# CODBENCH: A Critical Evaluation of Data-driven Models for Continuous Dynamical Systems

**Priyanshu Burark**
Department of Computer Science & Automation
Indian Institute of Science, Bangalore
Bengaluru, 560012, India
priyanshub@iisc.ac.in

**Karn Tiwari, Prathosh A P**
Department of ECE
Indian Institute of Science, Bangalore
Bengaluru, 560012, India
{karntiwari, prathosh}@iisc.ac.in

**Meer Mehran Rashid**
Department of Civil Engineering
Indian Institute of Technology, Delhi
New Delhi, 110016, India
mrashi12@jhu.edu

**N M Anoop Krishnan**
Yardi School of Artificial Intelligence
Indian Institute of Technology, Delhi
New Delhi, 110016, India
krishnan@iitd.ac.in

## Abstract

Continuous dynamical systems, characterized by differential equations, are ubiquitously used to model several important problems: plasma dynamics, flow through porous media, weather forecasting, and epidemic dynamics. Recently, a wide range of data-driven models has been used successfully to model these systems. However, in contrast to established fields like computer vision, limited studies are available analyzing the strengths and potential applications of different classes of these models that could steer decision-making in scientific machine learning. Here, we introduce CODBENCH, an exhaustive benchmarking suite comprising 11 state-of-the-art data-driven models for solving differential equations. Specifically, we comprehensively evaluate 4 distinct categories of models, *viz.*, feed forward neural networks, deep operator regression models, frequency-based neural operators, and transformer architectures against 8 widely applicable benchmark datasets encompassing challenges from fluid and solid mechanics. We conduct extensive experiments, assessing the operators' capabilities in learning, zero-shot super-resolution, data efficiency, robustness to noise, and computational efficiency. Interestingly, our findings highlight that current operators struggle with the newer mechanics datasets, motivating the need for more robust neural operators. All the datasets and codes are shared in an easy-to-use fashion for the scientific community. We hope this resource will be an impetus for accelerated progress and exploration in modeling dynamical systems. For codes and datasets, see: https://anonymous.4open.science/r/cod-bench-7525.

## 1 Introduction

Nature is in a continuous state of evolution. "Rules" governing the time evolution of systems in nature, also known as dynamics, can be captured mathematically through partial differential equations (PDEs). In the realm of science and engineering, PDEs are widely used to model and study several challenging real-world systems, such as fluid flow, deformation of solids, plasma dynamics, robotics, mechanics, and weather forecasting, to name a few [6, 22, 25]. Due to their highly non-linear and coupled nature, these PDEs can be solved analytically only for trivial or model systems. Thus, accurate numerical solutions for the PDEs are the cornerstone in advancing scientific discovery. Traditionally,

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

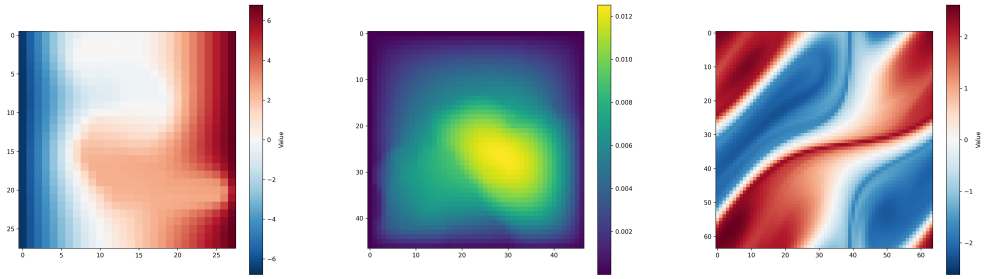

Figure 1: A glimpse of the data diversity under consideration. From left to right, we present visual representations of three distinct datasets — Biaxial, Darcy Flow, and Navier-Stokes.

the PDEs are solved using classical numerical methods such as finite difference, finite volume, or finite element methods [27]. However, these numerical methods exhibit major challenges in realistic systems in terms of system size, timescales, and numerical instabilities. Specifically, simulating the systems for longer timescale or for large domains is extremely computationally intensive to the extent that performing them in real-time for decision-making is a major challenge. Further, in the case of large/highly non-linear fields, these simulations often exhibit numerical instabilities, rendering them ineffective. [29]

The recent surge in artificial intelligence-based approaches suggests that neural models can efficiently capture continuous dynamical systems in a data-driven fashion [2]. These models are extremely time-efficient in comparison to traditional solvers and can capture highly non-linear input-output relationships. Earlier approaches in this direction relied directly on learning the input-output map through multilayer perceptrons (MLPs), convolutional neural networks, or graph neural networks. However, these approaches faced challenges in terms of generalizing to unseen initial or boundary conditions, geometries, or resolutions. This could be attributed to the fact that the neural models essentially learn the input-output relationship in a finite-dimensional approximation. To address this challenge, a seminal theory, extending the universal approximation theorem of neural networks [5] to neural operators was proposed, namely, the universal operator approximation theory [4]. This theory unveiled the neural networks' prowess in handling infinite-dimensional inputs and outputs.

Theoretically, directly learning the solution operator through specialized neural network architectures offers several key advantages. (i) They can directly learn input-output function mappings from data, thereby obviating the necessity for prior knowledge of the underlying PDE. (ii) They offer significantly improved time efficiency compared to traditional numerical solvers. (iii) They exhibit zero-shot generalizability, extending their applicability to systems of larger scale and complexity than those encompassed within the training dataset. (iv) They provide superior approximations of the solution operator compared to existing neural architectures, spanning from feed-forward networks to specialized models like convolutional networks and conditional generative adversarial networks (GANs). Thus, the neural operators attempt [13] to combine the best of both data-driven and physics-based numerical models.

This motivated the exploration of neural operator architectures [1], [23], capable of directly learning the solution operator. For instance, consider DEEPONET [18], which leverages the universal approximation theorem introduced by Chen and Chen to directly address PDEs. On a different front, FNO [16], one of the most widely used Neural Operators, focuses on parameterizing the integral kernel within Fourier space. Moreover, a noteworthy study [3] highlights the notion that all transformers are essentially operators. This insight has sparked endeavors to create operator transformers. Given their proven effectiveness in sequence-to-sequence learning tasks, these transformer-based designs open avenues for enhancing the approximation of spatiotemporal PDEs. Prior studies, such as those by [8], have delved into the realm of PINNs [24] and some neural operator architectures, like DEEPONET, FNO, and their variants. However, unlike fields like computer vision, comprehensive comparative evaluations of these neural operators are absent. Moreover, due to the variations and incompatibilities in architectures, a direct comparison of all these architectures is extremely cumbersome. Such evaluations are pivotal to discerning the distinctive advantages of diverse architectural paradigms, especially when addressing equations from a wide spectrum of scientific domains.

This study aims to bridge this gap by rigorously evaluating data-driven models that encompass a wide range of classes and methods, including the foundational deep operator regression model,

frequency domain parameterization models, and transformer-based architectures, to achieve state-of-the-art performance comparison on selected PDE datasets. Moreover, we integrate conventional neural architectures to underscore the merits of PDE-specialized structures. Our dataset selection is methodical, designed to challenge each model with equations from various scientific disciplines. We incorporate four prevalent equations from fluid dynamics and four standard differential equations from solid mechanics into the neural operator domain, ensuring a holistic comparison within the realm of neural operators.

**Our Contribution:** In this work, we critically analyze 11 data-driven models, including operators and transformers, on 8 PDE datasets. The major contributions of our research are as follows:

1. **CODBENCH:** We present a package that allows seamless analysis of several data-driven approaches on PDEs. We thoroughly assess state-of-the-art data-driven neural models for solving PDE datasets across diverse scientific realms, such as fluid and solid mechanics, shedding light on their precision and efficacy.
2. **Super-resolution:** We analyze the ability of neural operators' to generalize to systems of different resolutions than that of their training sets' discretizations.
3. **Data efficiency and robustness to noise:** We critically assess the efficiency of these models to learn from small amounts of data or noisy data. This is an important aspect since the data available can be scarce and noisy in practical applications.
4. **Out-of-distribution task:** A novel task to gain insights into what these models are truly learning to determine whether the underlying operator is genuinely being learned or if the training dataset is simply being fitted. Two closely related STRESS and STRAIN datasets are interchanged during training and testing to dig deeper into whether the solvers are actually operators.

## 2 Preliminaries

This section provides a concise mathematical framework to illustrate how traditional PDE solving can be transitioned and addressed using data-driven methodologies via neural networks.

1. **Function Domains**: Consider a bounded open set, represented as $\mathcal{D} \subset \mathbb{R}^d$. Within this domain, we define $\mathcal{F} = \mathcal{F}(\mathcal{D}; \mathbb{R}^{d_f})$ and $\mathcal{G} = \mathcal{G}(\mathcal{D}; \mathbb{R}^{d_g})$ as separable Banach spaces. These spaces correspond to input and output functions, which represent elements in $\mathbb{R}^{d_f}$ and $\mathbb{R}^{d_g}$, respectively.
2. **The Solution Operator**: In our exploration, we introduce $T^\dagger : \mathcal{F} \to \mathcal{G}$, a mapping that is typically nonlinear. This mapping emerges as the solution operator for PDEs, playing a pivotal role in scientific computations.
3. **Data Generation**: For training purposes, models utilize PDE datasets constructed as $\mathcal{D} = \{(\mathcal{F}_k, \mathcal{G}_k)\}_{1 \le k \le D}$, where $\mathcal{G}_k = T^\dagger(\mathcal{F}_k)$. Given the inherent challenges in directly representing functions as inputs to neural networks, the functions are discretized using mesh generation algorithms [33] over domain $\mathcal{D}$. We sample both input and output functions on a uniform grid, as it ensures compatibility with all selected solvers. For the input function $\mathcal{F}_k$, we discretize it on the mesh $\{x_i \in \Omega\}_{1 \le i \le R}$, and the discretized $\mathcal{F}_k$ is $\{(x_i, f_{ik})\}_{1 \le i \le R}$, where $f_{ik} = \mathcal{F}_k(x_i)$. Similarly, For the solution function $\mathcal{G}_k$, we discretize it on the mesh $\{y_i \in \Omega\}_{1 \le i \le R}$, and the discretized $\mathcal{G}_k$ is $\{(y_i, g_{ik})\}_{1 \le i \le R}$, where $g_{ik} = \mathcal{G}_k(y_i)$. It's worth noting that models such as POD-DEEPONET and SNO utilize only the function values for representation, excluding grid locations from the model input.
4. **Objective**: The overarching goal for each model is to craft an approximation of $T^\dagger$. This is achieved by developing a parametric mapping, denoted as $T : \mathcal{F} \times \Theta \to \mathcal{G}$ or, in an equivalent form, $T_\theta : \mathcal{F} \to \mathcal{G}$, where $\theta \in \Theta$. This mapping operates within a bounded parameter space, $\Theta$.
5. **Metric**: Evaluating the efficacy of the parametric mapping involves comparing its outputs, $T_\theta(\mathcal{F}_k) = \{\widetilde{g}_{ik}\}_{1 \le i \le R}$, with the actual data, aiming to minimize the relative L2 loss, given by:

$$\min_{\theta \in \Theta} \frac{1}{D} \sum_{k=1}^{D} \frac{1}{R} \frac{\|T_\theta(\mathcal{F}_k) - \{\widetilde{g}_{ik}\}_{1 \le i \le R}\|_2^2}{\|\{\widetilde{g}_{ik}\}_{1 \le i \le R}\|_2^2}, \tag{1}$$

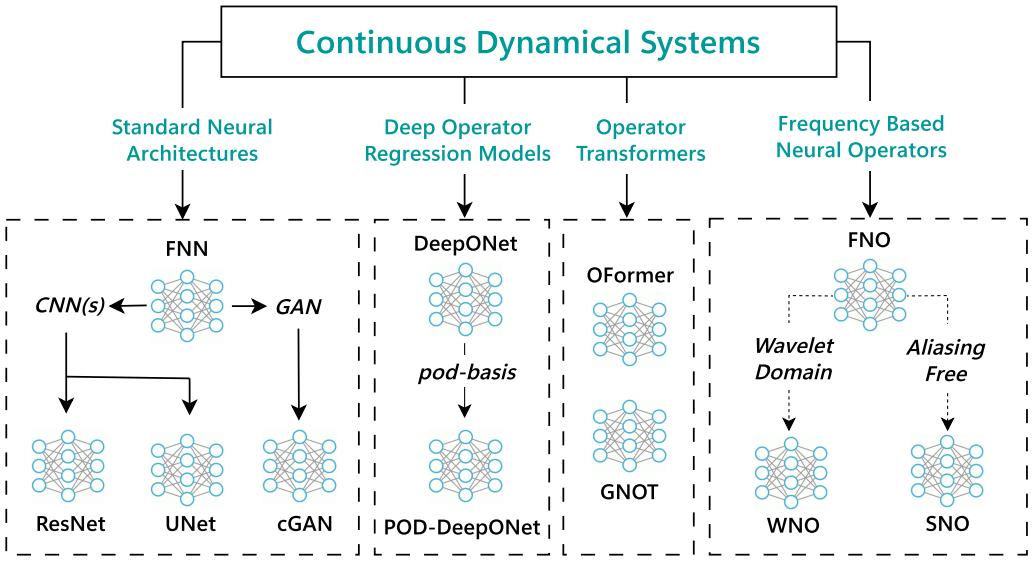


Figure 2: An overview of the various models being benchmarked and the relationship between them. The term 'pod-basis' denotes the basis of the output function, derived directly from proper orthogonal decomposition, as opposed to being learned through a neural network.

## 3 Model Architectures

**Standard Neural Network Architectures:** This study encompasses a broad spectrum of architectures, as illustrated in Figure 2. The Feed-Forward Neural Network (FNN) serves as the foundational component, distinguished by its pointwise configuration. Prevalent CNN-based architectures like UNET, RESNET, and CGAN are also incorporated. UNET, delineated in [26], employs a U-shaped encoder-decoder design augmented by skip connections, facilitating the capture of both granular and abstract features. RESNET, described in [11], consist of a series of residual blocks and are commonly used in computer vision tasks [31]. Conditional Generative Adversarial Networks (CGAN), introduced in [21], are an evolution of the GAN framework, facilitating conditional generation via the incorporation of label information in both the generator and discriminators.

**Deep Operator-Based Regression Models:** Neural Operators represent a novel ML paradigm, predominantly employed in scientific machine learning to decipher PDEs. These operators discern mappings between infinite-dimensional Euclidean spaces, relying solely on data and remaining agnostic to the underlying PDE. Within this study, we delve into deep operator regression models. DEEPONET bifurcates into two sub-networks: the branch net, which encodes the input function at fixed sensor locations, and the trunk net, encoding solution function locations [18]. The solution emerges from the inner product of the outputs from these nets. In POD-DEEPONET, the bases are determined by executing proper orthogonal decomposition (POD) on training data, replacing the self-learned basis of output functions [19]. This POD basis forms the trunk net, leaving only the branch net as the trainable component, which discerns the coefficients of the POD basis.

**Frequency-Based Operators:** Frequency-based solvers like FNO employ a finite-dimensional parameterization using truncated Fourier modes [16]. By integrating this with an integral operator restricted to convolution, and instantiated via a linear transformation in the Fourier domain, the FNO operator is conceived. WNO, or Wavelet Neural Operator, amalgamates the prowess of wavelets in time-frequency localization with an integral kernel. By learning the kernel in the wavelet domain, convolution operates on wavelet decomposition coefficients rather than direct physical space convolution [32]. SNO, the Spectral Neural Operator, addresses the often-overlooked aliasing error in the Fourier Neural Operator. By representing both input and output functions using coefficients in truncated Fourier or Chebyshev series, SNO offers an aliasing-free approach [7]. Any transformation between these coefficients can be executed using neural networks, and methods employing these series are termed spectral neural operators. In their approach, a straightforward feed-forward neural network architecture in the complex domain is utilized.

**Transformer Operators:** "GNOT introduces the Heterogeneous Normalized (linear) Attention (HNA) block and a geometric gating mechanism, specifically tailored for enhanced performance on PDE datasets [9]. This architecture effectively performs a soft domain decomposition [10], treating each decomposed domain independently and subsequently integrating them using a mixture-of-experts approach to predict the underlying truth. This design allows GNOT to serve as a versatile operator adept at handling a variety of PDE types. In contrast, the OFORMER model builds upon the seminal work presented in [34]. It incorporates random Fourier projection to counteract spectral bias, thereby enhancing its efficacy on PDEs [15].

## 4 Datasets

Here, we briefly describe the 8 datasets used in the present. While previous approaches have mostly focussed on fluid datasets, here we present 4 datasets on fluid flow and 4 on the deformation of solids; for complete dataset details, refer A.1, A.2.

1. **BURGERS**: This dataset models the one-dimensional flow of a viscous fluid. The input is the fluid's initial velocity distribution at time $t = 0$, and the output is the fluid's velocity at a time $t > 0$ [30].
2. **DARCY**: The Darcy Flow dataset describes the steady-state flow of a fluid through a porous medium in two dimensions. The input is the spatial distribution of the medium's resistance to flow (viscosity), and the output is the fluid's velocity distribution across the domain at steady-state [30].
3. **NAVIER STOKES**: This dataset models the time evolution of a 2D viscous, incompressible fluid. The input includes the fluid's initial swirling motion (vorticity) and external forces acting on the fluid. The output is the fluid's velocity distribution over a specified time period [30].
4. **SHALLOW WATER**: The shallow-water equations simulate the behavior of water that flows over a shallow surface in 2D. The input consists of the initial water depth and velocity distribution, and the output predicts the water flow dynamics in response to gravitational forces and varying underwater terrain (bathymetry) [30].
5. **STRESS**: This dataset models the stress distribution in a 2D binary composite material subjected to mode-I tensile loading. The input is the material microstructure (distribution of two materials), and the output is the stress field (STRESS) distribution of the digital composite [20].
6. **STRAIN**: The strain dataset describes the deformation of a 2D binary composite material subjected to mode-I tensilie loading. The input is the material microstructure and the output is the resulting strain fields (STRAIN) [20].
7. **SHEAR**: Part of the mechanical MNIST collection, this dataset simulates the deformation of a heterogeneous material block when forces are applied parallel to its surface (SHEAR). The input is the material microstructure, and the output captures element-wise displacements subjected to shear loading [14].
8. **BIAXIAL**: Another subset of the mechanical MNIST experiments, this dataset models the material's response when stretched equally in two perpendicular directions (equibiaxial loading). The input is the material microsturcture, and the output records the full field displacement under BIAXIAL stretching [14].

## 5 Benchmarking Results

We present the results of rigorous experimentation on PDE solvers across six tasks, each designed to showcase the unique capabilities and strengths of the models. The diversity of the selected PDEs, sourced from [30], [14], and [20], encompasses both time-dependent and time-independent challenges, capturing the intrinsic computational complexity inherent to these tasks. Specifically, the experiments conducted on novel mechanical datasets not previously encountered by the solvers, offer invaluable insights for the broader scientific community.

In alignment with established experimental protocols, the dataset was split as follows: $\sim 80\%$ for training, $\sim 10\%$ for validation, and $\sim 10\%$ for testing; refer 4 for more details. Our ensemble training methodology ensured a level playing field for each operator by defining a hyperparameter

| | Datasets | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Models | BURGERS | DARCY | NAVIER STOKES | SHALLOW WATER | STRESS | STRAIN | SHEAR | BIAXIAL |
| FNN | $5.853_{\pm 1.416}$ | $3.47_{\pm 0.14}$ | $34.77_{\pm 0.19}$ | $2.424_{\pm 0.656}$ | $25.69_{\pm 0.59}$ | $23.09_{\pm 7.08}$ | $1.11_{\pm 0.06}$ | $3.69_{\pm 0.01}$ |
| RESNET | $11.327_{\pm 1.208}$ | $5.14_{\pm 0.23}$ | $29.52_{\pm 0.14}$ | $0.287_{\pm 0.093}$ | $20.05_{\pm 0.19}$ | $14.64_{\pm 0.31}$ | $3.02_{\pm 0.95}$ | $13.58_{\pm 2.67}$ |
| UNET | $30.870_{\pm 2.000}$ | $2.10_{\pm 0.08}$ | $24.02_{\pm 0.95}$ | $0.295_{\pm 0.097}$ | $10.57_{\pm 0.19}$ | $9.05_{\pm 0.33}$ | $7.09_{\pm 0.46}$ | $16.63_{\pm 2.30}$ |
| CGAN | $34.906_{\pm 0.506}$ | $1.88_{\pm 0.04}$ | $24.00_{\pm 0.48}$ | $0.291_{\pm 0.027}$ | $\mathbf{6.66_{\pm 0.84}}$ | $6.12_{\pm 0.80}$ | $5.63_{\pm 0.50}$ | $15.74_{\pm 1.40}$ |
| FNO | $\mathbf{0.160_{\pm 0.004}}$ | $\mathbf{1.08_{\pm 0.06}}$ | $14.13_{\pm 0.34}$ | $0.128_{\pm 0.018}$ | $8.08_{\pm 0.15}$ | $\mathbf{5.61_{\pm 0.23}}$ | $2.25_{\pm 1.14}$ | $7.40_{\pm 1.91}$ |
| WNO | $7.332_{\pm 0.307}$ | $2.23_{\pm 0.14}$ | $37.08_{\pm 1.23}$ | $0.572_{\pm 0.036}$ | $17.24_{\pm 0.46}$ | $12.05_{\pm 0.26}$ | $4.37_{\pm 0.08}$ | $22.22_{\pm 2.86}$ |
| SNO | $40.623_{\pm 8.437}$ | $8.55_{\pm 1.03}$ | $98.46_{\pm 0.25}$ | $94.891_{\pm 0.060}$ | $51.31_{\pm 0.01}$ | $62.34_{\pm 1.17}$ | $4.37_{\pm 0.87}$ | $21.93_{\pm 0.57}$ |
| DEEPONET | $10.561_{\pm 1.182}$ | $4.27_{\pm 0.24}$ | $55.48_{\pm 1.06}$ | $8.602_{\pm 0.431}$ | $24.59_{\pm 0.98}$ | $23.75_{\pm 0.20}$ | $2.85_{\pm 0.18}$ | $8.28_{\pm 0.37}$ |
| POD-DEEPONET | $3.999_{\pm 0.654}$ | $3.43_{\pm 0.04}$ | $33.37_{\pm 1.30}$ | $1.503_{\pm 0.145}$ | $29.63_{\pm 0.52}$ | $18.31_{\pm 1.17}$ | $4.14_{\pm 0.44}$ | $30.46_{\pm 0.59}$ |
| OFORMER | $0.165_{\pm 0.016}$ | $3.21_{\pm 0.06}$ | $\mathbf{10.97_{\pm 3.03}}$ | $6.597_{\pm 0.352}$ | $27.33_{\pm 0.28}$ | $25.08_{\pm 1.36}$ | $41.75_{\pm 0.19}$ | $61.16_{\pm 0.49}$ |
| GNOT | $0.677_{\pm 0.021}$ | $2.04_{\pm 0.05}$ | $23.73_{\pm 0.97}$ | $\mathbf{0.102_{\pm 0.007}}$ | $13.02_{\pm 0.81}$ | $9.99_{\pm 0.62}$ | $\mathbf{0.43_{\pm 0.02}}$ | $\mathbf{0.71_{\pm 0.04}}$ |

Table 1: Performance of different models across diverse datasets from distinct domains. The Relative L2 Error, expressed as ($\times 10^{-2}$), is presented as the evaluation metric. Lower scores denote better performance. The optimal outcomes are highlighted in bold and dark blue, followed by the second-best in dark blue, and the third-best in light blue.

| | Models | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset Size | FNN | RESNET | UNET | CGAN | FNO | WNO | SNO | DEEPONET | POD-DEEPONET | OFORMER | GNOT |
| 25% | $4.80_{\pm 0.27}$ | $6.23_{\pm 0.23}$ | $2.60_{\pm 0.14}$ | $3.28_{\pm 0.13}$ | $\mathbf{1.87_{\pm 0.13}}$ | $2.94_{\pm 0.20}$ | $24.70_{\pm 1.08}$ | $7.50_{\pm 0.45}$ | $5.09_{\pm 0.20}$ | $3.94_{\pm 0.13}$ | $3.61_{\pm 0.20}$ |
| 50% | $3.95_{\pm 0.24}$ | $5.20_{\pm 0.29}$ | $2.10_{\pm 0.11}$ | $2.54_{\pm 0.13}$ | $\mathbf{1.32_{\pm 0.10}}$ | $2.37_{\pm 0.18}$ | $24.70_{\pm 1.12}$ | $6.15_{\pm 0.41}$ | $4.17_{\pm 0.28}$ | $3.32_{\pm 0.08}$ | $2.70_{\pm 0.13}$ |
| 100% | $3.47_{\pm 0.14}$ | $5.14_{\pm 0.23}$ | $2.10_{\pm 0.08}$ | $1.88_{\pm 0.04}$ | $\mathbf{1.08_{\pm 0.06}}$ | $2.23_{\pm 0.14}$ | $8.55_{\pm 1.03}$ | $4.27_{\pm 0.24}$ | $3.43_{\pm 0.04}$ | $3.21_{\pm 0.06}$ | $2.04_{\pm 0.05}$ |

Table 2: Data-Efficiency Analysis: The Relative L2 Error ($\times 10^{-2}$) is reported when trained with reduced subsets of 25% and 50% of the training dataset (left column). The testing and validation datasets remain consistent across all experiments.

range and selecting the best subset for experimentation refer 5 for the best set of hyperparameters for each model. Model optimization was achieved using the Adam [12] and AdamW [17] optimizers. Depending on the specific task, we employed either step-wise or cycle learning rate scheduling [28].

The training was conducted under optimal hyperparameter configurations, introducing variability through distinct random seeds and data splits. All experiments adhered to a fixed batch size of 20 and were executed on $1 \sim 8$ NVIDIA A6000 GPUs, with memory capacities of 48 GBs. To ensure fairness and accuracy in results, each experiment was replicated thrice with different seeds. We report the mean and deviation in Relative L2 Error.

### 5.1 Accuracy

Tab. 1 shows the performance of the models on the 8 datasets. FNO architecture stands out on all datasets, consistently delivering results among the best three. Its strength lies in its transformation in the frequency space. By capturing and transforming the lower frequencies present in the data, the FNO can approximate the solution operators of scientific PDEs. This approach, which uses the integral kernel in the Fourier space, facilitates a robust mapping between input and output function spaces, making it particularly adept at handling the complexities of the datasets in this study. Following FNO, GNOT, employing a mixture-of-experts approach, showcases exemplary performance on most (6/8) datasets. Its unique soft domain decomposition technique divides the problem into multiple scales, allowing it to capture diverse features of the underlying PDE. Each expert or head in the model focuses on a different aspect of the PDE, and their combined insights lead to a comprehensive understanding, especially for challenging datasets like SHEAR and BIAXIAL.

The OFORMER architecture, that employs an innovative approach to solving spatio-temporal PDEs, exhibits best results in NAVIER STOKES dataset. By unrolling in the time dimension and initiating with a reduced rollout ratio, it efficiently forwards the time step dynamics in the latent space. This method conserves significant space during training on time-dependent datasets. It also significantly enhances the approximation while tested on one of the most complex dataset in the study, the time-dependent navier stokes PDE.

Interestingly, most models, with the notable exception of GNOT, struggle to accurately learn the underlying PDE for the BIAXIAL and SHEAR datasets. Here, the simpler FNN architecture demonstrates significant proficiency in learning these datasets. Interestingly, architectures like CGAN, originally designed for 2D image data analysis with its U-Net encoder, demonstrate impressive performance across tasks. This underscores the versatility of such architectures, even when they aren't explicitly designed as operators.

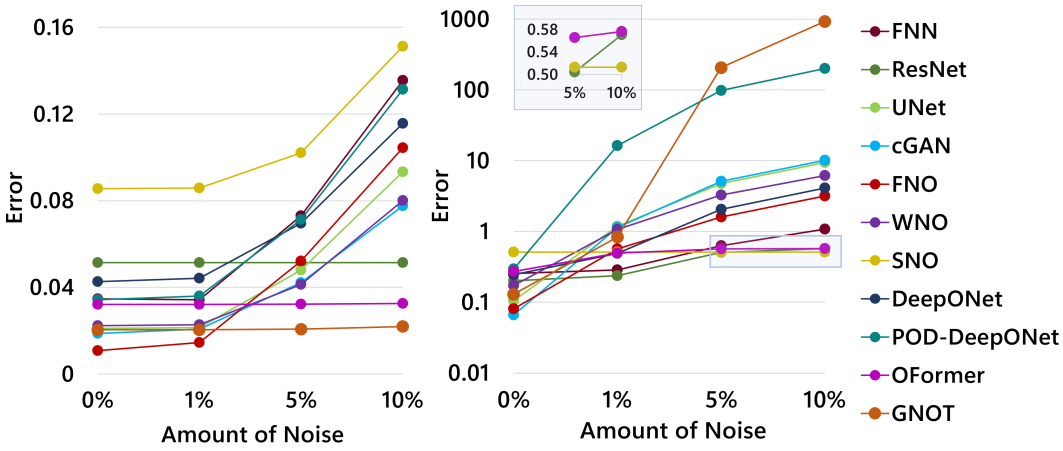


Figure 3: Robustness Analysis Against Noise: Performance metrics, in terms of Relative L2 Error, are presented for models subjected to random Gaussian noise. The evaluation encompasses the DARCY dataset (left) and the STRESS dataset (right). The right diagram provides a detailed comparison of the most noise-resilient models on the STRESS dataset, specifically SNO, OFORMER, and RESNET.

### 5.2 Data Efficiency

For the data-efficiency experiments, we utilized the DARCY dataset with 1700 samples of $47 \times 47$ dimensions. To assess the data efficiency of the models, we trained all models on reduced subsets: 25% (425 samples) and 50% (850 samples) of the original dataset, while maintaining the same testing and validation datasets.

The exceptional performance of frequency-based methods, notably FNO and WNO, even with limited data, is rooted in their operation within the frequency domain (see Table 2). The notable capability of these methods to capture the essential dynamics of the underlying PDEs through the lower frequencies present in the data enables data-efficient learning, a crucial feature for realistic data where the number of observations may be limited.

Transformer-based neural operator architectures have demonstrated potential in approximating operators. However, their efficacy diminishes when data is sparse. GNOT, which typically excels with a rich dataset, struggles to outperform even basic neural network architectures in a data-limited scenario. This trend underscores the inherent data dependency of transformer architectures, highlighting the challenges faced by many models, except frequency-based operators, when trained on limited data.

### 5.3 Robustness to Noise

In practical applications, it's common to encounter noise in measurements. To understand how various neural operators handle such real-world challenges, we simulated conditions with noisy data. During our testing phase, we intentionally introduced corrupted input function data to each model. The goal was to see how well these models could predict the ground truth amidst this noise.

Figure 3 shows the performance of the models on noisy data. Transformer-based architectures have shown commendable performance on the DARCY dataset. Even when noise is introduced, these models continue to perform well. However, their resilience is tested when faced with the STRESS dataset. In scenarios where they already find it challenging to learn the underlying PDEs, the addition of noise exacerbates their performance issues, causing a noticeable decline in accuracy.

On the other hand, the spectral neural operator SNO shows superior robustness to noisy data. While its performance in a noise-free environment is far from best, it performs remarkably when exposed to noisy dataset, especially on the STRESS dataset. This resilience can be attributed to its unique approach: unlike other frequency-based methods that transition between the time and frequency domains, SNO exclusively processes data in the frequency domain. This design choice allows it to effectively filter out noise, identifying it as a high-frequency disturbance, before it even begins its prediction process.

| Dataset | | Models | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Train | Test | FNN | ResNet | UNet | cGAN | FNO | WNO | SNO | DeepONet | POD-DeepONet | OFormer | GNOT |
| Stress | Stress | $25.69_{\pm 0.59}$ | $20.05_{\pm 0.19}$ | $10.57_{\pm 0.19}$ | $\mathbf{6.66_{\pm 0.84}}$ | $8.08_{\pm 0.15}$ | $17.24_{\pm 0.46}$ | $51.31_{\pm 0.01}$ | $24.59_{\pm 0.98}$ | $29.63_{\pm 0.52}$ | $27.33_{\pm 0.28}$ | $13.02_{\pm 0.81}$ |
| | Strain | $91.11_{\pm 0.04}$ | $89.83_{\pm 0.79}$ | $95.79_{\pm 3.40}$ | $95.34_{\pm 1.57}$ | $95.39_{\pm 0.89}$ | $93.71_{\pm 4.97}$ | $\mathbf{62.36_{\pm 0.46}}$ | $92.70_{\pm 3.30}$ | $596.33_{\pm 23.70}$ | $68.70_{\pm 0.76}$ | $94.35_{\pm 1.09}$ |
| Strain | Strain | $23.09_{\pm 7.08}$ | $14.64_{\pm 0.31}$ | $9.05_{\pm 0.33}$ | $6.12_{\pm 0.80}$ | $\mathbf{5.61_{\pm 0.23}}$ | $12.05_{\pm 0.26}$ | $62.34_{\pm 1.17}$ | $23.75_{\pm 0.20}$ | $18.31_{\pm 1.17}$ | $25.08_{\pm 1.36}$ | $9.99_{\pm 0.62}$ |
| | Stress | $75.63_{\pm 1.49}$ | $77.29_{\pm 0.72}$ | $77.41_{\pm 0.93}$ | $79.49_{\pm 1.07}$ | $79.50_{\pm 0.86}$ | $80.56_{\pm 1.27}$ | $\mathbf{51.65_{\pm 1.10}}$ | $77.49_{\pm 1.50}$ | $86.32_{\pm 2.24}$ | $80.26_{\pm 0.81}$ | $80.24_{\pm 0.95}$ |

Table 3: Out-of-Distribution Evaluation: Models are trained on the STRESS dataset and subsequently tested on both the STRESS dataset and the out-of-distribution STRAIN dataset. The experiment is reciprocated with STRAIN as the training set. Relative L2 Error ($\times 10^{-2}$) is reported, with top-performing results highlighted.

### 5.4 Zero-shot Super-resolution

Directly approximating the solution operator offers a theoretical advantage: the potential for a mesh invariant continuous dynamical system. Once trained, such a system can ideally maintain accuracy even when applied to much larger systems than those it was trained on. This capability is termed "zero-shot super-resolution". Note that FNO and GNOT enables zero shot super resolution without any modifications. Other models such as SNO and DEEPONET, upon closer examination, reveals that they cannot have a straightforward application on zero-shot super-resolution.

| Darcy | Models | |
|---|---|---|
| Resolution | FNO | GNOT |
| $\mathbf{47 \times 47}$ | $\mathbf{1.08_{\pm 0.06}}$ | $\mathbf{2.04_{\pm 0.05}}$ |
| $64 \times 64$ | $60.50_{\pm 5.49}$ | $55.32_{\pm 5.65}$ |
| $128 \times 128$ | $59.99_{\pm 5.48}$ | $55.42_{\pm 5.68}$ |

Table 4: Zero-shot super-resolution. Comparing on various resolutions (left column) with corresponding model performance (right column).

While certain modifications might enable super-resolution in their case, they diverge from the concept of zero-shot super-resolution from an architectural perspective. Accordingly, for our evaluation, we consider only FNO and GNOT. We trained both FNO and GNOT on the DARCY dataset at a resolution of $47 \times 47$. We then tested their performance on higher resolutions: $64 \times 64$ and $128 \times 128$. As seen in Table 4, both models exhibited a rapid decline in performance as the resolution increased. While GNOT fared slightly better, its results were still not up to the mark.

### 5.5 Out-of-distribution Generalization

The equations for STRESS and STRAIN are intrinsically linked, differing primarily by the coefficient of elasticity, commonly known as Young's modulus. Given that our training and testing processes utilize normalized data, it's reasonable to anticipate that the models trained on the STRESS dataset should be adept at predicting strain in the material microstructures, and vice versa. This expectation is particularly true for neural operators, which are designed to grasp the underlying partial differential equations (PDEs) governing such relationships. Table 3 shows the OOD evaluation on all the models. Interestingly, for SNO, the error on the strain test dataset remains consistent, whether it was trained on the strain dataset or the stress dataset. The same holds true when tested on the stress dataset. This consistency underscores SNO's ability to learn the underlying PDE. In stark contrast, other models don't exhibit this adaptability. Their accuracy levels decline when the testing set is swapped, indicating a potential limitation in their ability to generalize across closely related tasks.

### 5.6 Time Efficiency

Neural Operators are gaining traction over traditional numerical solvers due to their promise of rapid inference once trained. For this assessment, we've bench-marked various continuous dynamical systems on two criteria: the duration required to train on the DARCY dataset and the time needed to predict output function values for 200 test samples, each mapped on a uniform $47 \times 47$ grid. As anticipated, the FNN, with its straightforward architecture, stands out by requiring the least amount of time for both training and inference. However, when we delve into the other models, those based on deep operator regression methods, show training duration on par, with some of the complex but standard neural network architectures. For better visualization, see Figure 4. The narrative shifts when we consider inference time, a pivotal metric in practical applications. While FNO is relatively efficient during training, it, along with the transformer-based models, takes a longer inference stride. Though all these models show promising performance on different metrics, inference time efficiency remains a challenge for them. In stark contrast, most other models edge closer to offering real-time inference, highlighting the inherent time complexity trade-offs one must consider when opting for a particular neural operator.

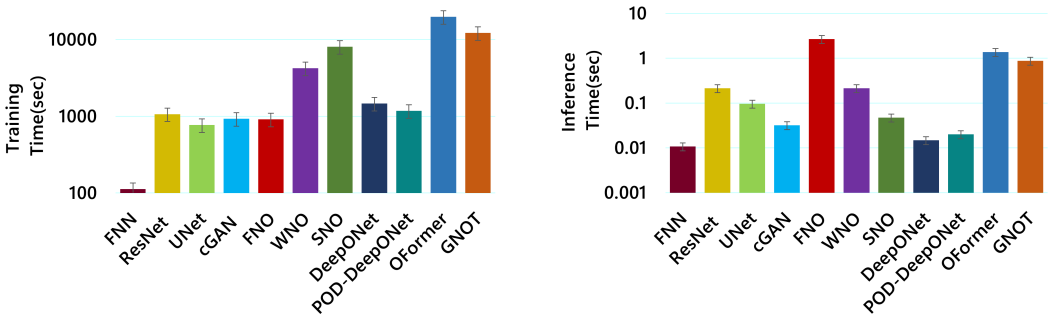


Figure 4: Time Efficiency: We report the time taken by each model during training (on left) and inference time on test set (on right). Results are collected during the training on DARCY dataset.

## 6 Concluding Insights

The key insights drawn from this work are as follows.

1. **Operator and transformer**: Both FNO and GNOT emerge as the superior models across various metrics, suggesting that the architectural novelty in these models can indeed capture the maps between infinite-dimensional functional spaces.
2. **Spectral resilience to noise and OOD**: Despite its underwhelming performance, SNO exhibits extreme resilience against noise. Similarly, SNO demonstrates impressive results on out-of-distribution dataset as well. The merit lies in its singular Fourier and inverse Fourier transformations, mapping input to output entirely in the frequency domain.
3. **Attention alone is not enough**: OFORMER, employing the attention-based transformer architecture, showcases notable advantages on the NAVIER STOKES dataset. It also demonstrates commendable results on specific PDEs like BURGERS, DARCY. However, a glaring limitation surfaces when these architectures are applied to other PDEs, whether of comparable complexity or even simpler ones. They fail to generalize. This shortcoming starkly contrasts with one of the primary advantages anticipated from data-driven PDE solvers: the capacity to discern the solution operator solely from data, independent of prior knowledge of the underlying PDE.
4. **Data-driven models work**: Surprisingly, the CGAN, a standard architecture for image tasks, excels in performance, even though it isn't inherently an operator. This prowess, however, wanes during cross-dataset evaluations, underscoring the importance of truly learning the underlying PDE rather than merely excelling on a given dataset.
5. **Challenges with SHEAR and BIAXIAL Datasets**: The collective struggle of most operators with the SHEAR and BIAXIAL datasets underscores the importance of studying complex deformation patterns. Specifically, it suggests clear and well-defined failure modes in operators, where future works can be focused.
6. **Time efficiency should be improved**: While the models give reasonable performance, they grapple with time efficiency. Especially, the best performing models such as transformer-based architectures are time-intensive both during training and inference, FNO is relatively swift in training, still intensive in inference.

**Limitations and future work:** Although FNO and GNOT exhibit superior results, their inconsistent results in cross-dataset evaluations and zero-shot super-resolution raise the questions whether they are truly learning approximate solution to the underlying PDE. Similarly, although resilient to noise and OOD, the internal neural network architecture SNO remains largely unexplored and often yields subpar outcomes. Future endeavors leveraging SNO might pave the way to operators with improved robustness. Failure modes of operators in datasets require further investigations to build more robust operators that can capture complex shear deformations. Finally, the inference time of the model requires improvement so that they can be applied to largescale real-world problems.

## References


[1] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

[2] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[3] Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

[4] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[5] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[6] Lokenath Debnath and Lokenath Debnath. *Nonlinear partial differential equations for scientists and engineers*. Springer, 2005.

[7] Vladimir Fanaskov and Ivan Oseledets. Spectral neural operators. *arXiv preprint arXiv:2205.10573*, 2022.

[8] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

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

[10] Ameya D Jagtap and George E Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. In *AAAI spring symposium: MLPS*, volume 10, 2021.

[11] S Jian, H Kaiming, R Shaoqing, and Z Xiangyu. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 770–778, 2016.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] 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.

[14] Emma Lejeune. Mechanical mnist: A benchmark dataset for mechanical metamodels. *Extreme Mechanics Letters*, 36:100659, 2020.

[15] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

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

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[18] 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.

[19] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em 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.

[20] Meer Mehran Rashid, Tanu Pittie, Souvik Chakraborty, and NM Anoop Krishnan. Learning the stress-strain fields in digital composites using fourier neural operator. *arXiv e-prints*, pages arXiv–2207, 2022.

[21] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[22] Shoichiro Nakamura. Computational methods in engineering and science, with applications to fluid dynamics and nuclear systems. 1977.

[23] Nicholas H Nelsen and Andrew M Stuart. The random feature model for input-output maps between banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021.

[24] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[25] D Robert. Partial differential equations and applications. *Sémin. Congr*, 15:181–250, 2007.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[27] Granville Sewell. *Analysis of a finite element method: PDE/PROTRAN*. Springer Science & Business Media, 2012.

[28] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[29] Pavel Ŝolín. *Partial differential equations and the finite element method*. John Wiley & Sons, 2005.

[30] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[31] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

[32] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

[33] Joseph R Tristano, Steven J Owen, and Scott A Canann. Advancing front surface mesh generation in parametric space using a riemannian surface definition. In *IMR*, pages 429–445, 1998.

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

# A Appendix

## A.1 Fluid Dynamics Datasets

### A.1.1 Burgers Equation

The 1D Burgers' equation is a non-linear PDE which is having the following form:

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial}{\partial x}\left(\frac{u^2(x,t)}{2}\right) = \nu \frac{\partial^2 u}{\partial x^2}(x,t), \quad x \in (0,1),\, t \in (0,1] \tag{2}$$

with periodic boundary conditions, where $u(x,0) = u_0(x)$, $x \in (0,1)$.

Here, $u_0 \in L^2_{\text{per}}((0,1);\mathbb{R})$ is the initial condition and $\nu \in \mathbb{R}^+$ is the viscosity coefficient.

### A.1.2 Darcy Flow Equation

We experiment with the steady-state solution of 2D Darcy Flow over the unit square, whose viscosity term $a(x)$ is an input of the system. The following equation defines the solution of the steady state.

$$\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in (0,1)^2, \tag{3}$$

$$u(x) = 0, \quad x \in \partial(0,1)^2. \tag{4}$$

For this paper, the force term $f$ is set to a constant value $\beta$, changing the scale of the solution $u(x)$. Instead of directly solving Equation 3, we get the solution by solving a temporal evolution equation:

$$\partial_t u(x,t) - \nabla \cdot (a(x)\nabla u(x,t)) = f(x), \quad x \in (0,1)^2, \tag{5}$$

The equation is solved with random initial conditions until it reaches a steady-state solution.

### A.1.3 Compressible NAVIER STOKES Equation

We experimented with the 2-D Navier-Stokes equation for a viscous, incompressible fluid in vorticity form on the unit torus having the following form:

$$\partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) = \nu \Delta w(x,t) + f(x), \quad x \in (0,1)^2,\, t \in (0,T] \tag{6}$$

$$\nabla \cdot u(x,t) = 0, \quad x \in (0,1)^2,\, t \in [0,T] \tag{7}$$

$$w(x,0) = w_0(x), \quad x \in (0,1)^2 \tag{8}$$

where $u \in C([0,T]; H^r_{\text{per}}((0,1)^2;\mathbb{R}^2))$ for any $r > 0$ is the velocity field, $w = \nabla \times u$ is the vorticity, $w_0 \in L^2_{\text{per}}((0,1)^2;\mathbb{R})$ is the initial vorticity, $\nu \in \mathbb{R}^+$ is the viscosity coefficient, and $f \in L^2_{\text{per}}((0,1)^2;\mathbb{R})$ is the forcing function.

### A.1.4 Incompressible NAVIER STOKES Equation

The incompressible Navier-Stokes equations represent a specialized form of the broader compressible fluid dynamics equations, tailored for scenarios involving subsonic flows. These equations are versatile and can be employed to analyze a wide range of systems, from hydromechanical processes to meteorological predictions and the exploration of turbulent behaviors.

### A.1.5 Shallow Water Equation

The shallow-water equations present a suitable framework for modeling free-surface flow problems. Its 2D hyperbolic PDEs with the following form:

$$\partial_t h + \nabla \cdot (hu) = 0, \tag{9}$$

$$\partial_t hu + \nabla \cdot \left(u^2 h + \frac{1}{2} g r h^2\right) = -g r h \nabla b, \tag{10}$$

where $u$ and $v$ are the velocities in the horizontal and vertical directions, $h$ denotes the water depth, and $b$ describes spatially varying bathymetry. $hu$ represent as the directional momentum components and $g$ represent the gravitational acceleration.

### A.2 Mechanical Datasets

#### A.2.1 Stress, Strains fields in 2D Digital in Composites

The constitutive relationship (generalized Hook's law) is defined as :

$$\{\sigma_{ij}\} = \mathbb{C}_{ijkl}\{\epsilon_{kl}\} \tag{11}$$

where $\sigma_{ij}$ and $\epsilon_{kl}$ are the stress and strain components, $\mathbb{C}_{ijkl}$ is the overall stiffness tensor. For 2D problems, we have $\sigma_{xx}$ , $\sigma_{yy}$ and $\sigma_{xy}$ satisfy the following equilibrium equations

$$\partial_x \sigma_{xx} + \partial_y \sigma_{xy} + F_x = 0; \partial_y \sigma_{yy} + \partial_x \sigma_{xy} + F_y = 0 \tag{12}$$

where $F_x$ and $F_y$ are the body forces in horizontal and vertical directions, respectively. Additionally, the strains $\epsilon_{xx}$, $\epsilon_{yy}$ and $\epsilon_{xy}$ are defined as

$$\epsilon_{xx} = \partial_x u_{xx}; \epsilon_{yy} = \partial_y u_{yy}; \epsilon_{xy} = \partial_y u_x + \partial_x u_x \tag{13}$$

where $u_x$ and $u_y$ are the displacements in horizontal and vertical directions, respectively
The 2D-Digital composites[20] are subjected to mode-I tensile loading with specific boundary conditions. The two-phase composite has a modulus ratio of 10 $(E_{stiff}/E_{soft})$, and both materials are assumed to be perfectly elastic. The simulations are run on an $8mm \times 8mm$ plate with negligible thickness. The two phases are randomly placed over the plate; however, the fraction of soft and stiff phases is equal.

#### A.2.2 Mechanical MNIST

The mechanical MNIST [14] has a collection of different tests such as shear, equibiaxial extension, confined compression, and other tests simulated on a heterogenous material subjected to large deformations. The famous MNIST digit dataset is generated by treating the bitmap images as heterogeneous material blocks modeled as a Neo-Hokean material of varying modulus. The material is subjected to fixed displacements in one direction, and the full-field displacements are recorded at each step. In this study, we select two tests from the mechanical MNIST experiments viz shear and equibiaxial.

### A.3 Result Generation Setting

| | | Time Steps | | Dataset Split | | | |
|---|---|---|---|---|---|---|---|
| Dataset | Grid Resolution | Input | Output | Total Samples | Training Data Size | Validation Data Size | Testing Data Size |
| BURGERS | (8192) | | | 2048 | 1700 | 148 | 200 |
| DARCY | $(47 \times 47)$ | | | 2000 | 1700 | 100 | 200 |
| NAVIER STOKES | $(47 \times 47)$ | 5 | 16 | 1200 | 900 | 100 | 200 |
| SHALLOW WATER | $(47 \times 47)$ | 10 | 40 | 1000 | 800 | 100 | 100 |
| STRESS | $(48 \times 48)$ | | | 1200 | 900 | 100 | 200 |
| STRAIN | $(48 \times 48)$ | | | 1200 | 900 | 100 | 200 |
| SHEAR | $(28 \times 28)$ | | | 70000 | 50000 | 10000 | 10000 |
| BIAXIAL | $(28 \times 28)$ | | | 70000 | 50000 | 10000 | 10000 |

Table 4: This table contains descriptions of datasets used with information about the data resolution, number of time steps, and dataset split.

### A.4 Hyperparameters For training Models

| Model | Optimizer | Scheduler | Learning Rate | Number of Parameters |
|---|---|---|---|---|
| FNN | Adam | StepLR | 0.0001 | 2330273 |
| RESNET | Adam | StepLR | 0.001 | 28060769 |
| UNET | Adam | StepLR | 0.0001 | 31031745 |
| CGAN | Adam | OneCycleLR | 0.0001 | 2240963 |
| FNO | Adam | StepLR | 0.001 | 1188353 |
| WNO | Adam | StepLR | 0.001 | 5337985 |
| SNO | AdamW | StepLR | 0.001 | 147321 |
| DEEPONET | AdamW | StepLR | 0.0001 | 36465408 |
| POD-DEEPONET | AdamW | StepLR | 0.0001 | 36413811 |
| OFORMER | Adam | OneCycleLR | 0.001 | 1476481 |
| GNOT | AdamW | OneCycleLR | 0.0001 | 6451201 |

Table 5: Best Hyperparameters Used for Training the Models On DARCY Dataset, Hyperparameters For Other Datasets Can Be Found Within The Codebase

.

### A.5 Training Details

All models undergo tuning of the learning rate through a grid search method, selecting rates from the set $\{10^{-4}, 10^{-3}, 10^{-2}\}$. In cases of unstable training trajectories across all rates in the initial set, additional experiments are conducted with rates from $\{10^{-5}, 10^{-1}\}$. Each experiment is executed once for every model-dataset pair. The learning rate yielding the best performance for a specific model on a given dataset is selected. Three experiments with different random seeds are then conducted using the optimal learning rate, and the mean and deviation of the results are reported. Table 6 displays the Relative L2 error achieved by the models when trained to convergence with three different learning rates. Note that most values lack a standard deviation as the initial learning rates are not optimal, and only one experiment is performed for each. However, the optimal learning rate includes both mean and standard deviation, derived from three experiments using distinct seeds. All experiments are executed until convergence, which is determined by two predefined criteria. The patience is set to 100 epochs, and $\epsilon$ is set to 1e-6. Convergence is considered achieved when the validation set error fails to decrease by at least the quantity $\epsilon$ from the 'best validation error yet' for the patience number of epochs. Once this criterion is met, the training process is halted, and the model is tested on the test dataset.

| | | Models | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | LR | FNN | RESNET | UNET | CGAN | FNO | WNO | SNO | DEEPONET | POD-DEEPONET | OFORMER | GNOT | |
| DARCY | 0.01 | 9.04 | 5.73 | 37.18 | 30.75 | 1.12 | 5.39 | 24.65 | 24.61 | 8.44 | 5.01 | 6.93 | 6.25 |
| | 0.001 | 3.87 | $5.14_{\pm 0.23}$ | 5.37 | 3.02 | $1.08_{\pm 0.06}$ | $2.23_{\pm 0.14}$ | $8.55_{\pm 1.03}$ | 4.89 | 3.54 | $3.21_{\pm 0.06}$ | 2.57 | 3.56 |
| | 0.0001 | $3.47_{\pm 0.14}$ | 5.33 | $2.10_{\pm 0.08}$ | $1.88_{\pm 0.04}$ | 1.60 | 2.87 | 25.47 | $4.27_{\pm 0.24}$ | $3.43_{\pm 0.04}$ | 7.82 | $2.04_{\pm 0.05}$ | $1.10_{\pm 0.11}$ |

Table 6: Hyper-parameter Tuning: Results of models on 3 different learning rates when tested on DARCY Flow dataset. Relative L2 Error ($\times 10^{-2}$) is reported.