type and associated boundary data.

4.4 Robust Optimization Objective

In Appendix F, we observe that existing models exhibit large performance disparities across PDE parameters. Beyond physical domain partitioning, we attribute this issue to standard empirical risk minimization under MSE loss. A model can achieve low average loss by perfecting easy, smooth cases, while failing catastrophically on difficult, high-frequency cases. This is especially pronounced in CAPE (Takamoto et al., 2023) variants. To build a generalized neural solver, we require uniform accuracy across PDE parameters to ensure reliability. To tackle this, we propose a Group Distributionally Robust Optimization (Kuhn et al., 2025; Gorissen et al., 2015; Rahimian & Mehrotra, 2022; Blanchet et al., 2024) training objective: $\mathcal{L}_{\text{DRO}}(\phi) = \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}'} \mathbb{E}_{(\boldsymbol{u}_0,\boldsymbol{u},\mathcal{B})\sim\mathcal{D}_{\boldsymbol{\theta}}}\left[\ell(\mathcal{M}_\phi(\boldsymbol{u}_0,\boldsymbol{\theta},\mathcal{B}),\boldsymbol{u})\right]$, where $\boldsymbol{\Theta}'$ denotes the PDE parameter set of a training batch, $\mathcal{D}_{\boldsymbol{\theta}}$ denotes the subset of training data corresponding to the PDE parameter $\boldsymbol{\theta}$, and ℓ represents the MSE. Since the hard maximization is non-differentiable, we approximate the objective using a differentiable Log-Sum-Exp (Zhang et al., 2023) relaxation: $\mathcal{L}_{\text{DRO}}(\phi) \approx \frac{1}{\tau}\log\left(\frac{1}{|\boldsymbol{\Theta}'|}\sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}'} e^{\left(\tau\mathbb{E}_{(\boldsymbol{u}_0,\boldsymbol{u},\mathcal{B})\sim\mathcal{D}_{\boldsymbol{\theta}}}[\ell(\mathcal{M}_\phi(\boldsymbol{u}_0,\boldsymbol{\theta},\mathcal{B}),\boldsymbol{u})]\right)}\right)$, where τ is the temperature parameter. When the cardinality of the parameter set $|\boldsymbol{\Theta}'|$ is prohibitively large, or the space is continuous, we employ a partitioning strategy. We discretize $\boldsymbol{\Theta}'$ into coarser bins, treating each bin as a distinct group for the DRO calculation.

Our training curriculum proceeds by iterating through each boundary condition type. In each step, we minimize the DRO objective. Detailed procedural steps are provided in Algorithm 1. We found that initiating the training directly with the DRO objective is overly restrictive. Thus, we adopt a two-stage curriculum. First, we train the network using standard MSE to establish a baseline approximation of the physics. After P epochs, we switch to the Group DRO loss to enforce uniform accuracy, specifically targeting and refining performance in the most challenging regimes.

5 Experiment

First, we individually validate the model's generalization ability across PDE parameters and boundary conditions by fixing the other to a specific setting. This ensures that we evaluate the model's robustness with respect to only one changing factor at a time. Then, we conduct extensive testing specifically covering simultaneous PDE parameters and boundary conditions variations. We demonstrate that our proposed method achieves superior performance while maintaining computational efficiency against numerical solvers.

Method	Heat	Advection	Burgers	Inc. NS
ViT-2	3.15×10^{-2}	8.95×10^{-1}	9.43×10^{-2}	3.84×10^{-1}
ViT-5	1.49×10^{-2}	2.90×10^{-1}	1.25×10^{-2}	4.45×10^{-1}
ViT-10	1.08×10^{-2}	1.40×10^{-1}	8.36×10^{-2}	5.43×10^{-1}
Concat	1.60×10^{-2}	9.48×10^{-1}	4.19×10^{-2}	6.12×10^{-1}
CAPE+Unet	3.05×10^{-1}	1.61×10^{-1}	1.30×10^{-2}	2.92×10^{-1}
CAPE+FNO	1.28×10^{-2}	1.69×10^{-1}	7.53×10^{-3}	1.67×10^{-1}
CAPE+Attn	1.52×10^{-2}	1.54×10^{-1}	1.39×10^{-2}	5.35×10^{-1}
Unisolver	6.89×10^{-3}	4.53×10^{0}	9.88×10^{-3}	9.09×10^{-1}
MoE-POT	6.84×10^{-1}	1.49×10^{-1}	5.29×10^{-2}	1.09×10^{0}
Ours	$\mathbf{4.20 \times 10^{-3}}$	$\mathbf{6.03 \times 10^{-2}}$	$\mathbf{2.82 \times 10^{-3}}$	$\mathbf{1.06 \times 10^{-1}}$

Table 2: Generalization across PDE parameters. We evaluate the normalized MSE on the Heat, Advection, Burgers, and Incompressible Navier-Stokes equations. For each equation, we vary the PDE parameters across the samples in the dataset to test generalization.

Method	Heat	Advection
ViT-2	2.33×10^{-1}	1.56×10^{-1}
ViT-5	1.57×10^{-1}	1.48×10^{-1}
ViT-10	9.18×10^{-2}	1.07×10^{-1}
Concat-Edge	1.48×10^{-1}	6.32×10^{-1}
Concat-Broadcast	3.02×10^{-1}	5.58×10^{-1}
Unisolver	1.15×10^{-1}	7.23×10^{-2}
MoE-POT	1.59×10^{-1}	4.14×10^{-1}
Ours	$\mathbf{7.80 \times 10^{-2}}$	$\mathbf{5.59 \times 10^{-2}}$

Table 3: Generalization across boundary conditions, including Dirichlet, Neumann, and periodic. We report the normalized MSE, averaged over different boundary conditions. For Dirichlet and Neumann boundaries, we randomly sample boundary values or fluxes.

To evaluate generalization across PDE parameters, we utilize four distinct partial differential equations: the Heat, Advection, and Burgers equations, as well as the incompressible Navier-Stokes equations. For the experiments concerning boundary condition generalization and joint pre-training, we focus specifically on the Heat and Advection equations. Detailed descriptions of these datasets and justifications for our benchmark selection are provided in Appendix C and Appendix D. All models predict the next-frame solutions. We evaluate trajectories using a 10-step rollout. We utilize the normalized MSE as the evaluation metric: $\text{nMSE} = \frac{\|\hat{\boldsymbol{u}} - \boldsymbol{u}\|^2}{\text{var}(\boldsymbol{u})}$, where $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ are the ground truth and predicted fields.

5.1 Generalization across PDE Parameters

We include *ViT-k* (Dosovitskiy et al., 2021), which aligns with recent foundational models by relying on initial frames to implicitly infer PDE parameters. k denotes the number of initial frames. We also include *Concat*, which follows Subramanian et al. (2023) and concatenates PDE parameters with initial conditions. We adapt the CAPE mechanism (Takamoto et al., 2023) to create *CAPE+Unet*, *CAPE+FNO*, and *CAPE+Attn*. Finally, we compare against *Unisolver* (Zhou et al., 2025) and *MoE-POT* (Wang et al., 2025b). We refer readers to the Appendix E for implementation details.

We present the results in Table 2. Our model achieves the lowest nMSE across all four PDE families, outperforming the strongest baseline on each. It reduces error by roughly 39% on Heat, 57% on Advection, 62% on Burgers, and 37% on Incompressible NS relative to the best competing baseline. We attribute the performance gain to the explicit physics encoding in our architecture. By using PDE parameters to govern kernel selection, we decouple control from prediction.

5.2 Generalization across Boundary Conditions

Existing boundary enforcement strategies often lack generalization, being limited to specific boundary types (Saad et al., 2023) or elliptic PDEs (Wang et al., 2024). For comparison, we utilize two standard baselines, *ViT-k* (Dosovitskiy et al., 2021) and *MoE-POT* (Wang et al., 2025b), alongside the boundary-aware *Unisolver* (Zhou et al., 2025). We also introduce two naive concatenation baselines that concatenate boundary information to the initial condition field. *Concat-Edge* maps boundary values strictly to the edge points while zeroing out the interior, and *Concat-Broadcast* broadcasts these values across the channel dimension.

Table 3 summarizes the results. We observe that increasing the number of initial frames enhances the performance of the *ViT* variants. We attribute this to the temporal smoothness of the boundary data, which allows the model to implicitly extrapolate future boundary conditions from the extended history. Nevertheless, explicit boundary encoding remains superior, as demonstrated by our proposed model achieving the highest accuracy.

5.3 Unified Generalization: PDE Parameters and Boundary Conditions

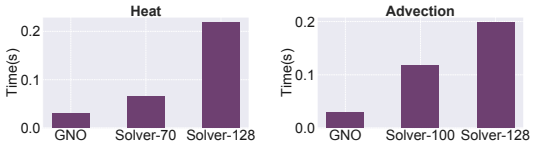
Table 4 presents main results, demonstrating that our proposed model outperforms baselines. This performance not only highlights the critical role of explicit encoding of PDE parameters and boundary conditions, but also validates the superiority of our design choices. The PDE parameters in the Heat equation generate drastically different physical regimes. Combined with varying boundary conditions, this creates a highly complex learning landscape. While several baselines exhibit large nMSE values, suggesting a failure to learn, our method successfully adapts to these varying physical conditions.

Method	Heat	Advection
ViT-2	2.60×10^{1}	3.33×10^{-1}
ViT-5	2.44×10^{0}	2.51×10^{-1}
ViT-10	2.06×10^{0}	2.28×10^{-1}
Unisolver	8.71×10^{-1}	5.02×10^{-1}
MoE-POT	1.85×10^{0}	3.80×10^{-1}
Ours	$\mathbf{3.62 \times 10^{-1}}$	$\mathbf{7.97 \times 10^{-2}}$

Table 4: Generalization across PDE parameters and boundary conditions. We evaluate the normalized MSE on the Heat and Advection equations.

Following McGreivy & Hakim (2024); Wang et al. (2025a), we compare the runtime of our neural model against numerical solvers calibrated to lower resolutions that yield equivalent error. Specifically, we progressively reduce the spatial resolution of the numerical solvers until their accuracy matches that of the neural solver. This approach serves as a fair-comparison methodology to rigorously evaluate the trade-off between accuracy and computational efficiency when comparing numerical and neural solvers.

As shown in Figure 2, we observe an approximate $2\times$ speedup for the Heat equation and a $4\times$ speedup for the Advection equation. While this speedup is more modest than the orders-of-magnitude improvements reported in some prior literature, under such strict evaluation protocols, many existing neural solvers actually perform slower than numerical baselines. As noted in McGreivy & Hakim (2024), even the most competitive models achieve only a marginal $7\times$ speedup on datasets with fixed PDE parameters and boundary conditions. Consequently, our model demonstrates a highly favorable accuracy-efficiency trade-off.



(a) Heat (b) Advection

Figure 2: Runtime comparison between our *GNO* and numerical solvers. The ground truth, operating at a 128×128 resolution, is denoted as Solver-128. For our accuracy-matched baselines, we utilize Solver-70 and Solver-100, which operate at 70×70 and 100×100 resolutions, respectively.

Performance Stability We report the standard deviation of the error on the Advection equation in Figure 4 in Appendix H. Our model exhibits the lowest standard deviation, demonstrating that our training curriculum and robust optimization loss ensure reliable performance suitable for real-world applications. Interestingly, *ViT-2* also shows low standard deviation. This is likely because the limited input context forces the model to predict averaged dynamics.

5.4 Additional Experiments

Ablation Study and Model Analysis: Appendix J.1

Additional Experiment on 3D dataset: Appendix K

Additional Experiment on Out-of-Distribution Generalization: Appendix L

Additional Comparison with Physics Informed Neural Operator (PINO): Appendix M

Additional Experiment on Complex PDE: Appendix N

Additional Analysis on Boundary Condition: Appendix O

6 Conclusion

We address the critical limitations of existing foundational models by proposing a theoretically grounded *Generalized Neural Operator*. We achieve superior generalization while maintaining the computational efficiency of numerical solvers.

7 AI USE STATEMENT

In this work, we used generative AI tools for debugging code and for improving the clarity, flow, and grammar of our writing. We have reviewed all AI-assisted work. Specifically, all AI-suggested text revisions were manually reviewed by the authors to ensure they accurately preserved our original scientific meaning and intent, and all AI-assisted code debugging was rigorously tested and verified for correctness in our execution environment. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Francesco Alesiani, Makoto Takamoto, and Mathias Niepert. Hyperfno: Improving the generalization behavior of fourier neural operators. In *Workshop on Machine Learning and the Physical Sciences (ML4PS) at the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022. URL https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_89.pdf.

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers. *arXiv preprint arXiv:2402.12365*, 2024.

Wolfgang Arendt, A. F. M. ter Elst, James B. Kennedy, and Manfred Sauter. The dirichlet-to-neumann operator via hidden compactness. *Journal of Functional Analysis*, 266:1757–1786, 2014.

Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Flow matching meets pdes: A unified framework for physics-constrained generation, 2025. URL https://arxiv.org/abs/2506.08604.

J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2019.

Jose Blanchet, Jiajin Li, Sirui Lin, and Xuhui Zhang. Distributionally robust optimization and robust statistics, 2024. URL https://arxiv.org/abs/2401.14655.

Alain Bossavit. The 'scalar' poincaré–steklov operator and the 'vector' one: algebraic structures which underlie their duality. In R. Glowinski (ed.), *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pp. 19–26, Philadelphia, PA, 1991. SIAM. ISBN 978-0-89871-278-0.

Lise Le Boudec, Emmanuel de Bezenac, Louis Serrano, Ramon Daniel Regueiro-Espino, Yuan Yin, and Patrick Gallinari. Learning a neural solver for parametric PDEs to enhance physics-informed methods. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=jqVj8vCQsT.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Tianyu Chen, Haoyi Zhou, Ying Li, Hao Wang, Chonghan Gao, Rongye Shi, Shanghang Zhang, and Jianxin Li. Omniarch: Building foundation model for scientific computing. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=UlprLwWYKP.

Dwyer Deighan, Jonas A. Actor, and Ravi G. Patel. Mixture of neural operator experts for learning boundary conditions and model selection, 2025. URL https://arxiv.org/abs/2502.04562.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Mohamed Elrefaie, Florin Morar, Angela Dai, and Faez Ahmed. Drivaernet++: A large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 499–536. Curran Associates, Inc., 2024.

Mohamed Elrefaie, Angela Dai, and Faez Ahmed. Drivaernet: A parametric car dataset for data-driven aerodynamic design and prediction. *Journal of Mechanical Design*, 147(4), 2025.

Bram L. Gorissen, Ihsan Yanıkoglu, and Dick den Hertog. A practical guide to robust optimization. *Omega*, 53:124–137, June 2015. ISSN 0305-0483. doi: 10.1016/j.omega.2014.12.006. URL http://dx.doi.org/10.1016/j.omega.2014.12.006.

Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In *International Conference on Machine Learning*, pp. 2415–2423. PMLR, 2019.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*, 2024.

Xiaodong He, Yinan Wang, and Juan Li. Flow completion network: Inferring the fluid dynamics from incomplete flow information using graph neural networks. *Physics of Fluids*, 34(8), August 2022. ISSN 1089-7666. doi: 10.1063/5.0097688. URL http://dx.doi.org/10.1063/5.0097688.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes, 2024.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation, 2024. URL https://arxiv.org/abs/2406.17763.

Gang Jing, Huan Wang, Xianting Li, Guijin Wang, and Yingying Yang. An airflow velocity field reconstruction method with sparse or incomplete data using physics-informed neural network. *Journal of Building Engineering*, 88:109231, July 2024. Published: 1 July 2024.

Luc F. Knockaert and Daniel De Zutter. On the complex symmetry of the poincaré-steklov operator. *Progress in Electromagnetics Research B*, 7:145–157, 2008. doi: 10.2528/PIERB08022102.

Armand Kassaï Koupaï, Lise Le Boudec, Louis Serrano, and Patrick Gallinari. ENMA: Tokenwise autoregression for continuous neural PDE operators. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=3CYXSMFv55.

Armand Kassaï Koupaï, Jorge Mifsut Benet, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Geps: Boosting generalization in parametric pde neural solvers through adaptive conditioning, 2024. URL https://arxiv.org/abs/2410.23889.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *JMLR*, 24(1), 2023.

J. Kruse, L. Ardizzone, C. Rother, and U. Köthe. Benchmarking invertible architectures on inverse problems. *arXiv preprint*, 2021.

Daniel Kuhn, Soroosh Shafiee, and Wolfram Wiesemann. Distributionally robust optimization, 2025. URL https://arxiv.org/abs/2411.02549.

Edward Li, Zichen Wang, Jiahe Huang, and Jeong Joon Park. VideoPDE: Unified generative PDE solving via video inpainting diffusion models, 2025a. URL https://openreview.net/forum?id=CtBKITrZgp.

Ruoyan Li, Zijie Huang, Haixin Wang, Guancheng Wan, Yizhou Sun, and Wei Wang. Self-guided diffusion model for accelerating computational fluid dynamics, 2025b. URL https://arxiv.org/abs/2504.04375.

Ruoyan Li, Guancheng Wan, Zijie Huang, Zixiao Liu, Haixin Wang, Xiao Luo, Wei Wang, and Yizhou Sun. Flow field reconstruction with sensor placement policy learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c. URL https://openreview.net/forum?id=1esFEjMUBS.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d pdes, 2023a. URL https://arxiv.org/abs/2309.00583.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations, 2023b. URL https://arxiv.org/abs/2111.03794.

B. List, L.-W. Chen, and N. Thuerey. Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *Journal of Fluid Mechanics*, 949:A25, 2022.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=AM7iAh0krx.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DKSI3bULiZ.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6 (10):1256–1269, September 2024. ISSN 2522-5839. doi: 10.1038/s42256-024-00897-5. URL http://dx.doi.org/10.1038/s42256-024-00897-5.

Yaxin Mo and Luca Magri. Reconstructing unsteady flows from sparse, noisy measurements with a physics-constrained convolutional neural network, 2024. URL https://arxiv.org/abs/2409.00260.

Roussel Desmond Nzoyem, David A.W. Barton, and Tom Deakin. Neural context flows for meta-learning of dynamical systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8vzMLo8LDN.

Assad A. Oberai, Manish Malhotra, and Peter M. Pinsky. On the implementation of the dirichlet-to-neumann radiation condition for iterative solution of the helmholtz equation. *Applied Numerical Mathematics*, 27(4):443–464, 1998. doi: 10.1016/S0168-9274(98)00024-5.

Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang,

Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

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

Alfio Quarteroni and Alberto Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, 05 1999. ISBN 9780198501787. doi: 10.1093/oso/9780198501787.001.0001. URL https://doi.org/10.1093/oso/9780198501787.001.0001.

Hamed Rahimian and Sanjay Mehrotra. Frameworks and results in distributionally robust optimization. *Open Journal of Mathematical Optimization*, 3:1–85, July 2022. ISSN 2777-5860. doi: 10.5802/ojmo.15. URL http://dx.doi.org/10.5802/ojmo.15.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=j3oQF9coJd.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Nadim Saad, Gaurav Gupta, Shima Alizadeh, and Danielle C. Maddix. Guiding continuous operator learning through physics-based boundary constraints. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gfWNItGOES6.

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

J. Sappl, L. Seiler, M. Harders, and W. Rauch. Deep learning of preconditioners for conjugate gradient solvers in urban water related problems. *arXiv preprint*, 2019.

M. S. Sauter. *Degenerate Elliptic Operators with Boundary Conditions via Form Methods*. PhD thesis, The University of Auckland, 2013. URL http://hdl.handle.net/2292/21318.

Junhong Shen, Tanya Marwah, and Ameet Talwalkar. UPS: Efficiently building foundation models for PDE solving via cross-modal adaptation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=0r9mhjRv1E.

Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W. Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=zANxvzflMl.

Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multi-operator learning and extrapolation, 2025. URL https://arxiv.org/abs/2404.12355.

Z. Sun, Y. Yang, and S. Yoo. A neural PDE solver with temporal stencil modeling. *arXiv preprint*, 2023.

Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. Learning neural pde solvers with parameter-guided channel attention, 2023. URL https://arxiv.org/abs/2304.14118.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2024. URL https://arxiv.org/abs/2210.07182.

Y. Teng and A. Choromanska. Invertible autoencoder for domain adaptation. *Computation*, 7(2):20, 2019.

Artur Toshev, Gianluca Galletti, Fabian Fritz, Stefan Adami, and Nikolaus Adams. Lagrangebench: A lagrangian fluid mechanics benchmarking suite. *Advances in Neural Information Processing Systems*, 36, 2024a.

Artur P. Toshev, Jonas A. Erbesdobler, Nikolaus A. Adams, and Johannes Brandstetter. Neural sph: Improved neural modeling of lagrangian fluid dynamics, 2024b. URL https://arxiv.org/abs/2402.06275.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tmIiMPl4IPa.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Haixin Wang, LI Jiaxin, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. BENO: Boundary-embedded neural operators for elliptic PDEs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZZTkLDRmkg.

Haixin Wang, Ruoyan Li, Fred Xu, Fang Sun, Kaiqiao Han, Zijie Huang, Guancheng Wan, Ching Chang, Xiao Luo, Wei Wang, and Yizhou Sun. Fd-bench: A modular and fair benchmark for data-driven fluid simulation, 2025a. URL https://arxiv.org/abs/2505.20349.

Hong Wang, Haiyang Xin, Jie Wang, Xuanze Yang, Fei Zha, Huanshuo Dong, and Yan Jiang. Mixture-of-experts operator transformer for large-scale pde pre-training, 2025b. URL https://arxiv.org/abs/2510.25803.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional pdes with latent spectral models. In *International Conference on Machine Learning*, 2023.

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

Vikas Yadav, Mario Casel, and Abdulla Ghani. Rf-pinns: Reactive flow physics-informed neural networks for field reconstruction of laminar and turbulent flames using sparse data. *Journal of Computational Physics*, 524:113698, March 2025. Published: 1 March 2025.

Liu Yang and Stanley J. Osher. Pde generalization of in-context operator networks: A study on 1d scalar nonlinear conservation laws, 2024. URL https://arxiv.org/abs/2401.07364.

Liu Yang, Siting Liu, Tingwei Meng, and Stanley J. Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39), September 2023. ISSN 1091-6490. doi: 10.1073/pnas.2310142120. URL http://dx.doi.org/10.1073/pnas.2310142120.

Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. PDEformer: Towards a foundation model for one-dimensional partial differential equations. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024. URL https://openreview.net/forum?id=GLDMCwdhTK.

Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, and Patrick Gallinari. Leads: Learning dynamical systems that generalize across environments, 2022. URL https://arxiv.org/abs/2106.04546.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai.

Y. Zhong, K. Fukami, B. An, et al. Sparse sensor reconstruction of vortex-impinged airfoil wake with machine learning. *Theor. Comput. Fluid Dyn.*, 37:269–287, 2023. doi: 10.1007/s00162-023-00657-y.

Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: PDE-conditional transformers are universal PDE solvers, 2025. URL https://openreview.net/forum?id=f3xXPDCh8Q.

A Broader Impact

This work significantly advances the field of scientific machine learning by establishing a theoretically rigorous framework for neural PDE solvers. By demonstrating that efficient, parameter- and boundary-aware architectures can outperform numerical solvers, we promote a more sustainable computational paradigm that reduces the energy consumption and environmental footprint associated with training large-scale networks. Furthermore, our emphasis on well-posedness enhances the reliability of AI-driven simulations in critical engineering applications, fostering greater trust in data-driven methods.

We do not anticipate any negative social or ethical consequences resulting from this work.

B Limitations and Future Work

A primary limitation of this study lies in the scarcity of established benchmark datasets that incorporate diverse boundary conditions. Existing datasets in the domain predominantly focus on fixed or simplistic boundary scenarios, which restricts the comprehensive evaluation of generalized neural operators. While we have mitigated this issue by generating a diverse set of boundary variations to rigorously validate our approach, the lack of a standardized community benchmark remains a challenge. We hope that future work will address this gap by developing high-quality, unified datasets that systematically capture the complexity of varying boundary interactions in physical systems. Establishing such comprehensive resources will require a sustained, long-term community effort and cannot be easily resolved within the scope of our current work.

C Datasets

Heat Equation The heat equation models the diffusion of temperature in a domain $\Omega \subset \mathbb{R}^d$:

$$\begin{cases} \dfrac{\partial u(\boldsymbol{x},t)}{\partial t} = \alpha \nabla^2 u(\boldsymbol{x},t), & (\boldsymbol{x},t) \in \Omega \times (0,T], \\ \mathcal{B}[u] = 0 & \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \end{cases}$$

where $u(\boldsymbol{x},t)$ is the temperature field, $\boldsymbol{x} \in \mathbb{R}^2$ denotes the spatial position, and t is time. The coefficient $\alpha > 0$ is the thermal diffusivity, defined as $\alpha = \frac{k}{\rho c_p}$, where k is the thermal conductivity, ρ is the density, and c_p is the specific heat capacity. Here, the PDE parameter is the thermal diffusivity α.

To evaluate generalization across PDE parameters, we examined thermal diffusivity $\alpha \in [0.01, 1.0]$. For the training data, we defined a linearly spaced parameter grid $A_{\text{train}} = \{\alpha_i\}_{i=1}^{501}$ over this interval. At each $\alpha_i \in A_{\text{train}}$, we generated 10 distinct trajectories, with initial conditions sampled randomly from a Gaussian field under strictly periodic boundary conditions. For validation and testing, we evaluated the model's interpolation capabilities by sampling a denser grid $A_{\text{test}} = \{\tilde{\alpha}_j\}_{j=1}^{1001}$. For each $\tilde{\alpha}_j \in A_{\text{test}}$, we generated 2 trajectories.

Next, we assessed generalization across boundary conditions by fixing the thermal diffusivity and varying the constraints among Dirichlet, Neumann, and periodic types. For the Dirichlet case, we generated 100 setups with boundary values sampled uniformly from $[-10, 10]$. Similarly, for the Neumann condition, we created 100 setups with boundary fluxes sampled from the same range. For both types, we generated 10 trajectories per setup. Additionally, we generated 1,000 trajectories for the periodic boundary condition. The validation and test sets mirrored this structure but generated only 2 trajectories per specific boundary setup.

Finally, we conducted joint pre-training to handle simultaneous variations in parameters and boundary conditions. We utilized the same diffusivity range (0.01 to 1.0) with 501 evenly spaced training values. For each diffusivity value, we created 5 distinct boundary setups for each type (Dirichlet, Neumann, and periodic), where values and fluxes were randomly sampled. We generated 10 trajectories for each unique parameter-boundary combination using Gaussian-sampled initial conditions.

Advection Equation The 2D advection equation models the transport of a scalar quantity $u(x,t)$ (such as temperature, density, or concentration) within a flow field $\mathbf{v}(x,t) = (v_1, v_2)^\top$ over a spatial domain $\Omega \subset \mathbb{R}^2$:

$$\begin{cases} \dfrac{\partial u}{\partial t} + \boldsymbol{v}(\boldsymbol{x}) \cdot \nabla u = 0, & (\boldsymbol{x}, t) \in \Omega \times (0, T], \\ \mathcal{B}[u] = 0, & \\ u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \end{cases}$$

where $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ and t denotes time. The velocity $\boldsymbol{v}(\boldsymbol{x})$ is the PDE parameter.

To investigate generalization across PDE parameters, we focused on varying the velocity field $\boldsymbol{v}(\boldsymbol{x})$. We generated the training dataset by randomly sampling 500 distinct velocity fields from a Gaussian field. For each velocity configuration, we simulated 10 distinct trajectories with initial conditions also drawn from a random Gaussian field. All simulations in this phase utilized periodic boundary conditions. For validation and testing, we expanded the scope by sampling 1,000 new velocity fields. For each of these test cases, we generated 2 trajectories to assess the model's performance on unseen parameters.

Subsequently, we evaluated generalization across boundary conditions by fixing the velocity field $\boldsymbol{v}(\boldsymbol{x})$ and varying the boundary constraints among Dirichlet, Neumann, and periodic types. For the Dirichlet condition, we created 100 configurations with boundary values sampled uniformly from $[-10, 10]$. Similarly, for the Neumann condition, we established 100 setups with boundary fluxes sampled from the same range. In each of these bounded cases, we generated 10 trajectories. Additionally, we produced 1,000 trajectories for the periodic boundary condition. In all scenarios, initial conditions were sampled from a random Gaussian field. The validation and test datasets followed this structure, generating 2 trajectories for each specific boundary setup.

Finally, we undertook a joint pre-training approach to capture the coupled effects of varying parameters and boundary conditions. We sampled 500 velocity fields $\boldsymbol{v}(\boldsymbol{x})$ from a random Gaussian field for the training set. For every specific velocity field, we constructed 5 distinct boundary configurations for each type (Dirichlet, Neumann, and periodic), with random sampling applied to the Dirichlet values and Neumann fluxes. We then generated 10 trajectories for each unique combination of parameter and boundary settings. For validation and testing, we sampled 1,000 new velocity fields. Following the training protocol for boundary generation, we produced 2 trajectories per setup to evaluate the model's ability to generalize jointly across diverse physical parameters and boundary constraints.

Burgers' Equation The 2D viscous Burgers' equation describes the evolution of a velocity field $\boldsymbol{u} = (u_1, u_2)^\top$ in a domain $\Omega \subset \mathbb{R}^2$:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \nu \nabla^2 \boldsymbol{u}, & (\boldsymbol{x}, t) \in \Omega \times (0, T], \\ \mathcal{B}[\boldsymbol{u}] = 0, & \\ \boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \end{cases}$$

where $\boldsymbol{u}(\boldsymbol{x}, t) = (u_1(\boldsymbol{x}, t), u_2(\boldsymbol{x}, t))^\top$ is the velocity vector field, $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ is the spatial coordinate, and t is time. The coefficient $\nu > 0$ is the kinematic viscosity, controlling the strength of diffusive effects relative to nonlinear advection.

We examined the kinematic viscosity ν across distinct training and testing regimes. For the training data, we defined a linearly spaced parameter grid $V_{\text{train}} = \{\nu_i\}_{i=1}^{501}$ over the interval $[5 \times 10^{-4}, 5 \times 10^{-2}]$. At each $\nu_i \in V_{\text{train}}$, we generated 10 distinct trajectories of length $T = 100$ time steps. Initial conditions were sampled randomly from a Gaussian field, and all simulations strictly enforced periodic boundary conditions. For the validation and test datasets, we evaluated a shifted parameter space, defining a denser grid $V_{\text{test}} = \{\tilde{\nu}_j\}_{j=1}^{1001}$ over the interval $[0.01, 1.0]$. For each $\tilde{\nu}_j \in V_{\text{test}}$, we generated 2 trajectories of 100 time steps using the same initial and boundary condition protocols.

Incompressible Navier-Stokes Equation

The 2D incompressible Navier-Stokes equations govern the motion of a viscous, incompressible fluid within a domain $\Omega \subset \mathbb{R}^2$:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{f}, & (\boldsymbol{x},t)\in\Omega\times(0,T], \\ \nabla\cdot\boldsymbol{u} = 0, & (\boldsymbol{x},t)\in\Omega\times(0,T], \\ \mathcal{B}[\boldsymbol{u},p] = 0, & \\ \boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}_0(\boldsymbol{x}), & \boldsymbol{x}\in\Omega, \end{cases}$$

where $\boldsymbol{u}(\boldsymbol{x},t) = (u_1(\boldsymbol{x},t), u_2(\boldsymbol{x},t))^\top$ is the velocity field, $p(\boldsymbol{x},t)$ is the pressure field, $\nu = \frac{\rho u L}{Re} > 0$ is the kinematic viscosity, and $\boldsymbol{f}(\boldsymbol{x},t)$ represents an external body force such as gravity. ρ is the fluid density. u is the velocity of the fluid field. L is the characteristic length. Re is the Reynold number. The first equation expresses the conservation of momentum, while the second enforces the incompressibility condition $\nabla\cdot\boldsymbol{u} = 0$, ensuring volume preservation of the flow.

We examine the Reynolds number across a range of values from 100 to 5000, inclusive. To create our training data, we select 50 evenly spaced values within this range. For each specific kinematic viscosity, we generate 10 distinct trajectories, each with a length of 100 time steps. The initial conditions for these trajectories are sampled randomly from a Gaussian field. For the validation and test datasets, we again look at kinematic viscosity ranging from 100 to 5000. However, we sample this space more densely, selecting 197 evenly spaced values. For each of these values, we generate 2 trajectories, each with a length of 100 time steps, using initial conditions sampled from a random Gaussian field. All simulations use periodic boundary conditions.

D Justifications on Benchmark Dataset Selection

Our benchmark datasets include a variety of PDEs. For PDE parameters, we consider wide variations. For boundary conditions, we consider both local and nonlocal boundary conditions, including Dirichlet, Neumann, and periodic, which are commonly studied boundary conditions in existing works (Wang et al., 2024; Saad et al., 2023; Wang et al., 2025a; Takamoto et al., 2024; Ohana et al., 2024).

Existing benchmarks and datasets (Wang et al., 2025a; Takamoto et al., 2024; Ohana et al., 2024) generally lack diverse boundary variations. Thus, we believe that our datasets constitute a valuable contribution and represent a significant step toward building generalized solvers. We suggest that future work focus on providing more comprehensive datasets regarding boundary variations.

E Model Implementation Details

ViT-k We adopt the architecture design from Dosovitskiy et al. (2021), utilizing a latent size of 256 and a patch size of 16. The model comprises 8 attention blocks, each configured with 8 attention heads and an MLP ratio of 4.0. We concatenate the k initial conditions through the channel dimension.

Concat The PDE parameters are spatially broadcast and concatenated with the initial condition to form the input for a ViT-based architecture (Dosovitskiy et al., 2021). We employ a patch size of 16 and a latent dimension of 256. The model consists of 8 attention blocks, each featuring 8 attention heads and an MLP ratio of 4.0.

CAPE+Unet We combine the CAPE (Takamoto et al., 2023) with Unet (Ronneberger et al., 2015). The CAPE module is configured with a widening factor of 64, a kernel size of 5, and a normalized dimension of 128. Additionally, if_11cnv is enabled, and the number of parameter embedding channels is set to 3. For the U-Net architecture, we utilize 32 initial features. The CAPE module accepts a single initial condition frame along with the PDE parameter, expanding the channel dimension to generate latent initial conditions. This representation is then fed into the U-Net to predict the subsequent frame.

CAPE+FNO We combine the CAPE (Takamoto et al., 2023) with FNO (Li et al., 2021). The CAPE module is configured with a widening factor of 64, a kernel size of 5, and a normalized dimension of 128. Additionally, if_11cnv is enabled, and the number of parameter embedding channels is set to 3. For FNO, we use modes 12 and a width of 20. The CAPE module accepts a single initial condition frame along with the PDE parameter, expanding the channel dimension to generate latent initial conditions. This representation is then fed into the FNO to predict the subsequent frame.

CAPE+Attn We combine the CAPE (Takamoto et al., 2023) with ViT (Dosovitskiy et al., 2021). The CAPE module is configured with a widening factor of 64, a kernel size of 5, and a normalized dimension of 128. Additionally, if_11cnv is enabled, and the number of parameter embedding channels is set to 3. For ViT, we use a latent dimension of 256, a patch size of 16, and 8 attention blocks featuring 8 heads and an MLP ratio of 4.0. The CAPE module accepts a single initial condition frame along with the PDE parameter, expanding the channel dimension to generate latent initial conditions. This representation is then fed into the FNO to predict the subsequent frame.

Unisolver We adopt the architecture design from Zhou et al. (2025), utilizing a latent size of 256 and a patch size of 16. The model comprises 8 attention blocks, each configured with 8 attention heads and an MLP ratio of 4.0.

MoE-POT We adopt the architecture proposed by Wang et al. (2025b). The model is configured with a latent dimension of 256, a patch size of 16, and 8 attention blocks featuring 8 heads and an MLP ratio of 4.0. Additionally, each block incorporates 4 experts to choose from. We use the top 2 experts and a shared common expert.

Ours Our model is configured with a latent dimension of 256, a patch size of 16, and 8 attention blocks featuring 8 heads and an MLP ratio of 4.0. Each attention block contains 4 kernels (we use 3 kernels for the incompressible Navier Stokes). The gating function is a lightweight MLP with hidden size 32.

F WHY MIXTURE OF KERNEL?

Error Trends vs. PDE Parameters We plot the three strong baselines, *CAPE+FNO*, *CAPE+Attn*, and *UniSolver*, showing normalized MSE (nMSE) as a function of the PDE parameter on the Incompressible Navier–Stokes dataset. To extract actionable insights without the overhead of full-scale training, we evaluate on a reduced subset of the incompressible Navier-Stokes dataset. The results reveal that nMSE varies markedly across parameters. All models degrade at low Reynolds numbers, while CAPE+FNO and CAPE+Attn exhibit an approximately exponential decay in error as the Reynolds number increases, reaching extremely small errors in the high-Re regime. The variability is most pronounced for CAPE+FNO, where nMSE is roughly 0.4 at low Re but becomes nearly negligible at high Re. These patterns suggest that the single kernel employed in the CAPE framework effectively captures PDE dynamics only within certain parameter regimes and can fail substantially outside those regimes. In contrast to the CAPE variants, UniSolver exhibits a noticeably smoother nMSE curve across PDE parameters, with no large fluctuations. The trend is still roughly parabolic, however, indicating parameter-dependent variation. This supports the same conclusion as before. A single kernel can capture PDE dynamics well only within certain parameter regimes. The results are reported in Figure 3.

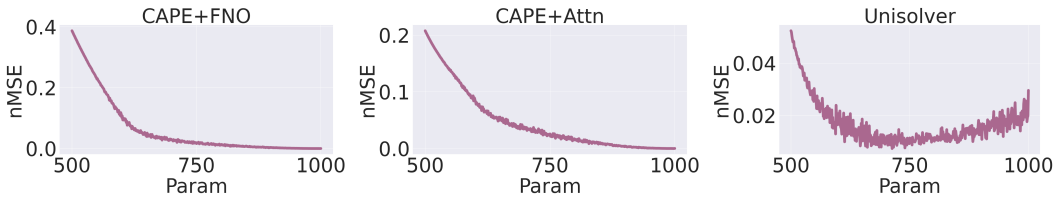


Figure 3: Error Trends vs. PDE Parameters on Incompressible Navier–Stokes dataset for *CAPE+FNO*, *CAPE+Attn*, and *UniSolver*.

G ADDITIONAL BASELINES FOR PDE PARAMETERS GENERALIZATION

FiLM Encoder: We apply FiLM (Perez et al., 2017) to the encoding stage of the model. $Z = \text{PatchEmbed}(x)$ and $\tilde{Z} = \gamma(\boldsymbol{\theta}) \odot Z + \beta(\boldsymbol{\theta})$, where $\gamma(\boldsymbol{\theta}) = W_\gamma \boldsymbol{\theta} + b_\gamma$ and $\beta(\boldsymbol{\theta}) = W_\beta \boldsymbol{\theta} + b_\beta$. Here W_γ, b_γ, W_β, and b_β are learnable model parameters. $\boldsymbol{\theta}$ is the PDE parameter and Z represent the latent embedding of initial condition.

FiLM Attn: We apply FiLM (Perez et al., 2017) before every multi-head attention layer. $\tilde{Z} = \gamma(\boldsymbol{\theta}) \odot Z + \beta(\boldsymbol{\theta})$ and $\tilde{Z} = \text{MHA}\Big(\text{Query} = \tilde{Z},\ \text{Key} = \tilde{Z},\ \text{Value} = \tilde{Z}\Big)$.

Pos Encoder: We apply parameter-guided positional encoding to the encoding stage of the model. $\tilde{Z} = Z + \text{PosEmbed}(\boldsymbol{\theta})$.

Pos Attn: We apply parameter-guided positional encoding before every multi-head attention layer. $\tilde{Z} = Z + \text{PosEmbed}(\boldsymbol{\theta})$ and $\tilde{Z} = \text{MHA}\Big(\text{Query} = \tilde{Z},\ \text{Key} = \tilde{Z},\ \text{Value} = \tilde{Z}\Big)$

Proj Encoder: We concatenate the input with the parameter and feed the result into an MLP in the encoding stage of the model. $\tilde{Z} = W[Z \mid\mid \boldsymbol{\theta}_{\text{expand}}] + b$. Here, W and b are learnable model parameters.

Proj Attn: We concatenate the input with the parameter and feed the result into an MLP before every multi-head attention layer. $\tilde{Z} = W[Z \mid\mid \boldsymbol{\theta}_{\text{expand}}] + b$ and $\tilde{Z} = \text{MHA}\Big(\text{Query} = \tilde{Z},\ \text{Key} = \tilde{Z},\ \text{Value} = \tilde{Z}\Big)$.

Method	Inc. NS
FiLM Encoder	8.90×10^{-1}
FiLM Attn	9.94×10^{-1}
Pos Encoder	9.26×10^{-1}
Pos Attn	8.43×10^{-1}
Proj Encoder	7.73×10^{-1}
Proj Attn	8.27×10^{-1}
Ours	$\mathbf{1.06 \times 10^{-1}}$

Table 5: Comparison of our proposed method against several additional baselines on conditional generation.

H Additional Results on Model Stability

We present the results in Figure 4.

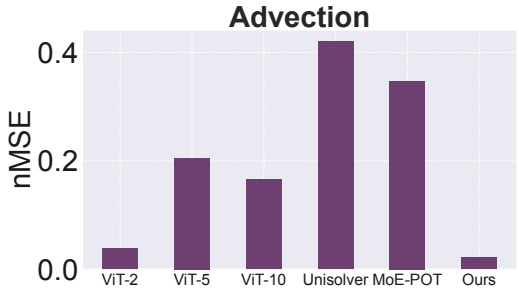


Figure 4: Standard deviation of model performance on the Advection equation in generalization across both PDE parameters and boundary conditions.

I Training Curriculum

We present the training curriculum in Algorithm 1.

Algorithm 1 Training Curriculum with Group DRO

Require: Dataset $\mathcal{D}$ partitioned by boundary types and parameter groups $\mathbf{\Theta}'$; Total epochs E; Phase 1 limit P.

for $e \leftarrow 1$ **to** E **do**
  for each boundary type **do**
    $\mathcal{D}' \leftarrow \{(\boldsymbol{u}_0, \boldsymbol{u}, \boldsymbol{\theta}, \mathcal{B}) \in \mathcal{D} \mid \text{boundary type}\}$
    if $e \leq P$ **then**
      ▷ **Phase 1: Warm-up (Standard MSE)**
      $\mathcal{L}_\phi \leftarrow \mathbb{E}_{(\boldsymbol{u}_0, \boldsymbol{u}, \boldsymbol{\theta}, \mathcal{B}) \sim \mathcal{D}'} \left[\ell(\mathcal{M}_\phi(\boldsymbol{u}_0, \boldsymbol{\theta}, \mathcal{B}), \boldsymbol{u}) \right]$
    else
      ▷ **Phase 2: Robust Training (Group DRO)**
      $L_{\boldsymbol{\theta}} \leftarrow \mathbb{E}_{(\boldsymbol{u}_0, \boldsymbol{u}, \mathcal{B}) \sim \mathcal{D}'_{\boldsymbol{\theta}}} \left[\ell(\mathcal{M}_\phi(\boldsymbol{u}_0, \boldsymbol{\theta}, \mathcal{B}), \boldsymbol{u}) \right] \; \forall \boldsymbol{\theta}$
      $\mathcal{L}_\phi \leftarrow \frac{1}{\tau} \log \left(\frac{1}{|\mathbf{\Theta}'|} \sum_{\boldsymbol{\theta} \in \mathbf{\Theta}'} \exp\left(\tau \, L_{\boldsymbol{\theta}}\right) \right)$
    end if
    Update $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}_\phi$
  end for
end for

J Ablation Study and Model Analysis

J.1 Full Ablation Study

Table 6 presents the results of our ablation study. The top row, "Full Model," incorporates all proposed components: the parameter-gated mixture of kernels, the generalized boundary transfer operator, and the robust optimization objective. Removing any of these modules consistently hurts performance, validating their individual contributions. While the model retains PDE parameters even without the mixture of kernels, removing the generalized boundary transfer operator completely deprives the network of boundary information. This specific removal causes the most severe drop in performance, strongly supporting our argument that explicitly incorporating physical inputs is highly beneficial.

Model	Advection
Full Model	$\mathbf{7.97 \times 10^{-2}}$
- Mixture of Kernels	2.15×10^{-1}
- Generalized Boundary Transfer Operator	3.36×10^{-1}
- Robust Optimization Objective	9.22×10^{-2}

Table 6: Ablation study on the Advection equation with simultaneous variations in PDE parameters and boundary conditions.

J.2 Analysis on Number of Kernels

We analyze the impact of kernel count on model performance, with results detailed in Table 7. Our findings confirm that increasing the number of kernels improves accuracy, validating the intuition that specialized kernels are necessary to handle distinct physical regimes arising from varied PDE parameters. Notably, the most significant performance gain occurs when increasing the number of kernels from 1 to 2. It is worth noting that since this mixture of kernels is present in every attention block, the increase in capacity scales with the depth of the model.

J.3 Analysis on Robust Optimization Objective Hyperparameters

We investigate the sensitivity of Algorithm 1 to the switching epoch P (the point at which the training objective transitions from standard MSE to robust optimization). Results are detailed in Tables 8. We find that while the model requires an initial MSE warm-up to prevent divergence, switching to the robust objective too late (e.g., epoch 90) degrades performance by limiting the time available for

# Kernels	Inc. NS
1	1.12×10^{0}
2	4.85×10^{-1}
3	3.30×10^{-1}
4	1.06×10^{-1}
5	1.03×10^{-1}

Table 7: Comparison of model performance with various kernel numbers.

robust optimization. Consequently, we select $P = 70$ to balance training stability with sufficient optimization time.

P	Inc. NS
0	Diverge
30	1.81×10^{-1}
50	1.76×10^{-1}
70	1.06×10^{-1}
90	1.34×10^{-1}

Table 8: Comparison of model performance on hyperparameter P on the incompressible Navier-Stokes dataset.

J.4 Analysis on Kernel Gating

In Table 9, we present the results for models where the kernel is gated by PDE parameters only, boundary conditions only, and both combined. The version gated exclusively by PDE parameters achieves the best performance, while the one gated solely by boundary conditions performs the worst. This aligns with our intuition that PDE parameters govern the intrinsic dynamics of the system and, therefore, should dictate the integration kernel.

Method	nMSE
combined	1.02×10^{-1}
boundary only	1.28×10^{-1}
parameters only	5.59×10^{-2}

Table 9: Analysis of gating components. We report the normalized MSE for different gating configurations.

K Additional Experiment on 3D dataset

Solving 3D PDEs with neural solvers is particularly difficult because the transition from 2D to 3D triggers a cubic explosion in data complexity. Beyond managing raw scale, the model must also capture intricate, non-local spatial interactions. To validate our model's performance in the 3D domain, we generated a 3D advection equation dataset on a 64^3 grid, incorporating variations in PDE parameters and boundary conditions. As reported in Table 10, purely data-driven methods, such as the ViT variants and MoE-POT, struggle significantly on this dataset. Specifically, the ViT variants require more initial condition frames to sufficiently infer the underlying physics. In contrast, our model achieves superior performance, outperforming baselines both with and without explicit physical inputs.

Method	Advection
ViT-2	1.10×10^{0}
ViT-5	4.22×10^{-1}
ViT-10	4.16×10^{-1}
Unisolver	1.67×10^{0}
MoE-POT	8.48×10^{-1}
Ours	$\mathbf{3.37 \times 10^{-1}}$

Table 10: Performance comparison on 3D Advection equation with both variations in PDE parameters and boundary conditions. We report the normalized MSE.

L Additional Experiment on Out-of-Distribution Generalization

We evaluate our model's out-of-distribution (OOD) generalization using the advection dataset, introducing variations in both PDE parameters and boundary conditions. To test parameter shifts, we first calculate the Frobenius norm of the velocity fields in the training set. We then sample testing velocity fields with norms strictly outside the training range, specifically, values either smaller than the training minimum or larger than the training maximum.

For boundary conditions, the Dirichlet values and Neumann fluxes are sampled from [-10, 10] during training. In our OOD evaluation, these values are instead drawn from either the [-15, -10] or the [10, 15] intervals. This ensures that the model is tested on magnitudes entirely unseen during the training phase.

The results, summarized in Table 11, demonstrate that our proposed model achieves optimal performance, highlighting its superior generalization capabilities compared to existing baselines.

While the velocity fields for the original testing and validation data in Table 4 are also unseen during training (as they are randomly sampled from a Gaussian field), their Frobenius norms may still fall within the range observed during training.

Method	Advection
ViT-2	4.48×10^{-1}
ViT-5	2.78×10^{-1}
ViT-10	3.45×10^{-1}
Unisolver	5.35×10^{-1}
MoE-POT	4.24×10^{-1}
Ours	$\mathbf{8.30 \times 10^{-2}}$

Table 11: Out-of-distribution comparison on the Advection equation with both variations in PDE parameters and boundary conditions. We report the normalized MSE.

M Additional Comparison with Physics Informed Neural Operator (PINO)

The Physics-Informed Neural Operator (PINO) (Li et al., 2023b) combines data-driven neural operators with physics-informed neural networks, leveraging both empirical data and symbolic equations via loss functions during training. However, PINO inherits a key limitation of traditional PINNs. It is tied to specific PDE parameters and boundary conditions. Because it does not explicitly receive physical parameters as inputs during inference, it struggles to generalize to new scenarios without retraining. To evaluate this, we trained PINO on the advection equation, using variations in both PDE parameters and boundary conditions purely to guide the training loss. As shown in Table 12, our proposed model consistently outperforms PINO.

Method	Heat	Advection
PINO	2.98×10^{0}	3.53×10^{-1}
Ours	$\mathbf{3.62 \times 10^{-1}}$	$\mathbf{7.97 \times 10^{-2}}$

Table 12: Performance comparison between PINO and our proposed model. We report the normalized MSE on the Heat and Advection equations with both PDE parameter and boundary condition variations.

N Additional Experiment on Complex PDE

Current benchmark datasets, such as those by Ohana et al. (2024); Takamoto et al. (2024); Toshev et al. (2024a), typically lack simultaneous variations in both PDE parameters and boundary conditions for a single PDE type. Consequently, the primary datasets used in this study were self-generated. To further evaluate our model's robustness, we also employ the 2D Turbulent Radiative Layer dataset from Ohana et al. (2024). This simulation features a cold, dense gas layer beneath a hot, dilute gas layer, moving at highly subsonic relative velocities. The setup utilizes periodic boundary conditions in the x-direction and zero-gradient Neumann conditions in the y-direction. The governing PDE parameter, the cooling time, is varied across the set $\{0.03, 0.06, 0.1, 0.18, 0.32, 0.56, 1.00, 1.78, 3.16\}$. We model the density, pressure, and x, y velocity fields. This validates our model on a public, computationally challenging PDE. We note that the boundary conditions remain fixed while only the PDE parameters vary.

We report the MSE in Table 13. Our model achieves a considerably lower MSE compared to the baselines. Notably, purely data-driven ViT models perform poorly in this configuration. We hypothesize that because this PDE involves multiple coupled fields, ViT-based architectures struggle to capture the underlying physics from concatenated data. This highlights a significant limitation of purely data-driven approaches and underscores the superiority of our framework.

Method	Advection
ViT-2	543.35
ViT-5	314.46
ViT-10	281.18
Unisolver	35.43
MoE-POT	24.82
Ours	**11.99**

Table 13: Performance Comparison on 2D Turbulent Radiative Layer dataset. This is a public dataset containing complex PDE with variations only in PDE parameters. We report the MSE over 10 steps of rollout.

O Additional Analysis on Boundary Condition

Extension to Other Boundary Conditions: Our method handles new boundary conditions only if they are included in the training data. This follows directly from our design philosophy. We explicitly condition on boundary conditions rather than implicitly inferring them from initial frames. While implicit approaches may appear to generalize zero-shot, they do so by solving a hidden inverse problem. This practice is inherently ill-posed, introduces irreducible epistemic error, and is causally impossible for time-dependent boundaries. Furthermore, implicit methods typically rely on monolithic networks, incurring an inference cost that undermines the primary speed advantage of neural operators over traditional numerical solvers. We view the requirement for representative training data as the necessary cost of a well-posed formulation.

Boundary nMSE: We present the boundary nMSE results in Table 14. Our model achieves the best performance across all three boundary conditions, validating the superiority of our design choices.

We wish to clarify that, by design, our model does not strictly satisfy boundary conditions (i.e., with zero nMSE error). However, we argue this is not a critical weakness, as exact boundary satisfaction remains a highly non-trivial challenge in the field. Existing foundational models rely on inference from initial conditions and inherently cannot guarantee exact boundary compliance. Physics-Informed Neural Networks (PINNs) typically enforce boundary conditions softly via penalty terms in the loss function, precluding absolute zero error. Similarly, Boundary-embedded Neural Operators (BENO) (Wang et al., 2024), despite focusing exclusively on elliptic PDEs, investigate boundary encoding but also fail to achieve exact satisfaction.

To the best of our knowledge, the Boundary-enforcing Operator Network (BOON) (Saad et al., 2023) is the only method capable of zero boundary error, yet this capability is strictly limited to Dirichlet conditions. BOON achieves this by making boundary-specific modifications to the operator's kernel. Consequently, their method is restricted to Dirichlet, Neumann, and periodic conditions, and this hardcoded kernel modification fundamentally prevents generalization across varying boundary types. Furthermore, even within BOON, Neumann and periodic conditions do not achieve zero error: because Neumann prescribes a derivative and periodic prescribes an inter-boundary relation, neither specifies an absolute value, causing both to inherit whatever errors exist in the model's interior predictions. Therefore, we believe the exceptionally low boundary nMSE achieved by our model represents a substantial success and a strong step forward.

Model	Periodic	Dirichlet	Neumann
ViT-2	4.69×10^{-1}	7.43×10^{0}	3.85×10^{-1}
ViT-5	3.65×10^{-1}	5.25×10^{0}	2.50×10^{-1}
ViT-10	3.86×10^{-1}	1.93×10^{0}	2.37×10^{-1}
Unisolver	5.51×10^{-1}	5.41×10^{0}	2.69×10^{-1}
Moe-PoT	1.84×10^{-1}	3.80×10^{0}	1.80×10^{-1}
Ours	$\mathbf{1.15 \times 10^{-1}}$	$\mathbf{4.92 \times 10^{-2}}$	$\mathbf{1.02 \times 10^{-1}}$

Table 14: Error across boundary conditions. We report the normalized MSE under periodic, dirichlet, and neumann boundary conditions.

Boundary Residual: In Table 15, we present the boundary residuals. Our model consistently achieves lower residuals, which validates the effectiveness of our proposed boundary transfer operator. We would like to clarify that, as previously discussed, achieving a strictly zero boundary residual is a non-trivial task. While the Boundary-enforcing Operator Network (BOON) (Saad et al., 2023) can accomplish this, it does so by making boundary-specific modifications to the operator's kernel, which inherently prevents generalization to novel boundary conditions. Thus, we believe our model's low boundary residual, combined with its low boundary nMSE, clearly demonstrates the efficacy of our proposed framework.

Model	periodic	dirichlet	neumann
ViT-2	2.60×10^{-1}	1.63×10^{0}	1.78×10^{-1}
ViT-5	1.83×10^{-1}	1.42×10^{0}	1.26×10^{-1}
ViT-10	2.14×10^{-1}	1.56×10^{0}	1.56×10^{-1}
Unisolver	2.23×10^{-1}	3.78×10^{0}	1.56×10^{-1}
MoE-POT	1.35×10^{-1}	5.04×10^{0}	1.14×10^{-1}
Ours	$\mathbf{1.09 \times 10^{-1}}$	$\mathbf{2.36 \times 10^{-1}}$	$\mathbf{1.01 \times 10^{-1}}$

Table 15: Boundary residual across boundary conditions. We report the boundary residual under periodic, dirichlet, and neumann boundary conditions.

P Model Parameters and Training Time

We present the model parameters and training times in Table 16. Because MoE-POT and our proposed model utilize a mixture-of-kernels architecture, they inherently require more total parameters. However, the number of active parameters per sample in our model remains comparable to the single-kernel ViT and Unisolver baselines. In contrast, MoE-POT activates multiple kernels per sample

alongside an always active common kernel, resulting in a higher total parameter count, substantially more active parameters per forward pass, and longer training times.

Furthermore, we emphasize that inference speed is the most critical practical constraint when evaluating neural solvers. As demonstrated in our paper, despite having a larger total parameter capacity, our model achieves a vastly superior accuracy-efficiency tradeoff because it selectively activates only one kernel at a time. Naively scaling the single-kernel baselines to match our total parameter count, whether by increasing depth or widening latent dimensions, would heavily burden their serial computation paths. This would drastically slow their inference speeds, rendering such a comparison fundamentally unfair.

Model	Parameters (M)	Training Time (s)
ViT	7.18	313.33
Unisolver	6.61	370.76
MoE-POT	31.82	1027.61
Ours	28.26	675.83

Table 16: Model size and training cost. We report the number of parameters (in millions) and per-epoch training time (in seconds).

Q Proof of Proposition 4.1

We begin by outlining and justifying our assumptions.

Assumption Q.1 (Existence)**.** For every initial condition $\boldsymbol{u}_0 \in \mathcal{X}_{\text{init}}$, a solution $\boldsymbol{u} \in \mathcal{Y}$ to the problem defined in Equation 1 **exists**.

Justification. This assumption is implicit in standard operator learning frameworks (Kovachki et al., 2023), which model the solution operator as a continuous map. The formulation of such a map inherently presumes that for every input initial condition, a realizable output exists. □

Assumption Q.2 (Uniqueness)**.** The solution to the problem defined in Equation 1 is locally **unique**. Specifically, for a fixed initial condition $\boldsymbol{u}_0$, there exists exactly one solution trajectory $\boldsymbol{u}$. This ensures that $\mathcal{S}_{\boldsymbol{\theta},\mathcal{B}}$ is a single-valued mapping rather than a one-to-many relation.

Justification. Uniqueness is a necessary condition for standard, deterministic neural operator architectures (Kovachki et al., 2023), which are designed to approximate single-valued functions. If the PDE admits non-unique solutions, they will fail to converge to a valid physical solution, often learning an unphysical average of the possible modes. Modeling one-to-many mappings requires probabilistic frameworks, such as diffusion-based neural operators (Huang et al., 2024; Li et al., 2025b;a; Baldan et al., 2025). □

Assumption Q.3 (Non-Degeneracy of PDE Configuration)**.** The family of PDE problems defined by the configuration space $\mathcal{C} = \Theta \times \mathbb{B}$ is **non-degenerate**. We assume that the PDE parameters and boundary conditions exert a non-trivial influence on the system dynamics. If an initial condition $\boldsymbol{u}_0$ sufficiently excites the system, then there exist at least two distinct configurations $c_1, c_2 \in \mathcal{C}$ whose corresponding solution trajectories are not identical:

$$\exists c_1, c_2 \in \mathcal{C}, \quad c_1 \neq c_2 \quad \text{such that} \quad \mathcal{S}_{c_1}[\boldsymbol{u}_0] \neq \mathcal{S}_{c_2}[\boldsymbol{u}_0]. \tag{3}$$

Justification. This assumption reflects the principle of model parsimony and physical relevance. It postulates that every component of the parameter vector $\boldsymbol{\theta}$ and every constraint in the boundary operator $\mathcal{B}$ exerts a measurable influence on the system dynamics. If a specific parameter or boundary condition could be altered without affecting the solution trajectory $\boldsymbol{u}$, that element would be physically non-descriptive and effectively irrelevant to the process. We therefore assume the problem is formulated such that no such redundant setup exists. □

We proceed by contradiction. Suppose the generalized neural operator $\mathcal{M}_\phi$ is well-defined as a function dependent only on the initial condition $\boldsymbol{u}_0$, and that it correctly represents the family $\mathfrak{S}$.

This hypothesis implies that for any valid PDE configuration $c \in \mathcal{C}$ and any initial condition $\boldsymbol{u}_0$, the operator satisfies:

$$\mathcal{M}_\phi[\boldsymbol{u}_0] = \mathcal{S}_c[\boldsymbol{u}_0]. \tag{4}$$

Consider the fixed initial condition $\boldsymbol{u}_0$ satisfying Assumption Q.3. By the assumption, there exist distinct configurations $c_1, c_2 \in \mathcal{C}$ such that $\mathcal{S}_{c_1}(\boldsymbol{u}_0) \neq \mathcal{S}_{c_2}(\boldsymbol{u}_0)$.

However, if we apply our hypothesized operator $\mathcal{M}_\phi$ to the initial condition $\boldsymbol{u}_0$, it must yield the correct solution for both configurations:

$$\mathcal{M}_\phi[\boldsymbol{u}_0] = \mathcal{S}_{c_1}[\boldsymbol{u}_0] \tag{5}$$

$$\mathcal{M}_\phi[\boldsymbol{u}_0] = \mathcal{S}_{c_2}[\boldsymbol{u}_0] \tag{6}$$

By definition, an operator is single-valued; it maps the input $\boldsymbol{u}_0$ to a unique output $\boldsymbol{y}^* \in \mathcal{Y}$. Substituting this into the equations above, we obtain:

$$\mathcal{S}_{c_1}[\boldsymbol{u}_0] = \boldsymbol{y}^* = \mathcal{S}_{c_2}[\boldsymbol{u}_0]. \tag{7}$$

This implies $\mathcal{S}_{c_1}[\boldsymbol{u}_0] = \mathcal{S}_{c_2}[\boldsymbol{u}_0]$, which directly contradicts Assumption Q.3. Thus, $\mathcal{M}_\phi$ cannot be solely a function of $\boldsymbol{u}_0$.

Remark: For inherently non-unique PDEs, a probabilistic framework, such as a diffusion-based neural operator, would be necessary to model the full solution distribution rather than a point estimate. In such non-unique cases, omitting explicit parameters and boundary conditions would force the model to marginalize over both physical configurations and solution branches, thereby conflating two distinct sources of ambiguity. Therefore, explicit conditioning on PDE parameters and boundary conditions still remains strictly necessary.

R HARDWARE SPECIFICATION

We implement all models in PyTorch (Paszke et al., 2019). All experiments are run on servers/workstations with the following configuration:

- 80 CPUs, 503G Mem, 8 x NVIDIA V100 GPUs.
- 48 CPUs, 220G Mem, 8 x NVIDIA TITAN XP GPUs.
- 96 CPUs, 1.0T Mem, 8 x NVIDIA A100 GPUs.
- 64 CPUs, 1.0T Mem, 8 x NVIDIA RTX A6000 GPUs.
- 224 CPUs, 1.5T Mem, 8 x NVIDIA L40S GPUs.
- 128 CPUs, 480G Mem, 8 × NVIDIA RTX 4090 GPUs.

S ALGORITHM PSEUDO CODE

```
INPUT:  x         (B, C, nx, ny)      # field snapshot (channels-first
    )
        param     (B, d)              # PDE parameter
        bc_type, bc_info, data_name   # boundary condition spec

------------------------------------------------------------

GNO.forward:
    x <- Encode(x)                          # -> (B, N, D)

    for blk in blocks (depth layers):
        x <- MoKBlock(x, param, bc_type, bc_info, data_name)

    x <- Decode(x)                          # -> (B, C, nx, ny)

    return x

------------------------------------------------------------

MoKBlock.forward(x, param, ...):            # x: (B, N, D)

    # 1. Route to one kernel per sample
    logits     <- Gate(param)               # MLP: (B, num_kernels)
    kernel_idx <- argmax(logits, dim=-1) # (B,)

    out <- zeros_like(x)                    # (B, N, D)

    # 2. Prepend conditioning tokens
    x <- BCTransfer(x, ...)                 # prepend BC token    -> (B,
    N+M, D)
    x <- ParamEmbed(x, param)               # prepend param token -> (B,
    N+M+1, D)
    x <- x + pos_embed                      # (1, N+M+1, D)

    # 3. Dispatch each sample to its assigned kernel.
    for e in 0 .. num_kernels-1:
        mask <- (kernel_idx == e)
        if any(mask):
            y_sub <- MLP(x[mask]) + SharedBlock( x[mask], attn =
    Kernel_e )
            out[mask] <- y_sub[:, M+1:, :]   # drop the prepended
    tokens

    return out                              # (B, N, D)

------------------------------------------------------------

SharedBlock.forward(x, attn):               # x: (B, L, D); attn = routed
     kernel
    x <- attn(x)
    return x

------------------------------------------------------------

BCTransfer.forward(x, bc_type, bc_info, data_name):
    edge_emb <- BoundaryEncoder(bc_info) # (B, M, D)
    cat_emb  <- Embedding(bc_type)       # (B, 1, D)  periodic/
    dirichlet/neumann

    seq <- concat([cat_emb, edge_emb, x])# (B, 1 + M + N, D)
    seq <- SelfAttention(seq, seq, seq)
```

```
    bc  <- seq[:, 1:M+1, :]

    return concat([bc, x])                # (B, N+M, D)

-----------------------------------------------------------------

ParamEmbed.forward(x, mu):
    token <- MLP(mu)                      # (B, 1, D)
    return concat([token, x])             # (B, N+M+1, D)
```