TIME-AWARE OPERATOR LEARNING WITH MULTI-FIDELITY SOLVER-EMBEDDED MIXTURE OF EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

In numerically solving PDEs, achieving both generalization and efficiency is crucial but challenging. While recent advancements in neural operators provide significant speed-ups, they often struggle to generalize to out-of-distribution scenarios. Hybrid models that integrate neural networks with conventional numerical solvers offer improved generalization ability, but they incur high computational costs. To address this efficiency-generalization trade-off, we propose the Multi-Fidelity Mixture-of-Experts (MF-MoE) framework. MF-MoE dynamically allocates resources based on physical parameters and an optimal routing solver across expert models of varying fidelity, optimizing both computational cost and predictive accuracy. This innovative design enables faster inference for in-distribution inputs while ensuring better generalization for out-of-distribution cases. Extensive experiments on fluid flow prediction governed by the incompressible Navier-Stokes equations demonstrate that MF-MoE consistently outperforms baseline approaches, offering an efficient solution for PDE surrogate modeling.

1 INTRODUCTION

Computational fluid dynamics governed by nonlinear partial differential equations (PDEs) are ubiquitous in scientific and engineering applications, including the earth system modeling (Palmer & Stevens, 2019; Gelbrecht et al., 2023), fluid flow prediction (Belbute-Peres et al., 2020; Ma et al., 2024), gas leak detection (Lee et al., 2024), urban water clarification (Li & Shatarah, 2024; Putra et al., 2024), and blood flow modeling (Schwarz et al., 2023; Csala et al., 2024). Accurately resolving these PDEs at high spatial and temporal resolutions often remains computationally prohibitive, as conventional numerical methods demand immense computational resources and long runtime.

Recently, neural operator methods have emerged as a promising alternative, offering data-driven PDE approximations that can exploit GPU acceleration for much faster training and inference (Li et al., 2020a; Wen et al., 2022; Janny et al., 2023; Azizzadenesheli et al., 2024; N et al., 2024; Xiong et al., 2024; Raonic et al., 2023; Liang et al., 2024; Wu et al., 2024; Li et al., 2023). Despite these advantages, a pure neural network commonly struggles when confronted with out-of-distribution inputs, as demonstrated by Belbute-Peres et al. (2020) and Sun et al. (2023). This limitation hampers their robustness and restricts their applicability in real-world scenarios.

To address these shortcomings, hybrid approaches that integrate classical PDE solvers into neural architectures have been proposed (Mishra, 2018; Um et al., 2020; Belbute-Peres et al., 2020; Zhuang et al., 2021; Dresdner et al., 2022; Sun et al., 2023; Wang et al., 2024; Cao et al., 2024; Liu et al., 2024). Instead of directly learning a full-resolution PDE solution, these solver-embedded strategies employ a neural network to refine an inaccurate solution produced by a lower-resolution solver. For instance, Belbute-Peres et al. (2020) introduced a graph-based model that corrects the up-scaled coarse-mesh outputs from computational fluid dynamics simulations, reducing the out-of-distribution prediction error of a pure graph neural network model in the airfoil air flow prediction task.

However, while hybrid approaches improve the generalization ability compared to pure neural network models, they inevitably slow down training and inference due to the integration of numerical

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.

PDE solver[1]. As the result, one can rely on solver-based refinement for improved accuracy on out-of-distribution data, but must then accept the associated computational overhead. This conflict motivates the following natural question:

> ***Q:*** *Can we design a model architecture that enjoys fast inference time for in-distribution samples while maintaining high generalization ability for out-of-distribution samples?*

We present an affirmative answer to this question by introducing the **Multi-Fidelity Mixture of Experts (MF-MoE)**, a mixture-of-experts framework that seamlessly integrates multiple solver-based hybrid models of varying fidelity with a pure neural operator.

Our contributions are summarized as follows:

- The MF-MoE architecture dynamically balances computational efficiency and prediction accuracy via an instance-wise routing strategy. Under tight time constraints, this architecture adaptively shifts probability mass toward fast-inference neural operators, minimizing prediction error while strictly adhering to the computational budget.
- We formulate the routing mechanism as a constrained convex optimization problem. A key advantage of this formulation is that it allows the model to adapt to arbitrary inference time constraints without re-training parameters, offering flexible deployment in time-critical scenarios.
- Extensive experiments on fluid flow prediction governed by the Navier-Stokes equations demonstrate that MF-MoE consistently achieves superior performance while maintaining acceptable inference time, validating the effectiveness and efficiency of the proposed MF-MoE framework.

2 BACKGROUNDS

Problem Setting In this work, we focus on the computational fluid dynamic problems characterized by the following form of *incompressible Navier-Stokes equations* (Temam, 1977):

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla \mathbf{p} + \mu\Delta\mathbf{u} + \mathbf{f}, \tag{1}$$

with $\nabla \cdot \mathbf{u} = 0$, where ∇ is the vector differential operator, $\Delta := \nabla \cdot \nabla$ is the Laplace operator, $\mathbf{u}$ is the velocity field with the no-slip Dirichlet condition, $\mathbf{p}$ is the pressure, ρ is the fluid density, μ is the constant viscosity, and $\mathbf{f}$ is an external force field. The incompressibility constraint is given by $\nabla \cdot \mathbf{u} = 0$. We consider a one-step flow prediction problem: given the physical fields from the previous K time steps, $[\mathbf{u}_1, \ldots, \mathbf{u}_K]$, the objective is to predict the physical fields $\mathbf{u}_{K+1}$ at the next time step. This problem is widely studied in closed-box fluid dynamics, and many works adopt this setting (Gupta & Brandstetter, 2022; Ruhe et al., 2023; Lippe et al., 2023; Brandstetter et al., 2022).

Numerical Solvers Solver-based hybrid models typically depend on external PDE solvers. In our work, we employ the PhiFlow framework (Holl & Thuerey, 2024), which has also been widely used in existing work (Um et al., 2020; Gupta & Brandstetter, 2022; Brandstetter et al., 2022; Schnell et al., 2022; Holl et al., 2022; Ruhe et al., 2023). This numerical solver employs the explicit forward Euler method to approximate the time evolution of the PDEs by discretizing the time derivative with a finite difference scheme. In general, if $\mathbf{u}^n$ denotes the velocity field at time t_n, then the forward Euler update from t_n to $t_{n+1} = t_n + \Delta t$ is given by

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t F\left(\mathbf{u}^n\right),$$

where $F(\mathbf{u}^n)$ represents the spatially discretized terms of the Navier-Stokes equations, including advection, diffusion, and external forcing. To enforce the incompressibility condition ($\nabla \cdot \mathbf{u}^{n+1} = 0$), a pressure projection step is performed. The method's accuracy is influenced by the time step size Δt and the spatial resolution (n_x, n_y). Larger Δt can introduce larger numerical errors or even instability due to the Courant-Friedrichs-Lewy (CFL) condition (Courant et al., 1967), while coarser grids reduce spatial accuracy.

[1] In some specific architectures (e.g., the Frozen Mesh Mode of the CFD-GCN model by Belbute-Peres et al. (2020)), the PDE solution over the training sample can be separately cached. In such cases, training time is not significantly affected, but the inference cost remains unavoidable, which is also our main focus.

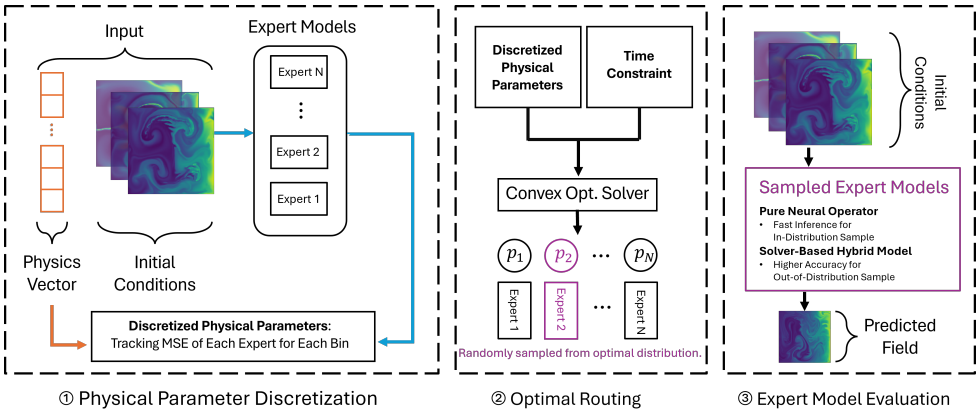

Figure 1: Overview of the MF-MoE framework. (1) Physical parameters, such as the viscosity μ in Equation (1), are discretized into bins. A look-up table is maintained to track the MSE of each expert within each bin. (2) The optimal gating strategy is determined via a convex programming problem designed to minimize MSE while satisfying expected computational cost constraints. (3) The selected expert (either a lightweight neural operator or a more expensive solver-based hybrid model) processes the input fields to generate the final predicted field.

3 The MoE Framework with Multi-Fidelity PDE Solvers

In this section, we present the structure of our proposed MF-MoE framework; its inference step is illustrated in Figure 1. The MF-MoE consists of two primary components: *(i)* ***Dynamic Gating Mechanism****:* This gating mechanism learns to determine whether using a physical solver is necessary for a given input. If a solver is required, the gating network also selects the appropriate fidelity level for the hybrid models with considering the specific time constraint. *(ii)* ***Multi-Fidelity Experts****:* In candidate expert models, we consider one pure neural operator such as Fourier Neural Operators (FNOs) (Li et al., 2020a), or UNO (Ma et al., 2021; Chen & Thuerey, 2021; Rahman et al., 2022), and multiple solver-based hybrid models with varying fidelity levels. The pure neural operator offers significantly faster inference speeds but may have limited generalization capabilities. In contrast, the hybrid models enhance generalization by incorporating solver-based refinements but incur higher time costs. We will discuss these two components and the training process in the following subsections.

3.1 Dynamic Gating Mechanism

The gating network determines the most appropriate expert for a given input field. Unlike conventional mixture-of-experts models in computational physics, whose routing processes focus solely on initial physical fields (Sharma & Shankar, 2024; Liang et al., 2024; Hao et al., 2023), our gating network incorporates two additional factors: *(i)* the inherent physical parameters governing the PDE problem (e.g., dynamic viscosity or buoyancy force), and *(ii)* the computational time constraint required to complete the task. To incorporate this information into the routing strategy, we proceed as follows:

1. *Physical parameter discretization.* During the training phase, we discretize the physical parameter space into multiple intervals. We then track the training loss and computational cost of each expert associated with each parameter interval. For more details of training this part, we refer to Section 3.3.
2. *Optimal routing strategy.* During the inference phase, for any input sample falling within a specific slot, we retrieve the historical error profile $[E_1, E_2, \ldots, E_N]$ and time cost $[C_1, C_2, \ldots, C_N]$ for the N experts. We then solve the following linear programming problem:

$$\min_{g} \quad \mathbb{E}_{i\sim g}[E_i]$$
$$\text{s.t.} \quad \mathbb{E}_{i\sim g}[C_i] \leq c,$$

where g represents the probability distribution over the experts. For more details of this formulation, we refer to Equations (2) and (3).

Table 1: Configuration details for different resolution levels. Each configuration specifies the number of timesteps (`nt`), spatial resolution (`nx`, `ny`), time cost per frame, and MSE of the numerical solution compared to the ground truth (GT). The ground truth configuration achieves the highest accuracy with the highest computational cost, while coarser resolutions progressively reduce time cost at the expense of higher error. The visualization of an example data pair is provided in Appendix D to better illustrate the difference among different resolution levels.

Resolution Level	`nt`	`(nx, ny)`	**Time Cost** (s)	**MSE Error**
Groundtruth (GT)	512	(256, 256)	1.58×10^2	< 0.001
Fine	64	(128, 128)	0.38×10^2	0.065
Medium	32	(64, 64)	0.20×10^2	0.104
Coarse	16	(32, 32)	0.12×10^2	0.135
XCoarse	8	(16,16)	0.08×10^2	0.151

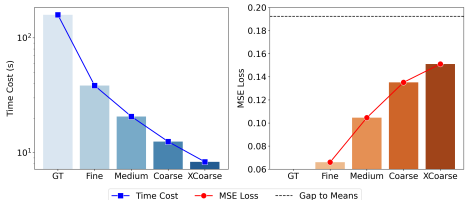

Figure 2: Computational cost (in seconds) versus spatial resolution in the numerical solver. Finer grids increase accuracy but require significantly more computational time. Details on the resolution configuration are provided in Appendix A.2. In the right panel, the black line labeled as "Gaps to Means", indicates the minimum MSE loss of using a constant field as the prediction.

This key mechanism ensures that the expected loss is minimized while satisfying the time constraint in expectation.

3.2 Multi-Fidelity Expert Models

Our MF-MoE system integrates two distinct types of experts:

1. *Pure Neural Operator*. This expert performs inference with no calls to a PDE solver. In our paper, we mainly use FNOs (Li et al., 2020a) and UNO (Ronneberger et al., 2015) as the base model; we also note that our framework is compatible with other state-of-the-art neural operators. These base models are selected as they are commonly used as the baselines in existing literature and excel in fast inference and high accuracy over in-distribution samples.

2. *Multi-Fidelity Solver-Based Hybrid Models*. This type of experts fuses a numerical PDE solver with a refinement neural network. In our paper, we use the hybrid model illustrated by Figure 7. Specifically, a numerical solver generates a low-resolution approximation $\hat{y}_c$, which is then processed by an up-sampling step to recover fine-scale details. The neural operators are used to learn the correction Δy. A convex combination of $\hat{y}_C$ and Δy is used as the final prediction.

Depending on resource availability, multiple solver-based models of varying fidelity (e.g., extremely coarse vs. moderately coarse mesh) can be included in the MoE to offer finer control over the accuracy-speed trade-off. The gating mechanism thus selects lower or higher fidelity solver-based experts as needed.

Up-Sampling After obtaining the coarse output $\hat{y}_c \in \mathbb{R}^{n_{w,c} \times n_{h,c}}$, we generate the high-resolution prediction $\hat{y}_f \in \mathbb{R}^{n_{w,f} \times n_{h,f}}$ by applying the k-nearest neighbor (k-NN) interpolation for each time channel (Dasarathy, 1991; Shakhnarovich et al., 2006; Belbute-Peres et al., 2020). For each scaled pixel location (i_f, j_f) in $\hat{y}_f$, we identify the k nearest neighbors $\{(i_c^{(n)}, j_c^{(n)})\}_{n=1}^k$ of (i_f, j_f) in $\hat{y}_c$ based on the given metric d. Then the high-resolution pixel value is computed as a weighted average of these neighbors:

$$\hat{y}_f(i_f, j_f) = \frac{1}{W} \sum_{n=1}^{k} w_n \cdot \hat{y}_c(i_c^{(n)}, j_c^{(n)}),$$

where $w_n = \frac{1}{d\left((i_f, j_f), (i_c^{(n)}, j_c^{(n)})\right)}$ and $W = \sum_{n=1}^k w_n$. In this paper, we always set $k = 4$. Appendix D further visualizes the performance of this up-sampling procedure on an example field.

3.3 TRAINING THE MF-MOE FRAMEWORK

In this section, we describe the training methodology for our proposed MF-MoE framework. Unlike most existing MoE approaches that focus on improving the prediction accuracy of neural operators or other hybrid models, our primary goal is to control the time cost introduced by incorporating an external numerical solver during inference while maintaining the prediction accuracy.

We formulate this trade-off as a constrained optimization problem:

$$\min_{\theta} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{\xi\sim \mathsf{G}_\phi(x)}\left\|\mathsf{E}_{\theta_\xi}(x)-y\right\|^2\right] \tag{2}$$

$$\text{subject to} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{\xi\sim \mathsf{G}_\phi(x)}\mathcal{T}\big(\mathsf{E}_\xi\big)\right] \le c, \tag{3}$$

where $\theta = \{\theta_1, \ldots, \theta_N\}$ denotes the trainable parameters of the N experts, and ϕ represents the parameters of the gating mechanism (specifically, the stored historical statistics). Here, (x, y) is a sample from the data distribution $\mathcal{D}$. The gating mechanism $\mathsf{G}_\phi(x)$ produces a probability distribution over the experts, E_{θ_ξ} is the expert selected by index ξ, and $\mathcal{T}(\cdot)$ measures the inference time.

To solve this effectively, we decouple the training process into two separate routines: optimizing all experts to minimize the prediction errors, and updating the gating parameters (historical statistics) to maintain the updated performance of each expert.

3.3.1 TRAINING OF EXPERT PARAMETERS

To ensure the gating mechanism has a robust selection of experts to choose from, we train the learnable experts (e.g., neural operators) to minimize their individual prediction errors. We minimize the MSE averaged across all experts. The loss function is defined as:

$$\mathcal{L}_{\text{experts}}(\theta) = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}_{\text{train}}}\left\|\mathsf{E}_{\theta_i}(x)-y\right\|^2, \tag{4}$$

where $\mathcal{N}$ is the set of indices for all experts. By minimizing this objective via gradient-based algorithms (more specifically, we deploy the standard AdamW optimizer to update these parameters), we ensure that every expert attempts to learn the optimal prediction that minimizes the empirical loss over the given dataset.

3.3.2 TRAINING OF GATING PARAMETERS (STATISTICAL UPDATE)

As introduced in Section 3.1, the parameters of our gating mechanism, denoted as ϕ, are not weights of a neural network, but rather the structured historical profiles of error and time cost associated with discretized physical parameters. The training of the gate, therefore, corresponds to the iterative update of these statistics during the learning process. We discretize the domain of the relevant physical parameter λ (e.g., viscosity) into K distinct slots (or bins), denoted as $\{S_1, \ldots, S_K\}$. The gating parameters consist of two matrices: the Error Profile $\mathbf{E} \in \mathbb{R}^{K\times N}$ and the Cost Profile $\mathbf{C} \in \mathbb{R}^{K\times N}$.

For an input batch with physical parameter $\lambda \in S_k$, we update the statistics for the k-th slot using the naive running average. The update rules at iteration t are:

$$\mathbf{E}_{k,i}^{(t)} = \frac{1}{k}\cdot\mathbf{E}_{k,i}^{(t-1)} + \frac{k-1}{k}\cdot\left\|\mathsf{E}_{\theta_i}(x)-y\right\|^2, \quad \mathbf{C}_{k,i}^{(t)} = \frac{1}{k}\cdot\mathbf{C}_{k,i}^{(t-1)} + \frac{k-1}{k}\cdot\tau_i(x), \tag{5}$$

where $i \in \{1, \ldots, N\}$ iterates over all experts, $\tau_i(x)$ is the measured wall-clock time for expert i.

These updated matrices $\mathbf{E}$ and $\mathbf{C}$ are then utilized directly in the Linear Programming solver during inference (as detailed in Section 3.1) to determine the optimal routing distribution g that satisfies the global time constraint c. In practice, we may also pre-calculate the solver's runtime at different resolutions in advance and maintain it as a look-up table.

4 EXPERIMENTS

4.1 FLUID FLOW DYNAMIC PREDICTION

In this section, we consider solving the fluid flow dynamic prediction problem given by the incompressible Navier-Stokes equation as defined in Equation (1). Given the input fields $[\mathbf{u}_t, \mathbf{u}_{t+1}, \mathbf{u}_{t+2}, \mathbf{u}_{t+3}]$,

Table 2: Performance comparison on the fluid-flow prediction task described in Equation (1) across various neural operator architectures (including the FNO and UNO) and our proposed MF-MoE framework. The table reports the number of activated parameters, MSE metrics, and time constraint violation. The MF-MoE approach consistently achieves superior accuracy compared to the baseline models while significantly reducing solver-inference time compared to the numerical solvers.

Ref.	Model[1]	Act. # Params	Train MSE	Valid MSE	Test MSE	< 10 secs?[2]
FNO (Li et al., 2020a)	FNO128-8$_{\text{modes16}}$	134 M	7.53×10^{-2}	15.9×10^{-2}	16.6×10^{-2}	✓
	FNO128-4$_{\text{modes16}}$	67.2 M	9.17×10^{-2}	15.3×10^{-2}	16.1×10^{-2}	
	FNO96-4$_{\text{modes32}}$	151 M	9.93×10^{-2}	17.2×10^{-2}	16.7×10^{-2}	
	FNO64-4$_{\text{modes32}}$	67.1 M	14.0×10^{-2}	20.1×10^{-2}	20.3×10^{-2}	
	MF-MoE (FNO128-8$_{\text{modes16}}$)	134 M	1.15×10^{-2}	17.2×10^{-2}	11.6×10^{-2}	✓
	MF-MoE (FNO128-4$_{\text{modes16}}$)	67.2 M	2.00×10^{-2}	14.2×10^{-2}	9.7×10^{-2}	
	MF-MoE (FNO96-4$_{\text{modes32}}$)	151 M	3.33×10^{-2}	14.0×10^{-2}	9.9×10^{-2}	
	MF-MoE (FNO64-4$_{\text{modes32}}$)	67.1 M	5.73×10^{-2}	14.9×10^{-2}	10.4×10^{-2}	
UNO[3] (Ma et al., 2021) (Chen & Thuerey, 2021) (Rahman et al., 2022)	UNO-128	440 M	0.26×10^{-2}	15.4×10^{-2}	17.0×10^{-2}	✓
	UNO-64	110 M	0.43×10^{-2}	17.8×10^{-2}	17.4×10^{-2}	
	U-F3Net$_{\text{modes16,8,4}}$	187 M	0.59×10^{-2}	11.8×10^{-2}	12.7×10^{-2}	
	U-F2Net$_{\text{modes16,8}}$	175 M	0.61×10^{-2}	12.8×10^{-2}	12.3×10^{-2}	
	U-F1Net$_{\text{modes16}}$	160 M	0.68×10^{-2}	11.3×10^{-2}	12.3×10^{-2}	
	MF-MoE (UNO-128)	440 M	0.18×10^{-2}	13.7×10^{-2}	10.9×10^{-2}	✓
	MF-MoE (UNO-64)	110 M	0.27×10^{-2}	15.1×10^{-2}	11.6×10^{-2}	
	MF-MoE (U-F3Net)$_{\text{modes16,8,4}}$	187 M	0.18×10^{-2}	15.2×10^{-2}	9.5×10^{-2}	
	MF-MoE (U-F2Net)$_{\text{modes16,8}}$	175 M	0.19×10^{-2}	14.5×10^{-2}	9.3×10^{-2}	
	MF-MoE (U-F1Net)$_{\text{modes16}}$	160 M	0.17×10^{-2}	17.4×10^{-2}	9.8×10^{-2}	
Numerical Solver (PhiFlow)	XCoarse (16×16)	-	-	-	15.1×10^{-2}	✓
	Coarse (32×32)	-	-	-	13.5×10^{-2}	✗
	Medium (64×64)	-	-	-	10.4×10^{-2}	✗
	Fine (128×128)	-	-	-	6.50×10^{-2}	✗
	Groundtruth (256×256)	-	-	-	$< 0.1 \times 10^{-2}$	✗

[1] In both the FNO and UNO model, the first number indicate the number of hidden layers, which is used to control the size of the model. In the FNO model, the subscript 64-4$_{\text{modes32}}$ indicates that it uses 4 blocks which retains 32 modes and have 64 channels in its hidden layers. In the MF-MoE model, the bracket name indicates the choice of the base model, which is used as the pure neural operator expert in the MoE structure and the residual approximation model used in the hybrid structure (illustrated in Figure 7).
[2] We consider the averaged time cost over the test set and report if this averaged time violates the time constraint 10.0 seconds.
[3] We also consider the U-Nets with Fourier blocks belongs to a general category of U-shaped Neural Operator (UNO). Here the subscript represents the modes of Fourier blocks included in the architecture.

the objective is to predict the next time-step field $\mathbf{u}_{t+4}$. For this task, we adopt two foundational neural operator architectures: FNOs (Li et al., 2020a) and UNOs (Ma et al., 2021; Chen & Thuerey, 2021; Rahman et al., 2022). These models are classical in the field of neural operators and are widely used as baselines in existing literature (Gupta & Brandstetter, 2022; Brandstetter et al., 2022; Ruhe et al., 2023). While numerous state-of-the-art models have been developed in recent years, most of them are based on these two fundamental models. Therefore, we focus on these models for their broad representativeness. Additional discussions of other advancing models are included in Appendix E.1.

Datasets We create a customized dataset by splitting the train set, validation set, and test set via the conditional parameters of viscosity μ. The split is described in Equation (6). The detailed dataset configuration and a sample trajectory visualization of each resolution setting are put in Appendix A.2 and Appendix D.

$$\begin{aligned} \text{(Train)} \quad & \mu \in \{3.2, 1.6, 0.8\} \times 10^{-2}, \\ \text{(Valid)} \quad & \mu \in \{1.6, 0.8, 0.4, 0.2\} \times 10^{-2}, \\ \text{(Test)} \quad & \mu \in \{0.8, 0.4, 0.2, 0.1\} \times 10^{-2}. \end{aligned} \tag{6}$$

Training We use the Mean Squared Error (MSE) loss on the predicted velocities as the evaluation metric. For the MF-MoE model, the number of experts $N = 4$, where one expert is a pure neural-operator model and the other three are solver-based hybrid models of varying fidelity (Fine, Medium, and Coarse); the time-cost constraint to $c = 10.0$, ensuring that a single solver-based hybrid model alone (though accurate) is excluded, as it exceeds this cost threshold. An additional discussion on the impact of the parameter c is included in Section 4.2. All models are trained using the AdamW optimizer (Loshchilov & Hutter, 2017; 2019) with a learning rate $\eta = 2 \times 10^{-4}$ for 8,000 training samples, a weight decay of 1×10^{-5}, and a batch size of 16. No further hyperparameter tuning is applied across different models.

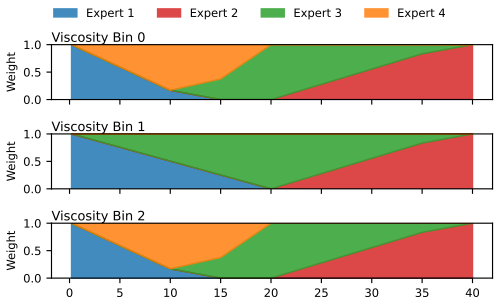

Figure 3: Distribution of expert selections under different time budget parameters c and viscosity bin (FNO128-4).

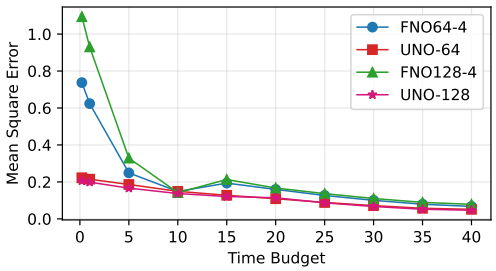

Figure 4: Impact of the Time Budget c on the Gating Mechanism.

Results In Table 2, we summarize the performance of our MF-MoE framework compared to baseline models. Here we present two primary observations: *(i)* The MF-MoE framework consistently outperforms neural operator baselines (FNO and UNO) across all evaluated scenarios. When using the same base architecture, MF-MoE achieves significant accuracy improvements over standalone neural operators. *(ii)* In contrast to numerical solvers, which fail to meet the time constraint at every fidelity level except the XCoarse resolution level, the MF-MoE model satisfies the required time constraint $c = 10.0$. This indicates that MF-MoE greatly enhances the efficiency of solver-based hybrid models. In traditional designs, the inference time cost of such hybrid models is dominated by the numerical solver; however, MF-MoE circumvents this bottleneck by dynamically choosing the desired numerical solver.

4.2 Impact of Time Budget on the Gating Mechanism

In this subsection, we examine the impact of the time budget c (referred to as the time constraint in Equation (3)) on the gating distribution during the inference phase. To analyze its impact, we vary the time budget c from 0 to 40, specifically investigating how it affects *(i)* the overall MSE, and *(ii)* the expert distribution.

4.2.1 Impact of Time Budget on the MSE

To evaluate the impact of the time budget on the MSE, we visualize the performance of MF-MoE across four different base models while varying the time budget c from 0 to 40.

Results In Figure 4, we observe a general trend where the MSE decreases as the time budget c increases. Specifically, the FNO-based models exhibit a sharp drop in error as the budget grows from 0 to 10, whereas the UNO models maintain a relatively stable low error rate even at tight constraints. At larger time budgets, the performance gap narrows, which is consistent with the intuition that relaxing constraints allows the gating mechanism to select the most accuracy hybrid model.

4.2.2 Impact of Time Budget on the Expert Distribution

To further analyze the behavior of the gating mechanism, we visualize the expert selection probabilities across different time budgets. Figure 3 illustrates the distribution of selected experts as c varies.

Results Figure 3 visualizes the evolution of expert selection probabilities across different viscosity bins. We observe: *(i)* As the time budget c increases from 0 to 40, the dominant expert shifts from the most efficient option (the pure neural operator, blue) to the most accurate one (the high-fidelity hybrid model, red). This validates our motivation for introducing a dynamic routing mechanism to adaptively select the optimal expert. *(ii)* For Viscosity Bins 0 and 2, the gating mechanism utilizes Expert 4 (orange) as a bridge between the low-cost and high-cost regimes. In contrast, for Viscosity Bin 1, the model bypasses Expert 4 entirely, transitioning directly from Expert 1 to Expert 3 (green)

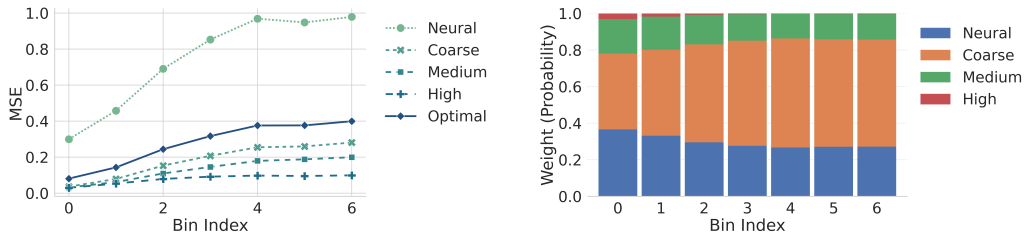

Figure 5: Impact of the buoyancy force on the gating mechanism in MF-MoE (FNO64-4_{modes32}). The left figure shows the overall MSE *vs.* bin indices. The right figure shows the distribution evolution over different buoyancy force parameters.

before converging to Expert 2. This discrepancy indicates that expert selection is driven not solely by the time budget, but also by the intrinsic complexity of the input data.

4.3 Extension to Buoyancy Force Parameters

To further evaluate the generalization capabilities of the MF-MoE framework, we extend our assessment beyond the viscosity parameter μ. As indicated in Equation (1), the external force field $\mathbf{f}$ is a governing term driving the fluid dynamics, which in this context is controlled by buoyancy. Consequently, we modified the data generation configuration to introduce varying buoyancy coefficients. We constructed a dataset split based on the buoyancy magnitude along the y-axis to create a distinct out-of-distribution scenario:

$$\text{(Train \& Valid)} \quad \mathbf{f} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}, \qquad \text{(Test)} \quad \mathbf{f} \in \{0.05, 0.55\}. \tag{7}$$

This setup tests the model's ability to extrapolate to highly perturbed flows that exhibit more chaotic turbulence than those in the training set. We compared the MF-MoE (utilizing FNO as the base) against a standalone FNO model. The results are summarized in Table 3.

Results As expected,the standalone neural operator exhibits limited generalization, yielding a high MSE range of 5.34–8.44×10^{-2}. In contrast, the MF-MoE consistently achieves an MSE of $\approx 3 \times 10^{-2}$, effectively halving the error of the baseline model. These results corroborate our earlier findings regarding viscosity and further validate the robustness of the MF-MoE architecture across diverse physical parameters.

Impact of Physical Parameters on Gating We further examine the influence of physical parameters on the gating distribution during the inference phase[2]. To investigate this, we vary the buoyancy force from 0.0 to 0.6, specifically analyzing its effect on *(i)* the overall MSE, and *(ii)* the expert distribution. The results indicate that when the buoyancy force is small, the problem difficulty is low; consequently, the pure neural operator is sufficient to achieve the desired accuracy. Therefore, as shown in Figure 5 (right), the neural operator has a higher probability of being selected in these regimes. It demonstrates the adaptability of the gating mechanism to the underlying physical complexity, confirming that our model effectively allocates computational resources based on the problem's hardness.

4.4 Extension to Multi-Step Prediction

To further evaluate the generalization capabilities of the MF-MoE framework, we extend our assessment beyond single-step prediction to the multi-step setting. In this setting, the model is trained and evaluated to directly predict the subsequent N states $\{y_{t+1}, y_{t+2}, \ldots, y_{t+N}\}$ autoregressively. We follow the same setting as Section 4.1 and evaluate the test set performance for $N = 4$ horizon length.

Results Test results are reported in Figure 6; all results are reported in Appendix A.7. As expected, both models exhibit performance degradation for multi-step prediction. However, the MF-MoE exhibits stable and consistent better performance compared to the standalone baseline. These results further validate the robustness of the MF-MoE architecture across diverse generalization scenarios.

[2]In this experiment, we adopt a modified experimental setup. See Appendix A.5 for details.

Table 3: Quantitative comparison of generalization performance on varying buoyancy parameters. We report the MSE on each dataset, comparing the proposed MF-MoE against the standalone FNO baseline. The full table along with the accuracy of numerical solver under different fidelity is presented in Appendix A.6.

Model	**Test MSE**	< 10 **secs?**
FNO128-8_{modes16}	6.90×10^{-2}	
FNO128-4_{modes16}	5.34×10^{-2}	✓
FNO96-4_{modes32}	5.74×10^{-2}	
FNO64-4_{modes32}	8.44×10^{-2}	
MF-MoE (FNO128-8_{modes16})	3.55×10^{-2}	
MF-MoE (FNO128-4_{modes16})	2.95×10^{-2}	✓
MF-MoE (FNO96-4_{modes32})	3.04×10^{-2}	
MF-MoE (FNO64-4_{modes32})	3.37×10^{-2}	

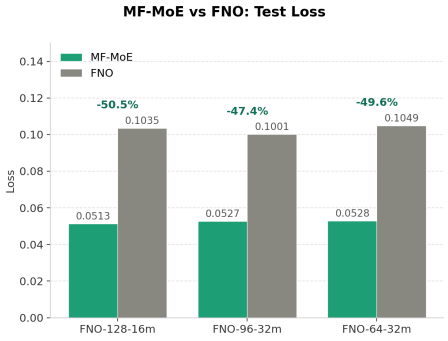

Figure 6: Visual comparison of generalization performance on multi-step prediction task. We consider the prediction horizon $N = 4$ and prediction the next-step field autoregressively. We report the averaged MSE over the test set, comparing the proposed MF-MoE against the standalone FNO baseline.

4.5 Extension to Other PDE Equations

To assess whether the MF-MoE framework transfers to PDE families beyond the incompressible Navier-Stokes setting, we further evaluate it on the viscous Burgers equation:

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \mu \, \Delta \mathbf{u}, \tag{8}$$

with $\mathbf{u} = (u_x, u_y)$ subject to boundary conditions. Compared to Equation (1), the Burgers system usually develops sharper gradients as the viscosity decreases, making it a complementary stress test for the gating mechanism. We adopt the same multi-fidelity protocol as in Section 4.1: four solver resolutions are generated to instantiate one pure neural-operator expert and three solver-based hybrid experts of varying fidelity, with an FNO as the shared base architecture. Full dataset specifications, measured expert costs, and training hyperparameters are reported in Appendix B.

Results Quantitative results are reported in Appendix B. Consistent with the observations on Navier-Stokes, MF-MoE achieves substantially lower MSE than the standalone FNO baseline while satisfying the prescribed time budget, despite reusing the hyperparameters from Section 4.1 without further tuning. This further validates that the multi-fidelity gating mechanism is not specific to a single PDE family.

5 Conclusion

In this work, we introduced the **Multi-Fidelity Mixture-of-Experts (MF-MoE)** framework, a novel approach to operator learning that seamlessly integrates pure neural operators with multi-fidelity solver-based hybrid models. Leveraging a dynamic gating mechanism to adaptively route inputs based on physical parameters and inference time budgets, the framework achieves an optimal balance between computational efficiency and predictive accuracy by formulating the routing as a constrained optimization problem. It enables fast inference for in-distribution data while ensuring robust generalization ability on out-of-distribution data through solver-based refinement. This was validated through the fluid flow prediction experiment governed by the incompressible Navier-Stokes equations, where the MF-MoE framework consistently outperformed its base model counterparts while strictly adhering to the time cost constraint. We believe this work provides an efficient approach for combining numerical solvers and pure neural operators, hence opening a new avenue for the development of time-aware, solver-based hybrid models.

AI USE STATEMENT

In this work, we used generative AI tools to propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat datasets, support qualitative or thematic data analysis, and interpret results. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, or assist with translation. Providing critical ingredients for mathematical proofs and assisting with proof writing are not applicable to this work.

Additionally, we used OpenAI Codex (and other AI tools) to assist with identifying relevant literature from citation context, checking bibliographic metadata and links against source records, removing uncited entries, and correcting and formatting references in BibTeX. Codex also assisted with LaTeX template adjustment, compilation checks, and drafting this statement.

We have reviewed all AI-assisted work (including this statement). The authors manually reviewed the AI-assisted research tasks listed above and the manuscript-preparation changes. Bibliographic metadata and links were checked against source records, and the LaTeX changes were checked by compiling the manuscript. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, Tz. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021. doi: 10.1016/j.camwa.2020.06.009.

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024. doi: 10.1038/s42254-024-00712-5.

Leah Bar and Nir Sochen. Unsupervised deep learning algorithm for PDE-based forward and inverse problems. arXiv preprint arXiv:1904.05417, 2019. URL https://arxiv.org/abs/1904.05417.

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

Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2402–2411. PMLR, 2020. URL https://proceedings.mlr.press/v119/de-avila-belbute-peres20a.html.

Roberto Bentivoglio, Elvin Isufi, Sebastiaan Nicolas Jonkman, and Riccardo Taormina. Multi-scale hydraulic graph neural networks for flood modelling. *Natural Hazards and Earth System Sciences*, 25(1):335–351, 2025. doi: 10.5194/nhess-25-335-2025. URL https://nhess.copernicus.org/articles/25/335/2025/.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model reduction and neural networks for parametric PDEs. arXiv preprint arXiv:2005.03180, 2020. URL https://arxiv.org/abs/2005.03180.

Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K. Gupta. Clifford neural layers for PDE modeling. arXiv preprint arXiv:2209.04934, 2022. URL https://arxiv.org/abs/2209.04934.

Oscar P Bruno, Jan S Hesthaven, and Daniel V Leibovici. FC-based shock-dynamics solver with neural-network localized artificial-viscosity assignment. *Journal of Computational Physics: X*, 15:100110, 2022. doi: 10.1016/j.jcpx.2022.100110.

Lianghao Cao, Thomas O'Leary-Roseberry, Prashant K Jha, J Tinsley Oden, and Omar Ghattas. Residual-based error correction for neural operator accelerated infinite-dimensional bayesian inverse problems. *Journal of Computational Physics*, 486:112104, 2023. doi: 10.1016/j.jcp.2023.112104.

Shuhao Cao. Choose a transformer: Fourier or galerkin. arXiv preprint arXiv:2105.14995, 2021. URL https://arxiv.org/abs/2105.14995.

Shuhao Cao, Francesco Brarda, Ruipeng Li, and Yuanzhe Xi. Spectral-Refiner: Fine-tuning of accurate spatiotemporal neural operator for turbulent flows. arXiv preprint arXiv:2405.17211v1, 2024. URL https://arxiv.org/abs/2405.17211v1.

Li-Wei Chen and Nils Thuerey. Towards high-accuracy deep learning inference of compressible turbulent flows over aerofoils. arXiv preprint arXiv:2109.02183, 2021. URL https://arxiv.org/abs/2109.02183.

Xu Chen, Shuo Liu, and Xuan Di. Physics-informed graph neural operator for mean field games on graph: A scalable learning approach. *Games*, 15(2):12, 2024. doi: 10.3390/g15020012.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. doi: 10.1090/s0025-5718-1965-0178586-1.

R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967. doi: 10.1147/rd.112.0215. URL https://ieeexplore.ieee.org/document/5391985.

Hunor Csala, Arvind Mohan, Daniel Livescu, and Amirhossein Arzani. Physics-constrained coupled neural differential equations for one dimensional blood flow modeling. arXiv preprint arXiv:2411.05631, 2024. URL https://arxiv.org/abs/2411.05631.

Belur V. Dasarathy (ed.). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991. ISBN 0818659300. URL https://ci.nii.ac.jp/ncid/BA19940413.

Gideon Dresdner, Dmitrii Kochkov, Peter Norgaard, Leonardo Zepeda-Núñez, Jamie A. Smith, Michael P. Brenner, and Stephan Hoyer. Learning to correct spectral methods for simulating turbulent flows. arXiv preprint arXiv:2207.00556, 2022. URL https://arxiv.org/abs/2207.00556.

Weinan E and Bing Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018. doi: 10.1007/s40304-018-0127-z.

Thomas D Economon, Francisco Palacios, Sean R Copeland, Trent W Lukaczyk, and Juan J Alonso. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 2016. doi: 10.2514/1.j053813.

Hugo Frezat, Julien Le Sommer, Ronan Fablet, Guillaume Balarac, and Redouane Lguensat. A posteriori learning for quasi-geostrophic turbulence parametrization. *Journal of Advances in Modeling Earth Systems*, 14(11):e2022MS003124, 2022. doi: 10.1029/2022ms003124.

Sergei Garmaev, Siddhartha Mishra, and Olga Fink. NOMTO: Neural operator-based symbolic model approximaTion and discOvery. arXiv preprint arXiv:2501.08086v1, 2025. URL https://arxiv.org/abs/2501.08086v1.

Maximilian Gelbrecht, Alistair White, Sebastian Bathiany, and Niklas Boers. Differentiable programming for earth system modeling. *Geoscientific Model Development*, 16(11):3123–3135, 2023. doi: 10.5194/gmd-16-3123-2023.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24048–24062. Curran Associates, Inc., 2021. URL https://papers.neurips.cc/paper_files/paper/2021/hash/c9e5c2b59d98488fe1070e744041ea0e-Abstract.html.

Jayesh K. Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized PDE modeling. arXiv preprint arXiv:2209.15616, 2022. URL https://arxiv.org/abs/2209.15616.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A general neural operator transformer for operator learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12556–12569. PMLR, 2023. URL https://proceedings.mlr.press/v202/hao23c.html.

Philipp Holl and Nils Thuerey. Φ_{Flow}: Differentiable simulations for PyTorch, TensorFlow and Jax. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18515–18546. PMLR, 2024. URL https://proceedings.mlr.press/v235/holl24a.html.

Philipp Holl, Vladlen Koltun, and Nils Thuerey. Scale-invariant learning by physics inversion. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5390–5403. Curran Associates, Inc., 2022. doi: 10.52202/068431-0389. URL https://papers.neurips.cc/paper_files/paper/2022/hash/240cc9ac4789351653d13cfcba4ee85c-Abstract-Conference.html.

Steeven Janny, Aurélien Béneteau, Madiha Nadri, Julie Digne, Nicolas Thome, and Christian Wolf. EAGLE: Large-scale learning of turbulent fluid dynamics with mesh transformers. arXiv preprint arXiv:2302.10803, 2023. URL https://arxiv.org/abs/2302.10803.

Yuchi Jiang, Zhijie Li, Yunpeng Wang, Huiyu Yang, and Jianchun Wang. An implicit adaptive Fourier neural operator for long-term predictions of three-dimensional turbulence. arXiv preprint arXiv:2501.12740, 2025. URL https://arxiv.org/abs/2501.12740.

Sharmila Karumuri, Lori Graham-Brady, and Somdatta Goswami. Physics-informed latent neural operator for real-time predictions of complex physical systems. arXiv preprint arXiv:2501.08428v1, 2025. URL https://arxiv.org/abs/2501.08428v1.

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

Steve Lawrence, C. Lee Giles, and Ah Chung Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 540–545. AAAI Press, 1997. URL https://clgiles.ist.psu.edu/papers/AAAI-97.overfitting.hard_to_do.pdf.

Hunggi Lee, Donghyeon Lee, Jaewook Lee, and Dongil Shin. Efficient gas leak simulation surrogate modeling and super resolution for gas detector placement optimization. *Computers & Chemical Engineering*, 181:108508, 2024. doi: 10.1016/j.compchemeng.2023.108508.

Haochen Li and Mohamed Shatarah. Operator learning for urban water clarification hydrodynamics and particulate matter transport with physics-informed neural networks. *Water Research*, 251: 121123, 2024. doi: 10.1016/j.watres.2024.121123.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28010–28039. Curran Associates, Inc., 2023. doi: 10.52202/075280-1216. URL https://papers.neurips.cc/paper_files/paper/2023/hash/590daf74f99ee85df3d8c007df9c8187-Abstract-Conference.html.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895, 2020a. URL https://arxiv.org/abs/2010.08895.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. arXiv preprint arXiv:2006.09535, 2020b. URL https://arxiv.org/abs/2006.09535.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. arXiv preprint arXiv:2003.03485, 2020c. URL https://arxiv.org/abs/2003.03485.

Aoming Liang, Zhaoyang Mu, Pengxiao Lin, Cong Wang, Mingming Ge, Ling Shao, Dixia Fan, and Hao Tang. M^2M: Learning controllable multi of experts and multi-scale operators are the partial differential equations need. arXiv preprint arXiv:2410.11617, 2024. URL https://arxiv.org/abs/2410.11617.

Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. In *Advances in Neural Information Processing Systems*, volume 36, pp. 67398–67433. Curran Associates, Inc., 2023. doi: 10.52202/075280-2946. URL https://papers.neurips.cc/paper_files/paper/2023/hash/d529b943af3dba734f8a7d49efcb6d09-Abstract-Conference.html.

Björn List, Li-Wei Chen, and Nils Thuerey. Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *Journal of Fluid Mechanics*, 949: A25, 2022. doi: 10.1017/jfm.2022.738.

Xin-Yang Liu, Min Zhu, Lu Lu, Hao Sun, and Jian-Xun Wang. Multi-resolution partial differential equations preserved learning framework for spatiotemporal dynamics. *Communications Physics*, 7 (1):31, 2024. doi: 10.1038/s42005-024-01521-z.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in Adam. arXiv preprint arXiv:1711.05101v1, 2017. URL https://arxiv.org/abs/1711.05101v1.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximaTion theorem of operators. arXiv preprint arXiv:1910.03193, 2019. URL https://arxiv.org/abs/1910.03193.

Hao Ma, Yuxuan Zhang, Nils Thuerey, Xiangyu Hu, and Oskar J. Haidn. Physics-driven learning of the steady Navier-Stokes equations using deep convolutional neural networks. arXiv preprint arXiv:2106.09301, 2021. URL https://arxiv.org/abs/2106.09301.

Shaocong Ma, James Diffenderfer, Bhavya Kailkhura, and Yi Zhou. End-to-end mesh optimization of a hybrid deep learning black-box PDE solver. arXiv preprint arXiv:2404.11766, 2024. URL https://arxiv.org/abs/2404.11766.

MFEM. MFEM: Modular finite element methods [Software]. Software, n.d. URL https://mfem.org/. Accessed September 25, 2026.

Siddhartha Mishra. A machine learning framework for data driven acceleration of computations of differential equations. arXiv preprint arXiv:1807.09519, 2018. URL https://arxiv.org/abs/1807.09519.

Navaneeth N, Tapas Tripura, and Souvik Chakraborty. Physics informed WNO. *Computer Methods in Applied Mechanics and Engineering*, 418:116546, 2024. doi: 10.1016/j.cma.2023.116546.

Nicholas H. Nelsen and Andrew M. Stuart. The random feature model for input-output maps between Banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021. doi: 10.1137/20M133957X. URL https://arxiv.org/abs/2005.10224.

Sheel Nidhan, Haoliang Jiang, Lalit Ghule, Clancy Umphrey, Rishikesh Ranade, and Jay Pathak. A domain decomposition-based autoregressive deep learning model for unsteady and nonlinear partial differential equations. arXiv preprint arXiv:2408.14461, 2024. URL https://arxiv.org/abs/2408.14461.

Tim Palmer and Bjorn Stevens. The scientific challenge of understanding and estimating climate change. *Proceedings of the National Academy of Sciences*, 116(49):24390–24395, 2019. doi: 10.1073/pnas.1906691116.

Ravi G Patel, Nathaniel A Trask, Mitchell A Wood, and Eric C Cyr. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373:113500, 2021. doi: 10.1016/j.cma.2020.113500.

Raphaël Pestourie, Youssef Mroueh, Chris Rackauckas, Payel Das, and Steven G. Johnson. Physics-enhanced deep surrogates for PDEs. arXiv preprint arXiv:2111.05841v1, 2021. URL https://arxiv.org/abs/2111.05841v1.

Refaldi I. D. Putra, Tatsuya Ishikawa, Naomi Simumba, and Michiaki Tatsubori. Sandwiched lo-res simulation for scalable flood modeling. In *ICASSP 2024 – IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2420–2424. IEEE, 2024. doi: 10.1109/ICASSP48485.2024.10447221. URL https://ieeexplore.ieee.org/document/10447221.

Md Ashiqur Rahman, Zachary E. Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. arXiv preprint arXiv:2204.11127, 2022. URL https://arxiv.org/abs/2204.11127.

Maziar Raissi, Paris Perdikaris, and George Em 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. doi: 10.1016/j.jcp.2018.10.045.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of PDEs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 77187–77200. Curran Associates, Inc., 2023. doi: 10.52202/075280-3376. URL https://papers.neurips.cc/paper_files/paper/2023/hash/f3c1951b34f7f55ffaecada7fde6bd5a-Abstract-Conference.html.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28.

David Ruhe, Jayesh K Gupta, Steven De Keninck, Max Welling, and Johannes Brandstetter. Geometric Clifford algebra networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29306–29337. PMLR, 2023. URL https://proceedings.mlr.press/v202/ruhe23a.html.

Subhankar Sarkar and Souvik Chakraborty. Spatio-spectral graph neural operator for solving computational mechanics problems on irregular domain and unstructured grid. arXiv preprint arXiv:2409.00604, 2024. URL https://arxiv.org/abs/2409.00604.

Patrick Schnell, Philipp Holl, and Nils Thuerey. Half-inverse gradients for physical deep learning. arXiv preprint arXiv:2203.10131, 2022. URL https://arxiv.org/abs/2203.10131.

Erica L Schwarz, Luca Pegolotti, Martin R Pfaller, and Alison L Marsden. Beyond CFD: Emerging methodologies for predictive simulation in cardiovascular health and disease. *Biophysics Reviews*, 4(1):011301, 2023. doi: 10.1063/5.0109400.

Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk (eds.). *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. Neural Information Processing. MIT Press, 2006. ISBN 9780262195478. URL https://people.csail.mit.edu/gregory/annbook/book.html.

Ramansh Sharma and Varun Shankar. Ensemble and mixture-of-experts DeepONets for operator learning. arXiv preprint arXiv:2405.11907, 2024. URL https://arxiv.org/abs/2405.11907.

Jonathan D. Smith, Kamyar Azizzadenesheli, and Zachary E. Ross. EikoNet: Solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59 (12):10685–10696, 2021. doi: 10.1109/TGRS.2020.3039165.

Michiel Straat, Thorben Markmann, and Barbara Hammer. Solving turbulent Rayleigh-Bénard convection using Fourier neural operators. arXiv preprint arXiv:2501.16209, 2025. URL https://arxiv.org/abs/2501.16209.

Zhiqing Sun, Yiming Yang, and Shinjae Yoo. A neural PDE solver with temporal stencil modeling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 33135–33155. PMLR, 2023. URL https://proceedings.mlr.press/v202/sun23o.html.

Roger Temam. *Navier–Stokes Equations: Theory and Numerical Analysis*, volume 2 of *Studies in Mathematics and Its Applications*. North-Holland, 1977. ISBN 9780720428407. URL https://www.jamstec.go.jp/opac/en/volume/4150.

Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. Physics-based deep learning. Online book, 2021. URL https://physicsbaseddeeplearning.org/.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier neural operators. arXiv preprint arXiv:2111.13802, 2021. URL https://arxiv.org/abs/2111.13802.

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6111–6122. Curran Associates, Inc., 2020. URL https://papers.neurips.cc/paper_files/paper/2020/hash/43e4e6a6f341e00671e123714de019a8-Abstract.html.

Charles Van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992. doi: 10.1137/1.9781611970999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://papers.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Chuwei Wang, Shanda Li, Di He, and Liwei Wang. Is L^2 physics informed loss always suitable for training physics informed neural network? In *Advances in Neural Information Processing Systems*, volume 35, pp. 8278–8290. Curran Associates, Inc., 2022. doi: 10.52202/068431-0602. URL https://papers.neurips.cc/paper_files/paper/2022/hash/374050dc3f211267bd6bf0ea24eae184-Abstract-Conference.html.

Qi Wang, Pu Ren, Hao Zhou, Xin-Yang Liu, Zhiwen Deng, Yi Zhang, Ruizhi Chengze, Hongsheng Liu, Zidong Wang, Jian-Xun Wang, Ji-Rong Wen, Hao Sun, and Yang Liu. P^2C^2Net: PDE-preserved coarse correction network for efficient prediction of spatiotemporal dynamics. arXiv preprint arXiv:2411.00040, 2024. URL https://arxiv.org/abs/2411.00040.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-FNO—an enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022. doi: 10.1016/j.advwatres.2022.104180.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. arXiv preprint arXiv:2402.02366, 2024. URL https://arxiv.org/abs/2402.02366.

Wei Xiong, Xiaomeng Huang, Ziyang Zhang, Ruixuan Deng, Pei Sun, and Yang Tian. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *Journal of Computational Physics*, 513:113194, 2024. doi: 10.1016/j.jcp.2024.113194.

Huaiqian You, Quinn Zhang, Colton J Ross, Chung-Hao Lee, and Yue Yu. Learning deep implicit Fourier neural operators (IFNOs) with applications to heterogeneous material modeling. *Computer Methods in Applied Mechanics and Engineering*, 398:115296, 2022. doi: 10.1016/j.cma.2022.115296.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016. URL https://arxiv.org/abs/1605.07146.

Chenyu Zeng, Yanshu Zhang, Jiayi Zhou, Yuhan Wang, Zilin Wang, Yuhao Liu, Lei Wu, and Daniel Zhengyu Huang. Point cloud neural operator for parametric PDEs on complex and variable geometries. arXiv preprint arXiv:2501.14475, 2025. URL https://arxiv.org/abs/2501.14475.

Hao Zhang, Yuting Weng, Zhiwei Zhao, and Dezhi Zhou. Learning transient evolution of multidimensional reacting flows by multiscale Fourier neural operators. *Proceedings of the Combustion Institute*, 40(1-4):105714, 2024. doi: 10.1016/j.proci.2024.105714.

Jiawei Zhuang, Dmitrii Kochkov, Yohai Bar-Sinai, Michael P Brenner, and Stephan Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, 6(6):064605, 2021. doi: 10.1103/physrevfluids.6.064605.

A EXPERIMENTAL SETTINGS

In this appendix, we provide a detailed description of the experimental setup, including hardware and software specifications, data generation configurations, and other training details. Since the solver-based hybrid model (and the MF-MoE models that include these hybrid models as experts) can require days for training and testing, we pre-generate the solver simulation results of the provided dataset for faster processing. The full codes and other omitted experimental settings are included in the supplymentary material.

Hardware Specifications All experiments were run on a single compute node with: Dual AMD EPYC 9124 CPUs (32 total CPU cores), 768 GB of DDR5 4800 MHz memory, and 8 NVIDIA RTX 6000 Ada Generation GPUs.

Software Requirements Throughout all experiments, we use Python 3.11.2, CUDA 12.4.1, and OpenSSL 1.1.1k FIPS (25 Mar 2021). All other required Python packages match those specified by the original repository, PDEArena (Gupta & Brandstetter, 2022; Brandstetter et al., 2022; Ruhe et al., 2023).

A.1 ILLUSTRATION OF SOLVER-BASED HYBRID MODELS

The MF-MoE model includes experts fusing a numerical PDE solver with a refinement neural network. In our paper, we use the hybrid model illustrated by Figure 7. Specifically, a numerical solver generates a low-resolution approximation $\hat{y}_c$, which is then processed by an up-sampling step to recover fine-scale details. The neural operators are used to learn the correction Δy. A convex combination of $\hat{y}_C$ and Δy is used as the final prediction.

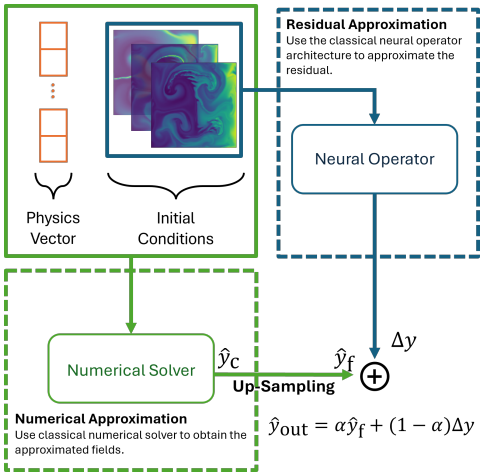

Figure 7: The illustration of the *solver-based hybrid model* used in this paper. The input sample is processed by the numerical solver and the pure neural network. The output of the numerical solver is the coarse prediction $\hat{y}_c$; it will be up-sampled to the fine prediction $\hat{y}_f$ with the desired dimension using the k-nearest neighbor methods. The output of the pure neural network is used to approximate the residual term Δy. The prediction $\hat{y}_{\text{out}}$ is a convex combination of two terms: $\hat{y}_{\text{out}} := \alpha\hat{y}_f + (1-\alpha)\Delta y$, where $\alpha \in [0, 1]$ is a hyper-parameter for tuning.

A.2 CONFIGURATION FOR DATA GENERATION

We adapt the existing data generation configuration from PDEArena's Github Repository, which has been widely used by Gupta & Brandstetter (2022); Brandstetter et al. (2022); Ruhe et al. (2023). More explicitly, we made the following modifications on the original configuration `navierstokes2dsmoke.yaml` to improve the accuracy of the PhiFlow's numerical solution: We reduced `skip_nt`, the initial timesteps to ignore, from 8 to 0, and reduced the `tmax`, the maximum time period, from 108.0 to 60.0; as the result, we are investigating the fluid dynamic over

the time period $[18.0, 60.0]$. We control the overal numerical accuracy by adjusting the number of timesteps and the spatial resolution. More explicitly, these configurations are summarized in Table 4. The time cost and the MSE error is evaluated per frame. For example, if a trajectory with setting $\texttt{nt} = 64$ and $\texttt{sampling_rate} = 8$, it outputs $\frac{\texttt{nt}}{\texttt{sampling_rate}} = \frac{64}{8} = 8$ frames; then its time cost is the total time cost t_{total} divided by 8. Similarly, we take the slice of the same time step t, and evaluate the MSE loss between the up-sampled low-resolution field and the groudtruth field.

Table 4: Configuration details for data generation at different resolution levels. Each configuration specifies the number of timesteps (`nt`), spatial resolution (`nx`, `ny`), time cost per frame, and MSE of the numerical solution compared to the ground truth (GT). The ground truth configuration achieves the highest accuracy with the highest computational cost, while coarser resolutions progressively reduce time cost at the expense of higher error. The visual illustration has been presented in Figure 2.

Resolution Level	**Configuration File**	`nt`	`(nx,ny)`	**Time Cost** (s)	**MSE Error**
Groundtruth (GT)	`navierstokes2dsmoke_high_res.yaml`	512	$(256, 256)$	1.58×10^2	< 0.001
Fine	`navierstokes2dsmoke_fine.yaml`	64	$(128, 128)$	0.38×10^2	0.065
Medium	`navierstokes2dsmoke_medium.yaml`	32	$(64, 64)$	0.20×10^2	0.104
Coarse	`navierstokes2dsmoke_coarse.yaml`	16	$(32, 32)$	0.12×10^2	0.135
XCoarse	`navierstokes2dsmoke_xcoarse.yaml`	8	(16,16)	0.08×10^2	0.151

For each viscosity μ (as shown in Equation (6) or the following dataset split), we generate 5 independent trajectories with the length 8 (as a result, each parameter corresponds to $20 = 5 \times (8 - 4)$ pairs of data for each μ in total, as we are using 4 frames to predict the next frame). As the result, the train, validation, and test set contain 60, 80, and 80 pairs of data, respectively. (for the case `nt` larger than 8, we set the `sampling_rate` parameter to omit the intermediate data). In Appendix D, we visualize a single trajectory for better illustrating the accuracy of the numerical solution over different fidelity.

$$\begin{aligned} \text{(Train)} \quad & \mu \in \{3.2, 1.6, 0.8\} \times 10^{-2}, \\ \text{(Valid)} \quad & \mu \in \{1.6, 0.8, 0.4, 0.2\} \times 10^{-2}, \\ \text{(Test)} \quad & \mu \in \{0.8, 0.4, 0.2, 0.1\} \times 10^{-2}. \end{aligned}$$

Other parameters are included in the supplementary material. We omit the full discussions on these parameters and the details of introduction can be found in the PDEArena (Gupta & Brandstetter, 2022).

A.3 Configuration for Training

In this section, we present the details on the training configuration. We use the AdamW optimizer (Loshchilov & Hutter, 2017; 2019) with learning rate $\eta = 2 \times 10^{-4}$ for 8,000 training samples[3], a weight decay of 1×10^{-5}, and a batch size of 16. No further hyperparameter tuning is applied across different models. For the MF-MoE model, we introduce several additional hyperparameters: the number of experts $N = 4$, where one expert is a pure neural-operator model and the other three are solver-based hybrid models of varying fidelity (Fine, Medium, and Coarse). Their relative accuracy and time costs appear in Table 4. We set the time-cost constraint to $c = 10.0$, ensuring that a single solver-based hybrid model alone (though accurate) is excluded, as it exceeds this cost threshold. A concise summary of all the important hyperparameters is provided in Table 5.

A.4 Configuration of Baselines

We list here a brief overview of the three types of model we have considered in this work. Their implementations are directly taken from the PDEArena (Gupta & Brandstetter, 2022) and we follow the same name used in their codes. These models are also compatible with our MF-MoE framework and can be served as the pure neural operator expert or the residual approximation network used in the solver-based hybrid expert.

- **Fourier Neural Operators (FNOs)** (Li et al., 2020a): FNOs are among the most widely used neural PDE surrogates, leveraging the Fast Fourier Transform (FFT) (Van Loan, 1992; Cooley

[3]When the dataset has less than 8,000 samples, some samples will be re-used.

Table 5: Hyperparameter overview for the main training process in the fluid flow dynamic prediction experiment. Unless further specified, all experiments follow this hyper-parameter setting.

Hyperparameter	Value
Optimizer	AdamW
Learning rate (η_θ)	2×10^{-4}
Number of training samples	8,000
Weight decay	1×10^{-5}
Batch size	16
Number of experts (N)	4
Time-cost constraint (c)	10.0

& Tukey, 1965). In this framework, low-frequency Fourier modes capture global features, while high-frequency modes capture local details. The first set of parameters (e.g., 128, 96, 64) specifies the number of hidden channels, which determines the model size. The second parameter indicates the number of modes and the hidden channels of each mode.

- **U-Net** (Ma et al., 2021; Chen & Thuerey, 2021; Rahman et al., 2022): The U-Net architecture, originally developed by Ronneberger et al. (2015) for biomedical image segmentation, has since been adapted for PDE surrogates (Ma et al., 2021; Chen & Thuerey, 2021). A special U-Net architecture with replacing all blocks with the Fourier blocks is called UNO, which is developed by Rahman et al. (2022). We adjust the number of hidden layers from 64 to 128 to control the size of UNO. If we only replace the blocks in downsampling paths, the model is called U-Nets with Fourier blocks. We consider different variants with including different modes in the Fourier blocks in our experiment.

A.5 Experimental Settings for Section 4.3

In this section, we describe the dataset splitting used in Section 4.3. For each buoyancy force $\mathbf{f}$ (as shown in Equation (7) or the following dataset split), we generate 20 independent trajectories with the length 8 (each trajectory contains 4 pairs of data). As the result, the train, validation, and test set contain 400, 400, and 160 pairs of data, respectively. (for the case `nt` larger than 8, we set the `sampling_rate` parameter to omit the intermediate data).

$$\begin{aligned} \text{(Train \& Valid)} \quad & \mathbf{f} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}, \\ \text{(Test)} \quad & \mathbf{f} \in \{0.05, 0.55\}. \end{aligned}$$

A.5.1 Specific Experimental Settings for Validating the Impact of Physical Parameters on Gating

In this experiment, we maintain the same experimental setting as described in Appendix A.5 while including an additional regularization term in the objective loss function in the gating mechanism; that is,

$$\begin{aligned} \min_{g} \quad & \mathbb{E}_{i\sim g}[E_i] - \lambda \mathrm{Ent}(g) \\ \text{s.t.} \quad & \mathbb{E}_{i\sim g}[C_i] \le c, \end{aligned}$$

where $\mathrm{Ent}(g)$ represents the entropy of the distribution g; explicitly, $\mathrm{Ent}(g) := -\sum_i g_i \log(g_i)$. We set $\lambda = 0.2$ to improve the smoothness of the distribution evolution as the buoyancy force varies. We note that this setting is only applied to our investigation for analyzing the impact of physical parameters on gating and is not applied to other experiments.

A.6 Extended Experimental Results to Table 3

In this section, we include the full results of the extended experiments on validating the performance of MF-MoE on the buoyancy force parameters.

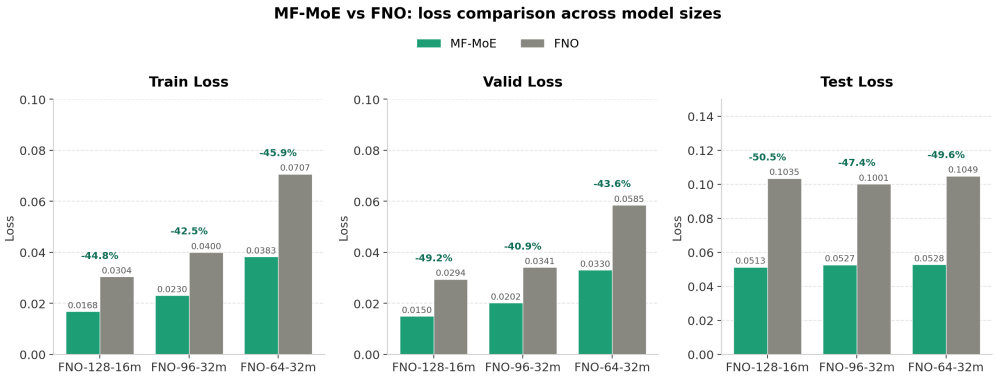

Figure 8: Visual comparison of generalization performance on multi-step prediction task. We consider the prediction horizon $N = 4$ and prediction the next-step field autoregres- sively. We report the averaged MSE over the train/validation/test sets, comparing the proposed MF-MoE against the standalone FNO baseline.

A.7 Extended Experimental Results to Figure 6

In this section, we include the full results of the extended experiments on validating the performance of MF-MoE on the multi-step prediction. The full results are presented in Figure 8.

B Additional Experimental Results

This appendix complements Section 4.5 with the dataset configuration, expert cost calibration, training hyperparameters, and detailed results for the Burgers experiment.

B.1 Dataset Configuration

We generate 2D viscous Burgers trajectories with the same multi-fidelity pipeline used for Navier-Stokes (Appendix A.2). The system is integrated via semi-Lagrangian advection followed by explicit diffusion under periodic boundary conditions on the domain $[0, 32]^2$ over the time interval $[18, 60]$. Five fidelity levels are produced; their grid sizes and temporal sub-sampling factors are summarized in Table 6. Of these, the Coarse, Medium, and Fine resolutions instantiate the three solver-based hybrid experts; Groundtruth (GT) provides the supervised target during training and evaluation.

Table 6: Resolution configurations for the 2D Burgers dataset. After temporal sub-sampling all fidelities share a fixed trajectory length of $T = 8$.

Level	Grid	Raw `NT`	`sample_rate`	Trajlen T
Coarse	32×32	16	2	8
Medium	64×64	32	4	8
Fine	128×128	64	8	8
Groundtruth (GT)	256×256	512	64	8

To probe trajectory-level generalization while holding the physical regime fixed, we use the same set of viscosities for both the training and test splits and partition by the random seed of the initial-condition sampler:

$$\text{(Train)} \quad \mu \in \{0.5, 1.0, 2.0\} \times 10^{-2}, \tag{9}$$

$$\text{(Test)} \quad \mu \in \{0.5, 1.0, 2.0\} \times 10^{-2}. \tag{10}$$

Each combination contributes 16 trajectories per fidelity, for a total of 96 training and 48 test trajectories. With input window `time_history` $= 4$ and prediction horizon `time_future` $= 1$, this yields the sliding-window (x, y) pairs used for single-step training.

B.2 Multi-Fidelity Experts and Cost Calibration

The MF-MoE module uses $N = 4$ experts: one pure neural-operator expert and three hybrid experts that combine the neural-operator output with the solver output at the corresponding fidelity. The wall-clock cost of each expert is calibrated from the data-generation runs of the corresponding solver (the pure neural-operator expert only incurs a forward pass) and serves as the cost vector of the LP gate (Equation (3)). The calibrated values are listed in Table 7.

Table 7: Burgers expert configurations and calibrated wall-clock cost. The cost values populate the LP cost vector and are matched in units to the per-step time budget c.

Expert	Fidelity	Type	Cost
1	-	Pure neural operator	0.2
2	Fine	Hybrid (NO + solver)	83.0
3	Medium	Hybrid (NO + solver)	41.0
4	Coarse	Hybrid (NO + solver)	20.0

B.3 Training Configuration

We optimize the four experts jointly with the same MoE objective as in Section 4.1, using the Mean Squared Error on the predicted velocity components as both the training loss and the evaluation metric. The pure neural-operator expert and the neural-operator branch of each hybrid expert share the same backbone. No further hyperparameter tuning is performed beyond what was used for the Navier-Stokes experiments.

B.4 Quantitative Results

Table 8 reports the test-set MSE on the Burgers dataset for the standalone FNO baseline and our MF-MoE under the time budget $c = 10.0$. Across all data sets, MF-MoE achieves a markedly lower error than the standalone neural operator while satisfying the prescribed time budget, demonstrating that the gating mechanism generalizes to a PDE family outside of incompressible Navier-Stokes without any retuning.

Table 8: Train / Validation / Test MSE ($\times 10^{-2}$) on the 2D Burgers equation under the time budget $c = 10.0$. Lower is better.

Model	**Train MSE**	**Valid MSE**	**Test MSE**
UNO-64	0.173×10^{-2}	0.140×10^{-2}	2.98×10^{-2}
DilResNet-128	0.613×10^{-2}	0.561×10^{-2}	0.899×10^{-2}
U-Net$_{\text{mod}}$-64	0.200×10^{-2}	0.183×10^{-2}	0.534×10^{-2}
FNO-128-8$_{\text{modes}}$	0.396×10^{-2}	0.352×10^{-2}	5.59×10^{-2}
FNO-64-32$_{\text{modes}}$	0.527×10^{-2}	0.530×10^{-2}	2.03×10^{-2}
MF-MoE (UNO-64)	0.101×10^{-2}	0.0815×10^{-2}	1.38×10^{-2}
MF-MoE (DilResNet-128)	0.422×10^{-2}	0.376×10^{-2}	0.574×10^{-2}
MF-MoE (U-Net$_{\text{mod}}$-64)	0.134×10^{-2}	0.127×10^{-2}	0.301×10^{-2}
MF-MoE (FNO-128-8$_{\text{modes}}$)	0.226×10^{-2}	0.194×10^{-2}	1.22×10^{-2}
MF-MoE (FNO-64-32$_{\text{modes}}$)	0.362×10^{-2}	0.332×10^{-2}	1.01×10^{-2}

C Limitations & Future Work

Despite the promising results demonstrated by the MF-MoE framework, several limitations remain, highlighting opportunities for future research: (1) This study primarily focuses on single-step predictions, which limits its applicability to scenarios requiring multi-step or long-horizon forecasts.

Future work could explore incorporating unrolled training techniques and developing time-evolving gating strategies to enhance performance in these more complex settings. (2) While the MF-MoE framework is designed to be flexible and compatible with advanced models, its empirical performance with such models has not yet been thoroughly evaluated. Investigating its integration with state-of-the-art neural operators and hybrid models is a promising direction for future studies.

D Visualization of Numerical Solution over Different Resolutions

In this section, we present the visualization of numerical solution over different resolution levels (including Fine, Medium, Coarse, and XCoarse). Their detailed resolution is provided in Table 4.

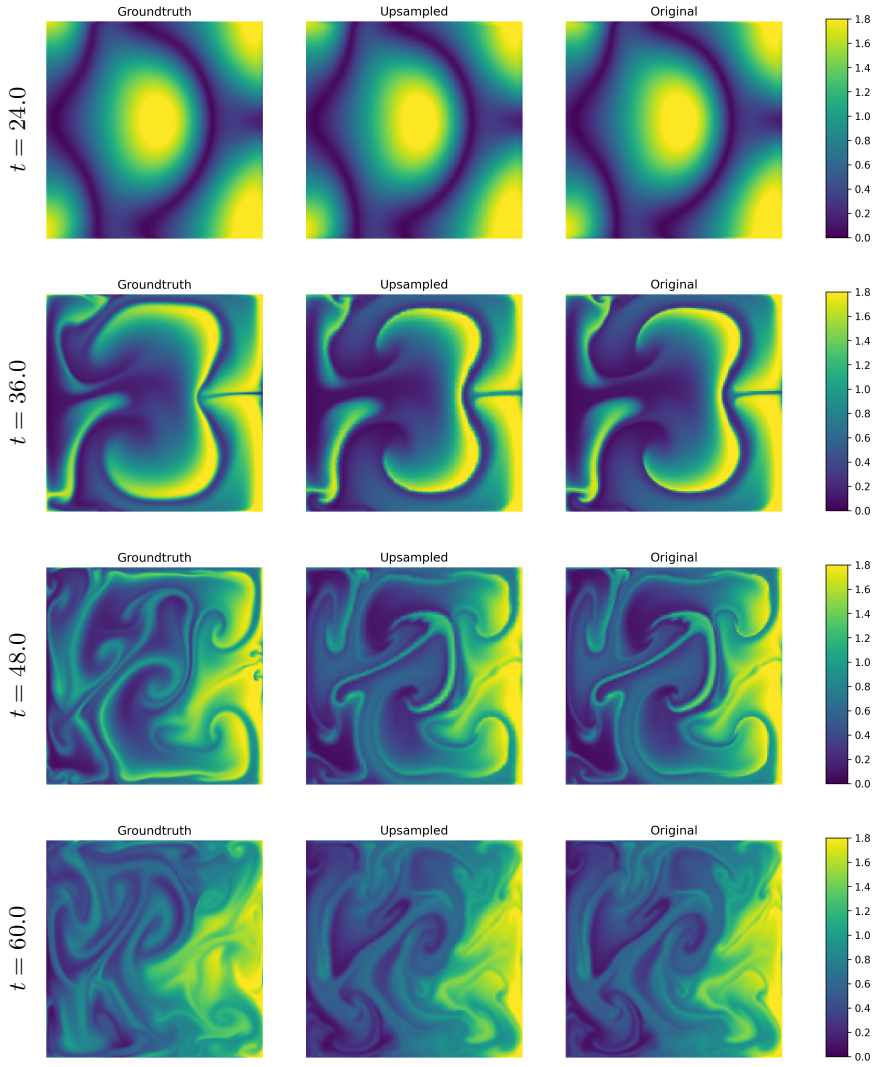

Figure 9: Illustration of the scalar field solved over the fine-level resolution at four time steps: $t = 24.0, 36.0, 48.0, 60.0$.

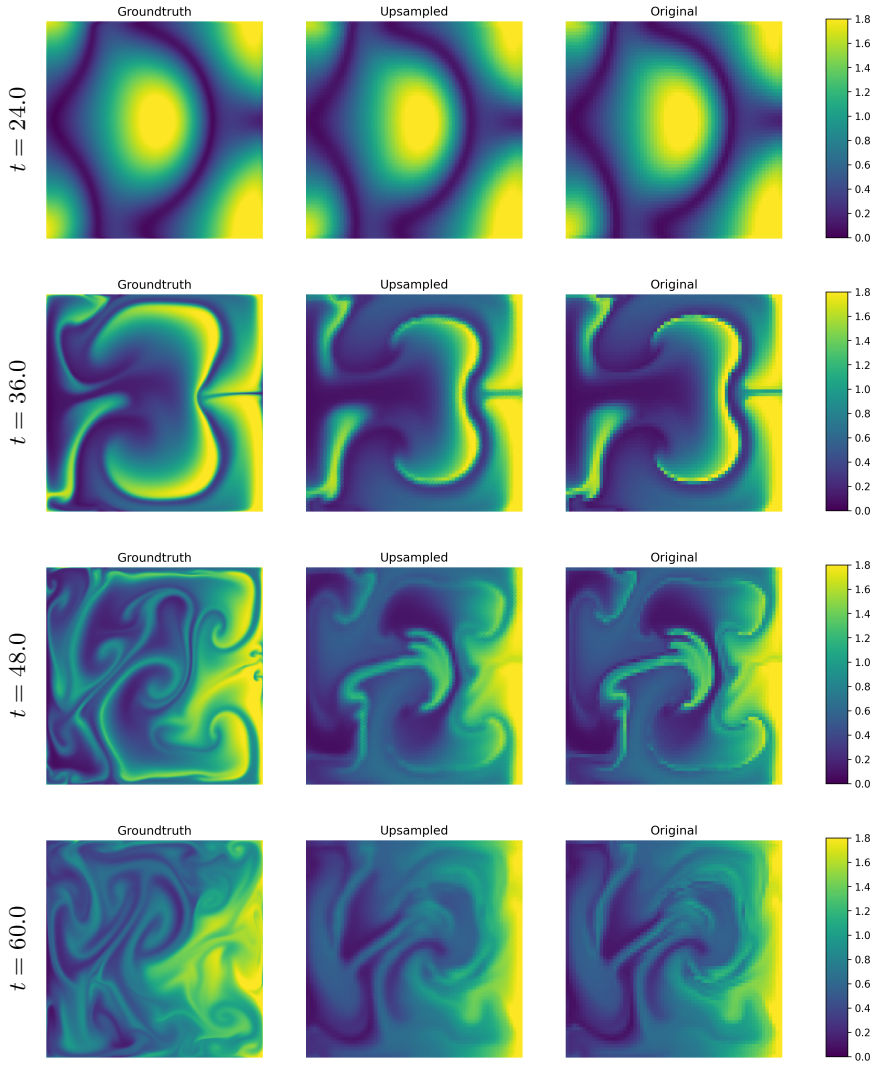

Figure 10: Illustration of the scalar field solved over the medium-level resolution at four time steps: $t = 24.0, 36.0, 48.0, 60.0$.

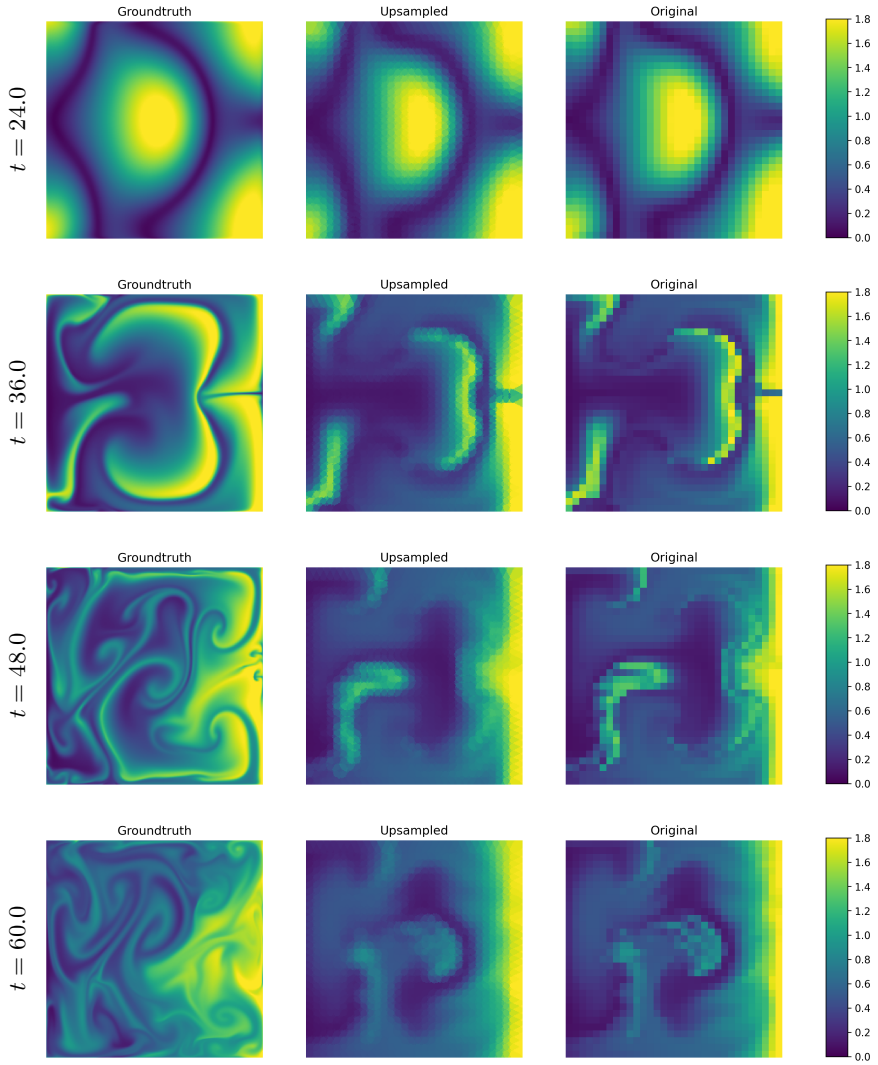

Figure 11: Illustration of the scalar field solved over the coarse-level resolution at four time steps: $t = 24.0, 36.0, 48.0, 60.0$.

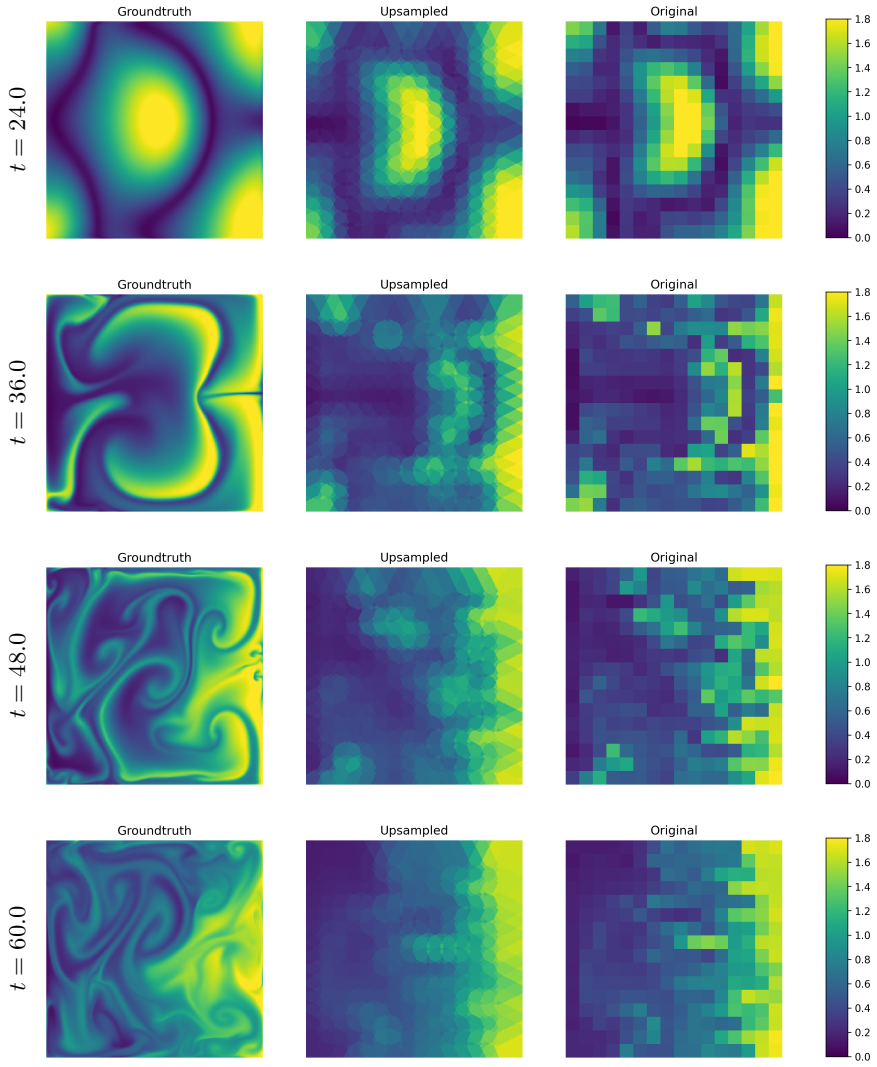

Figure 12: Illustration of the scalar field solved over the xcoarse-level resolution at four time steps: $t = 24.0, 36.0, 48.0, 60.0$.

E Related Work

Physics-Informed Neural Network (PINN) and Neural Operator Learning Physics-Informed Neural Networks (PINNs) provide a direct approach to solving PDEs by parameterizing the solution as a neural network, which is trained to satisfy the governing equations and boundary conditions (E & Yu, 2018; Raissi et al., 2019; Bar & Sochen, 2019; Smith et al., 2021; Wang et al., 2022). However, as noted by Sun et al. (2023), PINNs require re-optimization for every new setup, limiting their scalability, particularly for dynamic problems. An alternative neural network-based method for solving PDEs is neural operator learning (Lu et al., 2019; Bhattacharya et al., 2020; Patel et al., 2021; Li et al., 2020a; Nelsen & Stuart, 2021; Cao, 2021; Li et al., 2020c;b; Liang et al., 2024; Wu et al., 2024; Li et al., 2023). Unlike PINNs which parameterize the solution directly, neural operator methods learn a parameterized representation of a mapping from a field over the spatial-temporal domain (e.g., initial or boundary conditions) to another field over the same domain. These methods extend PDE solutions to functional mappings, allowing resolution-agnostic predictions through techniques such as Fourier transforms (Li et al., 2020a; Tran et al., 2021) and wavelet transforms (Gupta et al., 2021). However, as highlighted by Sun et al. (2023) and Belbute-Peres et al. (2020), purely neural network-based structures often struggle with out-of-distribution data due to the tendency of over-parameterized neural networks to overfit (Lawrence et al., 1997).

Solver-Based Hybrid Models Recent advancements have introduced various neural network designs that integrate with classical numerical simulators (Holl & Thuerey, 2024; Thuerey et al., 2021; Economon et al., 2016; Anderson et al., 2021; MFEM), broadly categorized into two approaches. The first approach employs neural networks to learn the stencils of advection-diffusion problems within the Finite Volume Method (FVM) framework (Bar-Sinai et al., 2019; Kochkov et al., 2021; Sun et al., 2023). The second approach leverages neural networks to correct numerical errors arising from low-resolution simulator outputs (Mishra, 2018; Um et al., 2020; Belbute-Peres et al., 2020; Pestourie et al., 2021; Dresdner et al., 2022; List et al., 2022; Frezat et al., 2022; Bruno et al., 2022; Ma et al., 2024). Our proposed MF-MoE framework falls into the latter category and aims to enhance time-efficiency by dynamically controlling the numerical solver's involvement.

Mixture-of-Expert Model in Operator Learning The application of Mixture-of-Expert (MoE) models in operator learning has shown promising results in recent studies. For instance, Sharma & Shankar (2024) introduced the ensemble DeepONet, which partitions the input domain into overlapping regions and blends outputs using a partition-of-unity approach. Similarly, Liang et al. (2024) proposed a model that employs Fourier Neural Operators (FNOs) at different scales to process spatial or temporal segments, which are then aggregated into a unified global solution. Hao et al. (2023) designs the GNOT which utilizes the gating network as the soft domain decomposition technique for dealing with multi-scale problems. These approaches have demonstrated significant improvements in the accuracy, generalization ability, and the flexiablity of the PDE surrogate model. Notably, our work differs from these MoE methods in two key aspects: (1) Instead of dividing the physical domain into multiple subregions, we use the entire physical field as the input, preserving the global context of the problem and avoiding the challenges associated with handling boundary conditions between segments. (2) While previous studies primarily focus on enhancing the flexibility of pure neural operators, our goal is fundamentally different: beside the improved prediction accuracy, we also care about the time-efficiency caused by integrating the numerical PDE solver.

E.1 Neural Operators

Recent advancements in neural operators have expanded beyond foundational architectures like FNOs (Li et al., 2020a) and U-Nets (Ronneberger et al., 2015; Ma et al., 2021; Chen & Thuerey, 2021; Rahman et al., 2022). Below, we summarize key developments in the field.

E.1.1 Fourier Neural Operators (FNOs)

The FNO architecture utilizes Fast Fourier Transforms (FFTs) to decompose inputs into low and high Fourier modes, capturing global and local information essential for modeling physical fields. Beyond classical 2D/3D applications, recent efforts focus on improving scalability, stability, and accuracy for highly non-linear PDEs. For instance, domain-decomposition-based FNOs (Nidhan et al., 2024) and

M2M (Liang et al., 2024) split large computational domains into subdomains to handle extreme-scale problems in fluid and solid mechanics. Multi-resolution variants, such as MFNO (Zhang et al., 2024), combine FNO blocks at different frequency levels to capture multi-scale phenomena, especially in turbulent flows. Notable extensions include U-FNOs (Wen et al., 2022) and IFNOs (You et al., 2022), which enhance the classical FNOs by extending their application from 2D to 3D predictions. Recent work by Straat et al. (2025) demonstrates FNO's effectiveness in solving turbulent Rayleigh-Bénard convection, outperforming traditional surrogates like Dynamic Mode Decomposition while achieving zero-shot super-resolution capabilities.

E.1.2 U-Shaped Neural Operators (UNOs)

U-Net-based neural operators for PDE surrogates have seen continuous innovations aimed at capturing complex, multi-scale structures in physical fields. When replacing the convolution blocks in both sampling paths with the Fourier blocks, the U-Net becomes the U-shaped Neural Operator (UNO). Alongside wide Residual Networks (Zagoruyko & Komodakis, 2016) and spatial attention mechanisms (Vaswani et al., 2017), the Point Cloud Neural Operator (PCNO) (Zeng et al., 2025) extends U-Net principles to handle parametric PDEs on complex geometries using adaptive meshing and rotation-invariant inputs. Recent innovations also include physics-informed latent architectures like PI-Latent-NO (Karumuri et al., 2025), which combines reduced-order modeling with operator learning for efficient training on high-dimensional PDEs. These developments underscore U-Nets' flexible design, enabling more accurate and efficient solution operators for a range of partial differential equations.

E.1.3 Graph Neural Operators

Graph Neural Operators (GNOs) extend the neural operator framework to irregular and graph-structured domains, enabling function space mappings on unstructured grids and complex geometries. By leveraging graph neural networks (GNNs), these architectures effectively model physical systems with non-Euclidean structures. The Spatio-Spectral Graph Neural Operator (Sp2GNO) (Sarkar & Chakraborty, 2024) integrates spatial and spectral GNNs to learn solution operators across arbitrary geometries. By combining local and global feature extraction, Sp2GNO can handle both time-dependent and time-independent PDEs on irregular computational domains. The Multipole Graph Neural Operator (MGNO) (Li et al., 2020b) addresses scalability challenges in PDE modeling by employing multipole expansions to capture long-range dependencies efficiently. The Physics-Informed Graph Neural Operator (PIGNO) (Chen et al., 2024) incorporates physical constraints into the operator learning process, ensuring solutions adhere to underlying laws governing the modeled system. Recent developments in GNOs highlight their versatility in handling PDEs on irregular domains, from computational mechanics to networked dynamical systems, further expanding the applicability of neural operators beyond structured grids.

E.1.4 Other Advancing Neural Operator Architectures

Beyond FNO and U-Net variants, several novel frameworks have emerged. The Neural Operator-based symbolic Model approximaTion and discOvery (NOMTO) (Garmaev et al., 2025) enables discovery of differential equations with singularities and special functions, significantly expanding symbolic regression capabilities. Hydraulic-based graph neural networks like mSWE-GNN (Bentivoglio et al., 2025) introduce multi-resolution modeling with ghost cells for flood prediction, achieving 700x speedups while handling time-varying boundary conditions. Additional architectures include Residual Networks for function space mapping (Cao et al., 2023), Wavelet Neural Operators (WNO) (N et al., 2024), Koopman Neural Operators (Xiong et al., 2024), the U-shaped Neural Operator (Azizzadenesheli et al., 2024), and Implicit Adaptive Fourier Neural Operators (IAFNO) (Jiang et al., 2025), demonstrating neural operators' growing versatility in handling diverse physical systems, from turbulent flows to symbolic equation discovery.