PDEInvBench: A Comprehensive Dataset and Design Space Exploration of Neural Networks for PDE Inverse Problems

Anonymous authors
Paper under double-blind review

Abstract

Inverse problems in partial differential equations (PDEs) involve estimating the physical parameters of a system from observed spatiotemporal solution fields, a fundamental task in numerous scientific domains. Neural networks, and particularly neural operators, are well-suited for PDE parameter estimation due to their capability to model function-to-function space transformations. While existing benchmarks of machine learning methods for PDEs primarily focus on the forward problem — mapping physical parameters to solution fields—to our knowledge, there are no similar comprehensive studies and benchmark datasets on PDE inverse problems - mapping solution fields to underlying physical parameters. We fill this gap by introducing *PDEInvBench* , a comprehensive benchmark dataset consisting of numerical simulations for both time-dependent and time-independent PDEs across a wide range of physical behaviors and parameters. Our dataset includes evaluation splits that assess performance in both in-distribution and various out-of-distribution settings. Using our benchmark dataset, we comprehensively explore the design space of neural networks for PDE inverse problems along three key dimensions: (1) optimization procedures, analyzing the role of supervised, self-supervised, and test-time training objectives on performance, (2) problem representations, where we study the value of architectural choices with different inductive biases and various conditioning strategies, and (3) scaling, which we perform with respect to both model and data size. Our experiments reveal several practical insights: 1) neural networks perform best with a two-stage training procedure: initial supervision with PDE parameters followed by test-time fine-tuning using the PDE residual, 2) incorporating PDE derivatives as input features consistently improves accuracy, and 3) increasing the diversity of initial conditions in the training data yields greater performance gains than expanding the range of PDE parameters. We make our dataset and evaluation codebase freely available to facilitate reproducibility and further development of our work.

1 Introduction

Inverse problems involve inferring unknown parameters or governing laws of a physical system using observed data, such as measurements of system behavior over space and time. They are ubiquitous in numerous domains including geophysics (Tarantola, 2005), nanophotonics (Molesky et al., 2018), biomedical imaging (Vlaardingerbroek and Boer, 2013), and fluid dynamics (Karnakov et al., 2024). In the case of systems modeled by partial differential equations (PDEs), the inverse modeling task is to map observed spatiotemporal solution fields to the underlying PDE parameters, which are typically physical constants such as viscosity, diffusivity, or the Reynolds number.

Recent advances in machine learning (ML) have shown promise in learning effective representations for PDE-related tasks (Rahman et al., 2023; Tran et al., 2023; Herde et al., 2024). In particular, neural operators (NO) (Li et al., 2020; 2021; Kovachki et al., 2023) have emerged as powerful tools for modeling the mapping between function spaces. While significant progress has been made in establishing effective NO recipes for the *forward* problem (Gupta and Brandstetter, 2022; Takamoto et al., 2022; Lu et al., 2022)—predicting spatiotemporal solution fields from known PDE parameters—there has been comparatively less investigation into PDE inverse problems. The ill-posedness of PDE inverse problems, including issues of non-existence, non-uniqueness, and

instability with respect to noisy observational measurements (Engl et al., 1996; Tarantola, 2005; Kitanidis, 2010; Isakov, 2017), represents a challenging regime with distinct challenges compared to the forward problem.

In this work, we seek a principled understanding of the design space of neural network approaches – both within and outside the neural operator framework – for solving PDE inverse problems. As the first step, we introduce *PDEInvBench*, which is, to our knowledge, the first comprehensive dataset dedicated to benchmarking the performance of ML approaches for solving PDE inverse problems. Our dataset consists of high-quality numerical simulations of 5 PDE systems, including Darcy flow, reaction-diffusion, unforced and forced Navier-Stokes, and Korteweg-De Vries, at a wide range of physical parameter values and initial conditions. The PDEs cover a range of mathematical forms, including elliptic, hyperbolic and parabolic (Evans, 2022). Similarly, parameter ranges are chosen to exhibit a variety of behaviors including Turing bifurcations, diffusion, steady-states, turbulence, and laminar flow. We construct evaluation splits for each PDE system to investigate performance in both in-distribution and various challenging out-of-distribution settings.

Our second main contribution is the comprehensive benchmarking of ML-based methods for PDE inverse problems. We organize our investigation around three fundamental design axes:

1. **Model optimization procedure:** What is the optimal strategy for training neural networks for PDE inverse problems? Is a supervised loss from paired parameter-solution data sufficient, or is it beneficial to incorporate physics-based PDE residual terms? How can test-time adaptation be used to improve generalization to new parameter regimes without explicit data?

2. **Problem representation and inductive bias:** The choice of neural architecture encodes implicit assumptions about the underlying problem. Do spectral biases (as in the Fourier Neural Operator), local convolutional patterns (as in ResNets), or global attention mechanisms (as in Transformers) best capture the relationship between solution fields and physical parameters? Should features like the derivative terms of the PDE be explicitly provided as input, or left for the network to learn implicitly?

3. **Scaling properties:** Resource constraints often force tradeoffs. Is it more beneficial to increase model size or to generate more training data? When generating data, should one prioritize covering more physical parameter values or more initial conditions?

We evaluate models using standard ML metrics such as relative error and the slope of scaling curves. We also perform physically motivated evaluations, such as checking whether the energy spectra of numerical simulations evolved with the predicted PDE parameter match that of reference simulations.

From our extensive experiments, we distill key insights that can guide practitioners in deploying neural networks for inverse problems.

- A two-stage training approach is preferable when learning neural networks for PDE inverse problems: 1) training with a supervised data loss, followed by 2) test-time training using the PDE residual.

- Explicitly conditioning networks on spatial and temporal derivatives significantly improves performance across all models and datasets.

- For time-dependent PDEs, the FNO architecture generally outperforms ResNet and Transformer-style architectures, potentially suggesting that, among other architectural factors, a spectral inductive bias may be useful for solving PDE inverse problems.

- For a fixed computational budget, generating multiple trajectories for each parameter value using different initial conditions yields greater performance gains than expanding parameter coverage.

We release the PDEInvBench dataset[1] and evaluation codebase[2] as a modular, standardized environment designed to facilitate systematic exploration of the PDE inverse problem design space and serve as a foundation for future methodological advancements by the community.

[1] https://huggingface.co/datasets/DabbyOWL/PDE_Inverse_Problem_Benchmarking
[2] https://huggingface.co/datasets/DabbyOWL/PDEInvBenchCode

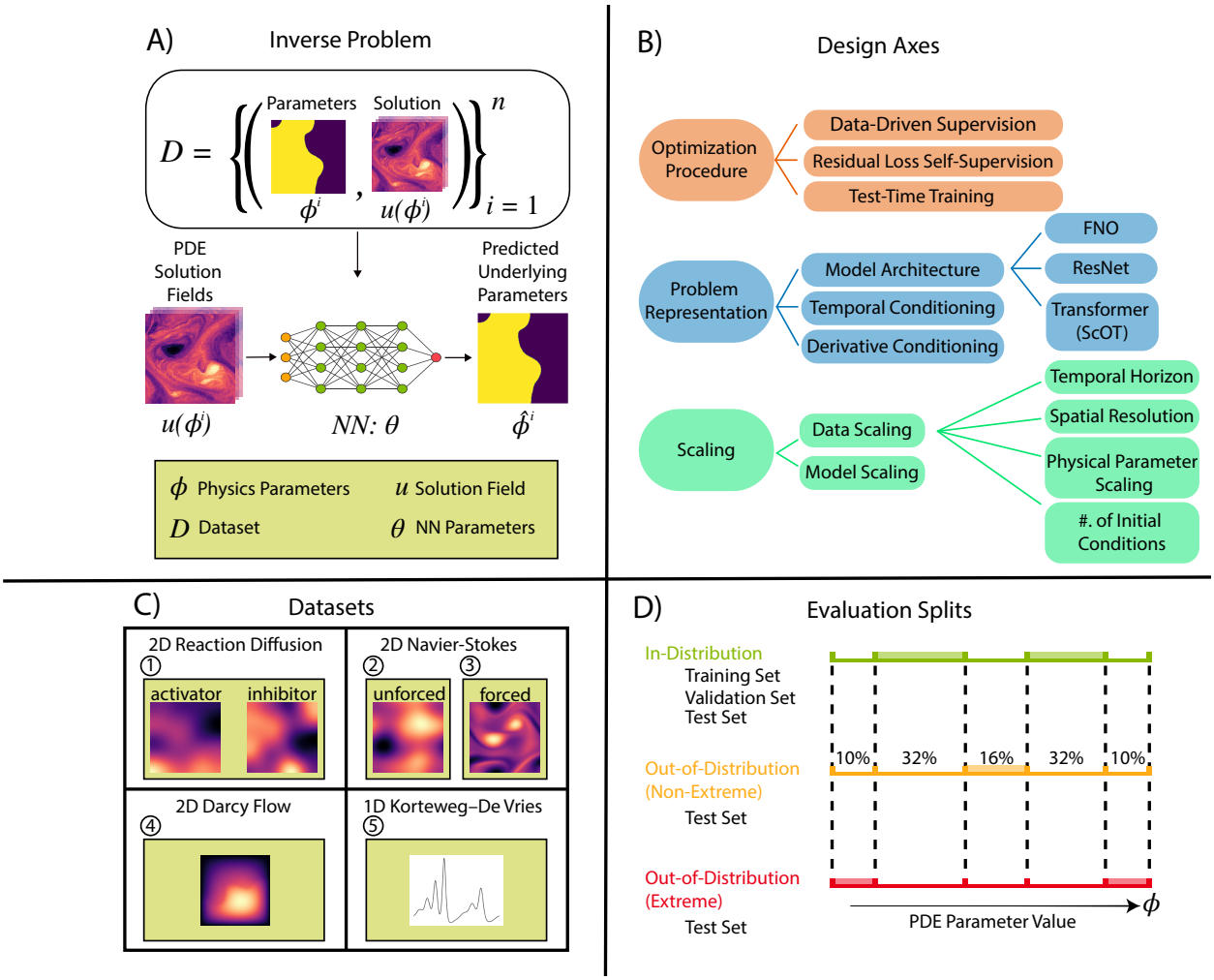

Figure 1: **Overview of design investigation of neural networks for PDE inverse problems. (A)** We consider the inverse problem setting in which a neural network learns to map from PDE solution fields $u(\phi)$ to predicted PDE parameters $\hat{\phi}$, using a dataset D consisting of parameter-solution pairs as supervision. **(B)** Our investigation is split across three key design axes: optimization procedures, problem representation, and scaling properties. **(C)** We benchmark on diverse datasets spanning different PDE families, including 2D Reaction Diffusion, 2D Navier-Stokes, 2D Darcy Flow, and 1D Korteweg-De Vries. **(D)** We perform evaluations on both in-distribution parameter regimes and various out-of-distribution scenarios to assess generalization capabilities.

2 Related works

Neural operators. Neural Operators (NOs) are a class of discretization-invariant, universal function approximators which learn mappings between infinite-dimensional function spaces, making them particularly well-suited for modeling PDE problems (Kovachki et al., 2023). NOs typically involve a point-wise lifting operation, multiple layers of a learnable iterative kernel integration, followed by another point-wise projection to the output function space. For a mathematical description of the NO framework, see Appendix A. Several types of NOs have emerged with different inductive biases. The Fourier Neural Operator (FNO) (Li et al., 2021), including its many extensions (Li et al., 2023; Tran et al., 2023; Kochkov et al., 2021) and generalizations (Du et al., 2024), leverages a spectral representation by parameterizing the integral kernel in the Fourier domain, enabling efficient modeling of global dependencies and smooth functions that naturally arise in physical systems. Transformers adapted for PDEs—like the scalable Operator Transformer (scOT) (Herde et al., 2024)

which modifies the Swin-Transformer (Liu et al., 2022)—use self-attention to model both local and global dependencies in a manner compatible with the NO framework. Multiple other attention-based architectures have been proposed (Cao, 2021; Hao et al., 2023a; Alkin et al., 2024; Wu et al., 2024), but we use scOT, which is implemented in the HuggingFace Transformers library (Wolf et al., 2020).

Inverse modeling using neural networks and Neural Operators. Several studies have explored neural networks for inverse PDE parameter estimation. Li et al. (2021) use a function space Markov chain Monte Carlo method to recover the distribution of initial conditions using FNO as a surrogate numerical solver. Li et al. (2024) extend this approach with a physics-informed losses during training and test-time refinement (similar to the test-time training (TTT) method we benchmark), enabling both direct inverse modeling and gradient-based parameter estimation using forward surrogates. MacKinlay et al. (2021) investigate regularization terms when using gradient descent through a neural network forward surrogate to recover system parameters. Molinaro et al. (2023) combine DeepONets (Lu et al., 2021) with FNOs for coefficient reconstruction from boundary measurements, and evaluate on electrical impedance tomography, inverse scattering, radiative transport and seismic imaging datasets. Jiao et al. (2024) use DeepONets to identify diffusion coefficients on unknown manifolds. Latent Neural Operators (Wang and Wang, 2024) introduce a "Physics-Cross Attention" to map geometric inputs into a latent space for coupled forward-inverse prediction. Cho and Son (2025) provide theoretical stability guarantees for inverse problems and introduce PI-DION, an architecture tailored for solution field reconstruction and parameter estimation from partial measurements. *PDEInvBench* advances this avenue of work by introducing a benchmark suite to accelerate model development and deriving relevant insights.

Existing datasets and benchmarks. Recent years have seen numerous PDE-specific benchmark datasets emerge, with most providing limited benchmarking of inverse problems, or consisting of numerical simulations performed with a limited range of PDE parameter values (Lu et al., 2022; Bhan et al., 2024; Toshev et al., 2024; Hao et al., 2023b). PDEBench (Takamoto et al., 2022) evaluates several PDE systems but only generate a few PDE parameters per system, making it ill-equipped to evaluate the performance of inverse models over a wide range of physical behaviors. PDEArena (Gupta and Brandstetter, 2022) examines Navier-Stokes across a range of bouyancy values, but focuses their analysis exclusively on the forward problem. BubbleML (Hassan et al., 2023), similar to PDEBench, only examines a limited number of parameters within the context of the forward problem. The Well (Ohana and others, 2024), a recent large dataset for the forward problem, covers a broad set of PDEs and focuses on generating a large number of initial conditions, but only covers a small range of physical parameter values. Kohl et al. (2023) focuses on fluid turbulence across a range of parameters, but benchmark autoregressive diffusion models only for the forward problem. By contrast, our dataset *PDEInvBench* is the only benchmark covering a wide range of PDE parameters (i.e., physical behaviors) and PDE systems in the inverse problem setting. Other works either exclusively focus on the PDE forward problem, or do not thoroughly study the inverse problem on multi-parameter datasets and across multiple design axes.

System identification. The PDE inverse problems we consider in this work are related to a broad line of literature on system identification Åström and Eykhoff (1971); Ljung (1998). Recent advances in multi-environment adaptation for system identification (Blanke and Lelarge, 2024; Kirchmeyer et al., 2022; Koupaï et al., 2024) focus on meta-learning task-agnostic representations with test-time adaptation of task-specific components according to the underlying parameters. However, these approaches primarily focus on solving the forward problem and do not directly predict underlying parameters. In contrast, our focus is on directly predicting the underlying PDE parameters while leveraging knowledge of the governing PDE equations for test-time adaptation. This allows for straightforward test-time tuning without the need to distinguish between task-agnostic and task-specific representations.

3 Preliminaries

Partial differential equations (PDEs). We consider the PDEs of the form,

$$\begin{aligned} &\mathcal{F}_\phi(u(x,t)) = 0 && x \in \mathbb{X}, t \in [0,T], \phi \in \Phi \\ &\mathcal{B}(u(x,t)) = 0 && x \in \partial\mathbb{X} \\ &u(x,0) = u_0 && x \in \mathbb{X} \end{aligned} \tag{1}$$

$\mathcal{F}$ is a differential operator characterizing a family of PDEs, with solutions u defined on the spatial domain $\mathbb{X}$ and temporal range $[0,T]$. x,t define spatial-temporal points, and ϕ denotes physical parameters of the PDE family (e.g., diffusion coefficient, viscosity, density) drawn from a distribution of possible parameters Φ. In general, ϕ may be an arbitrarily complex function of space and time, but in the systems we evaluate ϕ is typically a constant scalar or a spatially varying scalar field $\phi:\mathbb{X}\rightarrow\mathbb{R}$. $\mathcal{F}_\phi$ and u are potentially highly non-linear in x and t. $\mathcal{B}$ and u_0 denote boundary and initial conditions. For brevity, we abbreviate the entire solution over the spatial domain at time t as u_t, i.e., $u_t = u(\cdot,t)$.

Inverse problem and data loss. The inverse problem denotes the task of learning a function f_θ with parameters θ that maps k steps of observed PDE dynamics $u_{t-k},...,u_t$ from a PDE of the form of Equation 1 to the underlying PDE parameter ϕ. Formally, the optimization problem is posed as,

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\ \mathbb{E}_{\phi\sim\Phi}\|f_\theta(u_{t-k},...,u_t)-\phi\|_2, \tag{2}$$

where u satisfies $\mathcal{F}_\phi(u)=\mathbf{0}$. In practice, the expectation is approximated by an average over a finite dataset $\mathcal{D}_{\text{train}}=\{\phi^i,u^i\}_{i=1}^N$ of N PDE parameters paired with solution trajectories from a numerical simulation, where each u^i satisfies $\mathcal{F}_\phi(u^i)=\mathbf{0}$. We refer to supervised learning using paired PDE parameter data as *data-driven supervision*. The data-driven inverse problem objective is then:

$$\mathcal{L}_{\text{data}} = \frac{\|f_\theta(u^i_{t-k},...,u^i_t)-\phi^i\|_2}{||\phi^i||_2}. \tag{3}$$

Note that we use the relative error instead of the standard L_2 loss as physical parameters may span several orders of magnitude (e.g., viscosity $\nu\in[10^{-5},10^{-2}]$ in 2D turbulent flow).

Our problem setting is challenging due to the presence of a range of PDE parameters Φ, which induces a wide variety of physical behaviors in the solution u. For example, given two parameter settings $\phi_1,\phi_2\in\Phi$, the solution to $\mathcal{F}_{\phi_1}(u(x,t))=\mathbf{0}$ may be diffusive, while the solution to $\mathcal{F}_{\phi_2}(u(x,t))=\mathbf{0}$ may converge to a nontrivial steady-state. This is distinct from prior works on neural PDE solvers which typically consider a single or a few values of ϕ and only vary the initial or boundary conditions.

PDE residual loss. We can incorporate physics-based constraints via the PDE residual $\|\mathcal{F}_\phi(u(x,t))\|_2$, where the derivatives in $\mathcal{F}_\phi$ can be computed using autodifferentiation or a finite difference scheme. We elect to use the latter; henceforth, when $\mathcal{F}$ appears in a loss function, this refers to a finite difference approximation of the true differential operator on a finite-resolution mesh. We can use the PDE residual as a self-supervised learning signal Li et al. (2024). Given an inverse model f_θ that predicts parameters $\hat{\phi}=f_\theta(u^i_{t-k},...,u^i_t)$, the residual loss is,

$$\mathcal{L}_{\text{res}} = \|\mathcal{F}_{\hat{\phi}}(u^i_{t-k},...,u^i_t)\|_2^2 = \|\mathcal{F}_{f_\theta(u^i_{t-k},...,u^i_t)}(u^i_{t-k},...,u^i_t)\|_2^2 \tag{4}$$

Test-time training. Test-time training (TTT) leverages the self-supervised nature of the residual loss for adaptation to specific parameters that may fall outside the training distribution. Given observed dynamics $u_{t-k},...,u_t$ at test time, we can adapt the inverse model f_θ via gradient updates on the following loss:

$$\mathcal{L}_{\text{Tailor}} = \mathcal{L}_{\text{res}} + \alpha\mathcal{L}_{\text{anchor}} \tag{5}$$

where $\alpha\in[0,1]$ is a weighting coefficient and

$$\mathcal{L}_{\text{anchor}} = \frac{\|f_\theta(u_{t-k},...,u_t)-f_{\theta_{\text{frozen}}}(u_{t-k},...,u_t)\|_2}{||f_{\theta_{\text{frozen}}}(u_{t-k},...,u_t)||_2}, \tag{6}$$

where θ_{frozen} are model weights obtained after the initial training process. $\mathcal{L}_{\text{anchor}}$ prevents excessive deviations from the original model which we find helps stabilize tailoring.

4 Datasets and evaluation metrics

PDEInvBench contains five different PDE systems, each of which are simulated with a predefined range of PDE parameters chosen to span various physical behaviors including turbulence, steady-states, laminar flows,

Turing patterns, and diffusion. A summary of the datasets are given in Table 1. For each sampled parameter value within the range, a fixed set of Gaussian random fields, serving as initial conditions, are evolved using a numerical solver according to the governing equations. After a fixed initial burn-in phase, solutions are recorded until reaching convergence (e.g., Turing bifurcations (Borckmans et al., 1995)) or for a preset time horizon. Details about the numerical solver parameters used, including the type of solver, time-stepping scheme, and burn-in periods, can be found in Appendix B.2 alongside a complete description of the governing equations in Appendix B.1. We also provide analysis of numerical convergence and energy spectra of the datasets in Appendix B.3 and Appendix B.4. Here, we provide a high-level description of all the datasets.

2D Reaction Diffusion (RD) [parabolic]. 2D RD models two chemical species coupled non-linearly through the Fitzhugh-Nagumo equations (Klaasen and Troy, 1984; Takamoto et al., 2022). We focus on *activator-inhibitor* systems where the activator promotes the production of both species while the inhibitor suppresses both. There are 3 physical parameters k, D_u, and D_v. D_u and D_v are the diffusion coefficients for the activator and inhibitor, respectively. Meanwhile, k represents the balance between both species or the threshold for excitement. Our dataset contains simulations with a range of values for k, D_u, and D_v, spanning a wide range of induced physical behaviors, from completely dissipative to Turing bifurcations (Borckmans et al., 1995). For simplicity, when training inverse models, we only learn over *one* parameter at a time (i.e., either k or D_u) while treating the others as known. When k is unknown, we refer to the dataset as 2D RD-k and similarly, when D_u is unknown as 2D RD-D_u.

Unforced 2D Navier-Stokes (NS) [parabolic]. The unforced NS equations govern fluid flow with no external energy injection (Du et al., 2024). The parameter range covers the laminar flow regime. Both the data and the solver use the vorticity form of the PDE, with a single parameter, viscosity ν, serving as a proxy for internal friction.

Forced 2D Navier-Stokes (TF) [parabolic]. 2D TF models fluids across a range of Reynolds numbers in the turbulent regime. We use a Kolmogorov forcing function term on the second wavenumber (i.e., $-2\cos(2y)$) (Kochkov et al., 2021). As with unforced NS, the physical parameter of interest for the inverse problem is the viscosity ν.

Korteweg-De Vries (KdV) [hyperbolic]. KdV models waves on a shallow-water surface and includes a dispersive ($\partial_{xxx}u$) and an advection term ($u\partial_x u$) (Brandstetter et al., 2022). The physical parameter δ corresponds to the strength of the dispersive term (Zabusky and Kruskal, 1965).

Darcy Flow (DF) [elliptic]. Darcy Flow models the movement of a fluid through a porous medium (Li et al., 2021). All solutions converge to a non-trivial steady-state solution based on the diffusion coefficient field, making the system time-independent unlike previously described systems. The diffusion coefficient field, which is the parameter we learn over for the inverse problem, is a spatially varying scalar parameter rather than a constant scalar.

Table 1: Summary of PDE systems simulated in *PDEInvBench*. The dataset includes 4 time-dependent PDEs and 1 time-independent PDE, with physical parameter regimes for each chosen to span a wide range of physical behaviors, including turbulence, steady-states, laminar flows, Turing patterns, and diffusion.

PDE System	Number of Partial Derivatives	Time-dependent	Spatial Resolutions	Parameters	Temporal Resolution (s)/ Time Horizon (s)	Number of Parameter Values	Number of ICs/Trajectories (per parameter combination)
2D Reaction Diffusion (2D-RD)	5	✓	128×128	$k \in [0.005,0.1]$, $D_u \in [0.01,0.5]$, $D_v \in [0.01,0.5]$	[0.049 / 5]	k - 28, D_u - 28, D_v - 27	5
2D Navier-Stokes Unforced (2D-NS)	3	✓	64×64	$\nu \in [10^{-4},10^{-2}]$ (Reynolds: 80-8000)	[0.0468 / 3]	101	192
2D Navier-Stokes Forced (2D-TF)	3	✓	64×64	$\nu \in [10^{-5},10^{-2}]$	[0.23 / 14.75]	120	108
Korteweg-De Vries (1D-KdV)	4	✓	256	$\delta \in [0.8,5]$	[0.73 / 102]	100	100
Darcy Flow (2D-DF)	4	×	241×241	Piecewise constant diffusion coefficient $a \in L^\infty((0,1)^2;\mathbb{R}^+)$	-	2048	-

Evaluation settings. Each dataset consists of a broad range of parameter values for the training distribution. For all time-dependent PDEs, we create three different evaluation splits which test generalization in different

settings. Splits are categorized in relation to the training parameter range. The *Out-of-Distribution (Extreme)* split, referred to as OOD Extreme, tests generalization to extreme parameter values. In the extreme set, parameters are drawn from the smallest and largest 10% of the parameter range. Similarly, the *Out-of-Distribution (Non-Extreme)* split, referred to as OOD Non-Extreme, tests generalization to out-of-distribution parameter values in the middle 16% of parameter values which are omitted from the training parameter range. Finally, the *In-Distribution* (ID) split consists of held-out parameters drawn from the training range. Figure 1(D) visualizes how splits are partitioned for the single parameter case. For systems with more than one PDE parameter (e.g., 2D Reaction Diffusion), we generalize the percentages to a hypercube with each axis representing a single parameter (see Appendix B.5) for a visual depiction.

For Darcy Flow (the only time-independent PDE in our dataset), the physical parameter is a spatial *field* which takes on one of two values, 3 or 12, in each position. We make only ID and OOD Extreme splits for Darcy flow. We define the splits by computing the fraction of grid points in each coefficient field that take on the maximum value (12), a statistic which is approximately normally distributed across the dataset. We partition coefficient fields according to this distribution using ± 1.5 standard deviations, with the central mass defining the ID test set and the tails defining the OOD Extreme split (see Appendix B.5 for additional details).

Evaluation metrics. We present results using two primary metrics: *relative error* ($\downarrow$) and *the negative slope of the line of best fit (NLS)* ($\uparrow$). Relative error is defined as $||\phi - \hat{\phi}||_2 / ||\phi||_2$ where $\hat{\phi}$ and ϕ denote the predicted and true physical parameter, respectively. This metric is used to assess the quality of an inverse model. For experiments where we compare scaling approaches (e.g., relative error performance with respect to scaling the number of initial conditions versus scaling the number of generated physical parameters in Figure 4D-F), we use the NLS. From a plot of relative error versus a particular scaling axis, we compute a line of best fit, and we extract the NLS as the negative slope of that line. NLS measures the effect of scaling on the relative error by capturing the *decrease* in the y-axis (i.e., relative error) per x-axis unit (i.e., method of scaling). More negative values of the NLS indicate better scaling (i.e., faster reduction of the relative error due to scaling). We select the best performing models on the validation splits using these metrics (see Appendix B.6). In AppendixD.4, we present an additional evaluation to test the physical consistency of the predicted PDE parameters. We roll out numerical simulations with predicted PDE parameters and compare the resulting energy spectra with that of reference simulations run with the true PDE parameters.

5 Selected investigations for PDE inverse problems

We now describe our methodology for exploring key design axes for solving PDE inverse problems using neural networks. We investigate 1) optimization procedure, 2) problem representation and inductive biases, and 3) scaling properties, with several experiments within each axis. For all experiments, we train models with 3 different random seeds and report error bars of the corresponding standard deviation. In this section, we select the most significant investigations, but include additional experiments and details in Appendices C and D.

5.1 Optimization procedure

We investigate how different optimization approaches affect performance. Our exploration focuses on two key aspects: loss formulations and test-time training strategies.

Loss functions. We consider two primary loss formulations: a supervised, purely data-driven setting with the data loss defined in Equation 3, and a purely self-supervised, residual-driven setting with the PDE residual loss defined in Equation 4. In Appendix D, we also consider a generalized "physics-informed" loss which combines the data-driven and PDE-residual setting.

Test-time training. As defined in Section 3, we adapt a pretrained inverse model at inference time using a self-supervised objective (Equation 5). We take 50 gradient steps with a batch size of 32. We provide further details on test-time training in Appendix C.2

5.2 Problem representation and inductive biases

The choice of input features and inductive biases can significantly impact downstream performance. We investigate this through a comparative analysis of architectures and input representations.

Architecture comparison. We compare three architectures in a parameter-controlled ($\sim$5 million learnable parameters) manner (hyperparameters in Appendix C.3). The *Fourier Neural Operator (FNO)* (Li et al., 2021) serves as our baseline architecture and uses a spectral representation, similar to classical pseudo-spectral numerical methods. *ResNet* (He et al., 2016) tests the usefulness of locality in the spatial domain. Finally, we use the *scalable Operator Transformer (scOT)* (Herde et al., 2024), based on the Swin Transformers (Liu et al., 2022), to test the effectiveness of modeling local and global spatial dependencies through hierarchical attention mechanisms. Both FNO and scOT are by construction neural operators, and we include both to examine different classes of models. While not strictly a neural operator, ResNet operates over arbitrarily sized inputs due to the nature of convolutions which makes it a useful point of comparison. However, note that it is not discretization invariant like FNO or scOT.

Input representations. We examine the impact of explicitly providing derivative information as conditioning to the inverse model. In addition to the past k observed frames u_{t-k},...,u_t, the partial derivatives appearing in the differential operator $\mathcal{F}$ are concatenated as additional channels to the input. Additionally, in Appendix D.2, we study the effect of varying the number of temporal conditioning frames k provided to the inverse model.

5.3 Scaling properties

We conduct data scaling experiments along the number of initial conditions, total number of parameters and training time horizon, and model scaling experiments along the channel width (Appendix C.4).

Data scaling. For each data scaling experiment, we evaluate the performance of FNO, ResNet, and scOT to isolate the effects of data quantity. We vary dataset size along three dimensions. In our *Initial Condition Scaling* experiments, we train on a variable number of unique initial conditions per PDE parameter value, namely 20%, 35%, 50%, 75%, and 100% of the total number of initial conditions. Similarly, in our *Physical Parameter Scaling* experiments, we vary the number of unique PDE parameters sampled from the predefined range, varying over 20%, 35%, 50%, 75%, and 100% of the full number of parameters given in Table 1, while using 100% of available initial conditions. Lastly, in our *Temporal Horizon Scaling* experiments (Appendix D.3) we train on solution frames within 10%, 25%, 50%, 60%, and 75 % of the total time horizon of the simulated trajectories and evaluate on the held-out final 25%. For temporal horizon scaling, we use 100% of the PDE parameter values and initial conditions. Additionally, we investigate how the performance of different architectures behaves as data is scaled along *Initial Condition Scaling* and *Physical Parameter Scaling* (Appendix D.3). We note that *initial condition scaling* and *temporal horizon scaling* experiments are not applicable in the Darcy flow case, as it is a time-independent system (Appendix C.5).

Model Scaling For our model scaling experiments, we logarithmically scale the model size across the total number of parameters: 0.5 million, 5 million, and 50 million parameters. We primarily scale the total number of hidden channel dimensions rather than the model depth to avoid the training difficulties associated with deep neural operators (Tran et al., 2023; Koshizuka et al., 2024; Qin et al., 2024).

6 Results

We present selected findings of our evaluation centered on the three fundamental design axes (Section 5). Our baseline model is an FNO with an encoder-downsampler structure (details in Appendix C.1). Complete results can be found in Appendix D.

6.1 Optimization

Data supervision vs. self-supervision. Figure 2 A–C compares purely data-driven training (Equation 3) with purely self-supervised training using the PDE residual (Equation 4). On all three test sets, direct supervision consistently outperforms self-supervised training, except in cases where both methods produce relative errors close to $\sim$100% (OOD Extreme split of 2D RD). In Appendix Section D.1, we perform additional experiments combining the data loss with the PDE loss, finding that purely data-driven training generally always outperforms any setting with a nonzero coefficient on the PDE loss term. We also perform experiments comparing classical methods such as Newton-CG, L-BFGS-B, and SLSQP against FSNO to better contextualize the results (Appendix Section D.1).

Test-time training (TTT). Figure 2 D–E shows the results of performing TTT (optimizing Equation 5) after purely data-driven, supervised training. We consider two settings: 1) training on trajectories resulting

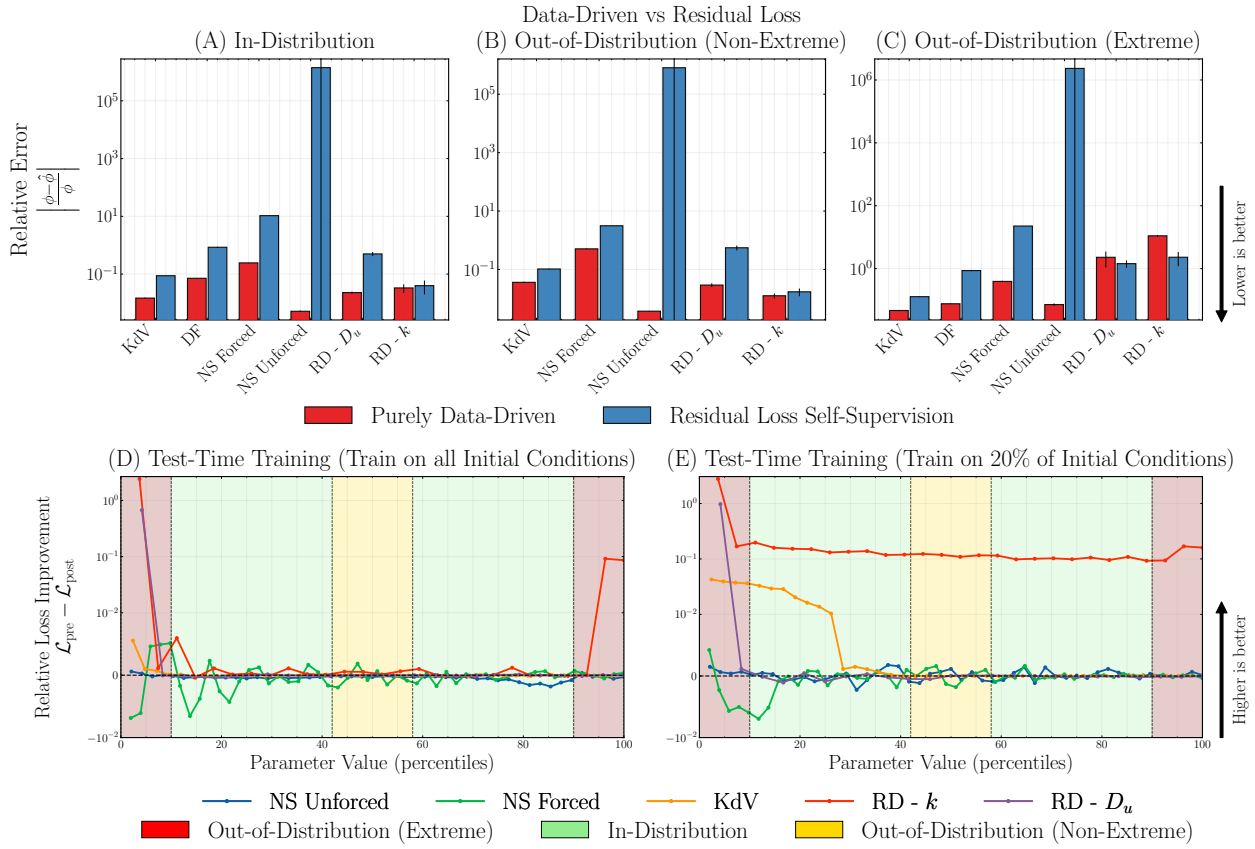

Figure 2: **Optimization approaches for NOs in PDE inverse problems**: (**A**–**C**) The performance of purely data-driven supervision versus self-supervision using only the PDE residual. For virtually all systems and evaluation settings, purely data-driven supervision consistently outperforms self-supervision using the PDE residual. As expected, performance degrades on the OOD Non-Extreme and OOD Extreme splits. (**D**–**E**) Change in relative error before and after test-time training (TTT), when the initial model is trained on trajectories from all (**D**) or 20% (**E**) of initial conditions in the dataset. Larger values indicate greater improvements from TTT. For the KdV and 2D RD systems, TTT yields noticeable performance improvements in the extreme out-of-distribution parameter regime, and the improvement is more pronounced when performing TTT on unseen initial conditions during test time.

from all initial conditions in the dataset, and subsequently evaluating on held-out trajectories from these same initial conditions (Figure 2D), and 2) training on 20% of initial conditions and evaluating on trajectories from unseen initial conditions (Figure 2E). In both figures, we show the improvement in PDE parameter relative loss resulting from TTT across the normalized parameter space (displayed in percentiles). We find that the effectiveness of TTT is generally higher when evaluating on trajectories from unseen ICs and at the extremes of the parameter range, but the degree of improvement is system-dependent. For example, we observe negligible improvements for 2D-NS and 2D-TF, while achieving 10-100% improvements for 2D-RD. The fact that TTT yields greater improvements in OOD initial condition or parameter settings is consistent with our intuition that incorporating information about the mathematical structure of the problem (via the PDE residual) would aid in OOD generalization. In general, since we observe that TTT never appreciably *hurts* performance, we recommend that it be used for PDE inverse problems.

6.2 Problem representation

We include model architecture and derivative conditioning results here, deferring temporal conditioning results to Appendix D.2

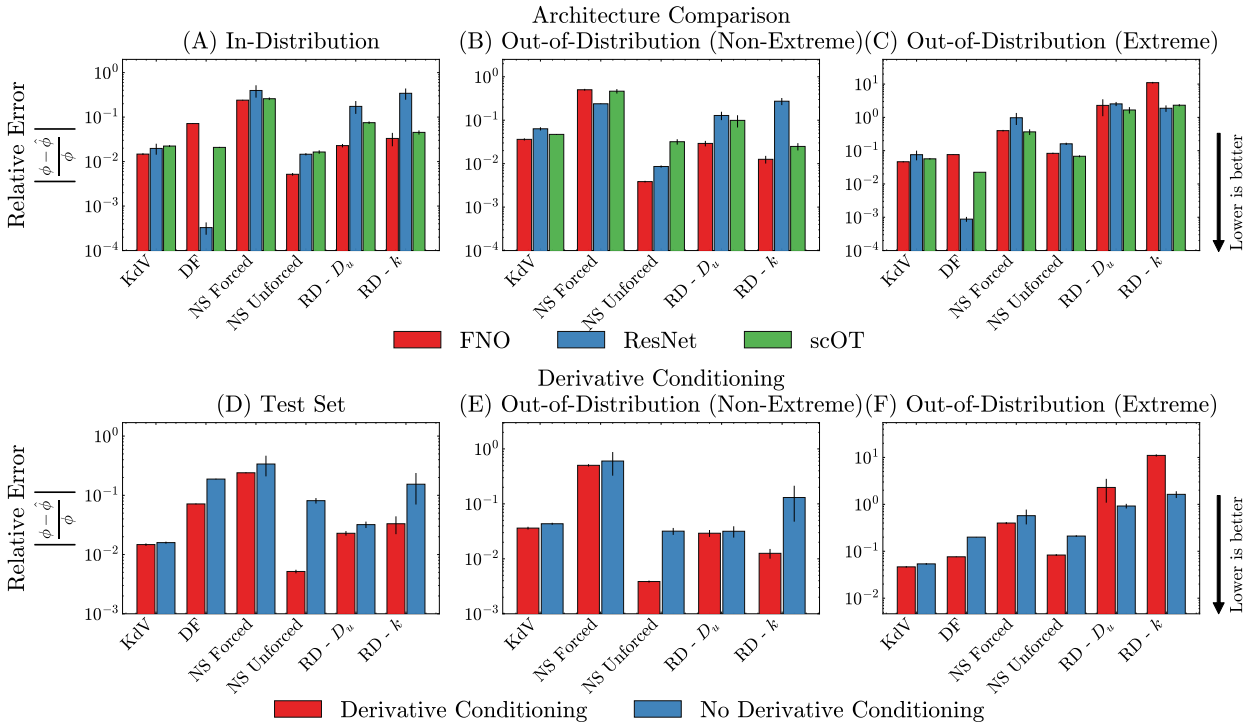

Figure 3: **Impact of problem representation on PDE inverse problem performance.** (**A**–**C**) Comparison of architectural inductive biases (FNO, ResNet, scOT) across evaluation splits. FNO generally outperforms ResNet and scOT on time-dependent PDEs, particularly in In-Distribution and Out-of-Distribution (Non-Extreme) regimes. (**D**–**F**) Effect of partial derivative conditioning on the FNO architecture, i.e., concatenation versus omission of partial derivatives as features. Conditioning the FNO architecture on partial derivatives consistently outperforms the unconditioned variant.

Architectural inductive biases. We compare three architectural inductive biases in a parameter-controlled manner: convolutional (ResNet), spectral (FNO), and global attention (scOT). For time-dependent PDEs, we find that FNO generally outperforms ResNet and scOT, particularly in OOD settings (Figure 3A-C). While there are differences between the architectures beyond their inductive biases, our results suggest that FNO's performance might be attributable to its spectral inductive bias, which provides a natural representation for modeling continuous solution fields. For Darcy Flow, which is time-independent, ResNet significantly outperforms both FNO and scOT (Figure 3A-C). The lack of time-dependency reduces the problem to a spatial modeling task, making it a good fit for convolutional methods.

Appending partial derivatives. Figure 3 D–F visualizes the effect of including partial derivatives as additional input features. Across all PDE systems and evaluation settings, models that receive derivative information generally outperform those that receive only the solution fields, demonstrating improved learning efficiency and generalization. While FNO should have the representational capacity to model the spatial derivative operator, we find it helpful to explicitly condition on partial derivatives. In the 2D RD OOD Extreme setting (Figure 3F), both appending and omitting partial derivative information fail to achieve good performance with relative errors $\sim$100%.

6.3 Scaling

We include data scaling results here, deferring model scaling results to Appendix D.3.

Initial condition scaling. We find that scaling the number of initial conditions from 20% to 100% of all initial conditions leads to a clear reduction in error, with every system improving both ID and OOD

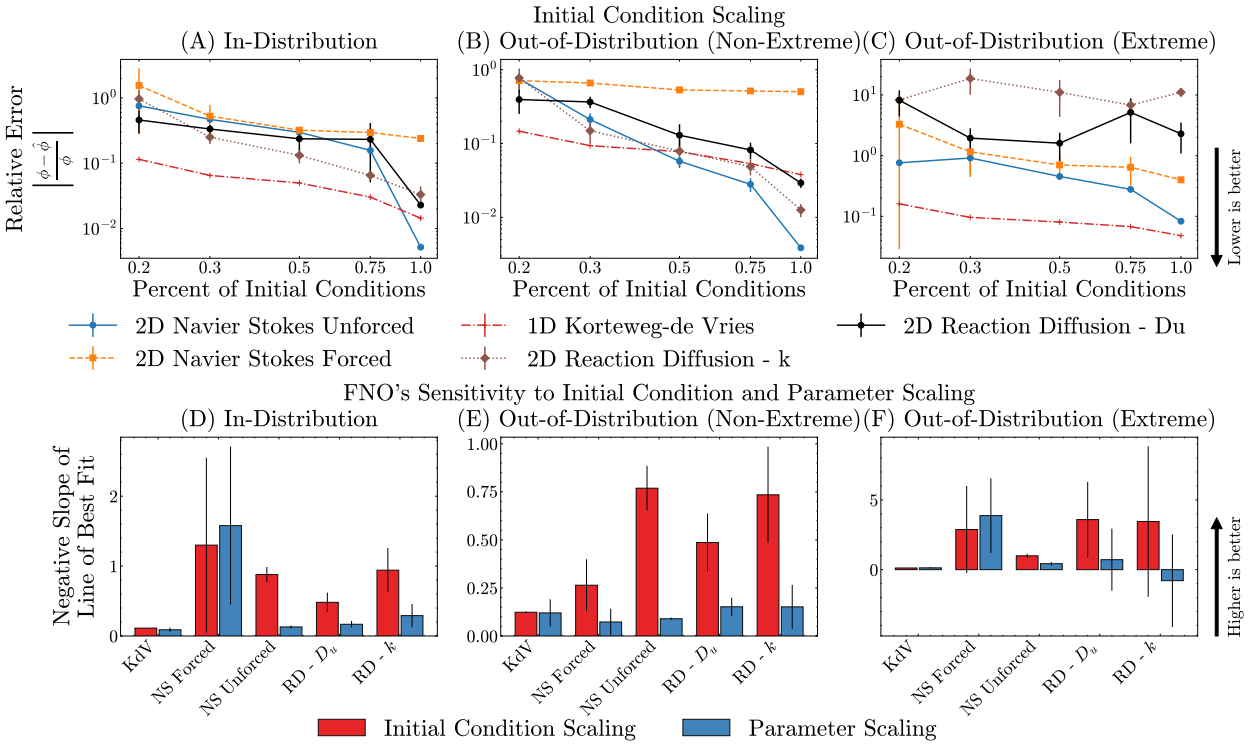

Figure 4: **Dataset scaling with the FNO architecture:** (**A**–**C**) Effect of initial condition scaling on FNO across evaluated systems. Increasing the percent of initial conditions used for training improves performance across all systems and evaluation settings. (**D**–**F**) FNO's sensitivity to initial condition scaling (while using 100% of available PDE parameters) versus scaling the number of generated PDE parameters (while using 100% of the available initial conditions) as measured by the negative slope of the line of best fit (NLS). The NLS is generated by fitting a line of best fit through each line in A–C, alongside parameter scaling, and taking the negative slope. Larger values indicate desirable faster scaling. FNO improves more quickly as the quantity of initial conditions is increased compared to when the quantity of PDE parameters in the training set is increased.

Non-Extreme errors (Figure 4A,B). The increase in initial conditions yields a modest reduction in relative error for the OOD Extreme case (Figure 4C) and lacks the strong trend seen on other splits.

Relative effect of data scaling. To discern which type of data scaling is the most impactful for model performance, we compare the relative effect of each type of data scaling on FNO in Figure 4(D-F). Increasing the number of initial conditions generally yields larger gains in performance compared to increasing the number of PDE parameters. Seeing new physical parameters does not provide the same amount of information as seeing the evolution of new initial conditions with the same physical parameters. In other words, in our setting, new initial conditions provide a better demonstration of the underlying mapping between solution fields and parameters.

7 Conclusion

We have introduced *PDEInvBench*, a comprehensive benchmark dataset for PDE inverse problems spanning diverse physical systems. We use *PDEInvBench* to systematically explore and identify best practices for solving PDE inverse problems with neural networks.

7.1 Key Insights

From our investigation of optimization procedures, problem representations, and scaling axes, we distill a set of practical, actionable guidelines for using neural networks to solve inverse problems.

Key insight 1: Practitioners should employ a two-stage training approach when learning neural networks for PDE inverse problems: 1) training with a supervised data loss, and 2) test-time training using the PDE residual to improve generalization to OOD PDE parameters and initial conditions.

Self-supervised training on the PDE residual is consistently worse than purely data-driven supervision (Section 6.1) across systems and irrespective of the weight β used for the residual loss term. Additionally, given an initial pre-trained network, test-time training (TTT) using the PDE residual leads to mixed results, with larger improvements when the initial condition and PDE parameter are OOD with respect to the training distribution (Section 6.1). However, since TTT never appreciably hurts performance, we recommend that it be used as long as the application is not highly compute constrained.

Key insight 2: For time-dependent PDEs, the FNO architecture generally outperforms ResNet and Transformer-style architectures, potentially suggesting that a spectral inductive bias is useful for solving PDE inverse problems.

This, however, does not hold true for time-independent PDEs, like Darcy Flow, where FNO was significantly outperformed by both ResNet and scOT (Section 6.2). Note that our findings are a result of our dataset size and the specific architectures we have chosen. Beyond inductive biases, there are more differences in architectures including normalization, skip connections, depth etc. For the specific systems, dataset regime, and training objective, we found FNO to generally outperform ResNet and scOT.

Key insight 3: Conditioning on the PDE partial derivatives via concatenation consistently improves model performance for the Fourier Neural Operator.

While expressive neural networks should be able to implicitly learn the differential operator, explicitly conditioning on PDE partial derivatives improves performance across all systems (Section 6.2).

Key insight 4: Generating trajectories with more initial conditions per PDE parameter is more beneficial than increasing the number of unique PDE parameters.

This suggests which axis to prioritize when generating data on a strict compute budget. Nonetheless, we find that increasing the amount of data along *any generation axis* universally improves performance (Section 6.3).

7.2 Future Work

Future work could extend our benchmarking study in several important directions. One avenue is to consider more PDEs beyond Darcy Flow with coefficients that vary freely in space, testing whether ML models can capture heterogeneous parameter fields rather than just scalar parameters (Leforestier et al., 1991). Another is to investigate inverse problems under sparse, irregularly sampled, or noisy measurements, which would better reflect realistic experimental conditions (Stuart, 2010). In these settings, it would be interesting to predict a distribution of PDE parameters with rigorous uncertainty quantification, rather than single point estimates of PDE parameters as done in the present work. Beyond the 2D settings explored here, the inclusion of 3D benchmark datasets such as Navier–Stokes turbulence would further challenge models on higher-dimensional dynamics while also probing scalability under memory and compute constraints (Du and Krishnapriyan, 2025). Our dataset also currently only contains systems with a uniform (square or linear) geometry, while many realistic systems have more complex topologies (e.g., spherical). We plan to include such systems in future work. Finally, incorporating solver-in-the-loop losses (e.g., running a PDE solver with the predicted parameters and enforcing consistency with a reference trajectory) alongside data and PDE residual losses could enforce stronger physics consistency and improve generalization across regimes and boundary conditions (Um et al., 2020). We believe *PDEInvBench* will act as a solid foundation for such further advances in ML approaches to PDE inverse problems.

References

B. Alkin, A. Fürst, S. L. Schmid, L. Gruber, M. Holzleitner, and J. Brandstetter. Universal Physics Transformers: A Framework For Efficiently Scaling Neural Operators. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=oUXiNX5KRm`.

K. J. Åström and P. Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1814058116. _eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1814058116.

L. Bhan, Y. Bian, M. Krstic, and Y. Shi. PDE Control Gym: A Benchmark for Data-Driven Boundary Control of Partial Differential Equations, 2024. _eprint: 2405.11401.

M. Blanke and M. Lelarge. Interpretable meta-learning of physical systems. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nnicaG5xiH.

P. Borckmans, G. Dewel, A. De Wit, and D. Walgraef. Turing Bifurcations and Pattern Selection. In R. Kapral and K. Showalter, editors, *Chemical Waves and Patterns*, pages 323–363. Springer Netherlands, Dordrecht, 1995. ISBN 978-94-011-1156-0. doi: 10.1007/978-94-011-1156-0_10. URL https://doi.org/10.1007/978-94-011-1156-0_10.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

J. Brandstetter, M. Welling, and D. E. Worrall. Lie Point Symmetry Data Augmentation for Neural PDE Solvers. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2241–2256. PMLR, July 2022. URL https://proceedings.mlr.press/v162/brandstetter22a.html.

N. B. Budanur and H. Kantz. Scale-dependent error growth in navier-stokes simulations. *Phys. Rev. E*, 106:045102, Oct 2022. doi: 10.1103/PhysRevE.106.045102. URL https://link.aps.org/doi/10.1103/PhysRevE.106.045102.

C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*. Scientific Computation. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-30728-0. URL https://books.google.com/books?id=iDckv0W52cQC.

S. Cao. Choose a Transformer: Fourier or Galerkin. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ssohLcmn4-r.

S. W. Cho and H. Son. Physics-Informed Deep Inverse Operator Networks for Solving PDE Inverse Problems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=0FxnSZJPmh.

J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980. ISSN 0377-0427. doi: https://doi.org/10.1016/0771-050X(80)90013-3. URL https://www.sciencedirect.com/science/article/pii/0771050X80900133.

G. Dresdner, D. Kochkov, P. C. Norgaard, L. Zepeda-Nunez, J. Smith, M. Brenner, and S. Hoyer. Learning to correct spectral methods for simulating turbulent flows. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wNBARGxoJn.

Y. Du and A. Krishnapriyan. EddyFormer: Accelerated Neural Simulations of Three-Dimensional Turbulence at Scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. doi: 10.48550/arXiv.2510.24173. URL http://arxiv.org/abs/2510.24173.

Y. Du, N. Chalapathi, and A. Krishnapriyan. Neural Spectral Methods: Self-supervised learning in the spectral domain. *The Twelfth International Conference on Learning Representations*, 2024.

H. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer, 1996. ISBN 0-7923-4157-0.

L. C. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 2nd edition, 2022. ISBN 978-0-8218-4974-3.

J. K. Gupta and J. Brandstetter. Towards Multi-spatiotemporal-scale Generalized PDE Modeling. *arXiv preprint arXiv:2209.15616*, 2022.

E. Hairer and G. Wanner. Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111(1):93–111, 1999. ISSN 0377-0427. doi: https://doi.org/10.1016/S0377-0427(99)00134-X. URL https://www.sciencedirect.com/science/article/pii/S037704279900134X.

Z. Hao, Z. Wang, H. Su, C. Ying, Y. Dong, S. Liu, Z. Cheng, J. Song, and J. Zhu. GNOT: a general neural operator transformer for operator learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023a. Place: Honolulu, Hawaii, USA.

Z. Hao, J. Yao, C. Su, H. Su, Z. Wang, F. Lu, Z. Xia, Y. Zhang, S. Liu, L. Lu, and others. PINNacle: A Comprehensive Benchmark of Physics-Informed Neural Networks for Solving PDEs. *arXiv preprint arXiv:2306.08827*, 2023b.

S. M. S. Hassan, A. Feeney, A. Dhruv, J. Kim, Y. Suh, J. Ryu, Y. Won, and A. Chandramowlishwaran. BubbleML: A Multi-Physics Dataset and Benchmarks for Machine Learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=0Wmglu8zak.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459. ISSN: 1063-6919 Place: Las Vegas, NV, USA.

M. Herde, B. Raonić, T. Rohner, R. Käppeli, R. Molinaro, E. de Bézenac, and S. Mishra. Poseidon: Efficient Foundation Models for PDEs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 72525–72624. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/84e1b1ec17bb11c57234e96433022a9a-Paper-Conference.pdf.

V. Isakov. *Inverse Problems for Partial Differential Equations*, volume 127 of *Applied Mathematical Science*. Springer Cham, 3 edition, 2017. ISBN 978-3-319-51658-5. URL https://doi.org/10.1007/978-3-319-51658-5.

A. Jiao, Q. Yan, J. Harlim, and L. Lu. Solving forward and inverse PDE problems on unknown manifolds via physics-informed neural operators, 2024. URL https://arxiv.org/abs/2407.05477. _eprint: 2407.05477.

P. Karnakov, S. Litvinov, and P. Koumoutsakos. Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks. *PNAS Nexus*, 3(1):pgae005, Jan. 2024. ISSN 2752-6542. doi: 10.1093/pnasnexus/pgae005. URL https://doi.org/10.1093/pnasnexus/pgae005. _eprint: https://academic.oup.com/pnasnexus/article-pdf/3/1/pgae005/56347037/pgae005.pdf.

M. Kirchmeyer, Y. Yin, J. Dona, N. Baskiotis, A. Rakotomamonjy, and P. Gallinari. Generalizing to new physical systems via context-informed dynamics model. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11283–11301. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/kirchmeyer22a.html.

P. K. Kitanidis. Bayesian and Geostatistical Approaches to Inverse Problems. In *Large-Scale Inverse Problems and Quantification of Uncertainty*, pages 71–85. John Wiley & Sons, Ltd, 2010. ISBN 978-0-470-68585-3. doi: https://doi.org/10.1002/9780470685853.ch4. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470685853.ch4. Section: 4 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470685853.ch4.

G. A. Klaasen and W. C. Troy. Stationary Wave Solutions of a System of Reaction-Diffusion Equations Derived from the FitzHugh–Nagumo Equations. *SIAM Journal on Applied Mathematics*, 44(1):96–110, 1984. doi: 10.1137/0144008. URL https://doi.org/10.1137/0144008. _eprint: https://doi.org/10.1137/0144008.

D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2101784118. _eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2101784118.

G. Kohl, L.-W. Chen, and N. Thuerey. Benchmarking Autoregressive Conditional Diffusion Models for Turbulent Flow Simulation. *arXiv*, 2023. doi: 10.48550/arXiv.2309.01745. URL https://doi.org/10.48550/arXiv.2309.01745. Publisher: arXiv _eprint: 2309.01745.

T. Koshizuka, M. Fujisawa, Y. Tanaka, and I. Sato. Understanding the expressivity and trainability of fourier neural operator: A mean-field perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=QJr02BTM7J.

A. K. Koupaï, J. M. Benet, Y. Yin, J.-N. Vittaut, and P. Gallinari. Boosting generalization in parametric PDE neural solvers through adaptive conditioning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=GuY0zB2xVU.

N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(1), Jan. 2023. ISSN 1532-4435. Publisher: JMLR.org.

A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

M. Lee, G. Park, C. Park, and C. Kim. Improvement of grid independence test for computational fluid dynamics model of building based on grid resolution. *Advances in Civil Engineering*, 2020(1):8827936, 2020. doi: https://doi.org/10.1155/2020/8827936. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2020/8827936.

C. Leforestier, R. H. Bisseling, C. Cerjan, M. D. Feit, R. Friesner, A. Guldberg, A. Hammerich, G. Jolicard, W. Karrlein, H.-D. Meyer, N. Lipkin, O. Roncero, and R. Kosloff. A comparison of different propagation schemes for the time dependent Schrödinger equation. *Journal of Computational Physics*, 94(1):59–80, 1991. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(91)90137-A. URL https://www.sciencedirect.com/science/article/pii/002199919190137A.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar. Multipole Graph Neural Operator for Parametric Partial Differential Equations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6755–6766. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4b21cf96d4cf612f239a6c322b10c8fe-Paper.pdf.

Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Z. Li, N. B. Kovachki, C. Choy, B. Li, J. Kossaifi, S. P. Otta, M. A. Nabian, M. Stadler, C. Hundt, K. Azizzadenesheli, and A. Anandkumar. Geometry-Informed Neural Operator for Large-Scale 3D PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=86dXbqT5Ua.

Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-Informed Neural Operator for Learning Partial Differential Equations. *ACM / IMS J. Data Sci.*, 1(3), May 2024. doi: 10.1145/3648506. URL https://doi.org/10.1145/3648506. Place: New York, NY, USA Publisher: Association for Computing Machinery.

Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo. Swin Transformer V2: Scaling Up Capacity and Resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11999–12009, 2022. doi: 10.1109/CVPR52688.2022.01170.

L. Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. In *Nature Machine Intelligence*, volume 3, pages 218–229, Mar. 2021. doi: 10.1038/s42256-021-00302-5.

L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.114778. URL https://www.sciencedirect.com/science/article/pii/S0045782522001207.

D. MacKinlay, D. Pagendam, P. M. Kuhnert, T. Cui, D. Robertson, and S. Janardhanan. Model Inversion for Spatio-temporal Processes using the Fourier Neural Operator. In *Neurips Workshop on Machine Learning for the Physical Sciences*, page 7, 2021.

S. Molesky, Z. Lin, A. Y. Piggott, W. Jin, J. Vucković, and A. W. Rodriguez. Inverse design in nanophotonics. *Nature Photonics*, 12(11):659–670, Nov. 2018. ISSN 1749-4893. doi: 10.1038/s41566-018-0246-9. URL https://doi.org/10.1038/s41566-018-0246-9.

R. Molinaro, Y. Yang, B. Engquist, and S. Mishra. Neural Inverse Operators for Solving PDE Inverse Problems. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25105–25139. PMLR, July 2023. URL https://proceedings.mlr.press/v202/molinaro23a.html.

R. Ohana and others. The Well: A large-scale dataset collection for scientific machine learning. In *38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks*, 2024.

A. D. Polyanin and V. F. Zaitsev. *Handbook of Nonlinear Partial Differential Equations*. Chapman & Hall/CRC Press, Boca Raton-London-New York, 2nd edition, 2012. ISBN 1-58488-355-3.

S. Qin, F. Lyu, W. Peng, D. Geng, J. Wang, X. Tang, S. Leroyer, N. Gao, X. Liu, and L. L. Wang. Toward a better understanding of fourier neural operators from a spectral perspective, 2024. URL https://arxiv.org/abs/2404.07200.

M. A. Rahman, Z. E. Ross, and K. Azizzadenesheli. U-NO: U-shaped Neural Operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=j3oQF9coJd.

P. Schober, C. Boer, and L. A. Schwarte. Correlation coefficients: Appropriate use and interpretation. *Anesthesia & Analgesia*, 126(5), 2018. ISSN 0003-2999. URL https://journals.lww.com/anesthesia-analgesia/fulltext/2018/05000/correlation_coefficients__appropriate_use_and.50.aspx.

A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, 2010. doi: 10.1017/S0962492910000061.

M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert. PDEBENCH: An Extensive Benchmark for Scientific Machine Learning. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022. URL https://openreview.net/forum?id=dh_MkX0QfrK.

A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005. doi: 10.1137/1.9780898717921. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898717921. _eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898717921.

A. Toshev, G. Galletti, F. Fritz, S. Adami, and N. Adams. Lagrangebench: A lagrangian fluid mechanics benchmarking suite. *Advances in Neural Information Processing Systems*, 36, 2024.

A. Tran, A. Mathews, L. Xie, and C. S. Ong. Factorized Fourier Neural Operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tmIiMPl4IPa.

K. Um, R. Brand, Y. Fei, P. Holl, and N. Thuerey. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. *Advances in Neural Information Processing Systems*, 2020.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

M. T. Vlaardingerbroek and J. A. Boer. *Magnetic Resonance Imaging: Theory and Practice*. Springer Berlin, Heidelberg, 3 edition, 2013. ISBN 978-3-662-05252-5. URL `https://doi.org/10.1007/978-3-662-05252-5`.

T. Wang and C. Wang. Latent Neural Operator for Solving Forward and Inverse PDE Problems. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 33085–33107. Curran Associates, Inc., 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/39f6d5c2e310a5a629dcfc4d517aa0d1-Paper-Conference.pdf`.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-Art Natural Language Processing. In Q. Liu and D. Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL `https://aclanthology.org/2020.emnlp-demos.6/`.

H. Wu, H. Luo, H. Wang, J. Wang, and M. Long. Transolver: A Fast Transformer Solver for PDEs on General Geometries, June 2024. URL `http://arxiv.org/abs/2402.02366`. arXiv:2402.02366 [cs, math].

N. J. Zabusky and M. D. Kruskal. Interaction of "Solitons" in a Collisionless Plasma and the Recurrence of Initial States. *Phys. Rev. Lett.*, 15(6):240–243, Aug. 1965. doi: 10.1103/PhysRevLett.15.240. URL `https://link.aps.org/doi/10.1103/PhysRevLett.15.240`. Publisher: American Physical Society.