Exploring Conditioning Strategies for Fourier Neural Operators

Wenxi Cai, Yimin Wang
caiwenxi@umich.edu,wyimin@umich.edu
University of Michigan
USA

ABSTRACT

Learning neural operators for parameter-dependent PDEs requires integrating physical parameters into the operator architecture. We systematically compare three conditioning strategies for Fourier Neural Operators (FNOs): local feature-wise modulation, global channel-wise scaling, and input-level concatenation. Using identical backbones, we evaluate all mechanisms on four PDE systems—Advection, Burgers, Elasticity, and Navier–Stokes—and show that conditioning consistently improves parameter generalization. Local modulation performs best on Advection and Burgers, while global scaling is most effective for linear elasticity. Input-level conditioning proves sufficient for Navier–Stokes but lacks expressivity for varying wave speeds. In a second stage, we apply the PDE-Refiner training scheme to the strongest method on each system and find that refinement improves some PDEs but has limited effect on Navier–Stokes due to its strongly advective dynamics. Our results provide a clear comparison of conditioning designs and practical guidance for building parameter-aware neural operators.

KEYWORDS

Neural Operators, PDE Forecasting, Parameter Generalization, Long-horizon Stability

1 INTRODUCTION

Learning efficient surrogate models for partial differential equations (PDEs) is an important goal in scientific machine learning. Many tasks in fluid dynamics, elasticity, and transport require repeated PDE evaluations under varying physical conditions, which makes traditional numerical solvers expensive for large ensembles or real-time deployment. Neural operators (NOs)[1, 3, 4, 8] provide a data-driven alternative by learning mappings between function spaces, avoiding the cost of running iterative numerical schemes.

Among existing operator-learning models, the Fourier Neural Operator (FNO)[7] is one of the most widely used due to its ability to capture long-range spatial interactions through spectral convolution. FNO-based methods have achieved strong performance in diverse applications, including weather forecasting[12] and seismic modeling[5].

In the broader context of deep learning, conditioning mechanisms are established tools for adapting model behavior across different tasks. While widely recognized in generative modeling through Conditional Generative Adversarial Networks (cGANs) [11] and Latent Diffusion Models [15], these strategies are also fundamental in computer vision and style transfer. Techniques such as Feature-wise Linear Modulation (FiLM) [13] and Adaptive Instance Normalization (AdaIN) [2] enable external inputs to modulate internal feature maps directly. This allows a single neural network to process data differently based on the context.

However, standard FNO architectures have no clear mechanism for integrating physical parameters such as advection speeds, viscosity, forcing coefficients, or wave speeds. These parameters significantly influence solution structure.

This motivates a central question:

How should physical parameters be incorporated into FNO to improve accuracy on parameter-varying PDEs?

We propose architectural modifications to the FNO that define how physical parameters interact with its spectral blocks. Our design is instantiated through three conditioning mechanisms that inject parameter information at different stages of the operator:

- **Input Concatenation.** The parameters are broadcast across the grid and appended as extra channels before the first Fourier layer. This gives the model access to parameter information without altering the internal spectral transformations.
- **Local Condition.** The parameter vector is encoded by an MLP to produce channel-wise scale and shift values. These values are broadcast over spatial coordinates and applied to intermediate spectral features through pointwise linear modulation, allowing the parameters to adjust each Fourier block directly.
- **Global Condition.** The parameters are mapped to a latent vector of size C_h, broadcast spatially, and used as channel-wise multiplicative factors on the learned feature map. This introduces a global adjustment per channel without modifying spatial patterns.

These mechanisms represent the primary strategies through which parameter information can enter an operator-learning model: direct input augmentation, local feature modulation, and global channel scaling. To compare their behavior across diverse physical regimes, we evaluate all three mechanisms on four PDE benchmarks with different dynamics and parameter dependencies:

- **Advection–Diffusion** with random velocities (c_x, c_y);
- **Burgers** equation with viscosity ν;
- **Elastic Wave** propagation with wave speeds (c_p, c_s);
- **Navier–Stokes** in vorticity form with forcing coefficients $(\text{coef}_x, \text{coef}_y)$.

We train sixteen models in total: three conditioned variants and one unconditioned baseline on each benchmark, to assess how each conditioning strategy interacts with different PDE dynamics, including transport, diffusion, oscillations, and nonlinear fluid flow. For the best-performing strategy on each benchmark, we apply the refinement technique introduced in [9], to illustrate the effects and current limitations.

Project page: https://pinkbubblebubble.github.io/Conditional-FNO/.

This study examines how different conditioning mechanisms influence the behavior of FNO. By treating conditioning as the central design variable and evaluating three representative strategies across several PDE systems, we aim to provide a clear comparison that can inform the construction of parameter-aware operator models.

2 RELATED WORK

Neural operators for PDEs. Neural operators learn mappings between function spaces and have become a versatile framework for PDE surrogate modeling. DeepONet[10] represents nonlinear operators through a branch–trunk decomposition, while graph-based neural operators[14] extend operator learning to irregular meshes. Kernel-based and attention-guided variants[1, 3, 4] further enhance expressiveness for fluid dynamics, elasticity, and multiscale transport. These models focus primarily on architectural flexibility, whereas the question of how physical parameters should be integrated into operator layers remains relatively unstudied.

Fourier Neural Operators. The FNO [7] performs spectral convolution through Fourier-space weights, enabling global spatial coupling and strong resolution transfer. Extensions include multi-resolution variants[6], geometry-aware formulations[4], and domain-specific adaptations used in forecasting and seismic inversion[5, 12]. These works improve spatial modeling but generally treat physical parameters as auxiliary inputs appended to the initial condition. As a result, the way parameters interact with spectral representations is rarely analyzed. Our work focuses on this aspect by examining how parameter signals can be injected at different points within the FNO computation.

Conditioning mechanisms. Conditioning strategies such as FiLM [13], channel-wise modulation, and input augmentation are widely used in generative modeling and supervised prediction. In PDE surrogate models, parameters are often introduced through input concatenation or global embeddings [16]. Recent studies have applied conditioning ideas to FNO-like architectures in specific applications. Zeng et al.[19] propose a conditional FNO for optical fiber propagation, and Zhu et al.[20] introduce FiLM and spectral gating inside FNO blocks as part of an equation-aware emulator. These approaches demonstrate the potential value of conditioning but integrate it with additional architectural changes tailored to particular domains. In contrast, our work isolates conditioning itself as the design variable and evaluates three representative mechanisms—local modulation, global channel scaling, and input concatenation—within an otherwise standard FNO architecture.

3 PROBLEM DEFINITION

We study parameter-dependent partial differential equations (PDEs) of the form

$$\mathcal{F}(u; \mathbf{p}) = 0,$$

where $u : \Omega \to \mathbb{R}^C$ is the solution field defined on a spatial domain $\Omega \subset \mathbb{R}^2$, and $\mathbf{p} \in \mathbb{R}^d$ denotes a vector of physical parameters such as advection velocities, viscosity, wave speeds, or forcing coefficients. In many scientific and engineering settings, the parameters $\mathbf{p}$ vary across instances, and the goal is to construct a surrogate model that predicts the corresponding solution fields efficiently.

Let u_0 be an initial condition and let u denote the resulting solution field at a fixed target time $t^\star$ after numerical integration of the PDE. We consider datasets of the form

$$\mathcal{D} = \{(u_0^{(i)}, \mathbf{p}^{(i)}, u^{(i)})\}_{i=1}^N,$$

where each sample is generated by solving the PDE under a specific set of parameters $\mathbf{p}^{(i)}$. Our objective is to learn an operator

$$\mathcal{G}_\theta : (u_0, \mathbf{p}) \mapsto \hat{u},$$

where $\hat{u}$ is the model prediction of the PDE solution at time $t^\star$. We optimize the network parameters θ by minimizing the loss function defined as the relative L_2 error between the predicted solution $\hat{u}$ and the exact solution u averaged over the training dataset.

We now extend this setting to time-dependent dynamics. Instead of observing only the solution at a single target time $t^\star$, we consider trajectories sampled on a uniform temporal grid. For each PDE instance, we observe the solution at times

$$t_k = k\Delta t, \quad k = 0, 1, \ldots, K,$$

so that each sample provides a sequence $\{u^{(i)}(t_k)\}_{k=0}^K$ together with the corresponding parameter vector $\mathbf{p}^{(i)}$. The learning problem is to construct a surrogate that uses the current state and parameters to predict the evolution of the system along this grid.

To make use of the temporal structure, we formulate the problem as learning a one-step operator

$$\mathcal{G}_\theta : (u(t_k), \mathbf{p}) \mapsto \hat{u}(t_{k+1}).$$

We train this operator on pairs $\left(u^{(i)}(t_k), \mathbf{p}^{(i)}\right) \mapsto u^{(i)}(t_{k+1})$ extracted from all trajectories. At inference time, we generate a rollout by applying $\mathcal{G}_\theta$ autoregressively. Starting from the initial condition $u(t_0)$, we compute

$$\hat{u}(t_1) = \mathcal{G}_\theta(u(t_0), \mathbf{p}), \hat{u}(t_2) = \mathcal{G}_\theta(\hat{u}(t_1), \mathbf{p}), \ldots, \hat{u}(t_K) = \mathcal{G}_\theta(\hat{u}(t_{K-1}), \mathbf{p}).$$

Rollout introduces an additional source of error growth beyond single-step prediction. Suppose that the relative error of a single step is bounded by ϵ, i.e.,

$$\frac{\|\mathcal{G}_\theta(u(t_k), \mathbf{p}) - u(t_{k+1})\|}{\|u(t_{k+1})\|} \approx \epsilon.$$

If local errors accumulate multiplicatively through repeated application of $\mathcal{G}_\theta$, the deviation factor after n rollout steps scales approximately as $(1 + \epsilon)^n$. Such error amplification is common in time-dependent PDEs where dynamics are sensitive to perturbations. This makes it necessary to minimize the one-step prediction error as much as possible to ensure stability over long horizons.

4 METHOD

We build on the FNO and study how physical parameters should be integrated. This section first recalls the baseline FNO architecture and then introduces three conditioning mechanisms that inject parameters at different stages of the architecture. In what follows, we focus on the two-dimensional case for clarity, while the same structure extends to any spatial dimension.

4.1 Baseline Fourier Neural Operator

Let $v_i \in \mathbb{R}^{B\times\Omega}$ denote the input field on a regular grid, where Ω is the shape of each sample, and let $\mathcal{F}$ be the 2D discrete Fourier transform applied channel-wise. A standard FNO layer computes

$$v_{i+1} = \sigma\big(\mathcal{F}^{-1}\big(R_\theta(\mathcal{F}(v_i))\big) + W v_i\big), \tag{1}$$

where R_θ multiplies a fixed set of low-frequency modes by learnable weights and filters out the higher modes, W is a pointwise 1×1 convolution, and σ is a nonlinear activation. The complete FNO architecture is composed of a lifting layer $\mathcal{P}$, a stack of L Fourier layers $\mathcal{K}_1,\dots,\mathcal{K}_L$, and a projection layer $\mathcal{D}$, forming the operator:

$$\mathcal{G}_\theta = \mathcal{D}\circ\mathcal{K}_L\circ\cdots\circ\mathcal{K}_1\circ\mathcal{P}.$$

Our first conditioning mechanism augments the network input, while the subsequent two modify the spectral layer defined in Eq. (1).

4.2 Input Concatenation

The first conditioning strategy injects the parameter vector directly at the input layer by concatenating it with the spatial field a. This keeps the internal Fourier blocks unchanged and serves as the architecture with the minimal modification.

Concatenation with the input field. The parameter vector $\mathbf{p}$ is spatially broadcasted to match the grid resolution and concatenated with the original input:

$$a_{\text{aug}} = \text{concat}(a,\ \mathbf{p}_{\text{broadcast}}).$$

The augmented field a_{aug} is then passed to a standard FNO:

$$u = \mathcal{G}_\theta(a_{\text{aug}}).$$

Operator interpretation. In this setting the parameters are treated as extra input channels that are constant in space. The network must infer how $\mathbf{p}$ affects the solution purely through the subsequent Fourier layers, without any explicit structure in the way $\mathbf{p}$ interacts with the spectral kernels.

4.3 Local Conditioning

This mechanism modifies the intermediate feature representation of each Fourier block through feature-wise affine modulation. This design introduces parameter dependence directly into the spectral path, allowing the operator to adjust its response for different physical regimes.

Parameter embedding. The physical parameter vector $\mathbf{p}$ is first expanded using a sinusoidal feature map [17, 18] with a fixed bank of frequencies and then processed by a learnable encoder η_{loc} (e.g., an MLP) to produce the modulation coefficients:

$$\mathbf{c}_{\text{loc}} = \eta_{\text{loc}}(\mathbf{p}) \in \mathbb{R}^{C_h},$$

where C_h is the hidden channel width.

Two-step spectral path with modulation. To allow conditioning to influence spectral computation at a deeper level, we replace the standard layer with a two-step spectral path. The first spectral transformation produces an intermediate state h_1:

$$h_1 = \mathcal{F}^{-1}\big(R_\theta^{(1)}(\mathcal{F}(v_i))\big),$$

where $R_\theta^{(1)}$ multiplies the Fourier modes by learned weights.

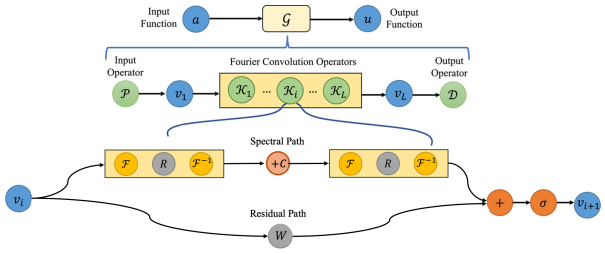

Figure 1: Illustration of the local conditioning mechanism. The physical parameters p are embedded into c, which controls feature-wise affine modulation inserted between two spectral transformations. The resulting operator is equivalent to a parameter-dependent convolution.

We then apply FiLM-style modulation based on $\mathbf{c}_{\text{loc}}$. This operation scales and shifts the features channel-wise:

$$h_{\text{mod}} = \alpha(\mathbf{c}_{\text{loc}}) \odot h_1 + \beta(\mathbf{c}_{\text{loc}}),$$

where $\alpha, \beta : \mathbb{R}^{C_h} \to \mathbb{R}^{C_h}$ are learned affine projections derived from the embedding.

The modulated representation flows into a second spectral block to produce the spectral output h_2:

$$h_2 = \mathcal{F}^{-1}\big(R_\theta^{(2)}(\mathcal{F}(h_{\text{mod}}))\big).$$

Operator interpretation. Since the modulation and the spectral transforms are linear in the feature space, the entire mapping forms a convolution whose kernel depends on $\mathbf{p}$. Formally:

Proposition 4.1. *Consider a spectral path*

$$v_i \xrightarrow{\mathcal{F}} R_\theta^{(1)} \xrightarrow{\mathcal{F}^{-1}} h_1 \xrightarrow{\text{FiLM}} h_{\text{mod}} \xrightarrow{\mathcal{F}} R_\theta^{(2)} \xrightarrow{\mathcal{F}^{-1}} h_2.$$

Then the overall mapping can be written as a parameter-dependent convolution

$$h_2 = v_i * f_{|\mathbf{p}} + b_{|\mathbf{p}},$$

where $f_{|\mathbf{p}}$ and $b_{|\mathbf{p}}$ are defined by the Fourier coefficients of $R_\theta^{(1)}$, $R_\theta^{(2)}$, and the FiLM parameters $\alpha(\mathbf{c}_{\text{loc}})$ and $\beta(\mathbf{c}_{\text{loc}})$.

A full derivation of Proposition 4.1 is provided in Appendix A. This mechanism yields the most expressive form of parameter injection: each hidden channel can independently reweight and shift its spectral activation based on $\mathbf{p}$.

4.4 Global Conditioning

We now introduce the global conditioning mechanism. Unlike the local strategy which inserts modulation between spectral transforms, this approach applies a global gating operation at the end of the linear stage of each FNO layer. The condition vector scales the amplitude of the features channel-wise, effectively reweighting the importance of different frequency modes and spatial features before they pass through the activation function.

Parameter embedding. Similar to the local conditioning setup, the physical parameters $\mathbf{p}$ are encoded into a global condition vector $\mathbf{c}_{\text{glob}} \in \mathbb{R}^{C_h}$.

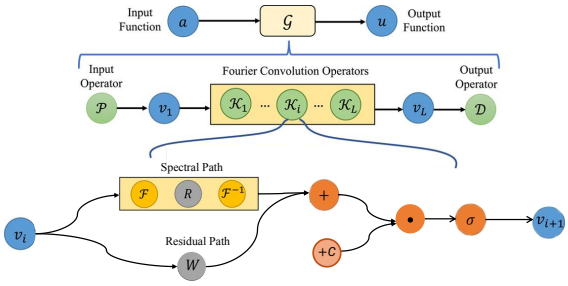

Figure 2: Illustration of the global conditioning mechanism. The physical parameters p are embedded into a global condition vector c that modulates the layer features through channel-wise multiplication. As shown in the zoomed view of the layer $\mathcal{K}_i$, this modulation is applied to the sum of the spectral and residual paths before the nonlinear activation σ.

Gated Fourier Layer. Let v_i be the input to the i-th layer. First, we compute the standard linear update, which is the sum of the spectral convolution and the residual path:

$$z_i = \mathcal{F}^{-1}\big(R(\mathcal{F}(v_i))\big) + W v_i.$$

Here, z_i represents the pre-activation features. To incorporate the physical parameters, we apply the condition vector $\mathbf{c}_{\text{glob}}$ via channel-wise multiplication (broadcasting over the spatial dimensions):

$$\tilde{z}_i = z_i \odot \mathbf{c}_{\text{glob}}.$$

Finally, the modulated features are passed through the activation function to produce the layer output:

$$v_{i+1} = \sigma(\tilde{z}_i).$$

Operator interpretation. This design mimics the branch-trunk architecture typical of DeepONet. Here, the FNO layer functions as a "trunk" that learns a fixed set of spatial basis features z_i, while the parameter encoder acts as a "branch" that outputs coefficients $\mathbf{c}_{\text{glob}}$. The element-wise multiplication combines these two streams, effectively allowing the physical parameters to reweight the spatial features learned by the spectral convolution. This acts as a dynamic gain controller where $\mathbf{p}$ determines the amplitude of specific frequency modes or spatial patterns without altering their fundamental shape.

4.5 Two-Stage Methodological Framework

Our study follows a two-stage methodological framework aimed at separating the effect of conditioning and evaluating its rollout performance under a refined training scheme.

Stage 1: comparison of conditioning mechanisms across PDE systems. In this stage, we train each conditioning mechanism. Each model uses the same backbone architecture, similar parameter counts, and the same spectral resolution. The goal is to determine, for each PDE system, which mechanism produces the strongest parameter-dependent operator.

Stage 2: refinement with enhanced training. In this stage, we take the best-performing mechanism for each benchmark and retrain it using the enhanced training scheme introduced in prior work [9], which uses iterative operator refinement with curriculum training.

4.6 Explanation of the Refinement Scheme

To clarify the enhanced training scheme introduced in [9], we briefly summarize the refinement mechanism. The method replaces pure one-step training with an iterative denoising objective that forces the model to correct its own prediction across multiple noise levels.

At refinement step $k = 0$, the model performs a standard one-step update:

$$u(t) \approx \mathcal{G}_\theta\big(0,\ u(t-\Delta t),\ 0,\ \mathbf{p}\big),$$

which recovers the classical MSE objective.

For $k > 0$, the ground-truth solution is perturbed by Gaussian noise whose variance decreases with the refinement index:

$$u_k(t) = u(t) + \varepsilon_k, \qquad \varepsilon_k \sim \mathcal{N}\big(0,\ \sigma_k^2 I\big), \qquad \sigma_k = \sigma_{\min}^{k/K}.$$

The training target becomes the noise itself:

$$\mathcal{G}_\theta\big(u_k(t),\ u(t-\Delta t),\ k,\ \mathbf{p}\big) \approx \varepsilon_k.$$

This denoising objective forces the operator to model solution components across the whole frequency spectrum, since different noise levels highlight different spatial frequencies of the signal.

During inference, the refinement loop iteratively improves an initial prediction. Starting from

$$\hat{u}_1(t) = \mathcal{G}_\theta\big(0,\ u(t-\Delta t),\ 0,\ \mathbf{p}\big),$$

each refinement step applies the inverse denoising update:

$$\hat{u}_{k+1}(t) = \hat{u}_k(t) - \mathcal{G}_\theta\big(\hat{u}_k(t),\ u(t-\Delta t),\ k,\ \mathbf{p}\big)\,\sigma_k, \qquad k = 1, \ldots, K.$$

As σ_k decreases, later iterations correct lower-amplitude frequency components, resulting in substantially improved long-time rollout stability.

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 PDE benchmarks. We evaluate the three conditioning mechanisms on four time-dependent PDE benchmarks. All datasets are generated on regular 64×64 spatial grids over the domain $[0, 2]^2$ with periodic boundaries, using Fourier pseudo-spectral solvers for spatial discretization and a fourth-order Runge–Kutta integrator in time. Below we summarize the governing equations and the specific physical parameters used for conditioning in each case.

Advection–Diffusion. This benchmark describes the transport and diffusion of a scalar field $u(x, y, t)$ under a spatially uniform but randomly sampled velocity field $v = (c_x, c_y)$ and a time-dependent source term:

$$\partial_t u = \nu \Delta u - v \cdot \nabla u + s(x, y, t),$$

with a fixed viscosity $\nu = 10^{-3}$. The velocity components (c_x, c_y) are sampled independently from $[-1, 1]$ and serve as the conditioning parameters, determining the direction and speed of the bulk transport.

Burgers. The Burgers benchmark models a two-component velocity field $u(x, y, t) = (u_1, u_2)$ governed by a viscous Burgers system:

$$\partial_t u + (u \cdot \nabla)u = \nu \Delta u.$$

The viscosity ν is sampled uniformly from $[0.01, 0.1]$ and used as the conditioning parameter. This coefficient controls the rate of energy dissipation and shock formation.

Elastic Waves. The elasticity benchmark simulates wave propagation of a displacement field $u(x, y, t) = (u_x, u_y)$ obeying a first-order formulation of the elastic wave equation:

$$\partial_t u = v, \qquad \partial_t v = (c_p^2 - c_s^2)\,\nabla(\nabla \cdot u) + c_s^2 \Delta u,$$

where v denotes velocity and (c_p, c_s) are the P-wave (compressional) and S-wave (shear) speeds. We treat (c_p, c_s), sampled from $[0, 1]^2$, as the conditioning parameters. These values determine the propagation speeds of the distinct compressional and shear wave modes relative to each other.

Navier–Stokes. For the incompressible flow benchmark, we consider the Navier–Stokes equations in vorticity form:

$$\partial_t \omega + u \cdot \nabla \omega = \nu \Delta \omega + f(x, y),$$

where ω is the scalar vorticity and u is recovered from a stream function via $\omega = \Delta \psi$. We fix the viscosity at $\nu = 10^{-3}$ and introduce a divergence-free forcing f built from low-frequency sinusoidal modes. The forcing amplitudes are controlled by two coefficients (α_x, α_y), sampled from $[0, 2]^2$, which serve as the conditioning parameters effectively influencing the intensity and structure of the driving force.

5.1.2 Model variants. We compare four models: the standard FNO without conditioning and the three conditioned variants. All models use the same hidden channel width of 32 and Fourier modes $(12, 12)$. For the local conditioning model, we use three blocks (each containing two spectral layers), while the other three models use six blocks (each containing one spectral layer), ensuring that all variants contain the same total number of spectral convolutions. All conditioning inputs are embedded into a 128-dimensional vector.

5.1.3 Training procedure. Models are trained using AdamW with a learning rate of 10^{-3} and cosine decay. Each model is trained until convergence. For both Stage 1 and Stage 2, we use the relative L_2 loss when $k = 0$; in Stage 2, we switch to MSE loss when $k > 0$ to match the noise distribution.

5.2 Scaling Law Analysis: Effect of Training Set Size

To understand how the proposed conditioning mechanisms behave as the amount of training data increases, we conduct a scaling study on the local conditioning model, which is the strongest performer in Stage 1 for most benchmarks. We fix all hyperparameters and architecture settings, and vary the number of training samples as

$$N \in \{1000,\ 2000,\ 4000,\ 8000\}.$$

For each value of N, we train the model to convergence and evaluate on the same held-out test set.

Figure 3 summarizes the scaling behavior. The error decreases in an approximately power-law manner as the sample size increases,

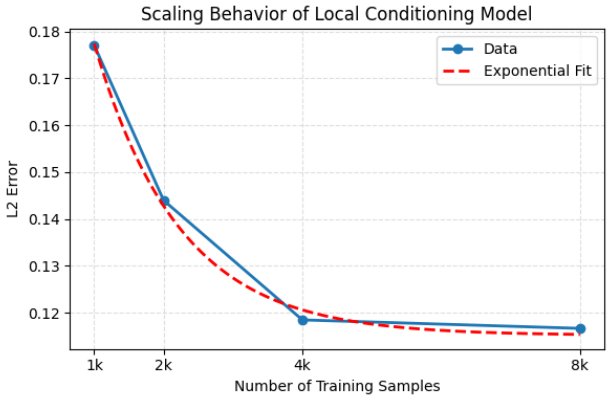

Figure 3: Scaling behavior of the local conditioning model as the number of training samples increases. Performance saturates at approximately 4000 samples, indicating a model capacity limit under fixed architecture size.

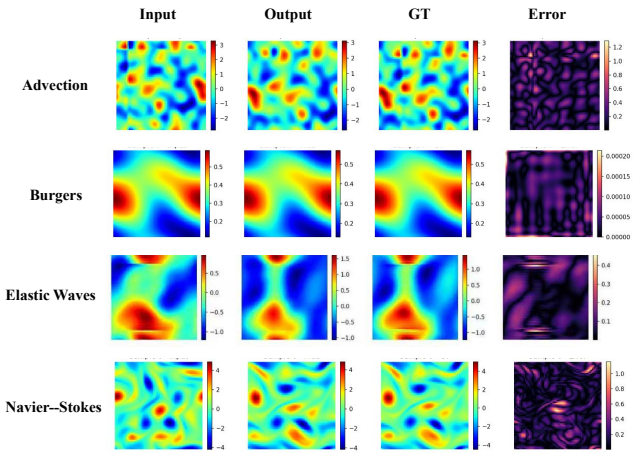

Figure 4: One-step predictions on all four PDE benchmarks using the best-performing conditioned FNO variant for each task. For every benchmark, we show the input field at the rollout start time, the predicted field, the ground-truth field at the same time, and the L^2 error.

and by 4000 samples the curve is already close to its limit. This suggests that, for this PDE family, the local conditioning mechanism reaches its model capacity limit at moderate dataset sizes, and additional data does not significantly improve generalization unless model width or depth is increased.

5.3 Results Across PDE Benchmarks

Table 1 summarizes the Stage 1 performance. Across all systems, explicit conditioning yields improvements over the unconditioned baseline, confirming the effectiveness of the proposed mechanisms.

Advection–Diffusion. This benchmark is the most sensitive to parameter shifts: the unconditioned FNO exhibits a large error of 0.9430, confirming that an operator trained on a limited velocity

Method	Adv.	Burg.	Elast.	N.–S.
Local	**0.1050**	**0.0016**	0.3232	0.1185
Global	0.1262	0.0019	**0.2967**	0.1359
Input	0.1343	0.0023	0.2979	**0.1175**
Vanilla	0.9430	0.0369	0.5057	0.1490

Table 1: Stage 1 comparison of conditioning strategies (L^2 error ↓).

distribution generalizes poorly when the advective field changes. All three conditioning mechanisms significantly reduce the error, with local conditioning achieving the best performance at 0.1050. Global and input-level conditioning follow with moderately higher errors, indicating that while they capture the bulk flow direction, the pixel-wise affine modulation of local conditioning better resolves the phase shifts induced by varying velocities.

Burgers. For Burgers' equation, viscosity controls the shock formation and dissipation rate. All conditioned models outperform the unconditioned baseline (0.0369) by a large margin. The best-performing mechanisms—local (0.0016) and global (0.0019)—achieve nearly identical accuracy. This reflects the global nature of viscosity: channel-wise spectral scaling already captures most of the necessary damping dynamics, so the additional expressivity of local modulation yields diminishing returns. Input-level conditioning is effective (0.0023) but remains the weakest, consistent with its limited ability to interact with the internal spectral features.

Elastic Waves. Because the elasticity system is a linear PDE, parameter dependence enters through the scaling of wave speeds (c_p, c_s). Global conditioning, which applies channel-wise scaling, aligns naturally with this scaling behavior and achieves the lowest error (0.2967). Input concatenation follows closely (0.2979), while local conditioning performs slightly worse (0.3232). The significant gap relative to the baseline (0.5057) underscores that parameter awareness is essential for capturing the correct propagation speeds.

Navier–Stokes. Input-level conditioning performs the best on the Navier–Stokes benchmark (0.1175). Unlike the other cases, the conditioning parameters here control the external forcing term $f(x, y)$, which appears as an additive source in the equation. Input concatenation is structurally well-suited for this, as it allows the network to construct the spatial forcing field immediately at the input layer. In contrast, the internal dynamics (advection and diffusion) remain governed by fixed coefficients, which may explain why modulating the internal spectral weights (Local/Global) offers no advantage over simple input augmentation.

5.4 Refinement Results

At this stage, we evaluated the refinement model on the Navier–Stokes dataset and observed that the results were unsatisfactory. As shown in Figure 5, varying the refinement depth k leads to almost no change in the model's prediction. Inspecting the training process reveals that the loss remains around 0.1, indicating a relative error level above 10% and a clear sign of underfitting. This behavior may stem from the highly variable conditioning in this system, which makes it difficult for the model to learn a stable correction direction. Since the refinement steps do not effectively reduce the error, the model eventually becomes unstable and breaks down at about the sixth time step, as shown in Figure 6.

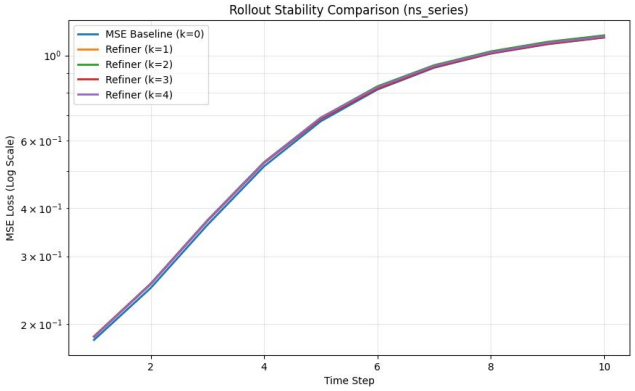

Figure 5: Across refinement depths $k = 1$ to $k = 4$, the error curves almost overlap with the baseline model ($k = 0$), indicating that refinement does not improve long-horizon stability for this system. The highly advective dynamics dominate the evolution, and the correction steps fail to reduce accumulated rollout error.

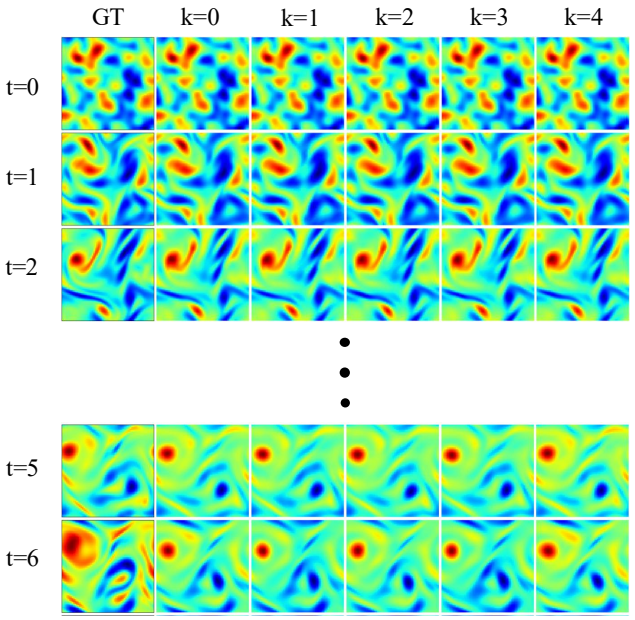

Figure 6: Across time steps $t = 0$ to $t = 6$, the predicted vorticity fields remain nearly identical for all refinement levels, showing no visible improvement over the baseline. This confirms that refinement does not alter the trajectory for this system, consistent with the dominance of the advective dynamics and the inability of the correction steps to reduce accumulated errors.

6 CONCLUSION AND DISCUSSION

This work provides a systematic comparison of how physical parameters should be incorporated into Fourier Neural Operators. By evaluating local modulation, global channel-wise scaling, and input-level concatenation across four PDE systems using identical backbones, we show that explicit conditioning is essential for parameter-dependent operator learning. Local modulation yields the strongest accuracy for parameter-sensitive transport systems such as Advection and Burgers, while global scaling is most effective for linear elasticity, where its structure aligns naturally with the PDE. Input-level concatenation offers a simple alternative that remains competitive for Navier–Stokes but performs less effectively on transport-dominated tasks. In the second stage, we applied the PDE-Refiner training scheme to the best method for each benchmark. Refinement improves certain systems but has limited effect on Navier–Stokes, reflecting the dominance of advective dynamics and the difficulty of learning stable correction steps. Overall, our study clarifies the roles of different conditioning strategies and offers practical guidance for designing parameter-aware neural operators.

REFERENCES

[1] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. 2023. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*. PMLR, 12556–12569.

[2] Xun Huang and Serge Belongie. 2017. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. arXiv:1703.06868 [cs.CV] https://arxiv.org/abs/1703.06868

[3] Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. 2023. Multi-grid tensorized fourier neural operator for high-resolution pdes. *arXiv preprint arXiv:2310.00120* (2023).

[4] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2023. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research* 24, 89 (2023), 1–97.

[5] Bian Li, Hanchen Wang, Shihang Feng, Xiu Yang, and Youzuo Lin. 2023. Solving seismic wave equations on variable velocity models with Fourier neural operator. *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), 1–18.

[6] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. 2023. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research* 24, 388 (2023), 1–26.

[7] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895* (2020).

[8] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485* (2020).

[9] Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. 2023. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems* 36 (2023), 67398–67433.

[10] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence* 3, 3 (2021), 218–229.

[11] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. arXiv:1411.1784 [cs.LG] https://arxiv.org/abs/1411.1784

[12] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. 2022. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214* (2022).

[13] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[14] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. 2020. Learning mesh-based simulation with graph networks. In *International conference on learning representations*.

[15] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752 [cs.CV] https://arxiv.org/abs/2112.10752

[16] Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. 2023. Learning neural pde solvers with parameter-guided channel attention. In *International Conference on Machine Learning*. PMLR, 33448–33467.

[17] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems* 33 (2020), 7537–7547.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[19] Yuanhang Zeng, Xiangyu Ma, Qingzhe Cui, Zhu Guangzhi, and Xiao Zhu. 2025. Conditional Fourier neural operator for prediction of ultrafast nonlinear dynamics in optical fibers. *Journal of Lightwave Technology* PP (01 2025), 1–8. https://doi.org/10.1109/JLT.2025.3633748

[20] Qian-Ze Zhu, Paul Raccuglia, and Michael P Brenner. 2025. Generalizing PDE Emulation with Equation-Aware Neural Operators. *arXiv preprint arXiv:2511.09729* (2025).

SUPPLEMENTARY MATERIAL

A PROOF OF PROPOSITIONS

Theorem A.1. *Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse. Consider a spectral path that applies two consecutive spectral transformations with a FiLM modulation in between:*

$$\mathbf{h} = \mathcal{F}^{-1}\big(M_1(\xi)\,\mathcal{F}\{x\}\big), \qquad \tilde{\mathbf{h}} = \alpha(\mathbf{c})\,\mathbf{h} + \beta(\mathbf{c}), \qquad y = \mathcal{F}^{-1}\big(M_2(\xi)\,\mathcal{F}\{\tilde{\mathbf{h}}\}\big),$$

where $M_1(\xi)$ and $M_2(\xi)$ are frequency-wise multipliers, and $\alpha(\mathbf{c}), \beta(\mathbf{c})$ are per-channel scalars broadcast over space (i.e., spatially constant), with

$$\mathbf{c} = [\,\gamma(\tau),\ \eta(\mathbf{p})\,].$$

Then y can be written as a condition-dependent convolution plus (optional) bias:

$$y = x * f_{|\mathbf{c}} + b_{|\mathbf{c}},$$

with

$$f_{|\mathbf{c}} = \mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\big), \qquad b_{|\mathbf{c}} = \beta(\mathbf{c})\,\mathcal{F}^{-1}\big(M_2(\xi)\big).$$

In particular, if $\beta(\mathbf{c}) \equiv 0$, then

$$y = x * f_{|\mathbf{c}}, \qquad f_{|\mathbf{c}} = \mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\big).$$

Proof. By definition of the first spectral transformation,

$$\mathbf{h} = \mathcal{F}^{-1}\big(M_1(\xi)\,\mathcal{F}\{x\}\big).$$

Applying FiLM with spatially constant scalars $\alpha(\mathbf{c}), \beta(\mathbf{c})$ yields

$$\tilde{\mathbf{h}} = \alpha(\mathbf{c})\,\mathbf{h} + \beta(\mathbf{c}).$$

Taking the Fourier transform and using linearity,

$$\mathcal{F}\{\tilde{\mathbf{h}}\} = \alpha(\mathbf{c})\,\mathcal{F}\{\mathbf{h}\} + \beta(\mathbf{c})\,\delta(\xi) = \alpha(\mathbf{c})\,M_1(\xi)\,\mathcal{F}\{x\} + \beta(\mathbf{c})\,\delta(\xi),$$

where $\delta(\xi)$ is the Dirac delta at $\xi = 0$. Passing this through the second spectral transformation,

$$\begin{aligned} y &= \mathcal{F}^{-1}\Big(M_2(\xi)\big[\alpha(\mathbf{c})\,M_1(\xi)\,\mathcal{F}\{x\} + \beta(\mathbf{c})\,\delta(\xi)\big]\Big) \\ &= \underbrace{\mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\,\mathcal{F}\{x\}\big)}_{\text{term (A)}} + \underbrace{\mathcal{F}^{-1}\big(\beta(\mathbf{c})\,M_2(\xi)\,\delta(\xi)\big)}_{\text{term (B)}}. \end{aligned}$$

Term (A) equals a convolution by the convolution theorem:

$$\text{(A)} = x * \mathcal{F}^{-1}\big(\alpha(\mathbf{c})\,M_2(\xi)M_1(\xi)\big) = x * f_{|\mathbf{c}}.$$

Term (B) reduces to a condition-dependent bias:

$$\text{(B)} = \beta(\mathbf{c})\,\mathcal{F}^{-1}\big(M_2(\xi)\big) = b_{|\mathbf{c}}.$$

Combining the two terms gives

$$y = x * f_{|\mathbf{c}} + b_{|\mathbf{c}}.$$

If $\beta(\mathbf{c}) \equiv 0$, the bias term vanishes, yielding the pure convolution form. □