# Diagnosing Failure Modes of Neural Operators Across Diverse PDE Families

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Neural PDE solvers have shown strong performance on standard benchmarks, but their robustness under deployment-relevant distribution shifts remains insufficiently characterized. We present a systematic stress-testing framework for evaluating neural PDE solvers across five qualitatively different PDE families—dispersive, elliptic, multi-scale fluid, financial, and chaotic systems—under controlled shifts in parameters, boundary or terminal conditions, resolution, rollout horizon, and input perturbations. The framework is instantiated on three representative architectures: Fourier Neural Operators (FNOs), a DeepONet-style model, and convolutional neural operators (CNOs). Across 750 trained models, we evaluate robustness using baseline-normalized degradation factors together with spectral and rollout diagnostics. This setup is designed to distinguish failure patterns that are shared across architectures from those that are architecture- or PDE-specific. Overall, the paper is framed as an evaluation study rather than a new architecture paper, with the goal of providing a clearer basis for assessing robustness claims in neural PDE solvers.

## 1 Introduction

Neural PDE solvers learn mappings between function spaces, enabling fast surrogates for families of partial differential equations rather than solving individual discretized instances (Kovachki et al., 2021). Architectures such as the Fourier Neural Operator (FNO) (Li et al., 2021), DeepONet (Lu et al., 2021), and more recent variants like convolutional neural operators (CNOs) (Raonić et al., 2023) have demonstrated strong performance on standard benchmarks. However, these models are typically evaluated under closely matched train–test distributions, limited perturbation settings, and short temporal horizons, leaving their robustness under deployment-relevant shift largely unexplored.

This limitation is significant in practice, where models are routinely exposed to changes in parameters, boundary or terminal conditions, discretization, and rollout length. In such settings, strong in-distribution accuracy does not guarantee reliable behavior. Similar concerns have been observed in scientific machine learning more broadly, including documented failure modes in physics-informed neural networks (Krishnapriyan et al., 2021). While recent operator-learning work increasingly emphasizes robustness and generalization, evaluation remains inconsistent and often architecture-specific.

In this work, we shift the focus from in-distribution accuracy to systematic robustness evaluation. We develop a stress-testing framework that evaluates neural PDE solvers under controlled shifts in parameters, boundary or terminal conditions, resolution, rollout horizon, and input perturbations. Rather than introducing a new architecture, our goal is to provide a unified setting for measuring and comparing robustness across models and PDE families.

We instantiate this framework on three representative architectures—FNO, a DeepONet-style model, and a CNO—capturing distinct inductive biases and enabling analysis of which failure patterns are shared across architectures and which are architecture-dependent.

We evaluate these architectures on five PDE families spanning qualitatively different regimes:

- **Nonlinear Schrödinger equation (NLS):** a dispersive, complex-valued wave equation where phase accuracy and long-time stability are central.
- **Poisson equation:** a canonical elliptic PDE strongly shaped by coefficient structure and boundary conditions.
- **Navier–Stokes equations (2D incompressible, vorticity form):** a multi-scale fluid system sensitive to viscosity, resolution, and rollout stability.
- **Black–Scholes equation:** a parabolic PDE from quantitative finance with terminal-condition dependence and sensitivity to volatility and payoff regularity.
- **Kuramoto–Sivashinsky (K–S) equation:** a chaotic fourth-order PDE with sensitive dependence on initial conditions and rollout error.

The framework supports analysis at multiple levels: it measures degradation relative to in-distribution baselines, includes stressor-specific diagnostics such as spectral error analysis and rollout amplification, and enables comparison across both architectures and PDE families.

Our contributions are:

**(1)** We introduce a systematic stress-testing framework for evaluating neural PDE solvers under deployment-relevant shifts.

**(2)** We instantiate this framework across FNO, DeepONet-style, and CNO architectures on five PDE families under a common protocol.

**(3)** We combine degradation-based metrics with spectral and rollout diagnostics to characterize where robustness breaks down.

**(4)** We use the framework to distinguish shared, architecture-dependent, and PDE-specific failure patterns.

Overall, the paper is an evaluation-framework study rather than a new architecture paper. Its aim is to provide a clearer basis for assessing robustness claims in neural PDE solvers and a reusable protocol for future comparisons.

## 2 Related Work

Neural operator methods learn mappings between function spaces rather than solution fields tied to a single discretization. Foundational work in this direction includes the general neural operator framework of Kovachki et al. (2021), the Fourier Neural Operator (FNO) of Li et al. (2021), and DeepONet (Lu et al., 2021), which established operator learning as a central paradigm for PDE surrogate modeling. These methods showed that learned operators can be discretization-aware or discretization-invariant and can achieve strong benchmark performance across canonical PDE families.

More recent work has focused on improving robustness, generalization, and physical consistency. Physics-Informed Neural Operators (PINO) combine operator learning with PDE residual constraints to improve fidelity under limited data and resolution extrapolation (Li et al., 2023). Convolutional Neural Operators (CNOs) provide a structure-preserving alternative to spectral architectures, emphasizing continuous–discrete equivalence and improved robustness across resolutions and distributions (Raonić et al., 2023). At larger scale, FourCastNet illustrates the practical importance of long-horizon stability and robustness for operator-style models in complex spatiotemporal forecasting settings (Pathak et al., 2022).

At the same time, understanding when scientific machine learning models fail remains an open problem. In the PINN literature, Krishnapriyan et al. (2021) showed that failure can arise from optimization and conditioning difficulties rather than lack of expressivity. More broadly, spectral bias remains a well-known limitation of neural networks, especially when high-frequency structure must be recovered under resolution shift or multi-scale dynamics (Tancik et al., 2020). Within PDE learning, related concerns also appear

when robustness claims are made primarily from in-distribution evaluations or limited perturbation settings (Karniadakis et al., 2021; Li et al., 2023; Raonić et al., 2023).

A particularly important but under-emphasized issue is conditional generalization under varying boundary or terminal conditions. Recent work has argued that, when such conditions vary across samples, neural PDE solvers are often better interpreted as learning boundary-indexed families of operators rather than a single boundary-agnostic solution map (Shikhman, 2026). This perspective helps explain why strong interpolation in forcing terms, coefficients, or discretization does not necessarily imply robustness under boundary-condition shift.

Taken together, prior work has produced increasingly expressive operator architectures and several mechanisms for improving generalization or physical consistency (Li et al., 2021; Lu et al., 2021; Kovachki et al., 2021; Li et al., 2023; Raonić et al., 2023). What remains less developed is a unified evaluation framework that probes deployment-relevant failure modes across multiple PDE families, architectures, and stress conditions. Our work is aimed at that gap: we develop a systematic stress-testing framework for comparing the robustness of neural PDE solvers under controlled shifts in parameters, boundary or terminal conditions, resolution, rollout horizon, and input perturbations.

## 3 Methodology

### 3.1 Neural PDE Solver Architectures and Training Setup

We evaluate three neural PDE solver architectures under a common stress-testing protocol: the Fourier Neural Operator (FNO) (Li et al., 2021), DeepONet (Lu et al., 2021), and a convolutional neural operator (CNO) (Raonić et al., 2023). These models represent distinct inductive biases: spectral convolution in FNO, branch–trunk operator approximation in DeepONet, and localized multi-scale convolution in CNO. The purpose of this paper is to demonstrate and validate a reusable stress-testing framework for neural PDE solvers, not to establish a definitive architecture leaderboard or a final benchmark comparison among these models. We therefore use representative, competent implementations under a common evaluation protocol to illustrate how the framework surfaces robustness differences across architectures, PDE families, and stress conditions.

For each PDE family, we construct paired input–output datasets from an in-distribution regime. Inputs include the relevant forcing terms, coefficients, initial conditions, payoff functions, or parameter channels, depending on the problem. Outputs are either solution fields for static PDEs or short-horizon targets for time-dependent PDEs. For PDEs with known scalar parameters, such as viscosity $\nu$ in Navier–Stokes, nonlinearity $\kappa$ in NLS, or volatility $\sigma$ in Black–Scholes, the parameter value is included as an additional input feature.

All models are trained on the same data-generation pipeline for a given PDE and then evaluated post hoc under the same stress-testing framework. We use a common training pipeline and broadly comparable model scales where possible, while allowing minor architecture-specific choices needed for stable optimization. Accordingly, the objective is not to maximize absolute performance for each architecture separately, but to obtain competent in-distribution models that can be stress-tested under controlled distribution shift. When this framework is used for claims about the superiority of a new architecture, a more exhaustive fairness-oriented comparison, including careful architecture-specific tuning and matched experimental budgets, should be carried out.

### 3.2 Stress-Testing Protocol

After training on the in-distribution regime, each model is evaluated under five classes of deployment-relevant stress:

(A) **Parameter and coefficient shifts:** PDE parameters are moved beyond the training range, such as larger $\kappa$ in NLS, lower viscosity $\nu$ in Navier–Stokes, higher volatility $\sigma$ in Black–Scholes, or rougher coefficients in Poisson.

(B) **Boundary or terminal-condition shifts:** Models are tested on boundary or payoff families that differ substantially from those seen during training, probing conditional generalization beyond the training support.

(C) **Resolution extrapolation:** Models trained at one discretization are evaluated at finer resolutions. We complement global error measurements with spectral diagnostics to assess whether additional error is concentrated in high-frequency modes.

(D) **Long-horizon rollout:** For time-dependent PDEs, one-step predictors are iteratively composed and compared against the true trajectory over longer horizons, revealing error accumulation and dynamical instability.

(E) **Input perturbation sensitivity:** Small perturbations are added to inputs to test whether prediction error is amplified or remains approximately stable under local input variation.

Not every stressor applies to every PDE family, but the evaluation protocol is fixed within each PDE and applied consistently across architectures. All stress tests are performed without fine-tuning on the shifted regime.

### 3.3 Multi-Seed Evaluation and Metrics

To reduce sensitivity to initialization and stochastic training effects, we use a multi-seed protocol. For each architecture and PDE family, we train 50 independent models, yielding a total of 750 trained models across 3 architectures and 5 PDE families. Every trained model is evaluated on the same PDE-specific family of stress tests.

Our primary summary statistic is the degradation factor, defined as the ratio between stressed and baseline error for a given model. Let $E^{(i)}_{\text{base}}$ denote the baseline relative $L^2$ error of model $i$ on in-distribution test data, and let $E^{(i)}_{\text{stress}}$ denote the worst-case error over the stress grid for a particular scenario. We define

$$D^{(i)} = \frac{E^{(i)}_{\text{stress}}}{E^{(i)}_{\text{base}}}.$$

This normalizes for absolute performance differences and measures how strongly a given stressor degrades accuracy.

We report means and 95% confidence intervals for degradation factors across seeds. In addition to degradation, we also analyze complementary diagnostics tailored to particular stressors, including absolute baseline error, rollout growth and amplification measures, and spectral error summaries under resolution shift.

### 3.4 Experimental Setup

We evaluate FNO, DeepONet, and CNO models on the same synthetic data-generation pipeline and stress-testing framework. Training and evaluation are deterministic for each seed. For each architecture–PDE pair, we train 50 independent models, yielding 750 trained models in total.

The training settings vary by PDE family. For Poisson, we train on 512 samples at resolution $n = 128$ with batch size 8 for 3000 optimization steps. For Black–Scholes, we use 512 samples at $n = 256$, batch size 8, and 3000 steps. For nonlinear Schrödinger, we use 256 samples at $n = 256$ and $n_t = 20$, batch size 4, and 4000 steps. For Navier–Stokes, we use 128 samples at $n = 64$ and $n_t = 20$, batch size 2, and 5000 steps. For Kuramoto–Sivashinsky, we use 256 samples at $n = 128$ and $n_t = 20$, batch size 8, and 3000 steps. Across all experiments, models are trained with Adam at learning rate $10^{-3}$.

Model sizes are architecture-appropriate but broadly comparable. FNO uses width 64 and depth 4, with 16 Fourier modes in 1D and 12×12 modes in 2D. CNO uses width 64 and depth 5. The DeepONet-style model uses width 128 and depth 2. All models use coordinate channels. Baseline evaluation is performed on 64 in-distribution test samples per seed; for time-dependent PDEs, baseline rollout summaries use 5 rollout steps. Stress-test grids are fixed within each PDE and shared across architectures.

# 4 Results

The results show that robustness cannot be inferred from baseline accuracy, nor from any single PDE or stress condition. While FNO achieves the lowest in-distribution error across all five PDE families, its advantage is not stable under shift. Instead, robustness depends jointly on the equation class and the type of perturbation applied, and model rankings frequently change across these dimensions.

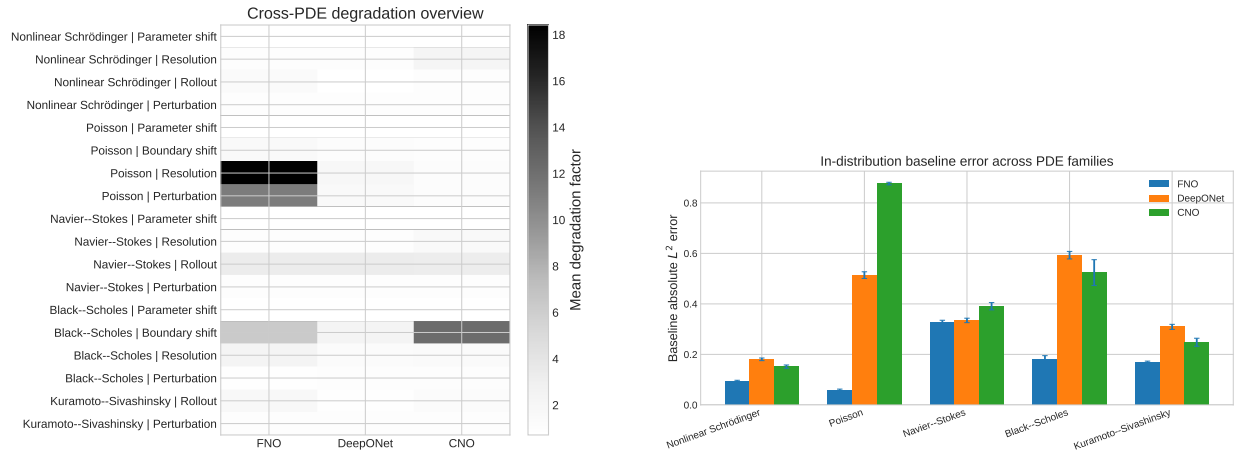


Figure 1: Left: mean degradation factors across PDE families, architectures, and stress tests. Darker cells indicate more severe degradation. Right: in-distribution baseline absolute $L^2$ error. FNO is strongest at baseline, but robustness patterns are far less uniform.

Figure 1 highlights two key observations. First, baseline rankings do not predict robustness under shift. Second, the dominant failure mode varies by PDE: Poisson is driven by perturbation and resolution sensitivity, Navier–Stokes by rollout instability, Black–Scholes by payoff-family shift, and nonlinear Schrödinger by parameter extrapolation. No single test captures this variability.

## 4.1 Nonlinear Schrödinger and Poisson

For nonlinear Schrödinger, parameter shift in $\kappa$ is a shared difficulty across all three architectures, but resolution transfer separates them sharply. FNO and DeepONet remain essentially unchanged, while CNO degrades substantially. This contrast shows that some weaknesses are intrinsic to the problem, while others depend strongly on model design.

Poisson provides the most striking reversal. FNO is clearly best at baseline, yet becomes the least robust model under perturbation and resolution shift. CNO shows the opposite behavior, remaining stable across all stressors despite the worst baseline error. This inversion would be invisible under standard in-distribution evaluation and illustrates how model rankings depend on the intended deployment regime.

## 4.2 Navier–Stokes and Black–Scholes

Navier–Stokes does not produce a strong ranking inversion. Instead, it reveals a shared limitation: rollout instability. Parameter and perturbation shifts are mild for all models, and resolution transfer only moderately separates them. However, all three degrade substantially under long-horizon rollout. Here the main takeaway is not which model is best, but that current architectures struggle with iterative prediction regardless of baseline performance.

Black–Scholes exhibits a different pattern. The primary challenge is conditional generalization under payoff shift. DeepONet is consistently the most robust model across volatility, payoff, and resolution changes, despite having the worst baseline error. CNO, in contrast, fails severely under payoff shift. This indicates

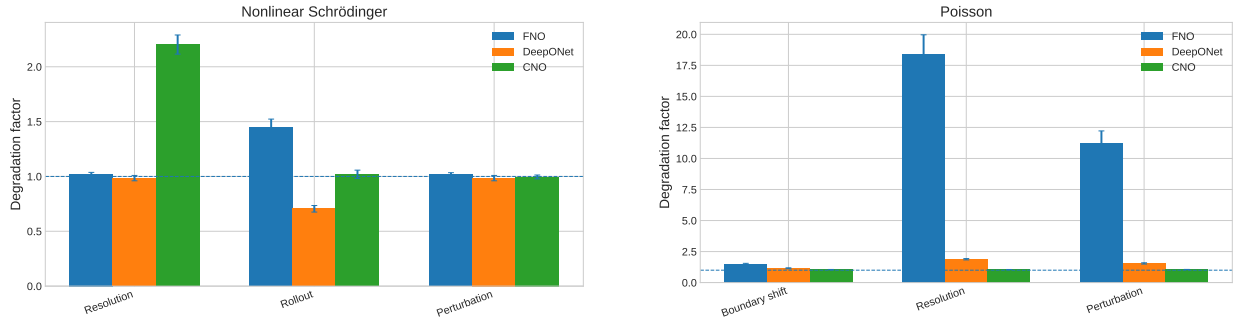


Figure 2: Left: nonlinear Schrödinger. Parameter shift is broadly difficult, while resolution transfer is strongly architecture-dependent. Right: Poisson. FNO is best at baseline but least robust under perturbation and resolution shift, whereas CNO is the most stable.

that different PDE families probe different generalization mechanisms: dynamical stability for Navier–Stokes versus functional extrapolation for Black–Scholes.

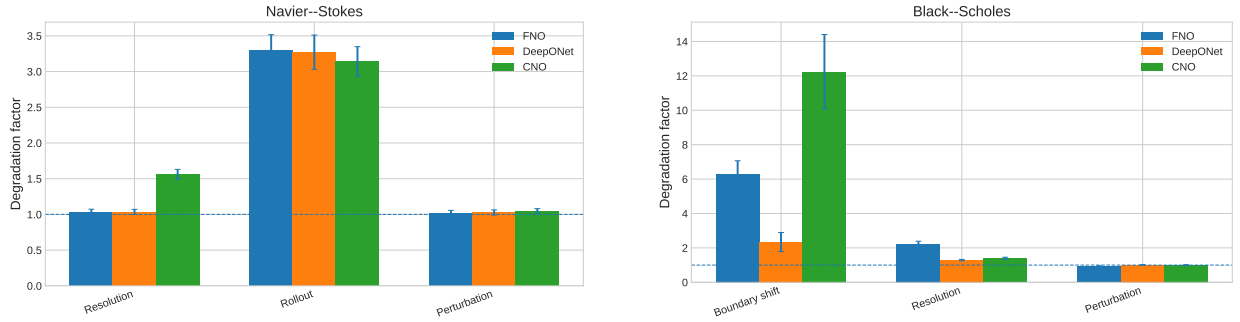


Figure 3: Left: Navier–Stokes. Rollout instability dominates and affects all architectures. Right: Black–Scholes. DeepONet is most robust under payoff and volatility shift, while CNO exhibits severe degradation.

### 4.3 Kuramoto–Sivashinsky

Kuramoto–Sivashinsky isolates rollout behavior in a chaotic setting. FNO has the best baseline accuracy but the worst rollout degradation, while DeepONet shows the opposite pattern. This decoupling indicates that one-step accuracy and long-horizon stability are only weakly related, even in a simplified two-stressor setting.

### Takeaways

Three patterns emerge. First, baseline accuracy is not a reliable proxy for robustness. Second, some weaknesses are shared (notably rollout instability in Navier–Stokes), while others are highly architecture-dependent (e.g., Poisson resolution sensitivity or Black–Scholes payoff shift). Third, robustness varies significantly across PDE families: the stressor that matters most is problem-dependent.

Taken together, these results show that robustness claims do not transfer cleanly across equations or shift types. Evaluating models along a single axis—whether baseline error, one PDE, or one perturbation—can give a misleading picture of performance under realistic deployment conditions.

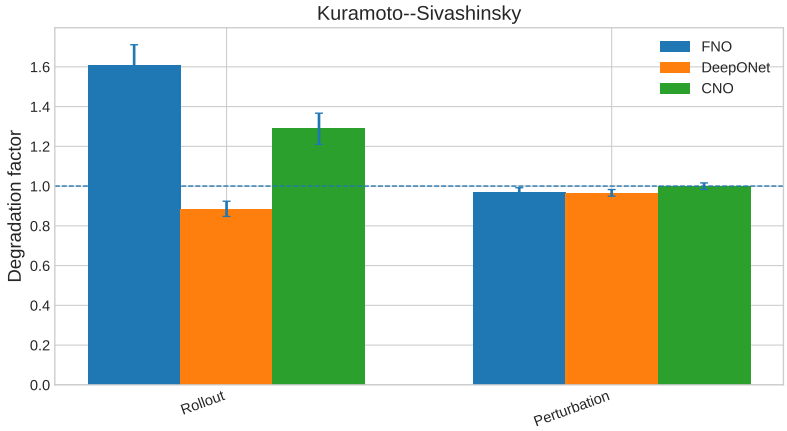


Figure 4: Kuramoto–Sivashinsky. The best baseline model and the most stable rollout model are different.

## 5 Discussion

The main lesson of this study is that robustness in neural PDE solvers is not a single property. It depends on the PDE family, the type of shift, and the architecture. This is why baseline accuracy alone is not enough: FNO is strongest in-distribution across all five PDE families, yet it is not uniformly the most robust under stress. In contrast, DeepONet is often more stable under the most important shifts, while CNO shows the highest variance across settings, performing extremely well in some cases and poorly in others. The main implication is that model choice depends not just on interpolation accuracy, but on which kinds of failures matter in deployment.

A second lesson is that some weaknesses are shared, while others are architecture-specific. Navier–Stokes rollout degradation is the clearest shared limitation: all three architectures deteriorate substantially under long-horizon prediction. By contrast, Poisson produces a strong ranking reversal, where FNO is best at baseline but least robust under perturbation and resolution shift, while CNO is the most stable. Black–Scholes exposes a different type of failure, centered on payoff-family generalization rather than rollout, and there DeepONet is the most robust architecture. These contrasts suggest that robustness should be understood as a structured profile rather than a single score.

The results also clarify what the current evaluation can and cannot claim. The five PDE families were chosen to span qualitatively different regimes, but they still represent only a small sample of the space of PDEs that a practitioner may care about. The goal is therefore not to claim universal coverage, but to provide a reusable testing setup that can be adapted to the application. For a specialized solver, a user can replace or extend the PDE suite with equation families that better match the target domain while keeping the same stress-based evaluation logic. In that sense, the value of the study is not only the particular benchmark we report, but also the general recipe for constructing comparable robustness evaluations.

For practitioners, the main implication is straightforward: a model that looks best on standard test error may not be the safest choice once the operating regime shifts. For researchers, the results point to several directions for improvement, including better multi-resolution representations, stronger handling of boundary or terminal-condition variation, and architectures or training schemes that improve long-horizon stability. More broadly, the study suggests that future claims about robust neural PDE solving should be tied to explicit stress conditions and equation families rather than to in-distribution accuracy alone.

## 6 Conclusion

We presented a comparative stress-based evaluation of neural PDE solvers across five PDE families, three architectures, and multiple deployment-relevant shifts. The main result is that robustness is not well described

by baseline accuracy alone. Although FNO achieved the lowest in-distribution error across all five PDE families, robustness rankings changed substantially under shift. DeepONet was often the most consistently stable model, while CNO showed the greatest variability across equation classes, performing very well in some settings and poorly in others.

More broadly, the results show that robustness depends strongly on both the PDE family and the type of shift being considered. Some weaknesses are shared, such as rollout instability in Navier–Stokes, while others are highly architecture-specific, such as Poisson sensitivity to perturbation and resolution or Black–Scholes sensitivity to payoff-family shift. This means that no single benchmark, PDE, or perturbation is sufficient for assessing reliability in neural PDE solvers.

The broader contribution of the paper is therefore not only the empirical comparison itself, but also the evaluation template it provides. The five PDE families used here span qualitatively different regimes, but they are still only a small sample of the equations a practitioner may care about. The same stress-based setup can be adapted to other PDE collections that better match a target application, making the methodology reusable beyond the specific benchmark reported here.

Taken together, these results suggest that future claims about robust neural PDE solving should be grounded in explicit testing across multiple shift types and equation families rather than in in-distribution accuracy alone. We hope this study provides a clearer basis for such evaluation and helps move the field toward more reliable and application-aware neural PDE solvers.

## References

George Karniadakis, Yannis Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, pp. 1–19, 05 2021. doi: 10.1038/s42254-021-00314-5.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. doi: 10.48550/arXiv.2109.01050. URL `https://arxiv.org/abs/2109.01050`.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *International Conference on Learning Representations (ICLR)*, 2021.

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

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karth Kashinath, and Anima Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Bogdan Raonić, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes, 2023. URL `https://arxiv.org/abs/2302.01178`.

Lennon Shikhman. One operator to rule them all? on boundary-indexed operator families in neural PDE solvers. In *AI&PDE: ICLR 2026 Workshop on AI and Partial Differential Equations*, 2026. URL `https://openreview.net/forum?id=lDjWQ9UxRy`.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, and Jonathan T. Barron. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:7537–7547, 2020.

## Acknowledgements

**Reproduciblity.** Code to reproduce all experiments, generate figures, and compute degradation metrics is available at `REDACTED`.

**Computational Resources.** The authors gratefully acknowledge the provision of computational resources that supported the experiments in this work. All experiments were conducted on a high-performance workstation equipped with modern CPU and GPU hardware. The views and conclusions expressed herein are those of the authors and do not necessarily reflect the views of the supporting organization.

## A Detailed Degradation Statistics

This appendix reports quantitative summaries of the degradation factors underlying the figures in the main text. For each architecture–PDE pair, we train 50 independent models and report the mean degradation factor, standard deviation, and 95% confidence interval across those 50 runs. Across runs, models differ in initialization and in the randomly generated training and evaluation data drawn from the same in-distribution regime. Unless otherwise noted, degradation factors correspond to the *worst-case* error over the fixed stress-test grid for each run (for example, the maximum degradation across parameter values or rollout horizons), yielding a conservative estimate of failure severity.

### A.1 Poisson Equation

Table 1: Poisson degradation summary (50 runs per architecture).

| Architecture | Stress test | Mean | Std | $CI_{95\%}$ (low) | $CI_{95\%}$ (high) |
|---|---|---|---|---|---|
| FNO | Parameter shift ($a$ scale) | 1.994 | 0.449 | 1.870 | 2.118 |
| | Boundary shift | 1.498 | 0.194 | 1.444 | 1.551 |
| | Resolution extrapolation | 18.417 | 5.572 | 16.872 | 19.961 |
| | Input perturbation | 11.199 | 3.682 | 10.178 | 12.219 |
| DeepONet | Parameter shift ($a$ scale) | 1.265 | 0.286 | 1.186 | 1.344 |
| | Boundary shift | 1.155 | 0.129 | 1.119 | 1.190 |
| | Resolution extrapolation | 1.882 | 0.195 | 1.828 | 1.936 |
| | Input perturbation | 1.545 | 0.184 | 1.494 | 1.596 |
| CNO | Parameter shift ($a$ scale) | 1.087 | 0.136 | 1.049 | 1.124 |
| | Boundary shift | 1.049 | 0.035 | 1.039 | 1.058 |
| | Resolution extrapolation | 1.051 | 0.018 | 1.046 | 1.056 |
| | Input perturbation | 1.055 | 0.024 | 1.048 | 1.062 |

### A.2 Nonlinear Schrödinger Equation

Table 2: Nonlinear Schrödinger degradation summary (50 runs per architecture).

| Architecture | Stress test | Mean | Std | $CI_{95\%}$ (low) | $CI_{95\%}$ (high) |
|---|---|---|---|---|---|
| FNO | Nonlinearity shift ($\kappa$) | 3.872 | 0.961 | 3.605 | 4.138 |
| | Resolution extrapolation | 1.022 | 0.053 | 1.008 | 1.037 |
| | Long-horizon rollout | 1.449 | 0.265 | 1.376 | 1.523 |
| | Input perturbation | 1.020 | 0.052 | 1.005 | 1.034 |
| DeepONet | Nonlinearity shift ($\kappa$) | 2.227 | 0.503 | 2.087 | 2.366 |
| | Resolution extrapolation | 0.985 | 0.088 | 0.960 | 1.009 |
| | Long-horizon rollout | 0.705 | 0.108 | 0.675 | 0.735 |
| | Input perturbation | 0.985 | 0.088 | 0.960 | 1.009 |
| CNO | Nonlinearity shift ($\kappa$) | 2.618 | 0.610 | 2.448 | 2.787 |
| | Resolution extrapolation | 2.203 | 0.315 | 2.115 | 2.290 |
| | Long-horizon rollout | 1.020 | 0.136 | 0.982 | 1.057 |
| | Input perturbation | 0.995 | 0.064 | 0.978 | 1.013 |

### A.3 Navier–Stokes Equation

Table 3: Navier–Stokes degradation summary (50 runs per architecture).

| Architecture | Stress test | Mean | Std | $CI_{95\%}$ (low) | $CI_{95\%}$ (high) |
|---|---|---|---|---|---|
| FNO | Viscosity shift ($\nu$) | 1.028 | 0.141 | 0.989 | 1.067 |
| | Resolution extrapolation | 1.034 | 0.138 | 0.996 | 1.072 |
| | Long-horizon rollout | 3.296 | 0.792 | 3.077 | 3.516 |
| | Input perturbation | 1.018 | 0.139 | 0.979 | 1.057 |
| DeepONet | Viscosity shift ($\nu$) | 1.036 | 0.131 | 1.000 | 1.073 |
| | Resolution extrapolation | 1.035 | 0.130 | 0.999 | 1.071 |
| | Long-horizon rollout | 3.271 | 0.866 | 3.031 | 3.511 |
| | Input perturbation | 1.027 | 0.131 | 0.991 | 1.063 |
| CNO | Viscosity shift ($\nu$) | 1.056 | 0.111 | 1.026 | 1.087 |
| | Resolution extrapolation | 1.562 | 0.248 | 1.493 | 1.630 |
| | Long-horizon rollout | 3.138 | 0.759 | 2.928 | 3.348 |
| | Input perturbation | 1.050 | 0.111 | 1.019 | 1.081 |

### A.4 Black–Scholes Equation

Table 4: Black–Scholes degradation summary (50 runs per architecture).

| Architecture | Stress test | Mean | Std | $CI_{95\%}$ (low) | $CI_{95\%}$ (high) |
|---|---|---|---|---|---|
| FNO | Volatility shift ($\sigma$) | 2.068 | 0.656 | 1.886 | 2.250 |
| | Payoff structure shift | 6.308 | 2.733 | 5.550 | 7.065 |
| | Resolution extrapolation | 2.236 | 0.552 | 2.083 | 2.389 |
| | Input perturbation | 0.912 | 0.170 | 0.864 | 0.959 |
| DeepONet | Volatility shift ($\sigma$) | 1.272 | 0.118 | 1.239 | 1.304 |
| | Payoff structure shift | 2.338 | 2.005 | 1.782 | 2.894 |
| | Resolution extrapolation | 1.287 | 0.154 | 1.244 | 1.330 |
| | Input perturbation | 1.012 | 0.059 | 0.995 | 1.028 |
| CNO | Volatility shift ($\sigma$) | 1.357 | 0.203 | 1.301 | 1.413 |
| | Payoff structure shift | 12.239 | 7.805 | 10.076 | 14.403 |
| | Resolution extrapolation | 1.395 | 0.229 | 1.331 | 1.458 |
| | Input perturbation | 0.997 | 0.096 | 0.970 | 1.024 |

### A.5 Kuramoto–Sivashinsky Equation

Table 5: Kuramoto–Sivashinsky degradation summary (50 runs per architecture).

| Architecture | Stress test | Mean | Std | $CI_{95\%}$ (low) | $CI_{95\%}$ (high) |
|---|---|---|---|---|---|
| FNO | Long-horizon rollout | 1.607 | 0.376 | 1.503 | 1.711 |
| | Input perturbation | 0.967 | 0.091 | 0.941 | 0.992 |
| DeepONet | Long-horizon rollout | 0.886 | 0.138 | 0.847 | 0.924 |
| | Input perturbation | 0.966 | 0.060 | 0.949 | 0.983 |
| CNO | Long-horizon rollout | 1.288 | 0.283 | 1.210 | 1.367 |
| | Input perturbation | 1.000 | 0.059 | 0.983 | 1.016 |