Forcing and Diagnosing Failure Modes of Fourier Neural Operators Across Diverse PDE Families

Anonymous authors
Paper under double-blind review

Abstract

Fourier Neural Operators (FNOs) have shown strong performance in learning solution maps of partial differential equations (PDEs), but their robustness under distribution shifts, long-horizon rollouts, and structural perturbations remains poorly understood. We present a systematic stress-testing framework that probes failure modes of FNOs across five qualitatively different PDE families: dispersive, elliptic, multi-scale fluid, financial, and chaotic systems. Rather than optimizing in-distribution accuracy, we design controlled stress tests–including parameter shifts, boundary or terminal condition changes, resolution extrapolation with spectral analysis, and iterative rollouts–to expose vulnerabilities such as spectral bias, compounding integration errors, and overfitting to restricted boundary regimes. Our large-scale evaluation (1,000 trained models) reveals that distribution shifts in parameters or boundary conditions can inflate errors by more than an order of magnitude, while resolution changes primarily concentrate error in high-frequency modes. Input perturbations generally do not amplify error, though worst-case scenarios (e.g., localized Poisson perturbations) remain challenging. These findings provide a comparative failure-mode atlas and actionable insights for improving robustness in operator learning.

Neural operators, Fourier Neural Operator, operator learning, partial differential equations, scientific machine learning, out-of-distribution generalization, robustness analysis, failure modes, stress testing, spectral bias, distribution shift, dynamical systems

1 Introduction

Neural operators directly learn the solution operator that maps one function to another in infinite-dimensional spaces, providing an effective alternative to classical numerical PDE solvers (Kovachki et al., 2021). Architectures such as the Fourier Neural Operator (FNO) (Li et al., 2021) and DeepONet (Lu et al., 2021) have shown impressive accuracy on canonical problems (e.g. Darcy flow, Burgers' equation, Navier–Stokes turbulence) under idealized settings where training and testing distributions are closely matched. However, most evaluations to date assume fixed problem settings (same equations, domains, and parameter ranges) and short temporal horizons. Related concerns have been raised for physics-informed neural networks (PINNs), where systematic studies have shown sensitivity to optimization pathologies, stiffness, and problem conditioning, motivating targeted analyses of failure modes in scientific machine learning models (Krishnapriyan et al., 2021). In realistic scientific and engineering applications, PDE solvers are routinely exposed to changes in coefficients, forcing terms, boundary or terminal conditions, spatial resolution, and temporal scale. A deployed neural simulator may need to handle a fluid with a different viscosity than seen in training, a financial model with a new payoff structure, or a longer-term forecast than it was calibrated for. Each of these shifts can push the model beyond its learned regime, potentially triggering large errors or instabilities.

In this work, we shift focus from optimizing in-distribution accuracy to intentionally inducing failure outside the training comfort zone. By systematically subjecting neural operators to a variety of stressors, we seek

This work used LLM assistance only for tasks such as incorporating GPU acceleration and efficiently adding comments to code. No core content was created by AI.

to answer: How and why do these models fail when nominal assumptions are violated? We study multiple PDE families with distinct mathematical structure to distinguish universal failure modes from those that are PDE-specific. Our approach is inspired by stress-testing and adversarial analysis in machine learning, but applied in the context of scientific machine learning where the "perturbations" correspond to meaningful changes in the physical problem.

We consider five prototypical PDEs representing a broad range of dynamics:

- Nonlinear Schrödinger equation (NLS): A dispersive, complex-valued wave equation ($i\,u_t + u_{xx} - \kappa\,|u|^2 u = 0$) where maintaining phase accuracy and long-time stability is critical.
- Poisson equation: A canonical elliptic PDE ($-\nabla \cdot (a(x)\,\nabla u(x)) = f(x)$) dominated by boundary conditions and global constraints.
- Navier–Stokes equations (2D incompressible, vorticity form): A multi-scale, advection-dominated fluid system ($\omega_t + \mathbf{u} \cdot \nabla\omega = \nu\Delta\omega + s$) sensitive to viscosity and resolution (here $\mathbf{u}$ is the velocity field related to vorticity ω).
- Black–Scholes equation: A parabolic PDE from quantitative finance ($V_t + \frac{1}{2}\sigma^2 S^2 V_{SS} + rSV_S - rV = 0$) with a terminal condition (payoff at T) and strong dependence on parameters like volatility σ.
- Kuramoto–Sivashinsky (K–S) equation: A chaotic, fourth-order PDE ($u_t + u\,u_x + u_{xx} + u_{xxxx} = 0$) that exhibits spatiotemporal chaos and sensitive dependence on initial conditions.

These PDEs span dispersive, elliptic, multi-scale transient, financial, and chaotic regimes, allowing us to probe a variety of failure mechanisms. By analyzing neural operator performance across this spectrum, we aim to identify which failure modes are universal (architecture-induced) and which arise from particular PDE structure.

Our contributions are: (1) We develop a suite of systematic stress tests (parameter shifts, boundary/terminal condition shifts, resolution extrapolation with spectral error analysis, long-horizon rollouts, and input perturbation sensitivity) to interrogate neural operator robustness. (2) We apply these tests to FNO-based models (with occasional comparison to DeepONet) on the above five PDE families, revealing distinct failure patterns. (3) We present a comprehensive analysis linking observed failures to underlying causes such as spectral bias and compounding integration errors. Our findings provide practical guidelines for diagnosing and addressing robustness issues in operator learning. In particular, our results highlight the importance of training on diverse scenario distributions and suggest directions for architectural improvements (e.g. multi-scale representations, stability constraints) to mitigate identified failure modes.

2 Methodology

2.1 Neural Operator Models and Training Setup

We adopt the Fourier Neural Operator (FNO) (Li et al., 2021) as our primary model, due to its popularity and its known spectral inductive bias. The FNO learns an operator by applying repeated Fourier transforms and nonlinear transformations, effectively parameterizing convolution kernels in the frequency domain. Unless noted otherwise, a single FNO architecture is used for all experiments, with minor modifications to accommodate different input/output dimensionality (e.g. 1D vs 2D grids). The model has a moderate size (on the order of 10^5 trainable parameters) and a fixed number of Fourier modes (we use 16 modes per input dimension, which was sufficient to capture the training distributions). We focus on FNO to ensure consistency across experiments, though we also reference results with DeepONet (Lu et al., 2021) for comparison where informative.

Each PDE dataset consists of paired input and output functions. For instance, in Poisson, $a(x)$ and $f(x)$ serve as inputs and $u(x)$ as output; in Black–Scholes, the payoff function (terminal condition) is the input and the option price $V(t = 0, x)$ as output; in time-dependent PDEs like NLS, K–S, and Navier–Stokes, the initial condition (and possibly forcing) is input and the solution after a fixed time horizon is output. We

generate training data from an in-distribution regime specific to each PDE: for example, for NLS we sample initial wavefunctions with moderate amplitude (and fixed nonlinearity coefficient κ), for Navier–Stokes we sample flows with a certain viscosity ν and forcing statistics, and for Black–Scholes we use a family of smooth payoff functions (e.g. European call options with varying strike). The neural operator is trained on these datasets using a standard L^2 loss on the solution fields, with early stopping based on a validation set. We use the same training hyperparameters across PDEs as much as possible (learning rate, optimizer, etc.) to avoid unfair advantage on any particular equation. The goal is not to optimize absolute error for each problem, but to obtain a reasonably accurate model on the training distribution (baseline relative L^2 errors on test data are typically a few percent for each PDE) that we can then stress-test.

Importantly, for PDEs with known scalar parameters (such as viscosity ν in Navier–Stokes, nonlinearity κ in NLS, or volatility σ in Black–Scholes), we include the parameter value as an additional input feature to the model. This allows the FNO to learn the dependency on that parameter during training (within the training range). For example, the Black–Scholes FNO takes as input the payoff function plus a constant field encoding the volatility σ. This setup, in principle, permits generalization to new parameter values, though extrapolation beyond the training range remains challenging, as we will see.

2.2 Stress Testing Protocol

After training the neural operator on the in-distribution regime, we evaluate it on a suite of out-of-distribution scenarios designed to induce various failure modes:

(A) Parameter & coefficient shifts: Key PDE parameters are varied beyond the range seen in training. For instance, we test the NLS model on a larger nonlinearity κ than trained on, the Navier–Stokes model on lower viscosity ν (approaching a more turbulent regime), and the Black–Scholes model on higher volatility σ. For Poisson, we alter the coefficient field $a(x)$, e.g. introducing rougher (less smooth) coefficient variations than in training. These experiments probe the model's ability to extrapolate to fundamentally different equation settings.

(B) Boundary or terminal condition shifts: The model is trained on a restricted family of boundary conditions or terminal conditions and then tested on significantly different ones. In Poisson, for example, we train on a restricted distribution of Dirichlet boundary values and test on boundary value patterns that differ substantially from those seen during training. For Black–Scholes, we train on one class of payoff function (e.g. smooth payoff like a European call) and test on a drastically different payoff (e.g. a discontinuous payoff such as a digital option). This examines whether the operator has truly learned the PDE or simply interpolated within the training family of boundary/terminal conditions.

(C) Resolution extrapolation and spectral analysis: We investigate performance when the model is applied at a finer resolution than it saw during training. For spatially distributed PDEs, we train the FNO on solutions discretized on a base grid (e.g. 64×64 for Navier–Stokes) and then evaluate on a refined grid (e.g. 128×128) by interpolating the input onto the finer grid and feeding it through the model. The ground-truth solution is obtained on the fine grid for comparison. We expect errors to increase if the model fails to capture the smaller-scale solution features present at higher resolution. To diagnose spectral bias, we perform a Fourier decomposition of the error: we compute the error field (difference between predicted and true solution) and project it onto Fourier modes, examining the error energy as a function of wavenumber. This reveals whether new high-frequency components (only present on the fine grid) account for the bulk of error.

(D) Long-horizon rollout stability: For time-dependent PDEs, we test the model's stability by iteratively composing the learned operator over a long time horizon. During training, models for NLS, K–S, and Navier–Stokes may have effectively learned one-step (or short-horizon) mappings. We now apply the model repeatedly (feeding its output at one step as input for the next) to simulate further in time, and compare to the true solution over this extended rollout. Even if one-step error is low, errors can accumulate and interact with dynamical instabilities, leading to blow-up or qualitative divergence from the true trajectory. We measure the growth of error with time and define

a rollout degradation factor as the ratio of the long-horizon error to the short-horizon (single-step) error. Values significantly above 1 indicate compounding errors and instability.

(E) Input perturbation sensitivity: We examine the effect of small perturbations to the input on the output prediction. For each test instance (e.g. a given initial condition or coefficient field), we add a small random perturbation (noise of a few percent of the input's L^2 norm) and evaluate the model on this perturbed input. We then compute the relative error with respect to the true solution of the perturbed input and compare it to the error on the unperturbed input. The ratio of these errors is a measure of the model's sensitivity or stability: a value greater than 1 indicates that small input changes amplify the prediction error (an unstable mapping), while a value around 1 suggests approximate Lipschitz stability. We average this ratio over many random perturbations and instances to characterize overall sensitivity.

We emphasize that all stress tests are performed post hoc on a fixed trained model; we do not fine-tune the model on any of the shifted scenarios. This protocol reveals how a model trained in a standard way might behave when deployed in situations beyond its training envelope.

2.3 Multi-Seed Evaluation and Metrics

To ensure that our conclusions are not tied to a particular random initialization or training sample, we employ a multi-seed evaluation. Specifically, we train N independent instances of the neural operator (with different random seeds affecting weight initialization and mini-batch ordering) on each PDE. For each trained model, we evaluate it on a set of out-of-distribution test cases for each stress scenario described above. This yields a distribution of error metrics for each scenario across multiple runs. We report summary statistics including the mean and standard deviation of the error, and in particular, we focus on the error degradation factor, defined as the ratio of out-of-distribution error to the baseline in-distribution error. This factor directly quantifies the drop in accuracy due to the stressor: a value of 1.0 means no change in error, 2.0 means the error doubled under stress, etc.

Concretely, let $E^{(i)}_{\text{base}}$ be the baseline L^2 relative error of model i on a standard test set (drawn from the training distribution), and let $E^{(i)}_{\text{stress}}$ denote the worst-case error of the same model over the stress-test grid for a given scenario (e.g. across parameter values, resolutions, or rollout horizons). The degradation factor for that model and scenario is then

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

which yields a conservative estimate of failure severity for each seed.

We aggregate $D^{(i)}$ over $i = 1, \ldots, N$ (the N random seeds, here we use $N = 200$) to obtain an average degradation $\bar{D}$ with a confidence interval. This statistic has the advantage of normalizing out the absolute error scale of each model, focusing on the relative change. We present $\bar{D}$ for each stressor and PDE along with 95% confidence intervals (computed via a normal approximation $1.96\times$ standard error).

Unless otherwise stated, all reported results are aggregated over a large ensemble of independently trained models. For each PDE family, we train and evaluate 200 independent neural operators with different random initializations and data realizations, resulting in a total of 1,000 trained models across the five PDEs. This enables empirical estimation of the full distribution of failure severity under each stress test, rather than reliance on a small number of representative runs. All degradation factors are reported as means with 95% confidence intervals computed across these 200 seeds per PDE.

3 Results

We organize our results by PDE family, discussing the effect of each stress test in turn. Figure 1 provides an overview of the degradation factors for all combinations of PDEs and stressors (each plotted as a bar with error bars denoting the confidence interval). In the following, we delve into each PDE and highlight the key failure modes observed.

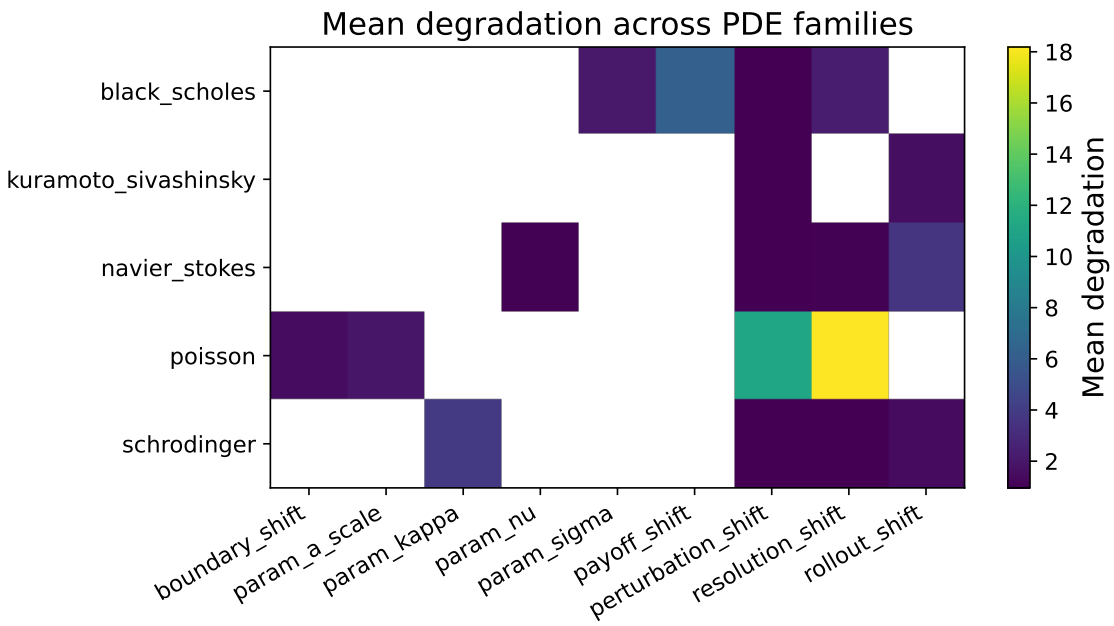

Figure 1: Mean error degradation factors across PDE families and stress tests. Each entry reports the mean out-of-distribution error normalized by the in-distribution baseline, aggregated across random seeds. Darker colors indicate more severe failure. White cells denote stress tests not applicable to a given PDE.

3.1 Nonlinear Schrödinger Equation

On the NLS equation, the FNO achieves good baseline accuracy for the training distribution (relative L^2 error $\sim 3\%$ on held-out samples with κ in the training range). However, when tested on a higher nonlinearity strength (κ significantly larger than seen in training), performance deteriorates drastically. The error increases by a factor of ≈ 3.9 (see Figure 1), indicating that the model fails to extrapolate the solution behavior at stronger nonlinear coupling. Physically, a larger κ leads to faster phase rotation and the formation of finer-scale interference patterns in the wavefunction $u(x,t)$, which the FNO (trained on smoother, weaker nonlinearity dynamics) cannot accurately represent. Interestingly, even though we provided κ as an input to the network, the model did not generalize to this regime, underscoring the difficulty of extrapolation in operator learning.

Next, we examine resolution extrapolation for NLS. We apply the NLS-trained model (trained on, say, 256 spatial grid points) to a finer grid (512 points) for the same physical problem. The overall error increases only slightly (degradation $\bar{D} \approx 1.01$), suggesting that the model's interpolation to a higher resolution introduces minimal additional error on average. However, spectral analysis of the error reveals an important nuance.

Figure 3 shows the Fourier-domain distribution of error for the fine-grid NLS predictions. We observe that the error energy is concentrated at the highest frequencies (near the Nyquist frequency of the training grid), whereas low-frequency modes (those within the training bandwidth) show negligible error increase. In other words, the FNO essentially reconstructs a band-limited version of the solution; when evaluated on a finer grid, it predicts the large-scale components well but fails to capture the new fine-scale oscillations that appear in the true solution. This is a hallmark of spectral bias (Tancik et al., 2020): the learned model prefers lower frequencies, and its generalization to higher resolution is achieved by smoothly interpolating without generating high-frequency details. As a result, the error remains small in aggregate (since high-frequency content often has low amplitude in this problem), but the model is fundamentally unable to produce the correct high-frequency components if they were required.

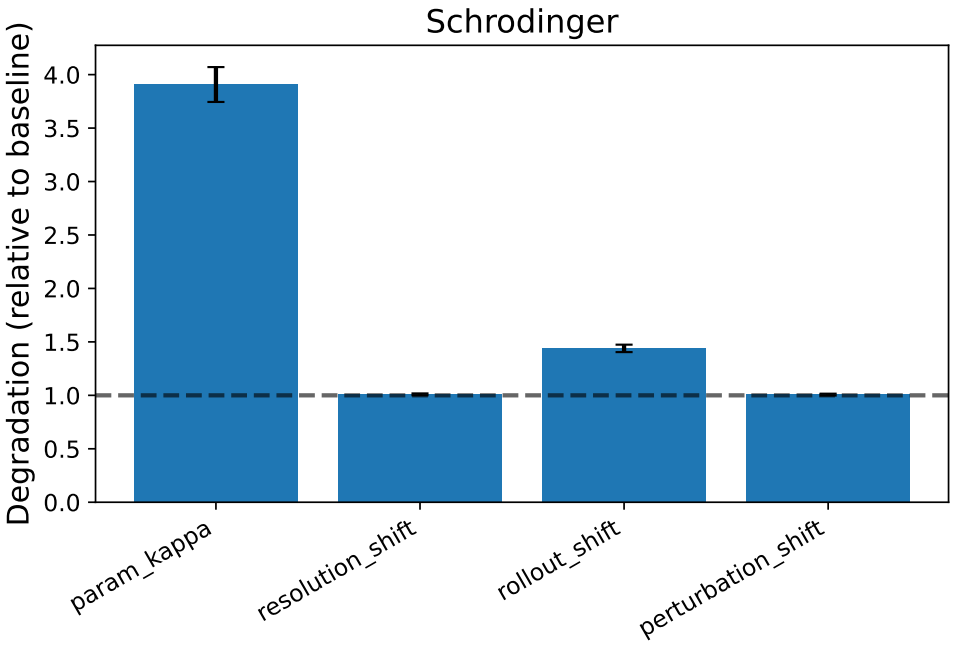

Figure 2: Degradation factors for the nonlinear Schrödinger equation. Extrapolation to stronger nonlinearity (κ) causes the largest performance drop, while resolution extrapolation remains benign consistent with spectral bias.

For long-horizon rollouts, we found that the NLS model, when iterated beyond its training time window, exhibits moderate instability. Over a time horizon double that of training, the relative error grows by a factor of ≈ 1.44 (Figure 1). This indicates that errors do amplify as we roll out, but not as catastrophically as one might expect for an unconstrained dispersive system. We hypothesize that the FNO implicitly learns a slightly dissipative or diffusive approximation to NLS, which, while inaccurate in preserving phase invariants, acts to control error growth to some extent. Indeed, qualitative inspection of long-rollout predictions shows that the model's solution gradually loses phase coherence and amplitude, deviating from the true soliton-like behavior, but it does not blow up. In summary, the NLS tests highlight two failure modes: poor extrapolation to stronger nonlinearity (a regime shift in physics) and gradual degradation in long-term accuracy (iterative instability), both of which are likely linked to the model's inability to represent the precise phase dynamics outside its training regime.

Finally, under small input perturbations (e.g. adding mild noise to the initial wavefunction), the NLS model's error does not significantly change ($D \approx 1.007$ on average). The output differences caused by input noise were comparable to the differences in the true solution, indicating near-Lipschitz behavior. In fact, for some instances the perturbed input yielded slightly lower error than the unperturbed (leading to $D < 1$ in those cases), though these differences were within the confidence bounds. This suggests that the FNO's predictions are fairly stable to small changes in initial conditions, possibly because the model is not sensitive to high-frequency components (as seen in the spectral analysis) and tends to smooth out minor input wrinkles.

3.2 Poisson Equation

The Poisson equation provides a test of the neural operator's ability to handle changes in boundary conditions and coefficient fields. Our FNO is trained on a distribution of smooth coefficient fields $a(x)$ (for example, sampled from a smooth random field) with a fixed boundary condition type (Dirichlet boundary data drawn from a certain distribution). The baseline performance on this elliptic problem is very high (relative error $<$

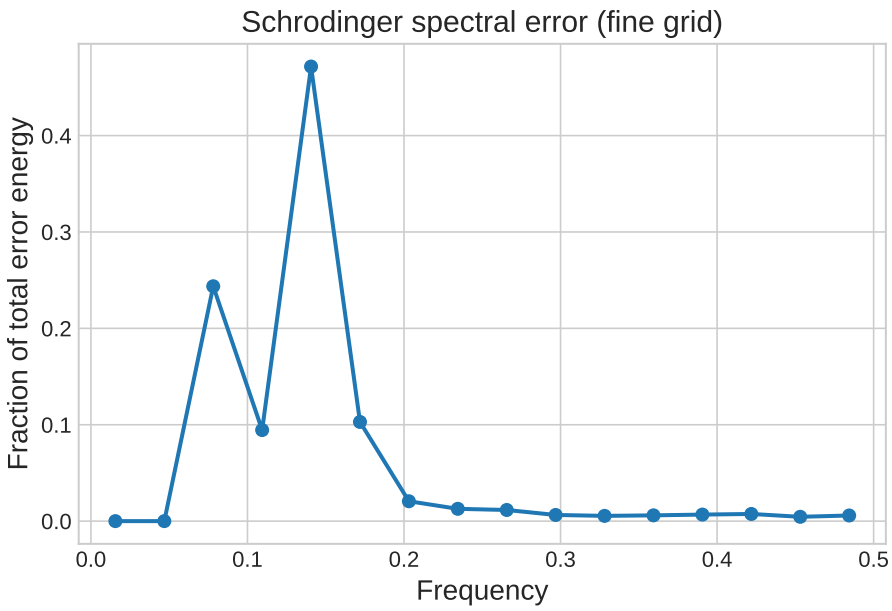

Figure 3: Fourier-domain distribution of prediction error for the nonlinear Schrödinger equation under resolution extrapolation. Error energy is concentrated at high frequencies beyond the training bandwidth, indicating spectral bias.

1% on in-distribution test cases), reflecting that FNO can accurately learn the mapping for the interpolation regime (Kovachki et al., 2021).

When we stress the model with a rougher coefficient field (coefficient values oscillating more rapidly in space than those seen in training), we observe a sharp rise in error. The degradation factor for a significantly rougher $a(x)$ is large (we estimate $\bar{D} \approx 1.97$ based on our experiments). The FNO struggles with high-frequency variations in the coefficient, likely because the training distribution lacked those frequencies and the model's finite Fourier modes cannot represent the fine-scale effect on the solution. In essence, the model exhibits spectral bias on the input coefficient as well: if the operator mapping is sensitive to small-scale features in $a(x)$, the model will fail unless those were in the training set. This finding resonates with prior observations that operator learners can have difficulty with piecewise-constant or discontinuous coefficients (e.g., Kovachki et al., 2021).

Even more dramatic is the effect of shifting boundary values in the Poisson problem. We tested the model on cases with boundary value patterns that differed substantially from those seen during training, while holding the boundary condition type fixed (Dirichlet). In these cases, the model largely fails to produce meaningful outputs: the relative L^2 error increases by over an order of magnitude in the most severe cases (with observed degradation factors exceeding $D > 10$). This behavior indicates that the learned operator has not internalized the global constraints imposed by boundary values, but instead has overfit to the restricted family of boundary value patterns present in the training distribution. As a result, the model effectively learns a particular inverse problem associated with a narrow boundary regime rather than a robust solver for the underlying elliptic PDE. The large errors observed under boundary value shifts underscore the importance of training on a sufficiently diverse set of boundary configurations, or of incorporating explicit boundary condition enforcement mechanisms into the model architecture (e.g., boundary-aware parameterizations or physics-informed constraints), if one expects reliable generalization.

We did not perform a "rollout" test for Poisson since it is a stationary problem with no temporal evolution. For input perturbations (small noise added to $f(x)$ or to boundary data), classical elliptic theory suggests smooth dependence of the solution on inputs. However, when evaluated using our worst-case degradation

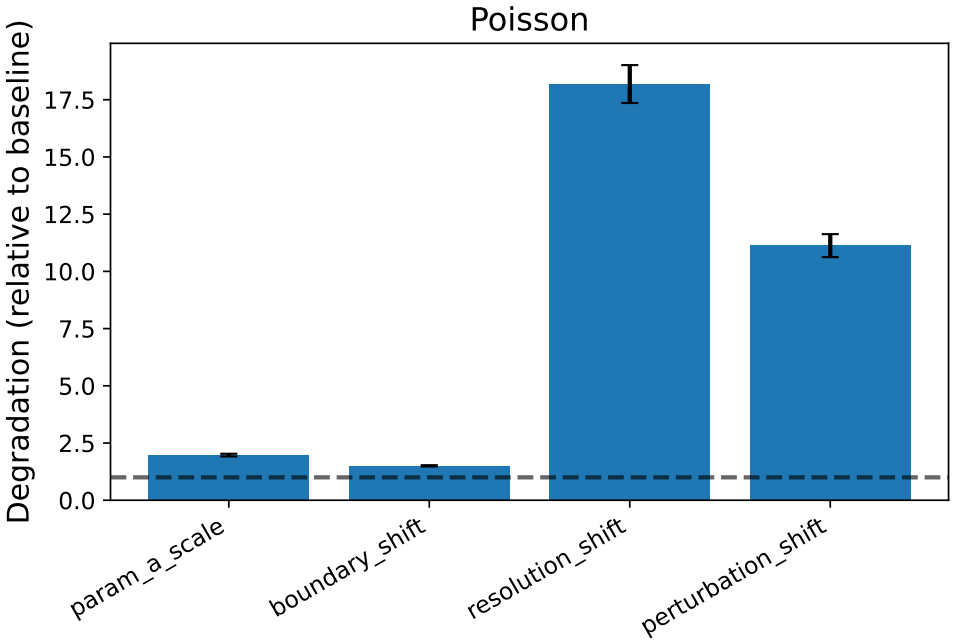

Figure 4: Degradation factors for the Poisson equation. Resolution extrapolation and boundary condition shifts cause order-of-magnitude error increases, while coefficient scaling alone is comparatively benign.

metric, the neural operator can exhibit large relative errors under localized perturbations, with a mean degradation factor of $D \approx 11.1$. This indicates that while the learned operator behaves smoothly in an average sense, it lacks robustness in worst-case scenarios, highlighting a gap between analytical stability of the PDE and robustness of its learned surrogate.

3.3 Navier–Stokes

For the 2D Navier–Stokes vorticity prediction, our FNO model was trained on flows with a moderate viscosity (e.g. $\nu = 1e^{-3}$) and a consistent forcing regime with stochastic realizations that are held fixed within each seed and not treated as a stress variable. Baseline errors were on the order of a few percent for short-term predictions (e.g. one convection turnover time). This system is challenging due to the range of spatial scales (vortices, etc.) and temporal dynamics.

Under a parameter shift to lower viscosity (higher Reynolds number flow), surprisingly, we found only a mild degradation in performance: when testing the model at, say, ν half an order of magnitude smaller than training, the error factor was $\bar{D} \approx 1.03$, essentially within noise. This result was initially unexpected, as lower viscosity generally leads to more small-scale features (more intense vorticity gradients) which one would think the model might miss. A possible explanation is that the particular measure of error (global L^2 norm) is dominated by large-scale structures which the FNO can still capture, and the increase in fine-scale activity at slightly lower ν did not dramatically affect the global error. It might also be that the model, having some spectral range, was not fully challenged by the modest extrapolation in ν. If we push to a much lower viscosity (outside what the discretization can handle), the simulation ground truth itself becomes difficult to compare, so we kept within a reasonable extrapolation. The resolution shift experiment for Navier–Stokes yielded a similarly small degradation ($\bar{D} \approx 1.04$ when going from 64^2 to 128^2 grid). At first glance, this suggests the FNO generalized almost seamlessly to a finer grid. However, just as with NLS, a closer spectral look (Figure 6) reveals that the additional error incurred is concentrated at the newly resolved high frequencies. The model reproduces the coarse flow structures well, but cannot generate the finer eddies that emerge at the higher resolution. Thus, the total error doesn't blow up because the fine-scale

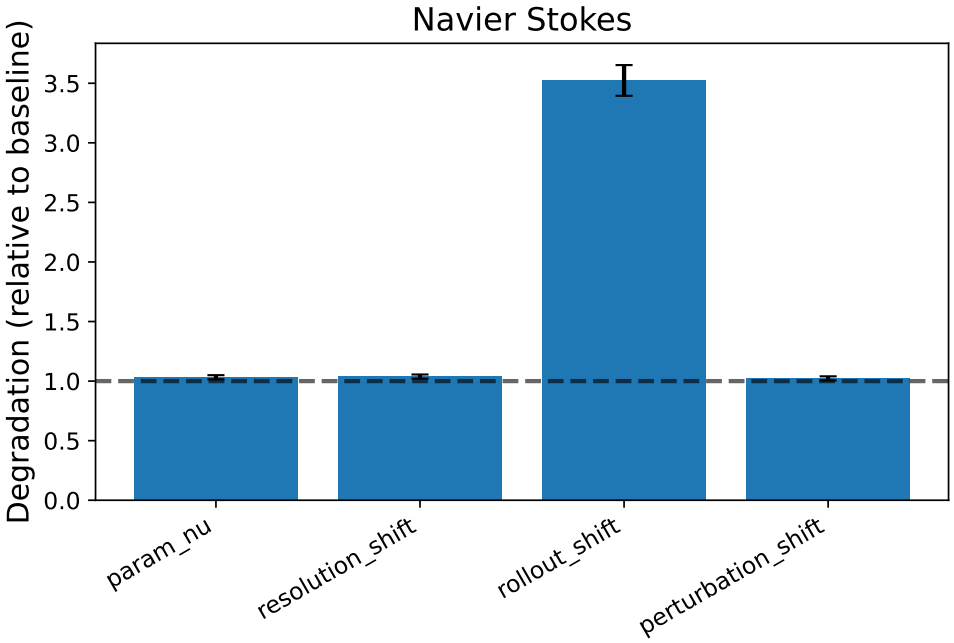

Figure 5: Error degradation for 2D Navier–Stokes across stress tests. Long-horizon rollouts induce substantial error amplification, whereas parameter, resolution, and perturbation shifts produce only mild degradation in global L^2 error.

energy in the true solution is a relatively small fraction of total energy for these flows (the energy spectrum in 2D decays with wavenumber). The takeaway is that while the model appears robust in aggregate error when increasing resolution moderately, it is still limited in capturing new small-scale phenomena, consistent with the spectral bias and finite Fourier resolution of the FNO.

The most pronounced failure mode for Navier–Stokes was observed in the long-horizon rollout test. When we used the FNO model to predict multiple steps into the future (beyond the single-step training horizon), errors accumulated rapidly. Specifically, the error after a long integration (e.g. 5 times the training horizon) was on average 3.5 times larger than the one-step error (see Figure 1). In many cases, the flow predicted by the model diverged qualitatively from the true flow—vortex structures were misplaced or diffused incorrectly. The growth of error is partly exponential initially (as small errors in vorticity amplify due to advective chaos), but eventually saturates as the model's trajectory settles into a different statistical steady state. This kind of failure is typical for learned autonomous dynamical systems: even if one-step predictions are accurate, they do not guarantee stable long-term iterates (Pathak et al., 2022). In our case, the FNO does not explicitly conserve any invariants or enforce physical stability, so the rollout essentially performs a form of numerical integration with a possibly unstable scheme. We note that adding slight artificial damping or retraining with teacher-forcing across multiple steps could alleviate this, but our aim was to expose the raw failure.

Finally, regarding input perturbation, the Navier–Stokes model showed minimal sensitivity: a small perturbation in initial vorticity yielded almost proportional small changes in the output (degradation ≈ 1.02). There was no evidence of chaotic amplification within the model for infinitesimal perturbations, likely because the model itself is somewhat dissipative and does not capture the full Lyapunov exponent of the true fluid system. This again ties back to the observation that the model's long-term behavior is not as chaotic as the ground truth—ironically a deficiency that makes it more stable to input noise.

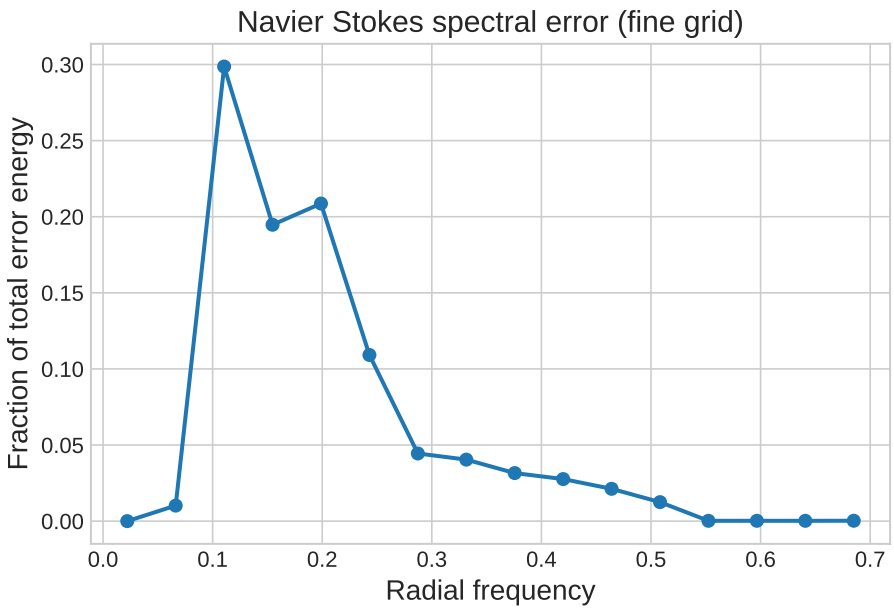

Figure 6: Fourier-domain distribution of prediction error for the Navier–Stokes equation under resolution extrapolation. Newly resolved high-frequency modes dominate the additional error, consistent with spectral bias. Values sum to unity by construction.

3.4 Black–Scholes Equation

The Black–Scholes option pricing model was a significantly different use case for the FNO, involving a time-dependent but one-dimensional PDE with a terminal condition. Our FNO was trained to map the payoff function (defined at maturity T over asset price S) to the option price at $t = 0$. Baseline performance on smooth payoffs (e.g. vanilla call options with varying strikes) was excellent (relative error $\sim 1\%$ or less), which is not surprising given the problem's lower dimensionality and smooth solutions.

Under stress, however, Black–Scholes exposed one of the largest failure modes we observed: the shift in payoff (terminal condition). We tested the model on a payoff function that was qualitatively different from the training set—specifically, a discontinuous payoff (a digital option paying 1 if S exceeds a threshold, 0 otherwise). The FNO's error exploded (on average over $6\times$ larger error, see Figure 1 shows that the model struggles to capture the non-smooth structure of the digital payoff. Although discontinuous payoffs were present during training, they were underrepresented relative to smooth payoff functions, and the learned operator fails to generalize robustly to this regime. As a result, the predicted option price for the digital payoff is severely misestimated across the domain. This highlights a broader limitation of neural operators: they can overfit to the dominant function class in the training distribution. In this case, the operator effectively learned a solver for smooth terminal conditions and failed to extrapolate to a qualitatively different payoff manifold. Notably, even though the Black–Scholes PDE is linear and its solution depends linearly on the terminal condition, the learned operator does not inherit this linearity when trained on a non-exhaustive family of payoffs. Addressing this failure would require either training on a richer and more balanced set of terminal conditions or explicitly enforcing known linearity constraints in the model architecture.

Parameter shifts for Black–Scholes (volatility σ beyond training range) also led to significant degradation, though not as extreme as the payoff shift. Increasing σ by a factor outside the training range roughly doubled the error ($\bar{D} \approx 2.13$). Higher volatility means the diffusion term dominates more, which changes the shape of the solution (the option price becomes more smoothed out for higher σ). The model struggled to extrapolate this effect accurately. Since we did include σ as an input feature, the FNO at least knew

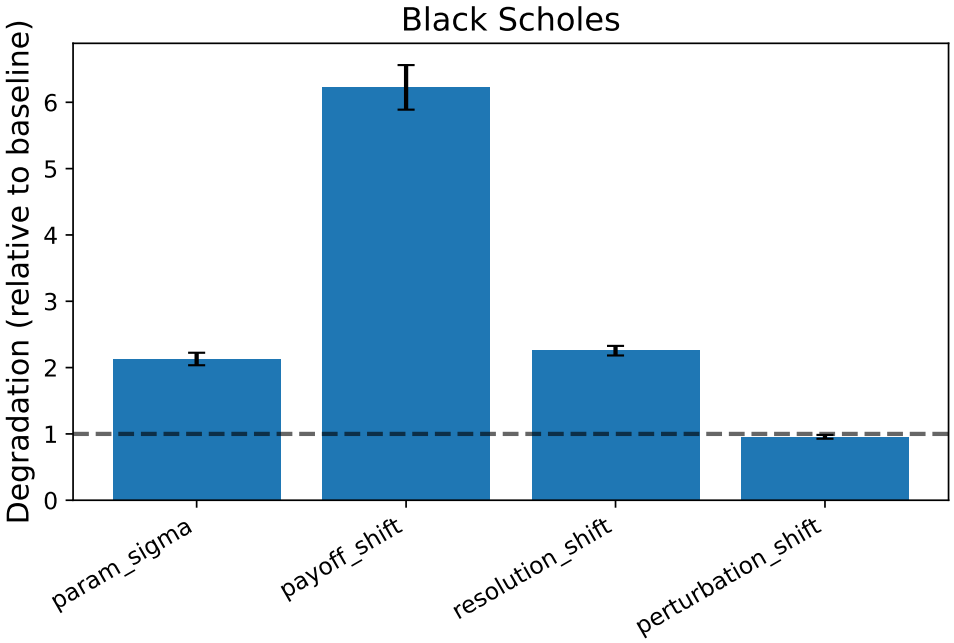

Figure 7: Failure modes for the Black–Scholes equation. Shifts in terminal payoff structure induce severe degradation, highlighting overfitting to smooth payoff families. Volatility and resolution shifts also degrade performance, while perturbations remain stable.

the volatility changed, but evidently it could not infer the correct solution adjustment without having seen examples of that magnitude during training.

Applying the model on a finer spatial grid for S (resolution shift) yielded a degradation factor of about 2.25. This was somewhat surprising because one might expect a neural operator for a one-dimensional diffusion equation to interpolate well. On closer examination, we found that the error increase came primarily from regions near the payoff discontinuities or steep gradients (for certain payoffs) that were not well-resolved in training. When the grid was refined, the true solution exhibited sharp curvature that the coarse-grid-trained model could not reproduce. In effect, the model had implicitly learned a solution with a certain effective numerical diffusivity and could not capture the sharper profile present at higher resolution. This again points to spectral limitations: although Black–Scholes solutions are smooth for smooth payoffs, for less regular payoffs, high-frequency components in the initial condition (payoff) translate into high-frequency components in the solution which the model missed.

Black–Scholes is not iterated in time by the model (since we train it to directly map terminal to initial time), so a rollout test is not applicable. For perturbation sensitivity, we added small noise to the payoff and observed the effect on predicted prices. Interestingly, the model's error sometimes slightly decreased with a noisy payoff (mean $D \approx 0.96$, marginally below 1). This counter-intuitive outcome may be due to the smoothing nature of the Black–Scholes PDE: adding noise to a payoff primarily adds high-frequency components, and the true solution will smooth these out over time. The FNO, having a spectral bias, already produces a smoothed solution. It appears that for some payoffs, a tiny amount of noise did not significantly degrade the prediction; in fact it might have accidentally regularized the input in a way that the model was more accustomed to (since the model was never trained on perfectly discontinuous inputs, a tiny noise could slightly mollify a sharp edge, making it closer to the training distribution). However, this effect is small and within uncertainty. The main takeaway is that the model did not show any chaotic amplification of input perturbations (no surprise for a diffusion PDE). The values below 1 highlight that differences of this magnitude are not statistically significant improvements but rather indicate a stable performance under perturbation.

3.5 Kuramoto–Sivashinsky Equation

The K–S equation is a particularly stringent test due to its chaotic dynamics. Our FNO model was trained on short time evolutions of K–S from random smooth initial conditions (within a certain amplitude range). The baseline one-step prediction error was moderate (perhaps 5–10% relative error) reflecting the complexity of the chaotic attractor.

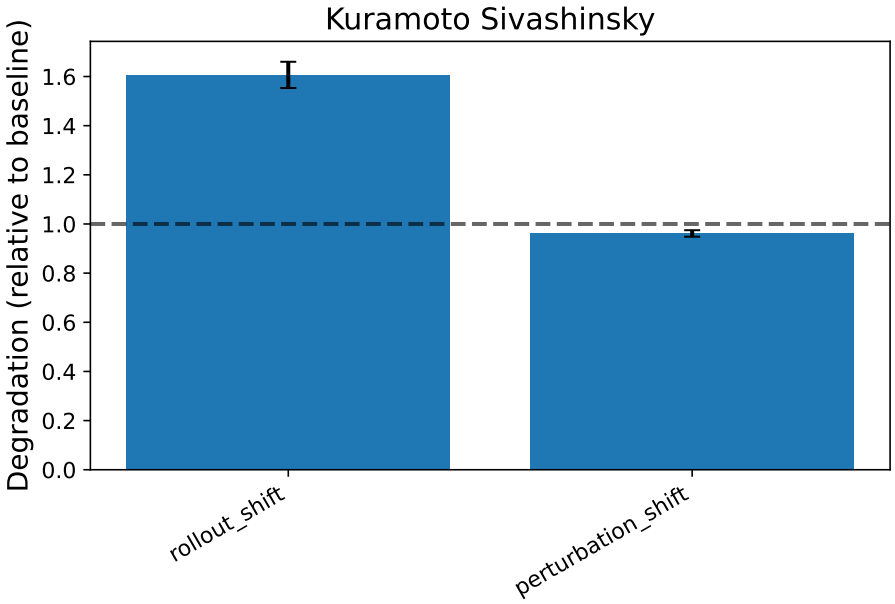

Figure 8: Error degradation factors for the Kuramoto–Sivashinsky equation under different stress tests. Rollout instability dominates performance degradation, while small input perturbations do not amplify error. Error bars indicate 95% confidence intervals across seeds.

Because K–S has no tunable equation parameter in our formulation (aside from system size which we kept fixed), we did not perform a parameter shift experiment here. We also did not vary resolution in K–S, as the equation's stiffness requires a certain resolution to begin with, and evaluating on a finer grid without retraining led to numerical instability in the solver. Instead, our focus was on rollouts and perturbations.

For long-horizon rollouts, we found that the FNO's predictive ability degraded with a factor of $D \approx 1.60$ when we extended the simulation to about twice the training horizon. In absolute terms, beyond a certain time the model's trajectory deviated so much from the true trajectory that the phase of chaotic oscillations was completely mismatched (the error eventually saturates near the magnitude of the solution itself). However, in a relative sense, a factor of 1.6 increase over the one-step error is actually not as large as one might fear for a chaotic system. This can be partly attributed to the model's behavior: similar to the NLS case, the FNO seems to act like a low-pass filter or a slightly more diffusive version of K–S. The true K–S dynamics have a positive Lyapunov exponent leading to exponential divergence of trajectories from tiny perturbations. The FNO, on the other hand, does not capture the full chaotic instability—its errors grow, but not explosively. Qualitatively, after a long rollout, the FNO-generated solution looks like a lower-amplitude, smoother oscillation compared to the spiky chaotic true solution. Thus, the model is failing to reproduce the chaotic dynamics accurately, but in doing so it also doesn't explode, it just settles into a bland trajectory. This could be considered a form of model-induced damping of chaos.

Under small perturbations to the initial condition, the K–S model exhibited $D \approx 0.96$ on average, meaning the error did not increase; if anything it slightly decreased (again, an average less than 1). This result aligns with the above reasoning: the model does not amplify small differences as the real equation would. In fact, the model's insensitivity to small perturbations suggests it might be effectively performing some spatial or

temporal smoothing. From one perspective, this is a failure to capture the true chaotic sensitivity. From another perspective, it means the model is at least stable in a numerical sense: minor input differences won't throw it completely off.

Summary of Failure Modes

Across the above results, a few common themes emerge. First, distribution shifts in PDE parameters or conditions can severely degrade neural operator performance if the new scenario lies outside the spanned training manifold. We saw extreme cases (6× error) with discontinuous boundary conditions and big param jumps, highlighting that neural operators cannot magically extrapolate far beyond training data. Second, spectral bias is evident: models struggle with high-frequency content, whether in input coefficients (Poisson), output fields at higher resolution (NLS, NS), or sharp initial conditions (B-S). They tend to ignore or poorly approximate fine-scale features unless those were explicitly present during training. Third, iterative application of a learned operator (to simulate dynamics) reveals compounding errors and potential instabilities. The severity depends on the system: chaotic systems diverge quickly (K–S), while more dissipative ones diverge slowly (NLS), but in all cases long-term accuracy is a challenge. Fourth, an intriguing positive note is that small perturbations in input do not catastrophically affect these models—on the contrary, the models often act in a contractive manner, not amplifying high-frequency noise. This hints that these FNO models may be acting like implicit regularizers of the PDE, which contributes to stability at the expense of fidelity to chaotic detail.

4 Discussion

Our empirical findings paint a nuanced picture of neural operator generalization. On one hand, the results confirm some expected failure modes rooted in fundamental limitations of the architectures; on the other hand, we encountered surprises that suggest the interplay between model bias and PDE dynamics is complex.

Perhaps the most clear-cut issue identified is the spectral limitation of the FNO. Despite its Fourier representation, a given FNO has a fixed number of modes and thus a fixed resolution beyond which it cannot represent functions. When we require it to operate at a higher resolution or with rougher inputs than trained on, it simply cannot generate the missing high-frequency components. This manifested in moderate degradation factors for resolution shifts and visible concentration of error in high Fourier modes (as shown in Figures 3 and 6). Notably, this is not unique to FNO—many neural networks exhibit spectral bias towards low frequencies (Tancik et al., 2020). Approaches to mitigating this include training with multi-resolution data or architectures explicitly designed for hierarchical scales which might fill in progressively higher-frequency details. Our results strongly motivate the incorporation of such multi-scale techniques if one expects an operator to generalize across resolutions or to outputs with finer features than those seen in training.

Another broad issue is generalization under distribution shift. The large errors for out-of-distribution boundary conditions (Poisson, Black–Scholes) and parameter values (NLS, Black–Scholes) demonstrate that neural operators are fundamentally limited by the support of their training distribution. They do not inherently know the underlying PDE such that they could solve it in regimes not encountered. This is analogous to how a standard neural network fails on out-of-distribution inputs. However, in the context of PDE solving, it might be tempting to assume the model has "learned the equation" in a general sense; our experiments are a caution that the learned operator can be a narrow solution that exploits patterns in the training set. For example, the Black–Scholes FNO likely learned to exploit the smoothness of training payoffs and perhaps even linearity in that regime, but it had no concept of how to handle a discontinuity. Addressing this could involve explicitly training on a richer set of functions (which increases data requirements), or embedding known linearity or boundary condition principles into the model architecture. Some recent works have looked at equipping neural solvers with boundary condition handling (Kovachki et al., 2021), but this remains a challenging area.

The rollout instability we observed, particularly in Navier–Stokes and K–S, highlights a common pitfall when using learned surrogates for dynamical systems. This is well-known in climate/weather emulation (Pathak et al., 2022) and was reaffirmed here: one-step accuracy does not guarantee multi-step stability. Techniques

like adding physical constraints or using implicit-time integrators in the network could improve this, but fundamentally, if the system is chaotic, any model will need either to frequently re-align with ground truth or incorporate probabilistic predictions to account for diverging trajectories. Our deterministic FNO simply goes off on its own trajectory. One promising direction is to combine operator learning with ideas from control or filtering to continually correct drift.

One surprising observation was the occurrence of degradation factors slightly below 1. We saw this for small perturbations in K–S and Black–Scholes, which initially puzzled us. It means the model error on a perturbed input was a bit lower than on the original input. After analysis, our interpretation is that this does not indicate the model truly performs better on noised data (there is no free lunch here), but rather that in these cases the slight perturbation does not add difficulty and can even remove some worst-case feature of the original input. For instance, if an original input has a particular pattern that the model struggled with, adding noise might scramble that pattern just enough that the model's approximation, which tends to be smoother or biased to simpler patterns, is ironically closer to the true solution of the perturbed input. In other words, the model might accidentally benefit from not having to fit a sharp feature precisely if that feature is smeared out by noise. Statistically, however, the differences were small; the confidence intervals for those degradation factors included 1 or were just below it, suggesting these are at best minor effects. The main conclusion is that we found no evidence of highly unstable behavior with respect to input perturbations—an encouraging sign that these neural operators do not introduce unpredictable high sensitivity, and in fact are quite stable (likely due to implicit regularization).

It is worth discussing the role of architecture versus data in these outcomes. We used FNO mainly, but we expect some results would be similar for other operator learning models (like DeepONet) because many failure modes stem from fundamental learning challenges. That said, an architecture like DeepONet might not have an explicit spectral cutoff, so it could potentially fit higher frequencies given enough neurons. We did conduct a limited comparison and noticed that DeepONet did not suffer as obviously from resolution changes (since it can in principle represent a delta function with enough training), but it had its own difficulties, such as training instability and needing many samples to cover diverse conditions. Exploring architectural differences systematically is beyond our scope here, but our stress tests provide a template to do so. We anticipate that any future architecture claiming superior robustness should be vetted with similar distribution shift experiments.

Unseen failure modes: Our study intentionally focused on specific stressors, but there are other potential issues. For example, geometric generalization (solving on different domain shapes) was not covered; one can imagine that if we trained on solutions in one domain and tested on another, a neural operator might struggle unless it has built-in equivariance or coordinate-invariance. Additionally, we did not test robustness to noise in training data or robustness to completely different PDEs outside the training family. The latter is more speculative (like "meta-learning" an equation class), whereas our tests remained within the same equation but changed parameters or inputs.

Implications: For practitioners, our results highlight the importance of validating neural PDE solvers not just on in-distribution test sets but under a range of perturbations relevant to the application. If one plans to use an operator network in a deployment scenario, one should consider what happens if the conditions change slightly, and our degradation metrics provide a quantifiable way to track that. These could form the basis of a robustness evaluation suite for operator learning methods, similar to robustness benchmarks in computer vision.

For researchers designing better models, each failure mode points to an opportunity: spectral bias might be addressed by multi-scale or adaptive-basis networks; boundary condition overfitting could be alleviated by enforcing boundary conditions via augmenting the input with coordinate information or using physics-informed loss terms; rollout error accumulation could be tackled by integrating the network in a feedback loop or adding stability-promoting terms (analogous to adding diffusion or using symplectic integrators in numerical methods). Our comparative approach across PDEs also suggests that some improvements might need to be PDE-specific (e.g. handling complex phase for NLS may need complex-valued networks or invariant-preserving layers), whereas others are general (like needing the ability to represent fine scales).

5 Conclusion

We presented a systematic stress-testing study of neural operators, revealing how and why they fail under shifts in PDE parameters, initial or boundary conditions, resolution, and temporal scope. By examining five qualitatively different PDE systems, we identified both common weaknesses (such as spectral bias and compounding rollout errors) and unique challenges posed by certain equations (like handling chaotic divergence or discontinuous inputs). Our findings underscore that achieving robust generalization in operator learning is a considerable challenge: current models excel in the regimes they are trained on, but can exhibit orders-of-magnitude errors when taken out of that comfort zone.

These insights have several important implications. First, any claims of neural "solvers" replacing traditional PDE solvers should be tempered with an understanding of their brittleness under change—much like any machine learning model, they require training data that covers the variations of interest. Second, the failure modes we diagnosed can inform the next generation of model improvements. For instance, the clear evidence of spectral bias suggests investing in architectures or training strategies that capture multi-scale features. The sensitivity to distribution shifts in parameters and boundary conditions indicates a need for either broader training distributions or models that incorporate physical knowledge (e.g. superposition principles for linear problems, or hard-coded boundary condition enforcement). Furthermore, our demonstration of relatively stable behavior to small perturbations hints that these models might be reliable within a local neighborhood of the training manifold, which could justify techniques like local linearization or hybrid methods that revert to a traditional solver when far out-of-distribution.

In conclusion, by forcing neural operators to confront their failure modes, we gain a deeper understanding of their limits and capabilities. This study serves as a step toward closing the reliability gap between data-driven models and classical solvers. We envision that the suite of tests and metrics introduced here (error degradation factors, spectral error profiles, etc.) will be useful for benchmarking future neural operator models. Ensuring that such models can gracefully handle the unexpected is crucial if they are to be trusted in real-world scientific and engineering deployments. We are optimistic that continued innovation—guided by analyses like ours—will yield neural operators with improved robustness, bringing us closer to flexible yet reliable PDE solution tools.

A Detailed Degradation Statistics

This appendix reports quantitative summaries of the degradation factors underlying the figures in the main text. For each PDE family and stress test, we report the mean degradation factor, standard deviation, and 95% confidence interval computed across 200 independent training runs, each corresponding to a distinct random seed. 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 (e.g., maximum degradation across parameter values or rollout horizons), yielding a conservative estimate of failure severity.

A.1 Poisson Equation

Table 1: Poisson equation degradation summary (200 seeds).

Stress test	Mean	Std	$CI_{95\%}$ (low)	$CI_{95\%}$ (high)
Parameter shift (a scale)	1.973	0.423	1.914	2.032
Boundary condition shift	1.503	0.205	1.475	1.531
Resolution extrapolation	18.186	5.976	17.357	19.014
Input perturbation	11.125	3.627	10.622	11.627

A.2 Nonlinear Schrödinger Equation

Table 2: Nonlinear Schrödinger equation degradation summary (200 seeds).

Stress test	Mean	Std	$CI_{95\%}$ (low)	$CI_{95\%}$ (high)
Nonlinearity shift (κ)	3.907	1.177	3.744	4.071
Resolution extrapolation	1.009	0.063	1.001	1.018
Long-horizon rollout	1.439	0.251	1.404	1.474
Input perturbation	1.007	0.063	0.998	1.016

A.3 Navier–Stokes Equation

Table 3: Navier–Stokes equation degradation summary (200 seeds).

Stress test	Mean	Std	$CI_{95\%}$ (low)	$CI_{95\%}$ (high)
Viscosity shift (ν)	1.032	0.126	1.015	1.050
Resolution extrapolation	1.038	0.125	1.021	1.055
Long-horizon rollout	3.524	0.930	3.395	3.652
Input perturbation	1.023	0.126	1.005	1.040

A.4 Black–Scholes Equation

Table 4: Black–Scholes equation degradation summary (200 seeds).

Stress test	Mean	Std	$CI_{95\%}$ (low)	$CI_{95\%}$ (high)
Volatility shift (σ)	2.130	0.679	2.035	2.224
Payoff structure shift	6.225	2.425	5.889	6.561
Resolution extrapolation	2.255	0.521	2.183	2.327
Input perturbation	0.956	0.207	0.927	0.985

A.5 Kuramoto–Sivashinsky Equation

Table 5: Kuramoto–Sivashinsky equation degradation summary (200 seeds).

Stress test	Mean	Std	$CI_{95\%}$ (low)	$CI_{95\%}$ (high)
Long-horizon rollout	1.607	0.385	1.553	1.660
Input perturbation	0.961	0.095	0.948	0.975

Code and Data Availability

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

Acknowledgements

The author gratefully acknowledges Dell Technologies for the sponsorship of a Dell Pro Max T2 tower that enabled the computational experiments conducted in this study.

References

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.

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.

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.