Neural Operator-based Curriculum Learning for Physics-Informed Neural Networks

Lingshi Meng **Haosen Shi** **Sinno Jialin Pan**
The Chinese University of Hong Kong
Hong Kong SAR, China

Abstract

In this paper, we tackle the critical failure modes of Physics-Informed Neural Networks (PINNs), such as spectral bias, which lead to poor convergence on complex PDEs. We identify two key shortcomings in existing curriculum learning methods for PINNs: unreliable knowledge transfer between stages and a reliance on manual, ad-hoc curriculum design. To overcome these limitations, we present Neural Operator-based Curriculum Learning (NOCL), a unified framework that combines Neural Tangent Kernel (NTK)-guided curriculum design with neural operators for robust, dynamic knowledge transfer across curriculum stages. By dynamically training neural operators and filtering data for PINN initialization, our approach ensures scalable and effective learning across progressively difficult tasks. Experiments verify that NOCL leads to marked gains in convergence and generalization relative to prior methods, resulting in substantially better performance on test suite.

1 Introduction

Neural networks have revolutionized scientific computing by introducing data-driven approaches for solving Partial Differential Equations (PDEs). Conventional numerical methods, such as the Finite Element Method (Belytschko et al., 2014) and Finite Difference Method (LeVeque, 2007), rely on spatial discretization and face exponentially growing computational and memory costs in high dimensions. Neural networks overcome this limitation using a mesh-free, scalable approach (Lagaris et al., 1998; Han et al., 2018). A prominent example is Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019; Karniadakis et al., 2021), which encode physical laws directly into the loss function. This framework effectively solves both forward and inverse problems, even with limited data, and has seen rapid adoption across scientific fields (Mao et al., 2020; Cai et al., 2021; Raissi et al., 2019).

However, training PINNs poses a particularly challenging optimization problem, often suffering from convergence failures in certain parameter regimes (Yan & He, 2024; Krishnapriyan et al., 2021; Chen et al., 2024). To address this, recent studies (Krishnapriyan et al., 2021; Bekele, 2025) have integrated curriculum learning strategies, which stabilize training by starting from easier problems and gradually increasing difficulty. These include defining a curriculum over the PDE's parameter space to progressively increase complexity, as well as causal learning, which structures the problem into sequential temporal stages (Guo et al., 2025; Wang et al., 2024b). While curriculum learning can improve convergence in some settings, it may fail to improve performance precisely in the challenging parameter regimes (Monaco & Apiletti, 2023).

Two fundamental challenges must be overcome to address these limitations effectively (Monaco & Apiletti, 2023). First, algorithmically selecting an optimal parameter-based curriculum for a given PDE system is highly non-trivial. Second, transferring across curriculum stages (i.e., across parameter regimes) is difficult, since PDE solutions can change sharply under small parameter perturbations (Hanna et al., 2024). In Section 4.1, we empirically demonstrate that previous curriculum

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

learning methods result in significant error. This underscores the critical need for a reliable knowledge transfer mechanism across curriculum stages.

In this paper, we propose a novel Neural Operator-based Curriculum Learning framework (NOCL) for training PINNs. Our approach leverages the complementary strengths of the Neural Tangent Kernel (NTK) theory for curriculum design and neural operators for robust knowledge transfer.

Our framework is motivated by two key insights:

1. **Neural Operators for Transfer:** Neural operators learn mappings between function spaces, exhibiting strong generalization across PDE instances (Li et al., 2021, 2024; Lu et al., 2021). This is ideal for transferring information between curriculum stages, providing a continuous and informative prior. However, although supervised training can yield cross-instance generalization, it typically relies on large, pre-computed datasets; when such data are scarce, performance in out-of-distribution regions cannot be guaranteed.
2. **NTK for Curriculum Design:** NTK analysis characterizes training dynamics (Tan & Liu, 2024; Jacot et al., 2018), and its eigenvalue spectra correlate with training difficulty for given PDE parameters (Wang et al., 2022), offering a foundation for systematic curriculum design. However, the variability of NTK spectra across different PDE systems necessitates a universal metric for quantifying prediction difficulty.

To improve the robustness of curriculum learning, we develop a strategy that dynamically trains the neural operator alongside the curriculum, augmented by a data-filtering algorithm to stabilize training. For curriculum design, we introduce a novel, universal metric, NTK eigenvalue variance, to consistently quantify PDE difficulty. Our **key contributions** are summarized as follows:[1]

1. **Identification of Pathological Failure in PINN Curriculum Learning:** We empirically demonstrate that conventional curriculum learning, which directly transfer model parameters across stages, can be ineffective and even detrimental, leading to catastrophic training failure in specific parameter regimes.
2. **Robust Knowledge Transfer via Neural Operators:** We introduce neural operators as a tool for knowledge transfer across curriculum stages. Our method ensures scalability via dynamic operator updates and data filtering, thereby mitigating parameter sensitivity, while remaining compatible with other PINN architectures and training tricks.
3. **Principled Curriculum Design via NTK Analysis:** We establish a principled curriculum design methodology by computing the variance of NTK eigenvalues across PDE parameters. This metric reliably reflects training difficulty, enabling NTK-guided construction of effective curricula, as validated by our experiments.

2 Problem Setup

Consider a general PDE system defined on a domain $\Omega \subset \mathbb{R}^d$ with boundary $\partial\Omega$:

$$\mathcal{L}[u(\mathbf{x})] = f(\mathbf{x}),\ \mathbf{x} \in \Omega,\ \text{and}\ \ \mathcal{B}[u(\mathbf{x})] = g(\mathbf{x}),\ \mathbf{x} \in \partial\Omega,$$

where $\mathcal{L}$ denotes the differential operator, $\mathcal{B}$ represents boundary conditions, f and g are known source terms, and u is an unknown solution function.

PINNs employ a deep neural network $\hat{u}(\mathbf{x};\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ to approximate the true solution $u(\mathbf{x})$. The network architecture typically comprises multiple fully connected layers with nonlinear activation functions, enabling the model to capture complex solution patterns across the domain. The training objective combines multiple physics-informed loss components to ensure the neural network satisfies both the governing equations and boundary constraints.

The Physics Loss enforces the PDE residual at interior points: $\mathcal{L}_{\text{phys}}(\boldsymbol{\theta}) = \frac{1}{N_p}\sum_{i=1}^{N_p} |\mathcal{L}[\hat{u}(\mathbf{x}_i;\boldsymbol{\theta})] - f(\mathbf{x}_i)|^2$, where $\{\mathbf{x}_i\}_{i=1}^{N_p}$ are collocation points sampled from Ω, and N_p denotes the number of physics points.

[1] Code is available at github.com/windindicator/NOCL-NeurIPS2026.

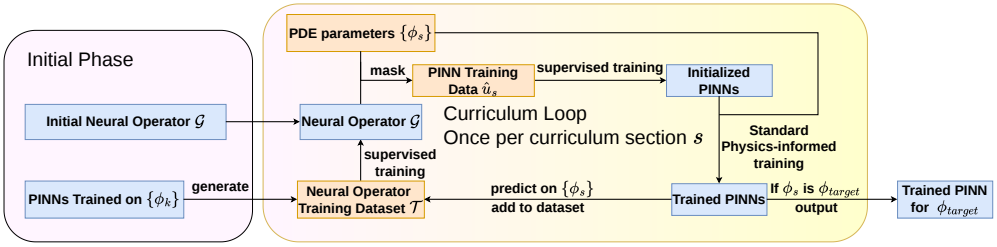

Figure 1: This figure illustrates the Neural Operator-based Curriculum Learning (NOCL). Orange denotes datasets, and blue denotes neural network models. The Initial Phase first trains PINNs on initial PDE parameter subsets ϕ_k to generate a training dataset $\mathcal{T}$ for the neural operator $\mathcal{G}$. The Curriculum Loop executes per curriculum section: neural operator $\mathcal{G}$ learns parameter-to-solution mapping via supervised training on $\mathcal{T}$; for PDE parameters ϕ_s in this Loop, predicted solutions from $\mathcal{G}$ undergo mask algorithm filtering to select low-residual points as quality initialization for PINN training; after physics-informed training, new solutions augment $\mathcal{T}$ to enhance the neural operator. When processing the target parameter ϕ_{target}, the algorithm outputs the final trained PINN.

The Boundary Loss ensures compliance with boundary conditions: $\mathcal{L}_{\text{bound}}(\boldsymbol{\theta}) = \frac{1}{N_b}\sum_{j=1}^{N_b} |\mathcal{B}[\hat{u}(\mathbf{x}_j;\boldsymbol{\theta})] - g(\mathbf{x}_j)|^2$, where $\{\mathbf{x}_j\}_{j=1}^{N_b}$ are boundary points, and N_b represents the number of boundary points.

The Composite Loss is a weighted sum of these two losses:

$$\mathcal{L}_{\text{total}}(\boldsymbol{\theta}) = \lambda_r \mathcal{L}_{\text{phys}}(\boldsymbol{\theta}) + \lambda_b \mathcal{L}_{\text{bound}}(\boldsymbol{\theta}), \tag{1}$$

where λ_r and λ_b are hyperparameters that balance the relative importance of physics and boundary constraints. The optimization process minimizes $\mathcal{L}_{\text{total}}(\boldsymbol{\theta})$ using gradient-based methods.

3 Methodology

3.1 Neural Operator-based Curriculum Learning

Given an input function $u(x)$ defined on a domain V, a neural operator G_θ (parameterized by θ) is defined to approximate the true operator G that maps $u(x)$ to another function $v(y)$ defined on a (possibly different) domain K:

$$G(u) \approx G_\theta(u) : u \mapsto v. \tag{2}$$

In Appendix A, we further show that for two common PDE classes in PINNs, parameterized elliptic equations (e.g., Poisson / diffusion) and parabolic equations (e.g., advection–diffusion/reaction–diffusion), under the stated assumptions, the following corollary holds.

Corollary 3.1 (Neural-operator generalization over the PDE parameter space)**.** *For any PDE parameter $\phi \in \Phi$, let $u_\phi := S(\phi)$ denote the exact solution (elliptic or parabolic, depending on the PDE class). Then the neural operator prediction satisfies*

$$\|G_\theta(\phi) - u_\phi\| \le \varepsilon_{\text{op}} + (L_G + L_S)\ \text{dist}\big(\phi, \text{supp}(\mu_\Phi)\big), \tag{3}$$

where ε_{op} is the (uniform) on-support operator approximation error on $\text{supp}(\mu_\Phi)$, L_G and L_S are Lipschitz constants of G_θ and the solution operator S, respectively, and

$$\text{dist}\big(\phi, \text{supp}(\mu_\Phi)\big) := \inf_{\tilde{\phi} \in \text{supp}(\mu_\Phi)} \|\phi - \tilde{\phi}\|_{\mathbb{R}^p}.$$

The norm $\|\cdot\|$ is the problem-dependent function-space norm: for elliptic problems, we use $\|\cdot\| = \|\cdot\|_{H_0^1(\Omega)}$; for parabolic problems, we use $\|\cdot\| = \|\cdot\|_{X_T}$, where

$$X_T := L^2(0,T;H_0^1(\Omega)) \cap C([0,T];L^2(\Omega)), \text{ and } \|v\|_{X_T} := \|v\|_{L^2(0,T;H_0^1(\Omega))} + \|v\|_{C([0,T];L^2(\Omega))}.$$

Proof. See the proof of Corollary 3.1 in Appendix A.

This bound provides a worst-case error guarantee; in practice, neural operators generalize well beyond the training support, enabling reliable extrapolation over the PDE parameter space and thereby serving as an effective transfer tool for curriculum learning in PINNs.

The overall procedure of NOCL is illustrated in Figure 1 and detailed in Algorithm 1. It comprises two main phases: an initialization phase and a curriculum-loop phase. The initialization phase bootstraps the process by training PINNs directly on a set of initial PDE parameters, typically chosen to facilitate training. After PINN training, we save its predictions on a predefined grid. Each saved data point forms a pair of PDE parameters and their corresponding solution snapshot. This collection of pairs constitutes the initial training dataset for the neural operator (typically a Fourier Neural Operator, FNO (Li et al., 2021)), which is initialized in this phase.

In the curriculum loop phase, the target PDE parameters are partitioned into a curriculum of progressively more difficult stages. For each stage: (i) the neural operator is trained in a supervised manner on the current dataset to learn the parameter-to-solution mapping; (ii) for each new PDE parameter, the trained neural operator predicts the corresponding solution function. The resulting $(x, u(x))$ pairs are used for supervised pretraining of the PINN; (iii) PINN continues training using the physics-informed loss defined in (1); (iv) finally, after PINN training is complete, its predictions are evaluated on the grid and added to the neural operator's training set, enhancing generalization for subsequent stages.

Algorithm 1 Neural Operator-based CL

Require: Curriculum parameters $\{\phi_k\}_{k=1}^{M}$, mask ratio τ
Ensure: Trained PINN for target PDE parameters

Step 1: Initialization Phase
Initialize neural operator $\mathcal{G}$ (typically FNO)
Initialize training set $\mathcal{T} = \emptyset$
for each parameter ϕ_k in initialization subset **do**
  Train PINN on PDE with parameters ϕ_k
  Generate predictions $u_k(\mathbf{x})$ on grid points
  Add $(\phi_k, u_k(\mathbf{x}))$ to $\mathcal{T}$
end for

Step 2: Curriculum Loop Phase
for each curriculum section s **do**
  Train neural operator $\mathcal{G}$ on training set $\mathcal{T}$
  for each parameter ϕ_s in section s **do**
    Prediction using neural operator: $\hat{u}_s = \mathcal{G}(\phi_s)$
    Compute PDE residual error by finite differences: $R(\mathbf{x}) = \mathcal{L}[\hat{u}_s(\mathbf{x})] - f(\mathbf{x})$
    Apply mask filtering: retain top τ fraction of points with smallest $|R(\mathbf{x})|$
    Initialize PINN with filtered predictions
    Train PINN using PDE loss (1)
    Generate predictions on grid points: $u_s(\mathbf{x})$
    Add $(\phi_s, u_s(\mathbf{x}))$ to $\mathcal{T}$
  end for
end for
Trained PINN for $\boldsymbol{\phi}_{\text{target}}$

Note that neural operator predictions can be inaccurate due to limited training data and resolution dependencies. To ensure only high-quality data initializes the PINN, we propose a mask-based filtering algorithm. This algorithm computes the PDE residual at each collocation point using finite differences. A mask then selects only the top–τ fraction of points with the smallest residuals, filtering out regions where predictions deviate significantly from the physics. This step is critical for providing a physically consistent starting point for PINN training. While our experiments demonstrate the effectiveness of this filtering strategy, choosing an optimal mask ratio τ remains a direction for future investigation.

Our algorithm offers several key advantages over methods that directly transfer model parameters across stages (denoted as CL in the sequel) (Krishnapriyan et al., 2021). First, neural operators learn the underlying solution operator, allowing them to make informed predictions for new parameters, even when the solution space changes significantly. This makes robust knowledge transfer possible. Second, the residual-based filtering prevents the propagation of unphysical errors, mitigating the overfitting and sensitivity issues common in parameter-transfer methods. Third, once trained, the neural operator and its dataset can be reused for new parameter configurations without restarting the entire curriculum, offering a substantial computational advantage over traditional methods that must repeat the process from scratch. As demonstrated in Section 4, NOCL achieves better performance than directly transferring parameters with standard curriculum designs, highlighting its efficiency.

3.2 NTK-based Curriculum Design

In this section, we propose a novel and principled curricula design method to prevent initialization failures. We propose using NTK eigenvalue analysis for systematic curriculum design. The NTK matrix Jacot et al. (2018) $K_{\boldsymbol{\theta}}$ for a neural network with parameters $\boldsymbol{\theta}$ is defined as $K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} f(\mathbf{x}'; \boldsymbol{\theta}) \rangle$, where $f(\mathbf{x}; \boldsymbol{\theta})$ is the neural network output and $\nabla_{\boldsymbol{\theta}} f$ is the parameter gradient. Training difficulties often arise from imbalanced NTK eigenvalues, as high-frequency solution components typically correspond to larger eigenvalues in the NTK spectrum (Wang et al., 2022). While this suggests the NTK condition number could indicate training difficulty, we find it unsuitable for our purpose. Specifically, the condition number is defined as $\kappa(K_{\boldsymbol{\theta}}(\boldsymbol{\phi})) = \frac{\lambda_{\max}(K_{\boldsymbol{\theta}}(\boldsymbol{\phi}))}{\lambda_{\min}(K_{\boldsymbol{\theta}}(\boldsymbol{\phi}))}$, where $\lambda_i(K_{\boldsymbol{\theta}}(\boldsymbol{\phi}))$'s are the eigenvalues of the NTK matrix for PDE parameters $\boldsymbol{\phi}$, and $\lambda_{\max}(K_{\boldsymbol{\theta}}(\boldsymbol{\phi})) := \max_i \lambda_i(K_{\boldsymbol{\theta}}(\boldsymbol{\phi}))$, $\lambda_{\min}(K_{\boldsymbol{\theta}}(\boldsymbol{\phi})) := \min_i \lambda_i(K_{\boldsymbol{\theta}}(\boldsymbol{\phi}))$. Using the NTK condition number for curriculum design is problematic: minimum eigenvalues often decay to zero with increasing collocation points, rendering the condition number unstable, while relying solely on maximum eigenvalues ignores the spectral imbalance that dictates training difficulty. To overcome these issues, we propose the variance of the NTK eigenvalues as a robust curriculum metric: $\mathcal{D}(\boldsymbol{\phi}) = \mathrm{Var}(\lambda_i^{\mathrm{res}}(K_{\boldsymbol{\theta}}(\boldsymbol{\phi}))) + \mathrm{Var}(\lambda_i^{\mathrm{bc}}(K_{\boldsymbol{\theta}}(\boldsymbol{\phi})))$, where λ_i^{res} and λ_i^{bc} are the eigenvalues of the NTK matrices computed using gradients of the residual (physics) and boundary loss terms, respectively.

The rationale for this design is that the NTK spectrum is known to decay rapidly (Wang et al., 2022). Motivated by this empirical observation, we model the NTK spectrum by a normalized monotone decay family with fixed effective rank and fixed trace, while explicitly allowing the presence of invalid eigenvalues that are numerically vanishing (treated as 0). Under this model, the eigenvalue variance computed on the full spectrum (including the vanishing tail) is a principled and robust proxy of spectral imbalance: it preserves the same monotone relation with the effective condition number, yet avoids the instability caused by invalid near-zero eigenvalues. Formally, Theorem B.1 in Appendix B establishes the following guarantee.

Theorem 3.2 (Monotone relation between eigenvalue variance and effective condition number)**.** *Under the spectral model in Appendix B, i.e., the NTK spectrum admits* $\widetilde{\Lambda}(\boldsymbol{\phi}) = \{\lambda_1 \geq \cdots \geq \lambda_m > 0, \underbrace{0, \ldots, 0}_{k}\}$ *with fixed* $m \geq 2$*,* $k \geq 0$*, fixed trace* $S = \sum_{i=1}^{m} \lambda_i$*, and the effective eigenvalues follow the rapid-decay family* (10) *for some concentration parameter* $\alpha = \alpha(\boldsymbol{\phi}) \geq 0$*, there exists a strictly increasing function* F *(depending only on* m, k, S*) such that*

$$\mathrm{Var}\big(\widetilde{\Lambda}(\boldsymbol{\phi})\big) = F\big(\kappa_{\mathrm{eff}}\big(K_{\boldsymbol{\theta}}(\boldsymbol{\phi})\big)\big)\,, \quad \kappa_{\mathrm{eff}} > 1, F'(\kappa) > 0, \tag{4}$$

where $\kappa_{\mathrm{eff}}(K_{\boldsymbol{\theta}}(\boldsymbol{\phi})) := \lambda_1/\lambda_m$ *is the effective condition number defined on the effective spectrum.*

Proof. See Appendix B.

Consequently, the variance of the NTK eigenvalues provides a more robust measure of spectral imbalance: The metric avoids being dominated by invalid zero-valued eigenvalues that can arise when too many samples are used, while still capturing the same spectral-imbalance behavior equivalent to the effective condition number on the non-vanishing spectrum. In Section 4.3, we empirically validate that $\mathcal{D}(\boldsymbol{\phi})$ is well aligned with practical training difficulty and yields reliable curricula across PDE parameters in our experiments.

Our curriculum design procedure is as follows. We begin by constructing a candidate set of curricula for the relevant PDE parameters. In our experiments, we adopt a leave-one-parameter-out protocol: when varying a PDE parameter to form the curriculum, we fix all other parameters at their target values and vary only one component of $\boldsymbol{\phi}$ at a time, typically sampling parameter values along a path from zero to the target value ϕ_{target}. Note that this is not a requirement of our method—more generally, $\boldsymbol{\phi}$ can vary along an arbitrary multi-parameter path, allowing multiple parameters to change simultaneously. For each sampled parameter value $\boldsymbol{\phi}_i$, we compute the corresponding NTK eigenvalue variance $\mathcal{D}(\boldsymbol{\phi}_i)$. We then filter this set, excluding any samples where the variance exceeds the target's variance, $\mathcal{D}(\boldsymbol{\phi}_{\text{target}})$. This ensures the curriculum consists only of scenarios that are simpler than or equally complex to the target.

Subsequently, we analyze the variance trends across the filtered candidate parameters. The parameter exhibiting the largest incremental change in variance is selected as the primary curriculum parameter. The curriculum for this parameter is then defined by the sequence of values where $\mathcal{D}(\boldsymbol{\phi}) \leq \mathcal{D}(\boldsymbol{\phi}_{\text{target}})$,

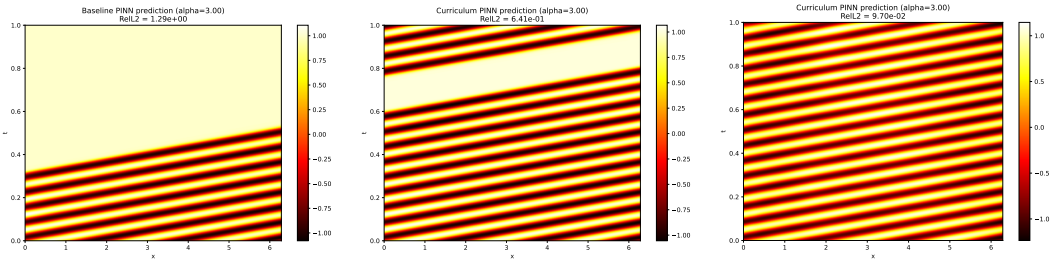

Figure 2: Visualization results for convection equation with varying initial condition parameter $\alpha = 3$. From left to right: baseline PINN predictions, CL PINN predictions, NOCL PINN predictions.

guaranteeing a progressive increase in difficulty. A key advantage of this NTK-based analysis is its ability to design effective curricula even when training difficulty does not increase monotonically with the parameter. In such cases of non-monotonic increase, the initial stage of the curriculum is not blindly set to a value near zero. Instead, we select the parameter region with the smallest variance, $\min \mathcal{D}(\boldsymbol{\phi})$, as the starting point, ensuring the learning process begins with the genuinely easiest instance. Finally, we compute the NTK only once under random initialization; therefore, the computational overhead of the NTK eigenvalue variance analysis in NOCL is negligible. A limitation is that the curriculum paths to select are still manually specified, and fully automated curriculum generation remains an important direction for future work.

4 Experiments on PDEs

To enable a more thorough comparison with CL (Krishnapriyan et al., 2021), validate the extensibility of NOCL, and provide a comprehensive ablation study, we select five PDEs as the primary benchmarks to cover a broad spectrum of behaviors commonly encountered in physics-informed learning: (i) High-frequency derivative-constrained PDE. (ii) Burgers' equation (nonlinearity and sharp regime change). (iii) Helmholtz (static constraints). (iv) Heat equation (diffusive, smooth dynamics). (v) Reaction–Diffusion system (coupled and challenging spatiotemporal patterns). Derivative-constrained/ Burgers/ Helmholtz primarily benchmark NOCL against CL, Heat supports ablation studies, and Reaction–Diffusion further shows that NOCL is compatible with training strategies beyond coefficient-based curricula (e.g., causal training). We use the relative ℓ_2 error: $\mathrm{Rel}L_2(u^*, u) = \frac{\|u^* - u\|_2}{\|u^*\|_2} = \frac{\left(\sum_{i=1}^{N}(u_i^* - u_i)^2\right)^{1/2}}{\left(\sum_{i=1}^{N}(u_i^*)^2\right)^{1/2}}$, a standard evaluation metric in PINN problems, as our performance measure. All experiments are conducted using the JAX (Bradbury et al., 2018) framework on an NVIDIA GeForce RTX 4090 GPU. Detailed training details are listed below after the description of each experiment.

4.1 PINN Curriculum Learning Failure Modes

To compare NOCL against CL that directly transfers model parameters across stages, we construct a derivative-constrained PDE system based on a high-frequency solution. Specifically, we define the governing system as

$$\begin{cases} \nabla u(x,t) = \mathbf{f}(x,t), & (x,t) \in \Omega, \\ u(x,0) = g(x), \end{cases} \tag{5}$$

where $\mathbf{f}(x,t)$ is analytically generated from the ground-truth solution $u^\star(x,t) = \sin(\alpha(x - 30t))$, i.e., $\mathbf{f}(x,t) = \nabla u^\star(x,t)$, and $g(x) = u^\star(x,0) = \sin(\alpha x)$.

In NOCL, since the convection equation system contains only a single parameter α, we only need to perform the second step of NTK analysis. Based on the NTK matrix generated from 64 data points, we observe that the eigenvalue variance of the NTK matrix increases monotonically with α. We design the corresponding curriculum based on this result, with detailed curriculum specifications provided in Appendix C.1. In all experiments, the final target value is set to $\alpha = 3$.

Table 1: Relative L_2 errors for Burgers and Helmholtz equations.

Method	Baseline	CL	NOCL
Burgers	1.32e-2	1.78e-2	**8.34e-3**
Helmholtz	4.30e-1	4.01e-1	**3.80e-2**

Table 2: Relative L_2 errors for the reaction-diffusion equation with different curriculum parameters and methods.

Method	Baseline	CL (ρ)	CL (ν)	NOCL (ρ)	NOCL (ν)
MLP	4.78e-1	9.93e-2	7.92e-2	**7.53e-2**	7.77e-2
Causal	1.00e-1	1.16e-1	1.00e-1	7.86e-2	**7.60e-2**

Since CL typically requires more optimization steps to achieve comparable convergence (Monaco & Apiletti, 2023), we allocate three times more training steps to the baseline methods to ensure sufficient learning capacity and a fair comparison. For other hyperparameters, we implement CL settings that are identical to NOCL, except for the neural operator-related components. The baseline settings use the same physics training configuration as our approach.

The experimental results, shown in Figure 2 for the case $\alpha = 3$, reveal a clear failure mode in CL. We attribute this behavior to feature mismatch across PDE coefficients: representations learned from solutions corresponding to previous parameter values do not necessarily align with the structural characteristics of the target-coefficient solution, leading to negative transfer when such representations are directly reused in subsequent training stages. This misalignment introduces biased inductive priors that hinder optimization and destabilize convergence. To rule out insufficient optimization as the bottleneck, we extend baseline training on the target task to 5×10^5 steps. Fig. 7 (App. C) shows that the solution remains inaccurate across the domain.

We further evaluate NOCL's stability and generality on two additional PDE systems: the nonlinear **Burgers** equation and the **Helmholtz** equation. Detailed experimental settings and PDE formulations are provided in Appendix C. As reported in Table 1, NOCL consistently achieves the lowest relative L_2 error across all tested PDEs. Notably, CL w/o NO performs worse than the baseline on the Burgers equation, further illustrating that it can exhibit failure modes on certain PDE systems. These results demonstrate that the generalization capability of neural operators enables NOCL to reliably transfer information between parameter regimes, preventing failure modes and ensuring robust performance across a diverse set of PDE systems.

Importantly, NOCL maintains computational efficiency: for each new parametric PDE, it can produce multiple predicted solutions at different parameter values, and the trained neural operator can be saved to provide a warm start for future runs. For the Burgers equation, FNO training time amortized to each PDE parameter is 8.33 seconds, the supervised training takes 2.90 seconds, and physics-based training takes 46.23 seconds (Baseline cost). The resulting added overhead is far lower than that of CL, which typically incurs roughly $3\times$ PINN training time. With all NOCL costs included, its accuracy advantage over longer-trained Vanilla PINNs and direct CL persists on all three benchmarks evaluated at matched wall-clock budgets under the tested configurations (Appendix F, Table 4).

4.2 Extensibility and validation of NTK variance

To demonstrate the extensibility of NOCL to other PINN training techniques and to validate the NTK-based curriculum design on multi-parameter PDEs, we conduct experiments on the reaction–diffusion equation. We employ the reaction-diffusion equation system from the study by Krishnapriyan et al. (2021), which investigates possible failure modes in PINNs. The reaction-diffusion equation is defined as follows:

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0, \quad x \in [0,1], \quad t \in [0,1] \tag{6}$$

with initial condition $u(x,0) = \sin(\pi x)$, where ν is the diffusion coefficient and ρ is the reaction coefficient. This equation combines diffusion processes (controlled by ν) with nonlinear reaction terms (controlled by ρ), creating complex dynamics that are particularly challenging for neural network approximation due to the interplay between these two competing mechanisms.

For this problem, we test three approaches: baseline without curriculum learning, CL, and NOCL. To evaluate the scalability of curriculum learning, we implement these approaches on both standard MLP models and MLP models trained using causal methods (Wang et al., 2024b). The detailed hyperparameters used in our experiments can be found in Appendix D.1.

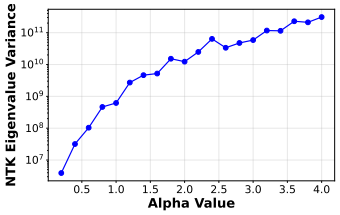

Figure 3: NTK eigenvalue variance analysis for 2D heat equation across different α values. The plot shows how the variance of NTK eigenvalues changes as the heat conduction parameter α varies from 0.2 to 4.0.

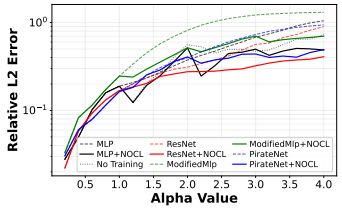

Figure 4: Relative L_2 error comparison between baseline models and their NOCL-augmented counterparts across different network architectures, including MLP, ModifiedMLP, ResNet, and PirateNet, evaluated over varying values of the diffusion parameter α.

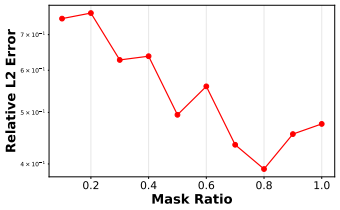

Figure 5: Ablation study on the mask ratio in the NOCL framework. The figure shows the final relative L_2 error at $\alpha = 4.0$ obtained using different mask ratios. A mask ratio of 1 corresponds to retaining all FNO-predicted data points for supervised learning.

We test vanilla PINN training method (Raissi et al., 2019), CL, and NOCL on a challenging PDE system with parameters $\nu = 4$ and $\rho = 5$. Through analysis of NTK matrix eigenvalues variance, as shown in Figure 6 in Appendix D, we discover distinct curriculum stages in the ν parameter, while no clear curriculum characteristics exist in the ρ parameter. We select ν as the curriculum parameter for our primary experiments. To compare curriculum strategies designed on different parameters, we also conduct curriculum learning experiments on the parameter ρ for both the traditional method and our proposed NOCL.

The experimental results in Table 2 show that NOCL achieves the lowest relative ℓ_2 errors for both MLP and causal PINNs. The table also provides a two-factor comparison of curriculum-path selection and cross-stage transfer. With direct weight transfer, selecting the NTK-guided ν curriculum instead of the manually specified ρ curriculum reduces the errors from 0.0993 to 0.0792 for MLP and from 0.116 to 0.100 for causal PINNs. Keeping the ρ curriculum fixed, replacing direct weight transfer with neural-operator-guided initialization reduces the errors to 0.0753 and 0.0786, respectively. These comparisons support distinct contributions from NTK-guided path selection and neural-operator-based transfer. Their benefits do not combine uniformly: NOCL(ν) performs best for causal PINNs, whereas NOCL(ρ) is slightly better than NOCL(ν) for MLP (0.0753 versus 0.0777). The smaller performance gap between the two paths under NOCL suggests that operator-based transfer reduces sensitivity to curriculum choice.

To compare the differences between the two curriculum designs, we examine the relative L_2 error of the PINN trained during the initial stage for curricula structured on parameters ν and ρ. We observe that in the initial stage of the curriculum on ρ (with $\rho = 1$, $\nu = 4$), the relative L_2 error reaches 1.84×10^{-1}, indicating that the PINN fails to provide high-quality training data for the neural operator at the initial stage. In contrast, for the curriculum on ν at its initial stage (with $\nu = 1$, $\rho = 5$), the relative L_2 error is only 5.97×10^{-2}. This significant performance gap explains why CL achieves a lower final relative L_2 error when the curriculum is designed on ν, thereby supporting the effectiveness of our proposed NTK eigenvalue variance metric. Regarding the fact that NOCL achieves a similarly low error even when the curriculum is designed on ρ, we hypothesize that this is due to the excellent generalization capability of the neural operator and the use of mask algorithms to filter out harmful information.

Furthermore, NOCL demonstrates superior extensibility compared to CL. When using causal learning, CL exhibits increased relative L_2 errors, while NOCL continues to show decreased relative L_2 errors. Therefore, we conclude that NOCL combined with NTK analysis exhibits excellent extensibility across various training tricks.

4.3 Ablation Study on NTK-based Curriculum Design

To systematically investigate the functional contributions of individual components in NOCL and assess its generalization across model architectures, we design controlled ablation and cross-architecture

experiments on the two-dimensional heat equation:

$$\frac{\partial u}{\partial t} - \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 0, \quad x, y \in [0, 1],\ t \in [0, 0.1] \tag{7}$$

subject to the initial condition $u(x, y, 0) = \sin(\pi x) \sin(\pi y)$ and Dirichlet boundary conditions $u(0, y, t) = u(1, y, t) = u(x, 0, t) = u(x, 1, t) = 0$. The thermal diffusivity is denoted by α. The analytical solution is $u^*(x, y, t) = \sin(\pi x) \sin(\pi y)\ \exp(-2\alpha\pi^2 t)$.

We begin by substantiating the utility of the NTK eigenvalue variance (Section 3.2) as a principled indicator of training difficulty. Although the heat equation involves only a single thermal coefficient α, we include this analysis as a sanity check. By evaluating NTK variance over $\alpha \in [0.2, 4.0]$, we confirm that its trend aligns closely with observed training error, thereby validating our proposed curriculum ordering. This experiment further shows that NOCL improves convergence even in this simple single-parameter setting. For 20 uniformly spaced values, we compute the NTK eigenvalue variance using the experimental setup described in Appendix E.1. As shown in Figure 3, the variance displays a variation pattern over α that is similar to that of the final relative L_2 error (green curve in Figure 4). This strong correlation supports NTK variance as an effective proxy for training difficulty. To rule out the confounding effect of uniform spectral scaling, which can also inflate variance, we further compute the variance of the logarithm of the eigenvalues, restricting to the top 75% to mitigate numerical instability from near-zero values. The resulting log-variance (Figure 8, Appendix E.2) aligns with both the standard variance and the error trend in Figure 4, indicating that NTK variance captures a meaningful notion of spectral imbalance suitable for curriculum construction.

To further evaluate the robustness of NTK variance, we conduct additional experiments across varying experimental configurations, including different numbers of sampled collocation nodes, different MLP widths, and different network architectures. Across all these settings, as shown in Figure 9 in Appendix E.2, the NTK eigenvalue variance consistently preserves the same monotonic alignment with training difficulty, demonstrating that the proposed NTK-based metric is not sensitive to sampling density, model capacity, or architectural choices. This invariance across discretization, network scale, and architecture provides strong empirical evidence that NTK variance captures an intrinsic property of the optimization landscape, rather than an artifact of specific experimental configurations.

We then compare NOCL with the baseline MLP model over the same range of α, holding all hyperparameters and training configurations fixed (Appendix E.1). As shown in Figure 4, NOCL consistently outperforms the baseline across the entire parameter range, except for the narrow cases of $\alpha = 1.8$ and $\alpha = 2.0$. The figure reports results for both baseline training and NOCL-based training under identical experimental settings. The curve labeled "No training" (small black dotted line) denotes the configuration in which FNO is only trained in the initial stage. While the performance gains are relatively modest in the low-α regime, they become increasingly pronounced as α grows, indicating that NOCL effectively mitigates the failure modes of standard PINN training in more challenging parameter regimes.

Effect of Mask Ratio and Residual-Based Data Filtering. To validate the necessity of curricular training for the neural operator, we conduct an ablation study in which the operator is trained only on the dataset generated in the initial stage (i.e., $\alpha \leq 1$), without any subsequent data expansion or operator updates during later curriculum stages. As α increases, we observe a systematic degradation in performance, demonstrating that a neural operator trained on a limited parameter range fails to generalize to more challenging regimes due to insufficient coverage of the solution manifold. In addition, as shown in Figure 5, we perform an ablation study on the mask ratio and observe that directly using all FNO-predicted points for supervised initialization leads to suboptimal performance. In contrast, residual-based data filtering consistently yields better convergence and accuracy, confirming that data sanitization is a necessary component of reliable knowledge transfer in NOCL. To stress-test robustness against catastrophic component failure, we further evaluate three worst-case initialization variants on the heat equation at $\alpha = 3$ *without* any data filtering: a random (untrained) NO, zero initialization, and a systematically shifted solution ($+0.2$), yielding relative L_2 errors of **0.630**, **0.604**, and **0.592** respectively, **all below the baseline (0.681)**, while NOCL itself reaches **0.498**. This confirms that NOCL's initialization-only intervention cannot harm training even under extreme failure of its internal components.

Cross-Architecture Scalability of NOCL. To evaluate the architectural scalability of NOCL, we conduct cross-architecture experiments using PirateNet (Wang et al., 2024a), Modified-MLP (Wang

et al., 2023), and ResNet-based MLP (He et al., 2016) as base models. Across all tested architectures, NOCL consistently delivers significant performance improvements over their respective baselines, demonstrating strong architectural scalability and effectiveness across diverse model families.

Table 3: Relative ℓ_2 errors for the heat equation and velocity errors for the Navier–Stokes equation with different neural operators.

NOCL operator	Heat, $\alpha = 3$	Heat, $\alpha = 4$	Navier–Stokes
Vanilla PINN (no operator)	$0.681^{\dagger}$	$1.050^{\dagger}$	0.004539
FNO	0.4416	**0.4602**	0.003092
DeepONet	0.7401	0.7987	0.003597
GNO adaptation	0.5355	0.5832	0.003460
Haar-WNO adaptation	0.4377	0.6814	0.003336
KNO-style adaptation	**0.3884**	0.4845	**0.002824**

† Previously reported Heat baselines.

Extensibility to Different Neural Operators. To evaluate the extensibility of NOCL to different neural operators, we replace FNO with DeepONet (Lu et al., 2021), GNO, a Haar-WNO adaptation, and a compact KNO-style model on the heat equation and the two-dimensional Taylor–Green vortex at $\mathrm{Re} = 100$. Detailed experimental settings are provided in Appendix G. As shown in Table 3, all five operator variants improve the final Navier–Stokes velocity error over the baseline. The heat experiments show the same compatibility but also clear architecture dependence. These results demonstrate that NOCL is not restricted to FNO and can exploit different neural operator architectures according to the PDE structure.

5 Conclusion

Curriculum learning can help PINNs solve challenging PDEs, but it often fails on certain curricula and still relies on prior knowledge to design progressions for multi-parameter PDEs. We propose NOCL, which trains a neural operator as a transfer tool across stages, uses a residual-based filtering algorithm to provide reliable initialization, and leverages the variance of NTK eigenvalues to assess curriculum rationality. Experiments across multiple PDE systems show that NOCL mitigates common curriculum-learning failure modes, is compatible with other PINN architectures and training tricks, and that NTK-based analysis is effective for curriculum construction. Future work includes automating curriculum generation along multi-parameter paths and developing principled methods for selecting the optimal mask ratio.

Acknowledgments and Disclosure of Funding

The research work described in this paper was conducted in the JC STEM Lab of Machine Learning and Symbolic Reasoning funded by The Hong Kong Jockey Club Charities Trust. We would also like to thank the reviewers for their thorough reading of our manuscript and for their constructive suggestions, which have undoubtedly improved the final version.

The authors declare no competing interests.

References

Bekele, Y. W. Physics-informed neural networks with curriculum training for poroelastic flow and deformation processes. In *Proceedings of the 19th Nordic Geotechnical Meeting*, volume 2, pp. 249–265, 2025.

Belytschko, T., Liu, W. K., Moran, B., and Elkhodary, K. I. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, 2nd edition, 2014.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: Composable transformations of Python+NumPy programs, 2018. Software, version 0.3.13.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

Chen, Y., Ji, Y., Narayan, A., and Xu, Z. TGPT-PINN: Nonlinear model reduction with transformed GPT-PINNs. *Computer Methods in Applied Mechanics and Engineering*, 430:117198, 2024.

Guo, Y., Fu, Z., Min, J., Lin, S., Liu, X., Rashed, Y. F., and Zhuang, X. Long-term simulation of physical and mechanical behaviors using curriculum-transfer-learning based physics-informed neural networks. *Neural Networks*, 191:107825, 2025.

Han, J., Jentzen, A., and E, W. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Hanna, J. M., Aguado, J. V., Comas-Cardona, S., Askri, R., and Borzacchiello, D. Sensitivity analysis using physics-informed neural networks. *Engineering Applications of Artificial Intelligence*, 135: 108764, 2024.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560, 2021.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

LeVeque, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics, 2007.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM / IMS Journal of Data Science*, 1(3):1–27, 2024.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3): 218–229, 2021.

Mao, Z., Jagtap, A. D., and Karniadakis, G. E. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

Monaco, S. and Apiletti, D. Training physics-informed neural networks: One learning to rule them all? *Results in Engineering*, 18:101023, 2023.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Tan, Y. and Liu, H. How does a kernel based on gradients of infinite-width neural networks come to be widely used: a review of the neural tangent kernel. *International Journal of Multimedia Information Retrieval*, 13(1):8, 2024.

Wang, S., Yu, X., and Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wang, S., Sankaran, S., Wang, H., and Perdikaris, P. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*, 2023.

Wang, S., Li, B., Chen, Y., and Perdikaris, P. PirateNets: Physics-informed deep learning with residual adaptive networks. *Journal of Machine Learning Research*, 25(402):1–51, 2024a.

Wang, S., Sankaran, S., and Perdikaris, P. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024b.

Yan, D. and He, L. DP-PINN: A dual-phase training scheme for improving the performance of physics-informed neural networks. In *Computational Science – ICCS 2024*, pp. 19–32, 2024.

A Neural Operator Generalization over PDE Parameter Space

This appendix provides a direct proof of Corollary 3.1 for two common PDE classes in PINNs: parameterized elliptic equations (e.g., Poisson / diffusion) and parabolic equations (e.g., advection–diffusion / reaction–diffusion).

A.1 An abstract distance-to-support bound

Lemma A.1 (From on-support accuracy and Lipschitzness to off-support control)**.** *Let $\Phi \subset \mathbb{R}^p$ and let $K \subset \Phi$ be nonempty and compact. Let $(\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$ be a normed space. Suppose $S : \Phi \to \mathcal{Y}$ and $G : \Phi \to \mathcal{Y}$ satisfy: (i) $\sup_{\phi\in K} \|G(\phi) - S(\phi)\|_{\mathcal{Y}} \le \varepsilon_{\mathrm{op}}$; (ii) $\|G(\phi_1) - G(\phi_2)\|_{\mathcal{Y}} \le L_G\|\phi_1 - \phi_2\|$ for all $\phi_1, \phi_2 \in \Phi$; (iii) $\|S(\phi_1) - S(\phi_2)\|_{\mathcal{Y}} \le L_S\|\phi_1 - \phi_2\|$ for all $\phi_1, \phi_2 \in \Phi$. Then for any $\phi \in \Phi$,*

$$\|G(\phi) - S(\phi)\|_{\mathcal{Y}} \le \varepsilon_{\mathrm{op}} + (L_G + L_S)\ \mathrm{dist}(\phi, K).$$

Proof. Fix $\phi \in \Phi$. Since K is compact, there exists $\tilde{\phi} \in K$ such that $\|\phi - \tilde{\phi}\| = \mathrm{dist}(\phi, K)$. By triangle inequality,

$$\|G(\phi) - S(\phi)\|_{\mathcal{Y}} \le \|G(\phi) - G(\tilde{\phi})\|_{\mathcal{Y}} + \|G(\tilde{\phi}) - S(\tilde{\phi})\|_{\mathcal{Y}} + \|S(\tilde{\phi}) - S(\phi)\|_{\mathcal{Y}}.$$

Bound the three terms by assumptions (ii), (i), and (iii), respectively, and use $\|\phi - \tilde{\phi}\| = \mathrm{dist}(\phi, K)$. □

A.2 Elliptic case

Problem setup. Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain and let $\Phi \subset \mathbb{R}^p$ be compact. For each $\phi \in \Phi$, consider the parameterized elliptic PDE

$$\begin{cases} -\nabla \cdot (A(x, \phi)\nabla u) = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{8}$$

where $f \in H^{-1}(\Omega)$ is fixed. Denote $S(\phi) = u_\phi$ the solution operator.

Elliptic assumptions (used below). Assume $A : \Omega \times \Phi \to \mathbb{R}^{d\times d}$ satisfies: (a) uniform ellipticity: $\lambda|\xi|^2 \le \xi^\top A(x, \phi)\xi \le \Lambda|\xi|^2$ for all $\xi, (x, \phi)$; (b) uniform boundedness: $A(\cdot, \phi) \in L^\infty(\Omega)$ uniformly in ϕ; (c) Lipschitz in parameters: $\|A(\cdot, \phi_1) - A(\cdot, \phi_2)\|_{L^\infty(\Omega)} \le L_A\|\phi_1 - \phi_2\|$. Assume the neural operator $G_\theta : \Phi \to H^1_0(\Omega)$ satisfies: (d) on-support approximation: $\sup_{\phi\in\mathrm{supp}(\mu_\Phi)} \|G_\theta(\phi) - S(\phi)\|_{H^1_0(\Omega)} \le \varepsilon_{\mathrm{op}}$; (e) Lipschitz continuity: $\|G_\theta(\phi_1) - G_\theta(\phi_2)\|_{H^1_0(\Omega)} \le L_G\|\phi_1 - \phi_2\|$.

Lemma A.2 (Elliptic solution operator is Lipschitz)**.** *Under the elliptic assumptions (a)–(c), for every $\phi \in \Phi$ there exists a unique weak solution $u_\phi \in H^1_0(\Omega)$ of* (8)*. Moreover, there exists $L_S > 0$ such that*

$$\|u_{\phi_1} - u_{\phi_2}\|_{H^1_0(\Omega)} \le L_S\|\phi_1 - \phi_2\|, \qquad \forall \phi_1, \phi_2 \in \Phi.$$

Proof. Existence and uniqueness follow from Lax–Milgram using uniform ellipticity and boundedness. Let $w := u_{\phi_1} - u_{\phi_2} \in H^1_0(\Omega)$. Subtract the weak formulations for ϕ_1 and ϕ_2 to obtain, for all $v \in H^1_0(\Omega)$,

$$\int_\Omega A(x, \phi_1)\nabla w \cdot \nabla v\, dx = \int_\Omega \big(A(x, \phi_2) - A(x, \phi_1)\big)\nabla u_{\phi_2} \cdot \nabla v\, dx.$$

Choosing $v = w$ and using uniform ellipticity yields

$$\lambda\|\nabla w\|^2_{L^2(\Omega)} \le \|A(\cdot, \phi_2) - A(\cdot, \phi_1)\|_{L^\infty(\Omega)}\|\nabla u_{\phi_2}\|_{L^2(\Omega)}\|\nabla w\|_{L^2(\Omega)}.$$

Using the parameter Lipschitz condition and the standard energy bound $\|\nabla u_{\phi_2}\|_{L^2(\Omega)} \le C_\Omega\|f\|_{H^{-1}(\Omega)}$ gives the claim with, e.g., $L_S = \frac{L_A C_\Omega}{\lambda}\|f\|_{H^{-1}(\Omega)}$. □

A.3 Parabolic case

Problem setup. Let $T > 0$. For each $\phi \in \Phi$, consider the parameterized parabolic PDE

$$\begin{cases} \partial_t u - \nabla \cdot \big(A(x,t,\phi)\nabla u\big) - b(x,t,\phi)\cdot \nabla u - c(x,t,\phi)u = f(x,t), & (x,t) \in \Omega \times (0,T), \\ u(\cdot,t,\phi) = 0, & \text{on } \partial\Omega \times (0,T), \\ u(\cdot,0,\phi) = u_0(\cdot), & \text{in } \Omega, \end{cases} \tag{9}$$

where $f \in L^2(0,T;H^{-1}(\Omega))$ and $u_0 \in L^2(\Omega)$ are fixed (independent of ϕ). Denote

$$S(\phi) = u_\phi \in X_T := L^2(0,T;H_0^1(\Omega)) \cap C([0,T];L^2(\Omega)).$$

Parabolic assumptions (used below). Assume the coefficients satisfy: (a) uniform parabolicity with constants λ, Λ; (b) uniform boundedness $\|A\|_{L^\infty} \le M_A$, $\|b\|_{L^\infty} \le M_b$, $\|c\|_{L^\infty} \le M_c$ and $c \ge 0$; (c) Lipschitz in parameters in L^∞ with constants L_A, L_b, L_c. Assume the neural operator $G_\theta : \Phi \to X_T$ satisfies: (d) on-support approximation: $\sup_{\phi \in \mathrm{supp}(\mu_\Phi)} \|G_\theta(\phi) - S(\phi)\|_{X_T} \le \varepsilon_{\mathrm{op}}$; (e) Lipschitz continuity: $\|G_\theta(\phi_1) - G_\theta(\phi_2)\|_{X_T} \le L_G \|\phi_1 - \phi_2\|$.

Lemma A.3 (Parabolic solution operator is Lipschitz)**.** *Under the parabolic assumptions (a)–(c), for every $\phi \in \Phi$ there exists a unique weak solution $u_\phi \in X_T$ to* (9)*. Moreover, there exists $L_S > 0$ (depending only on the coefficient bounds/Lipschitz constants and on $\Omega, T, \|u_0\|_{L^2(\Omega)}, \|f\|_{L^2(0,T;H^{-1}(\Omega))}$) such that*

$$\|u_{\phi_1} - u_{\phi_2}\|_{X_T} \le L_S \|\phi_1 - \phi_2\|, \qquad \forall \phi_1, \phi_2 \in \Phi.$$

Proof. (Proof sketch.) Let $w := u_{\phi_1} - u_{\phi_2}$. Subtracting the two PDEs yields an equation of the form

$$\partial_t w - \nabla \cdot (A_1 \nabla w) - b_1 \cdot \nabla w - c_1 w = \nabla \cdot \big((A_2 - A_1)\nabla u_{\phi_2}\big) + (b_1 - b_2)\cdot \nabla u_{\phi_2} + (c_1 - c_2)u_{\phi_2},$$

with homogeneous initial and boundary conditions, where $A_i = A(\cdot,\cdot,\phi_i)$ etc. Testing by w, using uniform parabolicity, Cauchy–Schwarz and Young inequalities, and absorbing terms yields an energy inequality

$$\frac{d}{dt}\|w(t)\|_{L^2(\Omega)}^2 + \lambda \|\nabla w(t)\|_{L^2(\Omega)}^2 \le C\,\|\phi_1 - \phi_2\|^2 \big(\|\nabla u_{\phi_2}(t)\|_{L^2(\Omega)}^2 + \|u_{\phi_2}(t)\|_{L^2(\Omega)}^2\big),$$

for a constant C depending only on the coefficient bounds and parameter Lipschitz constants. Integrating in time and using standard parabolic energy estimates for u_{ϕ_2} (depending on $\|u_0\|$ and $\|f\|$) gives the claim. □

A.4 Proof of the main-text corollary

Proof of Corollary 3.1. Let $K := \mathrm{supp}(\mu_\Phi)$ and note that K is assumed nonempty and compact.

Elliptic case. Take $\mathcal{Y} = H_0^1(\Omega)$ and $\|\cdot\|_{\mathcal{Y}} = \|\cdot\|_{H_0^1(\Omega)}$. By Lemma A.2, the solution operator S is L_S-Lipschitz. By assumptions (d)–(e) in the elliptic subsection, G_θ has on-support error $\varepsilon_{\mathrm{op}}$ and is L_G-Lipschitz. Applying Lemma A.1 yields (3).

Parabolic case. Take $\mathcal{Y} = X_T$ with $\|v\|_{X_T} = \|v\|_{L^2(0,T;H_0^1(\Omega))} + \|v\|_{C([0,T];L^2(\Omega))}$. By Lemma A.3, the solution operator S is L_S-Lipschitz in X_T. By assumptions (d)–(e) in the parabolic subsection, G_θ has on-support error $\varepsilon_{\mathrm{op}}$ and is L_G-Lipschitz in X_T. Applying Lemma A.1 again yields (3). □

B Monotone relation between eigenvalue variance and effective condition number

This appendix provides a simple theoretical justification that the eigenvalue variance is a principled proxy of spectral imbalance. In particular, under a rapidly decaying effective spectrum model and in the presence of k numerically vanishing (invalid) eigenvalues, the variance computed on the full spectrum is a strictly increasing function of the (effective) condition number.

B.1 Assumptions: effective spectrum, rapid decay, and vanishing tail

Let K be a symmetric positive semidefinite NTK matrix (for either residual or boundary terms). In practice, as the number of collocation points increases, many small eigenvalues become numerically indistinguishable from 0. To avoid instability, we consider the *effective* spectrum obtained by truncating vanishing eigenvalues: for a fixed $m \geq 2$,

$$\Lambda_{\text{eff}}(\alpha) = \{\lambda_1(\alpha) \geq \cdots \geq \lambda_m(\alpha) > 0\}, \qquad \sum_{i=1}^{m} \lambda_i(\alpha) = S \quad (S > 0 \text{ constant}).$$

We further assume a rapid-decay parametric family:

$$\lambda_i(\alpha) = \frac{S}{Z(\alpha)} e^{-\alpha(i-1)}, \qquad Z(\alpha) = \sum_{j=1}^{m} e^{-\alpha(j-1)}, \qquad \alpha \geq 0. \tag{10}$$

Here α is a *spectral concentration parameter*: larger α yields faster decay and hence stronger spectral imbalance. This exponential family is a canonical model for rapidly decaying spectra and is consistent with the empirical observation that NTK spectra often decay quickly.

We model the remaining invalid eigenvalues as $k \geq 0$ exact zeros appended to the effective spectrum:

$$\widetilde{\Lambda}(\alpha) = \{\lambda_1(\alpha), \ldots, \lambda_m(\alpha), \underbrace{0, \ldots, 0}_{k \text{ times}}\}, \qquad n := m + k.$$

We define the (effective) condition number using only the effective spectrum and the eigenvalue variance on the full spectrum as

$$\kappa_{\text{eff}}(\alpha) = \frac{\lambda_1(\alpha)}{\lambda_m(\alpha)}, \qquad \text{Var}(\widetilde{\Lambda}(\alpha)) = \frac{1}{n} \sum_{i=1}^{n} \left(\tilde{\lambda}_i(\alpha) - \frac{S}{n} \right)^2,$$

where $\{\tilde{\lambda}_i(\alpha)\}_{i=1}^{n} = \widetilde{\Lambda}(\alpha)$ and $\sum_{i=1}^{n} \tilde{\lambda}_i(\alpha) = S$.

B.2 Theorem and proof

Theorem B.1 (Variance is a strictly increasing function of the effective condition number)**.** *Under* (10) *with fixed $m \geq 2$, fixed $k \geq 0$, and fixed trace $S > 0$, both $\kappa_{\text{eff}}(\alpha)$ and $\text{Var}(\widetilde{\Lambda}(\alpha))$ are strictly increasing in α for $\alpha > 0$. Moreover, there exists a strictly increasing function F such that*

$$\text{Var}(\widetilde{\Lambda}(\alpha)) = F(\kappa_{\text{eff}}(\alpha)), \qquad F'(\kappa) > 0 \quad (\kappa > 1).$$

Proof. Let $q := e^{-\alpha} \in (0, 1]$. Then

$$Z(\alpha) = \sum_{j=0}^{m-1} q^j = \frac{1 - q^m}{1 - q} \quad (q \neq 1), \qquad \lambda_i(\alpha) = S \frac{1 - q}{1 - q^m} q^{i-1} \quad (q \neq 1).$$

(At $\alpha = 0$ we have $q = 1$ and $\lambda_i(0) = S/m$ by continuity.)

From (10),

$$\kappa_{\text{eff}}(\alpha) = \frac{\lambda_1(\alpha)}{\lambda_m(\alpha)} = \frac{1}{q^{m-1}} = e^{\alpha(m-1)},$$

hence $\kappa'_{\text{eff}}(\alpha) = (m-1)e^{\alpha(m-1)} > 0$ for all $\alpha \geq 0$.

Using $\sum_{i=1}^{m} \lambda_i(\alpha) = S$ and that $\widetilde{\Lambda}(\alpha)$ contains k zeros, we still have $\sum_{i=1}^{n} \tilde{\lambda}_i(\alpha) = S$, so

$$\text{Var}(\widetilde{\Lambda}(\alpha)) = \frac{1}{n} \sum_{i=1}^{n} \tilde{\lambda}_i(\alpha)^2 - \left(\frac{S}{n} \right)^2 = \frac{1}{n} \sum_{i=1}^{m} \lambda_i(\alpha)^2 - \left(\frac{S}{n} \right)^2.$$

Thus it suffices to show $M_2(\alpha) := \sum_{i=1}^{m} \lambda_i(\alpha)^2$ is strictly increasing for $\alpha > 0$.

For $q \in (0,1)$,

$$M_2(\alpha) = \sum_{i=1}^{m} \left(S \frac{1-q}{1-q^m} q^{i-1} \right)^2 = S^2 \left(\frac{1-q}{1-q^m} \right)^2 \sum_{i=0}^{m-1} q^{2i}$$
$$= S^2 \left(\frac{1-q}{1-q^m} \right)^2 \cdot \frac{1-q^{2m}}{1-q^2} = S^2 \frac{1-q}{1+q} \cdot \frac{1+q^m}{1-q^m}.$$

Define

$$h(q) := \frac{1-q}{1+q} \cdot \frac{1+q^m}{1-q^m}, \qquad \text{so that} \qquad M_2(\alpha) = S^2 h(q), \quad q = e^{-\alpha}.$$

We show $h'(q) < 0$ for $q \in (0,1)$. Let $\phi(q) := \log h(q)$. Then

$$\phi'(q) = \frac{d}{dq} \Big[\log(1-q) - \log(1+q) + \log(1+q^m) - \log(1-q^m) \Big] = -\frac{2}{1-q^2} + \frac{2mq^{m-1}}{1-q^{2m}}.$$

Thus $\phi'(q) < 0$ is equivalent to

$$mq^{m-1} < \frac{1-q^{2m}}{1-q^2} = 1 + q^2 + \cdots + q^{2m-2}.$$

By AM–GM applied to the m positive numbers $1, q^2, \ldots, q^{2m-2}$,

$$\frac{1}{m}\big(1 + q^2 + \cdots + q^{2m-2}\big) > \big(1 \cdot q^2 \cdots q^{2m-2}\big)^{1/m} = q^{(2(0+1+\cdots+(m-1)))/m} = q^{m-1},$$

where the inequality is strict since $q \in (0,1)$ and $m \geq 2$ imply the terms are not all equal. Multiplying by m yields $1 + q^2 + \cdots + q^{2m-2} > mq^{m-1}$, hence $\phi'(q) < 0$. Since $h'(q) = h(q)\phi'(q)$ and $h(q) > 0$, we have $h'(q) < 0$ on $(0,1)$.

Finally, $\frac{dq}{d\alpha} = -q < 0$ for $\alpha > 0$, so

$$\frac{d}{d\alpha} M_2(\alpha) = S^2 h'(q) \frac{dq}{d\alpha} > 0,$$

and therefore $\mathrm{Var}(\widetilde{\Lambda}(\alpha))$ is strictly increasing for $\alpha > 0$ because

$$\frac{d}{d\alpha} \mathrm{Var}(\widetilde{\Lambda}(\alpha)) = \frac{1}{n} \frac{d}{d\alpha} M_2(\alpha) > 0.$$

Because $\kappa_{\text{eff}}(\alpha) = q^{-(m-1)}$, we can invert

$$q = \kappa_{\text{eff}}(\alpha)^{-1/(m-1)}.$$

Substituting into the variance expression defines an explicit function F on $\kappa > 1$:

$$\mathrm{Var}(\widetilde{\Lambda}) = F(\kappa) = \frac{S^2}{n} h\Big(\kappa^{-1/(m-1)}\Big) - \left(\frac{S}{n}\right)^2.$$

Since h is strictly decreasing on $(0,1)$ and $\kappa \mapsto \kappa^{-1/(m-1)}$ is strictly decreasing on $(1,\infty)$, their composition is strictly increasing, hence $F'(\kappa) > 0$ wherever it exists. □

Theorem B.1 shows that, under a rapidly decaying effective spectrum model, eigenvalue variance is a monotone surrogate of spectral imbalance traditionally associated with condition number, while being more robust to vanishing minimum eigenvalues (which destabilize the raw condition number as collocation size grows).

C PINN CL Failure modes

C.1 Hyper-parameters

We use a 4-layer MLP with 64 hidden units as our base model architecture. For the alpha curriculum design, the initial curriculum is set as linspace(0.1,1,15), followed by two sections: $(1, 2]$ and $(2, 3]$,

with 5 PINNs trained in each section using corresponding alpha values. The selection of alpha and beta values remains independent of whether the neural operator is employed.

For each PINN training, we use 50,000 physics training steps with the Adam optimizer, initial learning rate of 1×10^{-3}, and exponential decay of 0.95 every 500 steps. The batch size consists of 256 residual loss data points and 100 initial condition data points. For the neural operator, we employ an 8-layer Fourier Neural Operator with 16 modes. The FNO training details are as follows: we train the FNO for 3,000 epochs using the Adam optimizer with an initial learning rate of 1×10^{-3}, exponential decay of 0.95 every 100 steps. When using FNO predictions to train PINN networks, the FNO predicts on a 128×128 grid, and we apply mask filtering with a ratio of 0.5. The PINN initialization uses the Adam optimizer with an initial learning rate of 1×10^{-3}, 10,000 training steps, and exponential decay of 0.9 every 2,500 steps. We use 0.5 as the keep ratio for the masking algorithm.

C.2 Experimental Design for Burgers' Equation

We evaluate our proposed method on the one-dimensional viscous Burgers' equation, which serves as a benchmark for capturing shock waves in fluid dynamics. The governing equation and its associated conditions are defined as:

$$\begin{cases} \frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, & x \in [-1, 1], \quad t \in [0, 1] \\ u(x, 0) = -\sin(\pi x), & u(-1, t) = u(1, t) = 0 \end{cases} \tag{11}$$

where $u(x, t)$ is the velocity field, the viscosity is fixed at $\nu = 0.01/\pi$, and α is the parameter controlling the non-linearity. Our objective is to solve the challenging full non-linear case at $\alpha = 1.0$. The Physics-Informed Neural Network (PINN) model is an MLP consisting of 4 hidden layers with 64 neurons each, employing the hyperbolic tangent activation function and weights initialized via a truncated normal distribution with variance scaling. For the neural operator within our NOCL framework, we utilize a 2D Fourier Neural Operator (FNO) with 32 Fourier modes, 64 hidden channels, and 8 FNO blocks, where the parameter α is injected through a 32-dimensional embedding generated by a 2-layer MLP.

The training process follows a curriculum learning strategy governed by the parameter α. Initially, we pre-train the FNO meta-learner using data from 15 PINNs optimized independently for low non-linearity values in the range $\alpha \in [0.01, 0.1]$. The curriculum then progresses through three main stages: α increases from 0.1 to 0.5 over 10 steps, then to 0.9 over another 10 steps, and finally reaches 1.0 in 2 steps. For each target α in this sequence, the FNO provides a warm-start prediction of the solution field. The PINN first undergoes supervised pre-training for 10,000 steps to match this FNO prediction using an Adam optimizer with a learning rate of 10^{-3}. Subsequently, the model is refined for 50,000 steps using the Physics-Informed loss, which incorporates 4,096 randomly sampled residual points and 768 boundary/initial points. Both the PINN refinement and FNO training employ the AdamW optimizer with an initial learning rate of 10^{-3} and an exponential decay rate of 0.95 every 500 steps. We compare this approach against a vanilla PINN baseline trained directly at $\alpha = 1.0$ and a traditional curriculum learning that performs sequential weight inheritance across the same α curriculum without the guidance of a FNO.

We extend our evaluation to the three-dimensional Helmholtz equation, a critical PDE for modeling wave propagation and vibration. The system is defined on a cubic domain $x, y, z \in [-1, 1]^3$ with homogeneous Dirichlet boundary conditions:

$$\begin{cases} \Delta u + k^2 u = q(x, y, z), & (x, y, z) \in (-1, 1)^3 \\ u(x, y, z) = 0, & (x, y, z) \in \partial[-1, 1]^3 \end{cases} \tag{12}$$

where $k = 1.0$ is the wavenumber. We prescribe an analytical solution $u(x, y, z) = \sin(a_1 \pi x) \sin(a_2 \pi y) \sin(a_3 \pi z)$ with fixed parameters $a_2 = 4.0$ and $a_3 = 3.0$, while a_1 serves as the varying curriculum parameter. The source term q is derived from the analytical solution as $q = (k^2 - (a_1\pi)^2 - (a_2\pi)^2 - (a_3\pi)^2)u$. Our objective is to solve the high-frequency case where a_1 reaches 4.0. The PINN architecture follows a modified MLP with 4 hidden layers of 64 neurons, utilizing a Fourier feature embedding layer (scale 2.0) to capture high-frequency components and $\tanh$ activations. The neural operator is a 3D Fourier Neural Operator (FNO) with 16 modes, 64 hidden channels, and 8 FNO blocks, which takes the spatial grid and a 32-dimensional embedding of a_1 as input.

The NOCL framework for the Helmholtz equation begins with a pre-training phase, where 10 PINNs are trained independently for $a_1 \in [0.1, 1.0]$. The curriculum then proceeds from $a_1 = 1.0$ to 4.0 through 15 sequential stages, each increasing a_1 by 0.2 and training 2 intermediate PINNs. For each new parameter value, the meta-learner provides a warm-start prediction, which is used for supervised pre-training of the PINN for 3,000 steps. This is followed by a physics-informed refinement phase of 100,000 steps using 4,096 residual points and 600 boundary points (100 per face). All optimizations use the AdamW optimizer with a learning rate of 10^{-3}. We compare NOCL against a vanilla PINN trained directly at $a_1 = 4.0$ and a traditional curriculum learning that performs sequential optimization over a 40-point a_1 sequence for 300,000 total steps without FNO-guidance.

D React-Diffusion Equation

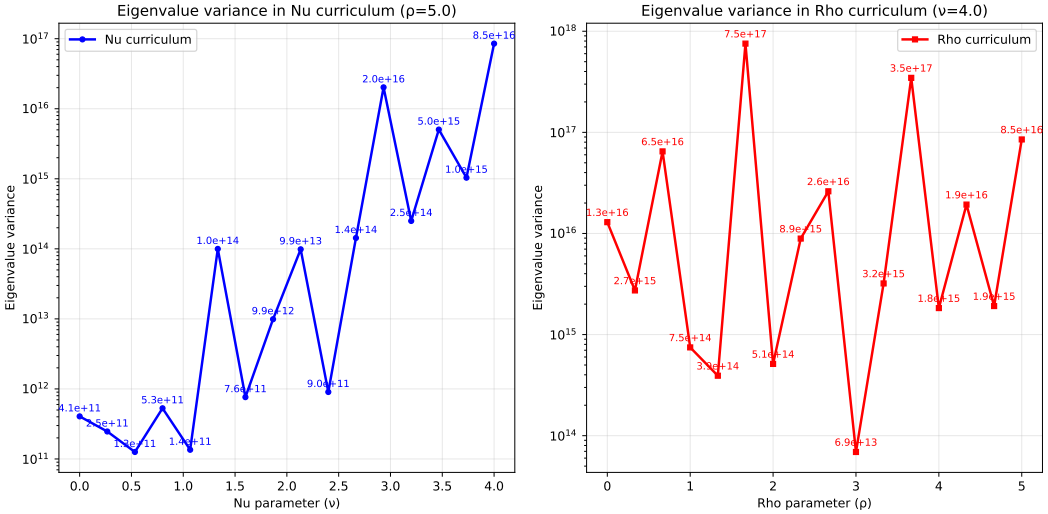

Figure 6: Eigenvalue variance of the NTK across parameter curricula. Left: varying ν with ρ fixed at 5.0. Right: varying ρ with ν fixed at 4.0. Curves show the mean variance over 5 random seeds (128 domain points, 32 initial-condition points); values are annotated at each marker, and the y-axis uses a logarithmic scale.

D.1 Hyper-parameters

Based on the variance trends observed in Figure 6, we designed the curriculum sections for the parameter ν. The initial section for ν was set as linspace(0, 1.0, 10), followed by subsequent curriculum sections covering the ranges $(1, 2]$, $(2, 3]$, and $(3, 4]$. Within each section, five distinct values of ν were selected for PINN training. Similarly, for the parameter ρ, the initial section was designed over the range $[0, 1]$ using linspace(0, 1.0, 10), with subsequent sections covering $(1, 2]$, $(2, 3]$, $(3, 4]$, and $(4, 5]$. In each of these sections, five values of ρ were used for PINN training.

For baseline MLP training, we use 50,000 physics training steps with the Adam optimizer, initial learning rate of 1×10^{-3}, and exponential decay of 0.95 every 500 steps. The batch size consists of 256 residual loss data points and 100 initial condition data points. For the neural operator, we employ an 8-layer Fourier Neural Operator with 16 modes. The FNO training details are as follows: we train the FNO for 3,000 epochs using the Adam optimizer with an initial learning rate of 1×10^{-3}, exponential decay of 0.95 every 100 steps. When using FNO predictions to train PINN networks, the FNO predicts on a 128×128 grid, and we apply mask filtering with a ratio of 0.5. The PINN initialization uses the Adam optimizer with an initial learning rate of 1×10^{-3}, 10,000 training steps, and exponential decay of 0.9 every 2,500 steps. We use 0.5 as the keep ratio for the masking algorithm.

When using causal learning as the base model, we divided the time domain $[0, 1]$ into 5 time windows. The total training steps were set to 50,000, with an additional time window added every 10,000 steps. When the i-th time window was introduced, the learning rate was reset to $(1e-3) * 0.9^{(i-1)}$.

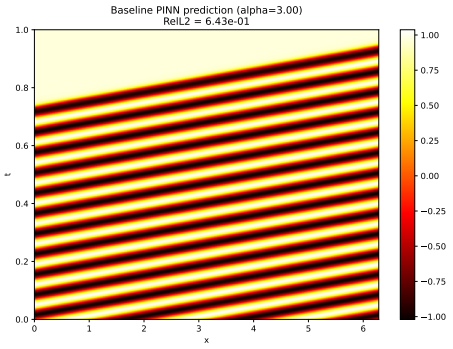

Figure 7: Prediction visualization using the baseline MLP after 500,000 training steps at $\alpha = 3$.

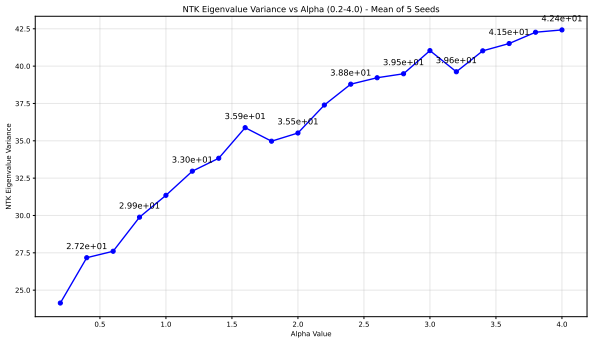

Figure 8: NTK log-eigenvalue variance analysis for the 2D heat equation across different α values. The plot shows how the variance of NTK log-eigenvalues changes as the heat conduction parameter α varies from 0.2 to 4.0.

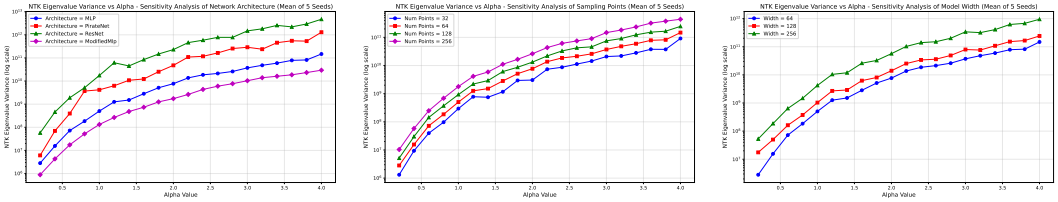

Figure 9: From left to right, the NTK eigenvalue variance as a function of $\alpha \in [0.2, 4.0]$ under (i) different network architectures, (ii) different numbers of sampled points, and (iii) different network widths.

E Ablation Study

E.1 Hyper-parameter

For the heat equation, we use a 4-layer MLP with a width of 64 as the base model. We use the Adam optimizer and train for 50,000 steps with an initial learning rate of 1×10^{-3}, applying an exponential decay of 0.95 every 500 steps. Each batch samples 512 residual points and 250 initial/boundary points. For curriculum learning, we divide the range 0.2 to 4.0 into 20 curriculum levels at intervals of 0.2. The range 0.2 to 1.0 is used as the initial curriculum, and every five curriculum levels thereafter form one section. The neural operator is an FNO with 8 layers and 16 modes. We use the Adam optimizer with an initial learning rate of 1×10^{-3}, training 3,000 steps between sections and applying an exponential decay of 0.95 every 30 steps. For FNO pretraining to initialize the MLP, we also use Adam with an initial learning rate of 1×10^{-3}, for a total of 10,000 steps, with an exponential decay of 0.9 every 2,500 steps. We use 0.3 as the keep ratio for the masking algorithm.

E.2 Log-eigenvalue curve

To show that the increased variance is not due to a uniform scaling of eigenvalues, we also computed the variance of the eigenvalues in the log domain. Due to numerical precision and outliers, we retained only the largest 75% of the eigenvalues. Result is shown in Figure 8

F Matched Wall-Clock Comparisons

We compare NOCL with Vanilla PINNs and direct CL at matched end-to-end wall-clock budgets on the high-frequency problem in Section 4.1, Burgers' equation, and the reaction–diffusion equation. Time is measured from process launch to exit and includes setup, compilation, initial-support and curriculum PINN training, neural-operator training and updates, filtering, supervised initialization, final physics training, evaluation, and serialization. The full NOCL cost is charged to each comparison without amortization. Each baseline budget is fixed from the corresponding NOCL run before the matched-time baseline experiments, and stopping depends on elapsed time rather than accuracy. The timing tolerance is the larger of 5 seconds and 0.25% of the budget.

Table 4: Final relative ℓ_2 errors at matched end-to-end wall-clock budgets. Physics steps count all PINN physics updates across the run; NOCL times also include operator training and supervised initialization. Bold values indicate the lowest error within each benchmark.

Problem	Method	Physics steps	Relative ℓ_2 error	Total time (s)
High-frequency	NOCL	1,250,000	**0.0880**	816.138
	Vanilla PINN	1,780,500	1.0170	815.635
	Direct CL	1,789,903	1.3361	815.634
Burgers	NOCL	1,850,000	**0.00313**	2115.417
	Vanilla PINN	2,256,550	0.00341	2113.854
	Direct CL	2,267,632	0.55985	2113.854
Reaction–diffusion	NOCL	1,250,000	**0.0121**	890.167
	Vanilla PINN	1,321,145	0.0171	888.140
	Direct CL	1,298,004	7.3420	888.137

Vanilla PINNs train directly at the target parameter. Direct CL follows the prescribed parameter path and then continues training at the target until the time budget is exhausted. Both baselines preserve their parameters, optimizer states, learning-rate counters, and sampling sequences throughout training.

Table 4 shows lower errors for NOCL on all three benchmarks even after accounting for its full cost and allowing the baselines more physics updates. This supports an accuracy advantage at matched total time for these configurations.

G Operator Substitution Protocols

Table 3 evaluates five operator architectures using seed 42. Within each benchmark, all operator variants load the same initial-support solution fields and use the same curriculum, PINN architecture, filtering rule, optimizer schedules, and update counts. Refined target fields subsequently depend on the operator and are added to its training dataset as prescribed by NOCL. Every PINN has four hidden layers of width 64 with $\tanh$ activations. The graph, Haar-wavelet, and compact-kernel operators are local adaptations; KNO denotes a kernel neural operator.

Heat. We use the PDE and curriculum in Section 4.3 and Appendix E.1, with a $32\times32\times32$ operator grid. Five initial-support PINNs at $\alpha = \{0.2, 0.4, 0.6, 0.8, 1.0\}$ and fifteen target PINNs each receive 50,000 physics updates. Each target also receives 10,000 supervised initialization updates, and each operator receives 3,000 updates per curriculum group. The operator batch sizes are 5, 10, and 2 across the three groups, with identical sampled support indices across variants. The lowest-residual 30% of predictions are retained; discarded labels are zero-filled and supervised initialization fits the full grid.

Navier–Stokes. We additionally evaluate a two-dimensional decaying Taylor–Green vortex governed by

$$\begin{aligned} u_t + uu_x + vu_y + p_x - \nu(u_{xx} + u_{yy}) &= 0, \\ v_t + uv_x + vv_y + p_y - \nu(v_{xx} + v_{yy}) &= 0, \\ u_x + v_y &= 0, \end{aligned} \tag{13}$$

on $(x, y, t) \in [0, 2\pi]^2 \times [0, 1]$, with initial and boundary values taken from

$$u = \sin(x)\cos(y)e^{-2\nu t}, \qquad v = -\cos(x)\sin(y)e^{-2\nu t}, \qquad p = \tfrac{1}{4}[\cos(2x) + \cos(2y)]e^{-4\nu t}. \tag{14}$$

The target is $\nu = 0.01$ ($\mathrm{Re} = 100$). The shared initial-support viscosities are $\{0.10, 0.08, 0.06, 0.05, 0.04\}$, followed by target groups $\{0.03, 0.025\}$, $\{0.02, 0.015\}$, and $\{0.0125, 0.01\}$. Each PINN receives 50,000 physics updates using AdamW with initial learning rate 10^{-3} and exponential decay 0.9 every 2,000 steps, sampling 128 interior points and 64 points per initial/boundary component. Each target receives 10,000 supervised updates using Adam with decay 0.9 every 2,500 steps. Each operator receives 3,000 updates per group on all current support fields using AdamW with decay 0.95 every 30 steps; both initial learning rates are 10^{-3}.

Operators predict on a 16^3 grid. Filtering retains the 30% of points with the lowest combined momentum and divergence residual scores, using fourth-order periodic spatial differences and second-order interior temporal differences with one-sided endpoint treatment. Only retained coordinate–value pairs enter supervised initialization. Final velocity error is $\|\widehat{\boldsymbol{u}} - \boldsymbol{u}\|_2 / \|\boldsymbol{u}\|_2$, where $\boldsymbol{u} = (u, v)$, evaluated on a $48 \times 48 \times 9$ grid. The Vanilla PINN receives 50,000 physics updates at the target only, whereas NOCL also trains the support and intermediate PINNs; this is an accuracy comparison, not a matched-compute comparison.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All our claims are supported by theoretical or experimental validation.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We identified a manually designed hyperparameter and honestly list it as future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Corollary 3.1 and Theorem 3.2 both state their assumptions explicitly, with full proofs provided in Appendices A and B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We report the full set of hyperparameters required for the experiments in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The NOCL code, configurations, and reproduction instructions for Convection and the rebuttal reaction–diffusion experiment are available at `https://github.com/windindicator/NOCL-NeurIPS2026`. Training points and reference solutions are generated by the code. To keep this release focused on the core method, the other PDE experiments and baseline implementations are omitted.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental details are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our experimental results are obtained over multiple random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the environment and computational resources in the experimental section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This work focuses on algorithmic improvements for PDE-based prediction models in abstract scientific computing scenarios.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The authors have properly credited and are the license and terms of use explicitly mentioned and properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code includes installation and execution instructions, experiment configurations, protocol descriptions, reference results, and an MIT license in the linked repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.