

---

# Data-Efficient Operator Learning via Unsupervised Pretraining and In-Context Learning

---

**Wuyang Chen\***

Simon Fraser University

**Jialin Song\***

Simon Fraser University

**Pu Ren**

Lawrence Berkeley National Laboratory

**Shashank Subramanian**

Lawrence Berkeley National Laboratory

**Dmitriy Morozov**

Lawrence Berkeley National Laboratory

**Michael W. Mahoney**

International Computer Science Institute

Lawrence Berkeley National Laboratory

University of California, Berkeley

## Abstract

Recent years have witnessed the promise of coupling machine learning methods and physical domain-specific insights for solving scientific problems based on partial differential equations (PDEs). However, being data-intensive, these methods still require a large amount of PDE data. This reintroduces the need for expensive numerical PDE solutions, partially undermining the original goal of avoiding these expensive simulations. In this work, seeking data efficiency, we design unsupervised pretraining for PDE operator learning. To reduce the need for training data with heavy simulation costs, we mine unlabeled PDE data without simulated solutions, and we pretrain neural operators with physics-inspired reconstruction-based proxy tasks. To improve out-of-distribution performance, we further assist neural operators in flexibly leveraging a similarity-based method that learns in-context examples, without incurring extra training costs or designs. Extensive empirical evaluations on a diverse set of PDEs demonstrate that our method is highly data-efficient, more generalizable, and even outperforms conventional vision-pretrained models. We provide our code at [https://github.com/delta-lab-ai/data\\_efficient\\_nopt](https://github.com/delta-lab-ai/data_efficient_nopt).

## 1 Introduction

Recent advancements in machine learning methodology have shown promise in solving partial differential equations (PDEs) [24, 67, 38, 45, 44, 50, 39, 25]. A significant development in this area is the concept of operator learning for PDEs. This approach differs from traditional neural network methods, which are restricted to fixed-dimension input and output, since neural operators focus on learning mappings between function spaces [38, 45, 44]. Like other neural network methods, neural operators are recognized to be universal approximators for any continuous operator [46, 38], enabling them to approximate any physical operator, including solution operators for various parametric PDE families. A solution operator is defined as a function that maps physical inputs to output solutions. Previous work has shown that in simple settings, neural operators can effectively capture complex, multi-scale dynamic processes [45, 46, 74, 73].

However, neural operators tend to suffer from a problem common to other deep networks, namely the *need for enormous quantities of data*. Limited availability of data is common in science and

engineering. High-fidelity numerical simulations are computationally costly or even infeasible for many applications [71]. For example, an extreme-scale simulation of magnitude 7.0 earthquake at frequencies up to 10 Hz in San Francisco requires 3600 Summit GPU nodes and 42.7 hour [53].

Motivated by this data-efficiency challenge, recent works, particularly in natural language processing (NLP) and computer vision (CV), have focused on unsupervised (or self-supervised) pretraining<sup>2</sup> to reduce the cost of collecting or generating labeled data. Such pretrained models have been shown to be highly data-efficient in downstream fine-tuning [7, 29, 8], and they can even become few-shot learners without any downstream data [3]. In CV, researchers collect large amounts of natural images without any manual labels, and then pretrain visual encoders with proxy tasks, such as Noise-Contrastive Estimation (NCE) [59], masked reconstruction [28], rotation and jigsaw prediction [58, 20]. In NLP, people typically pretrain models via next-word prediction or masked tokens [3, 12].

However, unsupervised pretraining is still largely underexplored in Scientific Machine Learning (SciML). Therefore, our core question is: *How can we design unsupervised pretraining for operator learning to reduce the data simulation costs?*

In this work, we resort to unsupervised pretraining for neural operator learning to achieve data efficiency in SciML. We overview our framework in Figure 1. First, we define unlabeled data for PDEs, which avoids heavy computation costs for simulating PDE solutions. We propose two physics-inspired reconstruction-based proxy tasks, and we pretrain neural operators on unlabeled PDE data. We demonstrate that with unsupervised pretraining, our neural operators not only improve counterparts trained with more simulated data, but also they outperform off-the-shelf pretrained checkpoints from other popular domains (such as CV) that are ready for fine-tuning. Then, to further improve the data efficiency during out-of-distribution (OOD) inference, we design a similarity-based method that learns in-context examples [3, 49, 80, 81, 47]. This approach introduces zero overhead during training: one just maintains the standard training pipeline, and it can be seamlessly plugged in for OOD inference, without further fine-tuning. In more detail, we summarize our main contributions:

1. We introduce unlabeled PDE data and unsupervised pretraining for data-efficient neural operator learning. We show that our method can achieve better performance than models trained with more simulated PDE solutions, or fine-tuned from public checkpoints pretrained on other benchmarks, demonstrating the importance of unsupervised pretraining on domain-specific PDE data.
2. We propose a similarity-based method to improve the OOD generalization of neural operators, which is flexible and can scale up to a large number of unseen in-context examples (“demos”).
3. We provide detailed empirical evaluations on both diverse PDE benchmarks and also several real-world scenarios, demonstrating that we can achieve both strong forward modeling performance and significant savings in PDE simulations.

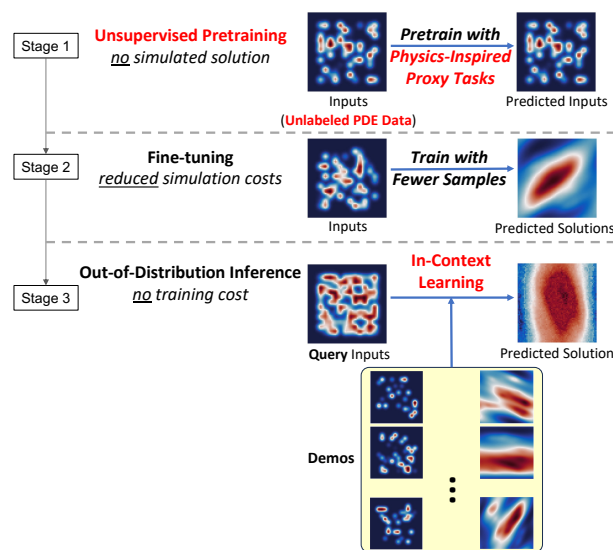


Figure 1: Overview of our framework for data-efficient neural operator learning (with our contributions highlighted in red). Stage 1: Unsupervised pretraining only on unlabeled PDE data. Stage 2: Fine-tuning with reduced simulation costs of PDE data. Stage 3: Test-time in-context examples can improve the neural operator’s out-of-distribution performance, without additional training costs.

<sup>2</sup>In this paper, we use the terms “unsupervised pretraining” and “self-supervised pretraining” interchangeably.

## 2 Related Works

### 2.1 Machine Learning for Scientific Modeling

There has been a long history of using learning-based methods to model physical scientific phenomena [41, 42, 6, 5]. A representative line of work is so-called physics-informed neural networks (PINNs) [67, 87, 19, 18, 69], which try to incorporate physics in neural networks by including the differential form of the PDE as an additional physics loss regularization term. However, this paradigm is confined to specific PDE scenarios (e.g., fixed PDE coefficients), instead of being more physics-agnostic. Moreover, recent work has highlighted several fundamental “issues” with PINN-based methods [40, 15]. On the other hand, operator learning methods, including Fourier Neural Operators [45, 44, 38] and Deep Operator Network [50], have achieved progress in approximating the solution operators of PDEs. Although these data-driven approaches show promise in learning PDE solutions, they (like many neural network-based methods) rely on vast quantities of high-fidelity labeled data. For operator learning, such data are usually computationally expensive to simulate [67, 2, 84]. More recently, people have tried to generate synthetic PDE solutions to train SciML models [27]. In contrast, our method works on unlabeled PDE data. This approach is distinct from the generation of synthetic PDE data, and it could be further combined as a semi-supervised learning strategy.

### 2.2 Unsupervised Pretraining and Foundation Models

Unsupervised (or self-supervised) pretraining is a key method in CV and NLP to achieve meaningful representations [7], data-efficient fine-tuning [31], and foundation models [1]. In CV, contrastive learning learns meaningful features by distinguishing between similar (positive) and different (negative) samples [59, 76, 7, 56, 29]. Masked Autoencoder (MAE) [28] uses a reconstructive approach where parts of the input are masked and the model learns to predict masked parts. In NLP, among the most prominent works are large language models (LLMs) such as GPT [3, 64, 65] and BERT [12, 66], which leverage token predictions for pretraining. Similar directions also show progress in SciML. For example, [2] and [54] propose to create augmented views in the solution space via Lie Symmetries; [73] study the scaling behavior of supervised pretraining and OOD generalization, charting directions for foundational models for SciML; [43] target learning astronomical foundation models with cross-modal contrastive learning; and [52] build large task-agnostic models with a broad understanding of common physical behavior to serve as foundation models for SciML.

### 2.3 In-Context Learning (ICL)

In-context learning (ICL) is a promising paradigm that helps deep networks generalize to unseen domains with a few in-context examples. Early works in CV seek to learn feature-level correspondence between the target and a few “shots,” such that models can generalize to open-set unseen objects [17, 83]. In NLP, people find LLMs are naturally few-shot learners [3], and thus tuning or optimizing prompts becomes extremely important to improve the in-context learning performance of LLMs [85, 72]. More recently, within SciML, a different operator learning strategy, termed “in-context operator learning,” has been proposed [80, 81, 47]. During both training and inference, the neural operator is asked to make predictions by explicitly leveraging a predefined number of so-called “demo” examples (pairs of physical parameters and simulated solutions). This approach provides a balance between model generalization and addressing data scarcity in the scenario of OOD testing.

## 3 Methods

In this section, we introduce our framework (outlined in Figure 1). We propose first to pretrain the model with unsupervised pretraining (Sec. 3.1), which will contribute to the data efficiency and reduced PDE simulation costs during standard training of neural operators. When we move to OOD scenarios during inference, we test our models with in-context examples (Sec. 3.2) to avoid further fine-tuning costs.

### 3.1 Unsupervised Pretraining

The core idea in unsupervised (or self-supervised) pretraining is to train a neural network with properly designed proxy tasks. These proxy tasks do not require labeled data, but they are designed

to be highly related to the supervised learning objectives of interest. While popular in CV and NLP, unsupervised pretraining on unlabeled data has never been explored in PDE operator learning in SciML. This is due to *two unresolved questions*: 1) What kinds of unlabeled data can we use to train neural operators? 2) How to design proxy tasks for PDE data? We address these questions below.

### 3.1.1 Unlabeled PDE Data

**How to Define Unlabeled PDE Data?** In general, when a neural operator is trained on PDE datasets [45, 74, 73], it learns to map inputs (physical parameters, coordinates, forcing functions, initial conditions, etc.) to PDE solutions. Therefore, given a set of PDE data (collected via simulations or observations), its unlabeled version is defined as the one without PDE solutions. Our unlabeled PDE data is a broader concept of related inputs in modeling PDE systems. Let us consider the second-order linear differential equation as a general example. It is formulated as

$$\sum_{i,j=1}^n a_{ij}(x)u_{x_i x_j} + \sum_{i=1}^n b_i(x)u_{x_i} + c(x)u = f(x),$$

where  $x \in \mathbb{R}^n$  represents physical space that varies in different systems (e.g.  $n = 3$  for 2D time-dependent PDEs); the coefficients (or physical parameters)  $a_{ij}, b_i, c$  are known from the physical process;  $u$  is the target solution; and  $f$  denotes an external forcing function [57]. We can consider two situations where solutions are unavailable:

- Time-independent equations: following [45, 73], our unlabeled PDE data include physical parameters ( $a_{ij}, b_i, c$ ), forcing functions ( $f$ ), and coordinates (grids of the discrete physical space).
- Time-dependent equations (e.g., forecasting problems [74, 52]): without simulating the temporal dynamics, our unlabeled PDE data only include initial snapshot  $u_0(x)$  that defines PDE systems. Note that collecting snapshots with temporal dynamics in large-scale scenes is more complex than capturing individual snapshots. For example, weather forecasting [30] and smoke dispersion [14] require continuous monitoring and multiple sensors, whereas single measurements are simpler and less resource-intensive. Long-term data collection often involves extensive networks and processing, unlike one-time measurements.

For concrete examples of different PDEs and unlabeled data that we will study, see Appendix A. There are two main reasons for pretraining only on unlabeled PDE data, as discussed below.

**Cheap Generation of Unlabeled PDE Data.** One critical reason that leads to expensive computational costs when collecting PDE data is the time marching scheme [23, 22] in numerical simulation. However, only generating unlabeled PDE data and snapshots without temporal dynamics will be much cheaper than simulating solutions (Table 4), making our unsupervised pretraining highly feasible in practice. Our pretraining strategy will be very data-efficient, and can avoid the heavy computational cost of simulating complex time-dependent equations for massive high-fidelity labeled solutions.

**Benefits of Pretraining on Unlabeled PDE Data.** Beyond the cheap generation, pretraining on unlabeled PDE data has the following benefits. First, *regularization against overfitting*. Unsupervised pretraining can strongly regularize the model towards better generalization. Second, *faster convergence*. Pretraining on unlabeled PDE data will provide neural operators with domain-adapted initializations and can accelerate training speed. Third, *meaningful representations*. Pretraining on unlabeled PDE data can help models extract useful representations for subsequent operator learning. We defer our results and experimental details in Figure 4 in Sec. 4.1.

### 3.1.2 Proxy Tasks

To illustrate our general approach of constructing proxy tasks, we choose two variants of reconstruction as our core proxy tasks. In particular, we will input unlabeled PDE data to our neural operators, and after a decoder network, we will force the output to be close to the input. We consider two perturbation variants (or augmented views). They are inspired by real-world settings when people collect scientific data, and they are important invariances we need to introduce to SciML models.

**Masked Autoencoder.** Masked autoencoders (MAEs) have been shown to be scalable self-supervised learners [28]. The method is conceptually simple: remove a portion of the input data, and

learn to predict the removed content. These methods enable training in NLP and CV of generalizable models containing over one hundred billion parameters [12, 3]. Here, we investigate the potential of MAE for scientific modeling.

**Motivation:** PDE dynamics are invariant to sparse sensing of the full field scientific data. It is very common [4, 16, 32] that scientific data need to be collected from sparse sensors, and that people need to reconstruct or generate data for domains without sensors. We enforce our model to learn sensor invariance via random masking, and we extract the invariant features over distorted views of the same unlabeled PDE data. The invariance to sparse sensing of scientific data will facilitate the robustness of the representations from MAE.

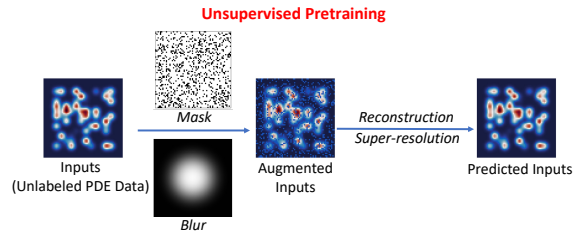


Figure 2: Overview: unsupervised pretraining via MAE and super-resolution. During pre-training, in the input unlabeled PDE data, a random subset (e.g., 70%) of spatial locations are masked, followed by a Gaussian blur. After the encoder and decoder, the full set of input is required to be reconstructed.

Therefore, we consider MAE as a proxy task. Specifically, our MAE is a straightforward autoencoding approach that reconstructs the original signal, given its partial observation. Like all autoencoders, our approach has an encoder that maps the observed signal to a latent representation, and a decoder that reconstructs the original signal from the latent representation. We randomly sample masks according to a certain masking ratio (i.e., the ratio of removed areas), and the values of masked areas of unlabeled PDE data are set to zero. Our loss function computes the mean squared error (MSE) between the reconstructed and original input. We compute the loss only on masked areas; if no mask is applied (i.e., a vanilla autoencoder), the MSE loss will be applied to all spatial areas.

**Super-resolution.** Super-resolution (SR) techniques have emerged as powerful tools for enhancing data resolution, improving the overall quality and fidelity of data representation, and retrieving fine-scale structures. SR is a task that involves recovering fine-scale data from corresponding coarse-grained data. It is also a popular task on PDE learning [9, 68], which is to train SciML models to preserve the inherent physical properties of scientific data.

**Motivation:** Numerical solutions of PDEs are expected to exhibit invariance to filtering blur or different resolutions of inputs [37, 79]. For instance, in turbulence simulations, the traditional numerical methods always fail to model the expected physical phenomenon with low-resolution meshes due to substantial numerical errors. SR has emerged as a powerful tool for subgrid modeling of PDE dynamics, especially helping to capture the critical patterns of turbulence [51, 37]. Given a specific input distribution, after fitting the SR objective, neural operators are expected to preserve the inherent physical properties and exhibit invariance to filtering blur.

Therefore, we introduce SR as another proxy task. Our objective shares the same motivation as recent SciML works for SR [68]. Specifically, we enforce the model to learn invariant features of unlabeled PDE data that are immune to resolution and blur. Blurry snapshots often occur when the resolution is too low to accurately represent the details in the original content. To do so, we apply a Gaussian filter to blur the unlabeled PDE data, and the autoencoder will reconstruct the high-resolution input with fine-grained details. Instead of applying a fixed blurring, we randomly sample the variance of the Gaussian filter from a certain range as augmentations.

### 3.1.3 PDEs

After pretraining on unlabeled PDE data, we fine-tune neural operators on simulated solutions of PDEs. We study two time-independent PDEs (Poisson, Helmholtz) and two time-dependent PDEs (Reaction-Diffusion, Navier-Stokes). We include details of these PDEs in Appendix A.

### 3.1.4 Model Architectures

We consider two popular architectures for fair comparisons with previous works. These are encoder-decoder architectures designed to reconstruct the original input given partial observations, where the encoder maps observed unlabeled PDE data to a latent space, and the decoder reconstructs the original conditions. We include visualizations of these architectures in Appendix D.

**Fourier Neural Operator.** Fourier Neural Operator (FNO) targets learning PDE data in the Fourier space. The original model backbone (encoder) employs Fourier transform and learns lower Fourier modes with linear transforms. The FNO backbone outputs features back to the spatial domain (i.e., the embeddings are on the pixel level). We refer readers to the original paper for details [45, 46].

- *Pretraining:* We build the decoder to be identical to the encoder (except for the input/output dimension). Unlabeled PDE data are randomly masked at the pixel level.
- *Fine-tuning:* After the pretraining, we discard the decoder, and we follow the original design to append two fully-connected layers (with ReLU activations) to predict final spatial-wise solutions.

**Transformer.** Transformers, which mainly employ self-attention and linear transform blocks, have shown promise in both NLP and CV [78, 13]. Different from FNO, which directly operates on grids, transformers tokenize and group grids into patches, i.e., each tokenized patch embeds a local neighborhood of subgrids. We follow the 3D transformer architecture of Video-MAE [77, 52]. Our encoder embeds patches by a linear projection with added positional embeddings (just as in a standard ViT), and it then processes the resulting set via a series of Transformer blocks. For the transformer, the unlabeled PDE data are randomly masked at the patch level.

- *Pretraining:* For the transformer encoder, we only apply it on the subset of tokens that are visible (i.e., unmasked patches), and masked patches are removed. This allows us to train encoders efficiently. The input to the MAE decoder is the full set of tokens consisting of (i) encoded visible patches, and (ii) the mask token. The mask token is a shared and learned vector that indicates the presence of a missing patch to be predicted. We add positional embeddings to all tokens in this full set. Without this, mask tokens would have no information about their location in the input. Following [28, 77], we adopt an asymmetric design where the decoder is more lightweight (shallower and narrower) than the encoder.
- *Fine-tuning:* After the pretraining, the decoder is preserved during fine-tuning, since we need to reconstruct the tokenized patches back to the input.

### 3.2 Similarity-based Mining of In-Context Examples

Out-of-distribution (OOD) generalization is a critical technical challenge, not only in SciML but also across multiple domains in AI for science [84]. To improve the OOD generalizability of neural operators and to reduce the extra effort of downstream fine-tuning, the following inference paradigm has been proposed: given a query input, the model is also provided with a few supporting examples (dubbed “demos”), together with their ground-truth solutions, to make the final prediction. This approach enables the “open-set” generalization of the model to make predictions on unseen samples.

Originally, in the literature on few-shot learning [82, 83, 55, 70, 35, 48, 61], people developed delicate architectures to find a correspondence between the input and the supporting examples. The purpose of this extra architecture/training design is twofold: first, *find similarities* between the target input and supporting examples; and second, *aggregate labels* of supporting examples for the final predictions. Recent ICL works [80, 81, 47] on learning PDE data also adopt this strategy, with transformers and cross-attention layers.

However, the ICL (in-context learning) of LLMs (large language models) enables a different strategy. The pretraining is still standard and simple (next/masked token prediction), without additional training costs. During inference, LLMs can auto-regressively take any number of few-shot examples, finding similarities between tokens in few-shot examples and those in the target query (via self-attention), and then generate responses by aggregating embeddings of tokens in few-shot examples. This ICL strategy used in LLM is highly scalable and training-efficient.

Motivated by this, we propose to leverage in-context examples via two steps (Algorithm 1).

**Similarity by Prediction.** We find spatial-wise and temporal-wise similar demos by calculating their distance in the output space. That means, for two input locations over the spatial and temporal domains, if we find their outputs of the trained neural operator similar, then we treat them as similar samples. Following [80, 81], we assume demos share the same distribution of physical parameters with the query.

**Aggregation.** For each spatial-temporal location of the query, after finding its similar samples in demos, we aggregate and average their solutions as the prediction.

---

**Algorithm 1:** Pseudocode of Similarity-based Mining of In-Context Examples.

---

- 1 **Data resolution:** x-axis ( $W$ ), y-axis ( $H$ ), output temporal steps ( $T$ ), output channel dimensions for the solution ( $C_{out}$ ).
  - 2 **Input:** Query input ( $x$ ). Paired unlabeled PDE data ( $X$ ) and solutions ( $Y \in \mathbb{R}^{J \times H \times W \times T \times C_{out}}$ ) as  $J$  demos. Trained Neural Operator Model  $\mathcal{M}$ . TopK ( $k$ ) demo solutions to aggregate.
  - 3  $\hat{y} = \mathcal{M}(x)$  ▷ Shape:  $H \times W \times T \times C_{out}$
  - 4  $\hat{Y} = \mathcal{M}(X)$  ▷ Shape:  $J \times H \times W \times T \times C_{out}$
  - 5  $\hat{\delta} = \hat{y}.\text{reshape}(-1, 1, C_{out}) - \hat{Y}.\text{reshape}(1, -1, C_{out})$  ▷ Query-Demo Distance. Shape:  $H \times W \times T \times (J \cdot H \cdot W \cdot T) \times C_{out}$
  - 6  $\hat{\delta} = \text{absolute}(\hat{\delta}).\text{sum}(-1)$
  - 7  $\text{index} = \text{argsort}(\hat{\delta}, -1)[:, :, :, : k]$  ▷ Spatial-wise and temporal-wise selection of demos similar to the query.
  - 8  $\hat{y}_{icl} = \text{take\_along\_dim}(\hat{Y}.\text{reshape}(-1, C_{out}), \text{index})$  ▷ Spatial-wise and temporal-wise aggregation of solutions from similar demos. Shape:  $H \times W \times T \times C_{out} \times k$ .
  - 9 **Return:**  $\hat{y}_{icl}.\text{mean}(-1)$
- 

## 4 Empirical Results

To illustrate the benefits of our approach, we perform empirical evaluations on both PDE benchmarks and real-world observations. *Most importantly*, our unsupervised pretraining (on unlabeled PDE data, followed by fine-tuning) outperforms neural operators trained from scratch, while requiring fewer PDE simulations (Sec. 4.1 and Sec. 4.2). *Moreover*, our in-context examples can help the model generalize better to OOD cases (Sec. 4.3). In our experiments, we trained models three times with different random seeds for statistical significance. For more experimental details, see Appendix B. We also include ablation studies about pretraining hyperparameters in Appendix J. For visualizations of our pretraining, see Appendix K.

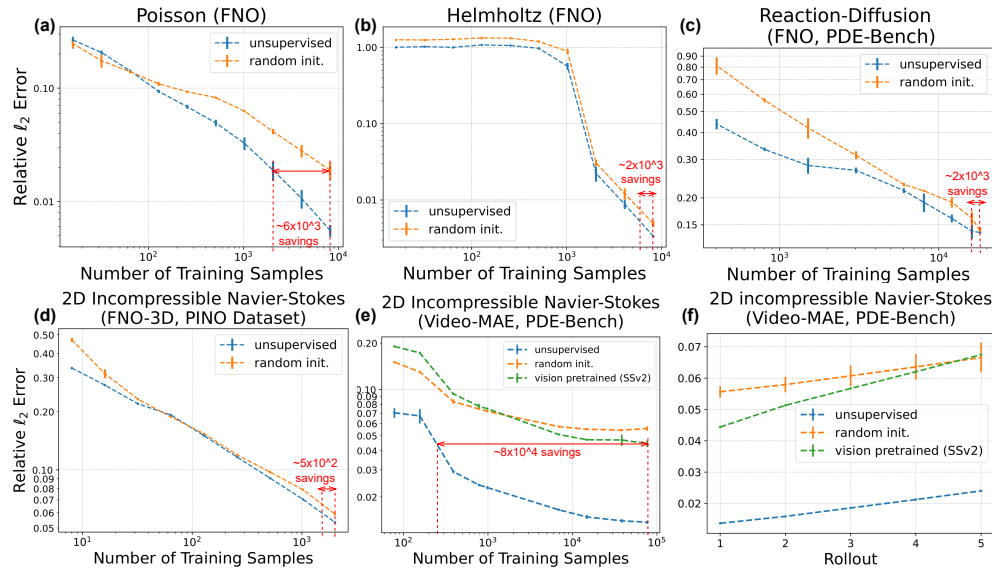


Figure 3: Pretraining neural operators on unlabeled PDE data improves its performance and data efficiency on Poisson (a), Helmholtz (b), Reaction-Diffusion (c), and Navier-Stokes (d and e, with relative errors at different unrolled steps shown on f). “random init.”: models are trained from scratch with random initialization. “vision pretrained (SSv2)”: fine-tuning from the publicly available checkpoint for Video-MAE (pretrained on computer vision dataset SSV2 [21] for video understanding). Savings of the number of simulated PDE data (when “random init.” achieves the best test error) are shown in red.

#### 4.1 Unsupervised Pretraining Enables Data-Efficient Operator Learning

**Data Efficiency.** We first demonstrate that, by leveraging unsupervised pretraining, neural operators can achieve improved errors with less simulated data. In Figure 3, on each PDE, compared with directly training from scratch (“random init.”), pretraining on unlabeled PDE data can help neural operators achieve better performance, which can further help reduce the amount of simulated data. Specifically, in our experiments, when we target achieving the best test error of the baseline (“random init.”), our method can save  $5 \times 10^2 \sim 8 \times 10^5$  simulated solutions across diverse PDE systems.

Among all PDEs, we find that Helmholtz (Figure 3 (b)) is the most challenging. Both two curves failed to improve the error until we increased the number of simulated data points to over 1024. Meanwhile, the generalization gaps remain high (Figure 12 (b)), indicating low training errors. We suspect that learning on the Helmholtz equation may be exhibiting the grokking issue [63, 86], where the network quickly memorizes the training data, but the improvement in generalizability is delayed.

**Unsupervised Pretraining Outperforms Off-the-Shelf Pretrained Checkpoints.** Pretraining on unlabeled PDE data is not the only way to save simulation costs. As pretraining is widely adopted in CV, pretrained checkpoints on vision data become publicly available and are ready for fine-tuning. We choose to compare with Video-MAE [77] for state-of-the-art video understanding pretrained on SSV2 [21]. As shown in Figure 3 (e), vision-pretrained Video-MAE can only outperform the random initialization with high volumes of simulated data, while its performance suffers when fine-tuned with limited simulations. In contrast, our unsupervised pretraining on unlabeled PDE data can save a significant amount of simulated data.

As errors during testing may quickly accumulate with further timestamps, we also report results with more unrolled steps. We use checkpoints trained with the largest amount of simulated data from above for this study. As shown in Figure 3 (f), at each rollout step, our unsupervised pretraining achieves much better performance. Similarly, vision-pretrained Video-MAE is eventually outperformed by the random initialization at the long rollout step.

**Benefits of Pretraining on Unlabeled PDE Data.** Pretraining on unlabeled PDE data is beneficial beyond achieving better performance with reduced simulations. *First*, when training on extremely low volumes of data, neural operators tend to overfit, resulting in poor generalization. In Figure 4-left, pretraining on unlabeled PDE data can reduce the generalization gap (testing error – training error). We further show that better generalization gaps persist across all PDEs we studied (see Figure 12). *Second*, pretraining on unlabeled PDE data can lead to faster convergence during fine-tuning. Unlike standard random initializations from Gaussian distributions, pretraining on unlabeled PDE data will provide neural operators with domain-adapted initializations and facilitate a much faster convergence rate, as shown in Figure 4-middle<sup>3</sup>. *Third*, unsupervised pretraining can also help models extract useful representations for subsequent operator learning. From Figure 4-right, we find that even with pre-extracted features (i.e., fixed encoder and only fine-tuned decoder), our neural operators can still outperform the baselines where both the encoder and decoder are updated during fine-tuning.

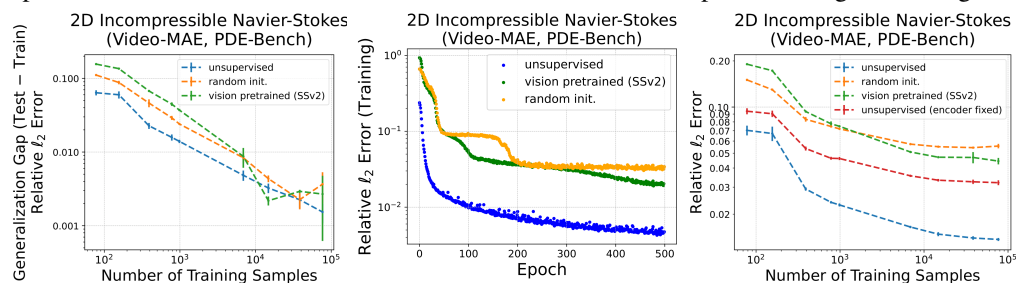


Figure 4: Benefits of our unsupervised pretraining. **Reduced overfitting** (left): our method consistently leads to smaller generalization gaps (test error – training error) across all PDEs we studied (Fig. 12). **Faster convergence** (middle): our unsupervised pretraining can accelerate model convergence than both random initialization and vision-pretrained checkpoint. **Meaningful representations** (right): fine-tuning Video-MAE with fixed encoder (pretrained on unlabeled PDE data, red line) can extract meaningful features and outperform the baseline and the vision-pretrained model (both encoder and decoder are updated during fine-tuning).

<sup>3</sup>Training curves when the number of training samples of 14760.

## 4.2 More Comprehensive Experiments on Real-World Data

We now move to a broader range of benchmarks. We will study real-world and noisy data instead of toy datasets, providing even more comprehensive experiments. These benchmarks are widely studied in previous works [60, 62, 52].

**Datasets.** We brief the background, with more details in Appendix L and visualizations in Fig. 15.

- **ECMWF Reanalysis v5 (ERA5)** [30] is a public extensive dataset, which delivers hourly data on multiple atmospheric variables spanning from 1979 to today. ERA5 represents a type of atmospheric reanalysis dataset [34], integrating observations from a variety of measurement sources with numerical models through data assimilation [33]. Essentially, it reconstructs the most accurate estimation of the Earth’s atmospheric conditions over time. This dataset has been extensively utilized in prior SciML studies [60, 68]. We focus on forecasting the important and challenging *temperature* atmospheric variable.
- **ScalarFlow** [14] is a reconstruction of real-world smoke plumes. It assembles the first large-scale dataset of realistic turbulent flows. The availability of a large, volumetric data set opens up a wide range of applications, including re-simulations, novel visualization, and metric evaluations. The dataset contains 104 real-world smoke flow density reconstructions. Each reconstruction is captured from five different viewpoints for 150 temporal frames spanning 2.5 seconds.
- **Airfoil** [75] is a large-scale dataset that contains the pressure and velocity simulations of the flow around real airfoils. This dataset has 53880 samples for training and 90 samples for testing. Each sample contains 6 channels. The 3 input channels include the binary spatial mask of the airfoil and the initial velocity of the freestream condition in the x and y directions. The output channels include the pressure and the x and y velocity components from the simulation at the steady state.

**Model and Training.** For time-dependent ERA5 (Sec. L.1) and ScalarFlow (Sec. L.2) datasets, we adopt the same VideoMAE architecture [77] used in Sec. 4.1. We use 15 consecutive temporal snapshots to forecast the next time step. We train VideoMAE with Adam, with other hyperparameters the same as in Table 3 column “N.S. (PDEBench)”. For the time-independent steady-state Airfoil dataset (Sec. L.3), we adopt the 2D-FNO architecture. We train 2D-FNO with Adam, with other hyperparameters the same as in Table 3 column “Poisson”.

**Results.** As shown in Figure 5, compared with directly training operators from scratch (“random init.”), pretraining on unlabeled data (2D snapshots of ERA5/ScalarFlow without temporal dynamics, or freestream velocities of Airfoil) can help neural operators achieve better performance on both temperature/flow forecasting and predictions of the steady-state pressure and velocity around airfoils.

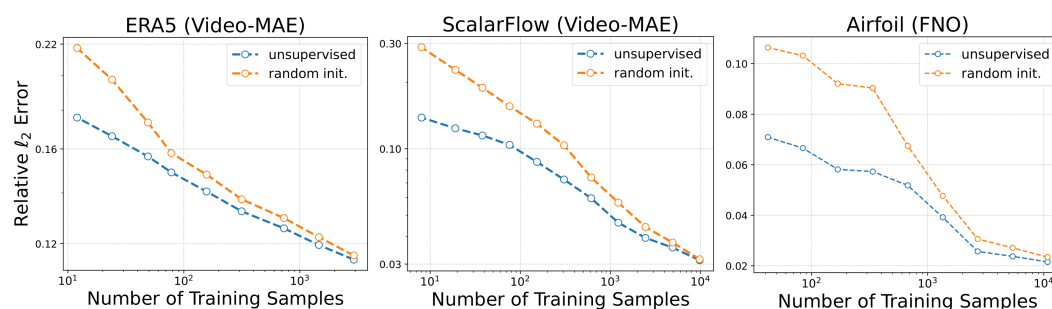


Figure 5: For **real-world scientific problems**, pretraining neural operators on unlabeled PDE data improves its performance and data efficiency. We study VideoMAE [77] pretrained with unlabeled snapshots (no temporal dynamics), and then fine-tune across different numbers of temporal snapshots on ERA5 (left) and ScalarFlow (middle). We also pretrain 2D-FNO [45] on freestream velocities and fine-tune on time-independent steady-state airflow pressure and velocities (right). “random init.”: models are trained from scratch with random initialization.

## 4.3 In-Context Examples Enable Data-Efficient OOD Generalization

We now move to OOD settings, where models will be tested on PDE data simulated with physical parameters unseen during fine-tuning/training. We include how to simulate OOD samples in

Appendix B.2. Neural operators suffer from poor OOD generalization [73, 80]. Traditionally, improvements heavily depend on further fine-tuning on simulated data, which requires extra simulation and training costs. We study the benefits of leveraging test-time in-context examples. As shown in Figure 6, when we flexibly scale up the number of demos, we can keep improving FNO’s OOD generalization on diverse PDEs. We follow [80, 81] that demos are randomly sampled from the same distribution used to generate the OOD test set. When the number of demos is 0, we have the baseline in the OOD setting. Notably, we introduce zero training overhead: we keep the standard training pipeline, and our mining of in-context examples can be seamlessly plugged in during OOD inference.

We further provide a baseline, which uses features extracted by the backbone of the neural operator (high-dimensional features before the final output layer) to find similar samples. As we can see, this baseline is worse than our method (both performance and confidence), indicating that the final output of the neural operator can more accurately indicate true similar samples.

See Appendix M for further discussions about the benefits of leveraging in-context examples, and Appendix N for visualizations.

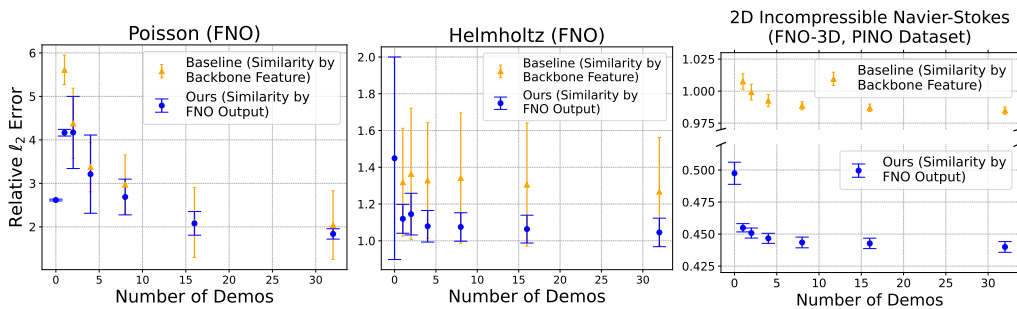


Figure 6: In-context examples for OOD testing. **Our method (blue) improves both  $\ell_2$  errors and confidence as we increase the number of demos.** “Ours (Similarity by FNO Output)”: we leverage the output (prediction) of neural operators to find similar samples. “Baseline (Similarity by Backbone Feature)”: the baseline uses features extracted by the backbone of the neural operator (high-dimensional features before the final output layer) to find similar samples.

## 5 Conclusion

In this work, we focus on improving the data efficiency of solving partial differential equations (PDEs) using deep learning, with a particular emphasis on unsupervised pretraining and in-context learning (ICL) methods. Our key contributions include introducing unsupervised pretraining for operator learning and a flexible ICL approach that enhances out-of-distribution (OOD) generalization without increasing training costs. Through extensive evaluations, we demonstrate that our method is not only more data-efficient, but it also achieves greater generalizability compared to existing approaches. By improving the data efficiency of neural operators for solving PDEs, our approach can significantly reduce the computational costs and energy demands of high-fidelity numerical PDE simulations. Additionally, by making advanced PDE solutions more accessible through efficient pretraining, our method has the potential to accelerate scientific and engineering progress across various fields, ultimately benefiting society. We hope our work will inspire the scientific machine learning (SciML) community to further address the high simulation costs and limited OOD generalization of neural operators, contributing to advancements that support both scientific innovation and environmental sustainability.

## 6 Limitations

Current limitations of our work: 1) We could design more physics-inspired proxy tasks and data augmentation methods for scientific data; 2) We could study more PDE systems in our unsupervised pretraining and in-context learning; 3) We could consider more different neural operator architectures. We expect that addressing these limitations will lead to broader impacts in future works.

## References

- [1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
- [2] Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, pages 2241–2256. PMLR, 2022.
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [4] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.
- [5] Tianping Chen and Hong Chen. Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks. *IEEE Transactions on Neural Networks*, 6(4):904–910, 1995.
- [6] 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.
- [7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [8] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.
- [9] Wai Tong Chung, Bassem Akoush, Pushan Sharma, Alex Tamkin, Ki Sung Jung, Jacqueline Chen, Jack Guo, Davy Brouzet, Mohsen Talei, Bruno Savard, et al. Turbulence in focus: Benchmarking scaling behavior of 3d volumetric super-resolution with blastnet 2.0 data. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.
- [10] Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. *arXiv preprint arXiv:2301.07733*, 2023.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [14] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.
- [15] C. Edwards. Neural networks learn to speed up simulations. *Communications of the ACM*, 65(5):27–29, 2022.
- [16] N Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L Brunton, Michael W Mahoney, and J Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238):20200097, 2020.

- [17] Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.
- [18] Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.
- [19] Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.
- [20] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.
- [21] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.
- [22] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations (Springer Series in Computational Mathematics)*, volume 31. Springer, 2006.
- [23] E Hairer, S P Nørsett, and G Wanner. *Solving ordinary differential equations I. nonstiff problems*, volume 29. Springer, 1987.
- [24] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.
- [25] D. Hansen, D. C. Maddix, S. Alizadeh, G. Gupta, and M. W. Mahoney. Learning physical models that can respect conservation laws. *Physica D: Nonlinear Phenomena*, 457:133952, 2024.
- [26] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*, 2024.
- [27] Erisa Hasani and Rachel A Ward. Generating synthetic data for neural operators. *arXiv preprint arXiv:2401.02398*, 2024.
- [28] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.
- [29] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.
- [30] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.
- [31] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.
- [32] Balaji Jayaraman and SM Abdullah Al Mamun. On data-driven sparse sensing and linear estimation of fluid flows. *Sensors*, 20(13):3752, 2020.
- [33] Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

- [34] Eugenia Kalnay, Masao Kanamitsu, Robert Kistler, William Collins, Dennis Deaven, Lev Gandin, Mark Iredell, Suranjana Saha, Glenn White, John Woollen, et al. The ncep/ncar 40-year reanalysis project. In *Renewable energy*, pages Vol1\_146–Vol1\_194. Routledge, 2018.
- [35] Dahyun Kang and Minsu Cho. Integrative few-shot learning for classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9979–9990, 2022.
- [36] Gene A Klaasen and William 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.
- [37] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.
- [38] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.
- [39] A. S. Krishnapriyan, A. F. Queiruga, N. B. Erichson, and M. W. Mahoney. Learning continuous models for continuous physics. *Communications Physics*, 6:319, 2023.
- [40] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.
- [41] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.
- [42] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.
- [43] Francois Lanusse, Liam Parker, Siavash Golkar, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Geraud Krawezik, Michael McCabe, Ruben Ohana, Mariel Pettee, et al. Astroclip: Cross-modal pre-training for astronomical foundation models. *arXiv preprint arXiv:2310.03024*, 2023.
- [44] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.
- [45] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.
- [46] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.
- [47] Jerry Weihong Liu, N Benjamin Erichson, Kush Bhatia, Michael W Mahoney, and Christopher Re. Does in-context operator learning generalize to domain-shifted settings? In *The Symbiosis of Deep Learning and Differential Equations III*, 2023.
- [48] Jie Liu, Yanqi Bao, Guo-Sen Xie, Huan Xiong, Jan-Jakob Sonke, and Efstratios Gavves. Dynamic prototype convolution network for few-shot semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11553–11562, 2022.

- [49] Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*, 2021.
- [50] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.
- [51] Romit Maulik, Omer San, Adil Rasheed, and Prakash Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019.
- [52] Michael McCabe, Bruno Régalo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for physical surrogate models. *arXiv preprint arXiv:2310.02994*, 2023.
- [53] David McCallen, Anders Petersson, Arthur Rodgers, Arben Pitarka, Mamun Miah, Floriana Petrone, Bjorn Sjogreen, Norman Abrahamson, and Houjun Tang. Eqsim—a multidisciplinary framework for fault-to-structure earthquake simulations on exascale computers part i: Computational models and workflow. *Earthquake Spectra*, 37(2):707–735, 2021.
- [54] Grégoire Mialon, Quentin Garrido, Hannah Lawrence, Danyal Rehman, Yann LeCun, and Bobak T Kiani. Self-supervised learning with lie symmetries for partial differential equations. *arXiv preprint arXiv:2307.05432*, 2023.
- [55] Juhong Min, Dahyun Kang, and Minsu Cho. Hypercorrelation squeeze for few-shot segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6941–6952, 2021.
- [56] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6707–6717, 2020.
- [57] AK Nandakumaran and PS Datti. *Partial differential equations: classical theory with a modern touch*. Cambridge University Press, 2020.
- [58] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.
- [59] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [60] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.
- [61] Bohao Peng, Zhuotao Tian, Xiaoyang Wu, Chengyao Wang, Shu Liu, Jingyong Su, and Jiaya Jia. Hierarchical dense correlation distillation for few-shot segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23641–23651, 2023.
- [62] Michael Poli, Stefano Massaroli, Federico Berto, Jinkyoo Park, Tri Dao, Christopher Ré, and Stefano Ermon. Transform once: Efficient operator learning in frequency domain. *Advances in Neural Information Processing Systems*, 35:7947–7959, 2022.
- [63] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.
- [64] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI*, 2018.

- [65] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [66] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- [67] 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.
- [68] Pu Ren, N Benjamin Erichson, Shashank Subramanian, Omer San, Zarija Lukic, and Michael W Mahoney. Superbench: A super-resolution benchmark dataset for scientific machine learning. *arXiv preprint arXiv:2306.14070*, 2023.
- [69] Pu Ren, Chengping Rao, Yang Liu, Jian-Xun Wang, and Hao Sun. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *Computer Methods in Applied Mechanics and Engineering*, 389:114399, 2022.
- [70] Xinyu Shi, Dong Wei, Yu Zhang, Donghuan Lu, Munan Ning, Jiashun Chen, Kai Ma, and Yefeng Zheng. Dense cross-query-and-support attention weighted mask aggregation for few-shot segmentation. In *European Conference on Computer Vision*, pages 151–168. Springer, 2022.
- [71] Justin Sirignano, Jonathan F MacArt, and Jonathan B Freund. Dpm: A deep learning pde augmentation method with application to large-eddy simulation. *Journal of Computational Physics*, 423:109811, 2020.
- [72] Alessandro Sordoni, Xingdi Yuan, Marc-Alexandre Côté, Matheus Pereira, Adam Trischler, Ziang Xiao, Arian Hosseini, Friederike Niedtner, and Nicolas Le Roux. Deep language networks: Joint prompt training of stacked llms using variational inference. *arXiv preprint arXiv:2306.12509*, 2023.
- [73] Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *arXiv preprint arXiv:2306.00258*, 2023.
- [74] 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.
- [75] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.
- [76] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.
- [77] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.
- [78] 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.
- [79] Ricardo Vinuesa and Steven L Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.
- [80] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

- [81] Liu Yang, Tingwei Meng, Siting Liu, and Stanley J Osher. Prompting in-context operator learning with sensor data, equations, and natural language. *arXiv preprint arXiv:2308.05061*, 2023.
- [82] Xiaolin Zhang, Yunchao Wei, Guoliang Kang, Yi Yang, and Thomas Huang. Self-produced guidance for weakly-supervised object localization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 597–613, 2018.
- [83] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas S Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. *IEEE transactions on cybernetics*, 50(9):3855–3865, 2020.
- [84] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*, 2023.
- [85] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.
- [86] Xuekai Zhu, Yao Fu, Bowen Zhou, and Zhouhan Lin. Critical data size of language models from a grokking perspective. *arXiv preprint arXiv:2401.10463*, 2024.
- [87] Yin hao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

## A PDEs

We consider two time-independent PDEs (Poisson, Helmholtz) and two time-dependent PDEs (Reaction-Diffusion, Navier-Stokes), as described below. We also describe types of unlabeled data for per-PDE examples. For a general definition of unlabeled PDE data, please refer to Section 3.1.1.

1. *Poisson*: We consider a two-dimensional (2D) elliptic PDE that arises in various physical situations, with periodic boundary conditions within a spatial domain  $\Omega = [0, 1]^2$ :

$$-\operatorname{div} \mathbf{K} \nabla u = f, \quad (1)$$

where  $u(x)$  represents the solution,  $f(x)$  acts as the source (e.g., forcing) function, and  $x$  denotes spatial coordinate. The diffusion coefficient tensor, denoted by  $\mathbf{K}$ , is employed to measure the physical properties of this system.

**Unlabeled PDE Data:** In [73], inputs to neural operators have four channels: the source function, and three diffusion coefficients (two diagonal elements and one off-diagonal element in the symmetric diffusion coefficient tensor) expanded to the whole spatial domain. We use this input as the unlabeled data to pretrain the neural operator.

2. *Helmholtz*: We consider the 2D inhomogeneous Helmholtz equation, which is a time-independent form of the wave equation, with periodic boundary conditions and a spatial domain  $\Omega = [0, 1]^2$ . The governing equation is given by

$$-\Delta u + \omega u = f \quad \text{in } \Omega, \quad (2)$$

where  $u(x)$  denotes the solution,  $f(x)$  represents the source function, and  $\omega > 0$  is the wavenumber (that defines the dynamic properties of the Helmholtz equation). This system produces high-frequency large wavenumber oscillatory patterns, which poses challenges in terms of generalization.

**Unlabeled PDE Data:** In [73], inputs to neural operators have two channels: the source function, and the wavenumber  $\omega$  expanded to the whole spatial domain. We use this input as the unlabeled data to pretrain the neural operator.

3. *Reaction-Diffusion*: The 2D Reaction-Diffusion (RD) equation involves the interaction between two nonlinear coupled variables, i.e., the activator  $u(t, x, y)$  and the inhibitor  $v(t, x, y)$ . The RD equation is given by

$$\begin{aligned} \partial_t u &= D_u(\partial_{xx} u + \partial_{yy} u) + R_u, \\ \partial_t v &= D_v(\partial_{xx} v + \partial_{yy} v) + R_v, \end{aligned} \quad (3)$$

where  $\{D_u, D_v\}$  are diffusion coefficients for  $u$  and  $v$ , respectively, and where  $R_u$  and  $R_v$  are reaction functions, which are defined as the Fitzhugh-Nagumo equation [36]:

$$\begin{aligned} R_u &= u - u^3 - k - v, \\ R_v &= u - v. \end{aligned} \quad (4)$$

The spatial domain considered for the 2D RD equation is  $\Omega = [-1, 1]^2$ , and the time duration is  $t \in (0, 5]$ .

**Unlabeled PDE Data:** In PDEBench [74], we forecast the activator  $u$  and the inhibitor  $v$  (i.e. two input channels) with  $T = 10$ . Since each individual snapshot can serve as an initial condition for forecasting, during our unsupervised pretraining we discard the temporal dynamics and use randomly shuffled snapshots of  $u$  and  $v$  with  $T = 1$  as unlabeled PDE data. That means, during pretraining the model only has access to single and individual snapshots without any awareness of temporal dynamics.

4. *Navier-Stokes Equation*: Lastly, we consider the 2D incompressible Navier-Stokes equation in vorticity form on the unit torus, which is formulated as

$$\begin{aligned} \nabla \cdot \mathbf{v} &= 0 \\ \rho (\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) &= -\nabla p + \nu \nabla^2 \mathbf{v} + \mathbf{f} \end{aligned} \quad (5)$$

Here, the velocity  $\mathbf{v}$  is defined within the time duration  $[0, T]$  and the spatial domain  $[0, 1]^2$  (the vorticity can be formulated as  $w = \nabla \times \mathbf{v}$ ). Moreover,  $\rho$  is the density, and the coefficient

$\nu$  signifies the viscosity of a fluid, and  $\mathbf{f}$  is the external forcing function. Dirichlet boundary conditions are employed in this system.

**Unlabeled PDE Data:** We follow the training settings from previous works. 1) For FNO [45], we focus on mapping the vorticity from the initial condition to the full solution ( $w_0 \mapsto w|_{[0,T]}$ , with  $T = 33$ ). The input includes four channels: the vorticity (which is the initial condition, duplicated for  $T$  times), and three (xy-spatial and temporal) linear mesh position embedding channels. Thus, similar to Reaction-Diffusion above, during pretraining, the model will take individual snapshots of vorticity and position embeddings as unlabeled data. 2) For PDEBench [74], we forecast both xy velocities and pressure (i.e. three input channels), with  $T = 15$ . The model will take individual snapshots of vorticity and xy velocities as unlabeled data.

We summarize detailed inputs and outputs in Table 1.

Table 1: Inputs and outputs for learning different PDEs. See Table 3 for resolutions. “NS”: Navier-Stokes. “RD”: Reaction-Diffusion.

| PDE Simulations    | Input                                                            | Input Shape                                    | Output                                               |
|--------------------|------------------------------------------------------------------|------------------------------------------------|------------------------------------------------------|
| Poisson [73]       | Source function ( $f$ ), diffusion coefficients ( $\mathbf{K}$ ) | $C \times H \times W (C = 4)$                  | Potential field ( $u$ )                              |
| Helmholtz [73]     | Source function ( $f$ ), wavenumber $\omega$                     | $C \times H \times W (C = 2)$                  | Wave function ( $u$ )                                |
| NS (FNO [45])      | Vorticity ( $w$ ), Spatiotemporal Coordinates                    | $H \times W \times T \times 4 (T = 33)$        | Vorticity ( $w \in [0, T]$ )                         |
| NS (PDEBench [74]) | Velocity ( $v_x, v_y$ ), pressure ( $p$ )                        | $T \times C \times H \times W (T = 15, C = 3)$ | Velocity ( $v_x, v_y$ ), pressure ( $p$ ) at $T + 1$ |
| RD (PDEBench [74]) | Activator ( $u$ ), inhibitor ( $v$ )                             | $T \times C \times H \times W (T = 10, C = 2)$ | Activator ( $u$ ), inhibitor ( $v$ ) at $T + 1$      |

## B Detailed Experiment Settings

### B.1 Distributions of Unlabeled PDE Data

In Table 2, for the purpose of OOD testing, we summarize the distribution of our unlabeled PDE data during pretraining, fine-tuning, and inference with in-context examples. Ranges of these physical parameters are inspired by [73]. During pretraining, we consider a wide distribution of unlabeled PDE data. When training (fine-tuning) our model, we consider in-distribution unlabeled PDE data. Finally, we test our similarity-based method that learns in-context examples in OOD settings.<sup>4</sup> For Helmholtz OOD, we choose a narrow range of coefficients ([15, 20]) mainly because its solution is very sensitive to the wavenumber, and FNO’s performance significantly drops when we move to more extreme OOD settings.

Table 2: Ranges of physical parameters (integers) for unsupervised pretraining, training (fine-tuning), and out-of-distribution (OOD) inference.

| Physics Parameters          | Poisson<br>(diffusion) | Helmholtz<br>(wave number) | Naiver Stokes<br>(Reynolds number) |
|-----------------------------|------------------------|----------------------------|------------------------------------|
| Unsupervised Pretraining    | [1, 20]                | [1, 20]                    | {100, 300, 500, 800, 1000}         |
| Training (or Fine-tuning)   | [5, 15]                | [5, 15]                    | 300                                |
| Out-of-Distribution Testing | [15, 50]               | [15, 20]                   | 10000                              |

### B.2 Data Generation

**Unlabeled PDE Data.** We generate data for Poisson and Helmholtz [73], Reaction-Diffusion on PDE-Bench [74] and 2D incompressible Navier-Stokes on PINO Dataset [46] following the procedure mentioned in the paper. For unlabeled data generation, we bypass the computation of solvers, which expedites the generation speed, as shown in Table 4.

**OOD Samples.** The OOD data generation procedure is similar to the unlabeled data, except for the changes in the physical parameters coefficients. For Poisson and Helmholtz, we consider changing the range of diffusion eigenvalue and waver number respectively. For Navier-Stokes equation, we change the Reynolds number. We list the coefficients in Table 2.

<sup>4</sup>For Navier Stokes from PDEBench, we use the original data. We could not generate our own pretraining/finetuning data with different ranges of physics parameters, due to a possible mismatch of the provided configuration files and the version of Phiflow used in PDEBench (see GitHub issue at <https://github.com/pdebench/PDEBench/issues/36>).

### B.3 Training Hyperparameters

We summarize our hyperparameters used during pretraining and fine-tuning/training in Table 3. These hyperparameters strictly follow previous works [45, 73, 74, 46, 52]. We conducted our experiments on four A100 GPUs, each with 40GB of memory.

Table 3: Hyperparameters for pretraining and training/fine-tuning. “N.S.”: 2D Incompressible Navier-Stokes. “DAdapt”: adaptive learning rate by D-adaptation [10]. “ns”: total number of simulated training samples. A batch size of “min(32, ns)” is because the total number of training samples might be fewer than 32.

| Stage ↓                   | PDEs →                                    | Poisson            | Helmholtz          | Reaction-Diffusion | N.S. (PINO)        | N.S. (PDEBench) | ERA5               | ScalarFlow         | Airfoil            |
|---------------------------|-------------------------------------------|--------------------|--------------------|--------------------|--------------------|-----------------|--------------------|--------------------|--------------------|
| Pretraining               | Number of Samples                         | 46,080             | 46,080             | 76,760             | 57,545             | 713,286         | 8,760              | 56,160             | 43,104             |
|                           | Learning Rate                             | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | DAdapt          | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ |
|                           | Batch Size                                | 32                 | 32                 | 5                  | 2                  | 8               | 32                 | 32                 | 32                 |
|                           | Resolution: ( $H \times W$ )              | 64×64              | 64×64              | 128×128            | 128×128            | 512×512         | 180×180            | 180×120            | 96×45              |
| Training /<br>Fine-tuning | Epochs/Iterations                         | 500 epochs         | 500 epochs         | 500 epochs         | 100,000 iters      | 500 epochs      | 500 epochs         | 500 epochs         | 500 epochs         |
|                           | Learning Rate                             | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | DAdapt          | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ | $1 \times 10^{-3}$ |
|                           | Batch Size                                | min(32, ns)        | min(32, ns)        | 5                  | 2                  | 8               | 4                  | 4                  | min(32, ns)        |
|                           | Resolution:                               |                    |                    |                    |                    |                 |                    |                    |                    |
|                           | ( $H \times W$ or $H \times W \times T$ ) | 64×64              | 64×64              | 128×128×10         | 64×64×33           | 512×512×15      | 180×180×16         | 180×120×16         | 96×45              |
|                           | Epochs/Iterations                         | 500 epochs         | 500 epochs         | 500 epochs         | 50,000 iters       | 500 epochs      | 500 epochs         | 500 epochs         | 500 epochs         |
|                           | Rollouts                                  | N/A                | N/A                | 91                 | N/A                | 1               | 1                  | 1                  | N/A                |

### C Examples of Simulation Costs

In Table 4, we demonstrate the cheap simulation of only unlabeled PDE data, versus simulating both unlabeled PDE data and solutions, on 2D incompressible Navier-Stokes on PINO Dataset [46] and Reaction-Diffusion on PDE-Bench [74]. We can see that unlabeled PDE data are extremely cheap to simulate. Therefore, our pretraining method can boost the performance and meanwhile save the heavy cost of data simulations.

Table 4: Simulation time costs on 2D Incompressible Navier-Stokes (“N.S.”) on PINO Dataset [46] and Reaction-Diffusion (“R.D.”) on PDE-Bench [74]. “Re”: Reynolds number. “ $D_u, D_v$ ”: diffusion coefficients.  $N$ : number of samples.  $T$ : temporal resolution.  $H \times W$ : spatial resolution.  $C$ : input channels (1 for the vorticity in N.S., 2 for velocities  $u, v$  in R.D.).

| Data | Physical Parameters                                  | Unlabeled PDE Data (sec.) | Data+Solutions (sec.) | Data Size                                                                                | CPU             | GPU                  |
|------|------------------------------------------------------|---------------------------|-----------------------|------------------------------------------------------------------------------------------|-----------------|----------------------|
| N.S. | $Re = 100$                                           | 5499.32                   | 11013.90              | $N \times T \times H \times W = 20 \times 2000 \times 512 \times 512$                    | 1 AMD EPYC 7763 | 1 NVIDIA A100 (40GB) |
|      | $Re = 300$                                           | 3683.02                   | 7625.82               |                                                                                          |                 |                      |
|      | $Re = 500$                                           | 4059.71                   | 8963.39               |                                                                                          |                 |                      |
|      | $Re = 800$                                           | 4829.3                    | 10811.15              |                                                                                          |                 |                      |
|      | $Re = 1000$                                          | 4957.24                   | 10788.69              |                                                                                          |                 |                      |
| R.D. | $D_u = 1 \times 10^{-3}$<br>$D_v = 5 \times 10^{-3}$ | 29.65                     | 6657.34               | $N \times T \times H \times W \times C = 1000 \times 101 \times 128 \times 128 \times 2$ | 1 AMD EPYC 7763 | N/A                  |

### D Model Architectures

In Figure 7, We show visualizations of architectures described in Sec. 3.1.4. Specifically, the FNO has 67.1M parameters, and the Video-MAE has 23.4M. During pretraining, FNO costs 20 GPU hours, and Video-MAE costs 18 GPU hours. During fine-tuning, FNO costs 4 GPU hours, and Video-MAE costs 6 GPU hours.

### E Comparison with Contrastive Learning

Contrastive learning is an important self-supervised pretraining technique studied in computer vision. For a fair comparison, we directly compare with MoCo v2 [8], a highly-cited self-supervised learning method also originally implemented in PyTorch, whose core method is closely related to SimCLR [7] (originally implemented in TensorFlow).

We compare MoCo v2 with our method on a broader real-world benchmark ERA5 [30]. As shown in Figure 8, our unsupervised learning method can largely outperform MoCo v2. This extra comprehen-

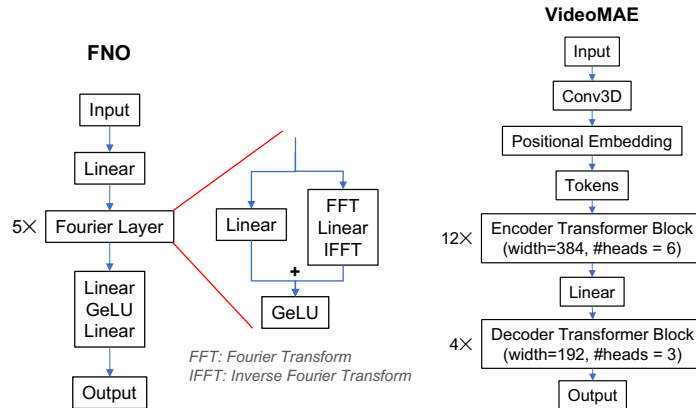


Figure 7: Visualizations of architectures we studied. Left: FNO [46]. Right: VideoMAE [77].

sive result demonstrates that our method can be widely adopted in real-world problems, outperforming previous unsupervised learning methods.

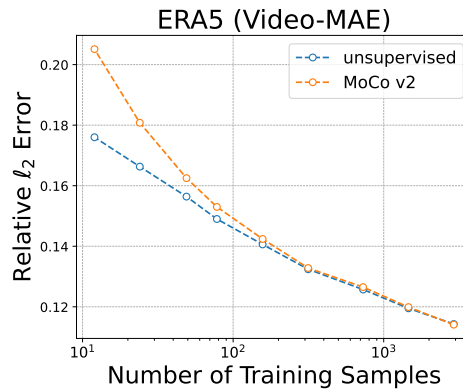


Figure 8: Comparison between our unsupervised pretraining method versus MoCo v2 [8].

## F More Comparisons with Vision Pretrained Models

Beyond Figure 3(e) for the transformer on time-dependent PDE (Navier-Stokes), we further verify that pretraining on unsupervised PDE data makes FNO outperform its off-the-shelf vision-pretrained checkpoints. Specifically, we first pretrain the 2D FNO model on ImageNet [11], then fine-tune on the downstream PDE simulation data. As shown in Figure 9, vision-pretrained FNO performs worse during downstream fine-tuning on time-independent PDEs (including both Poisson and Helmholtz equations), confirming that domain-specific pretraining, even on unsupervised PDE data, is more beneficial than conventional checkpoints pretrained on unrelated domains like computer vision.

## G Joint Pretraining Further Improves Performance

We also study if joint pretraining on unlabeled multiple PDE data can bring extra benefits. We combine all 46,080 unlabeled Poisson samples and all 46,080 unlabeled Helmholtz samples (see Table 3). We choose this setting because recent works on SciML foundation models [52, 26] also use all samples from each PDE for pretraining. We do zero-paddings for mismatched channels. From Figure 10, we can see that joint pretraining can further improve the performance of fine-tuning on different PDEs.

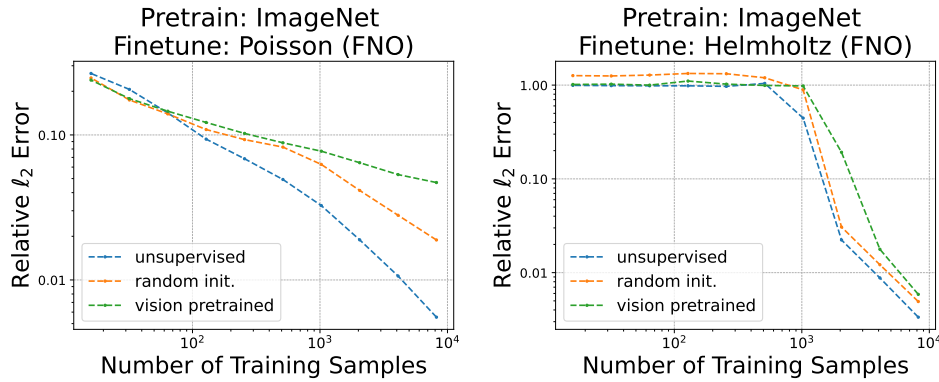


Figure 9: Pretraining neural operators on unlabeled PDE data improves its performance and data efficiency on Poisson (left), Helmholtz (right). “random init.”: models are trained from scratch with random initialization. “vision pretrained”: fine-tuning from the checkpoint pretrained on computer vision dataset ImageNet [11].

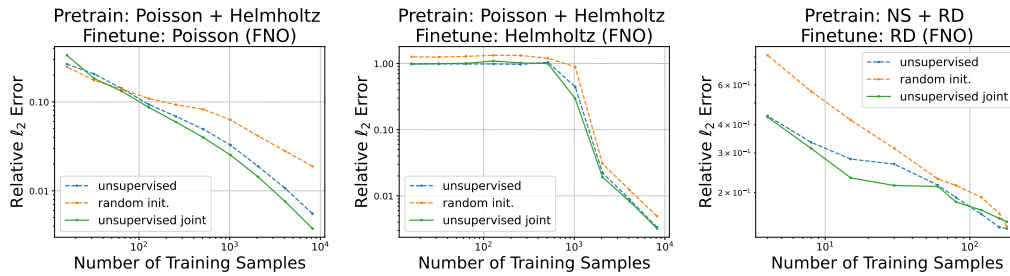


Figure 10: **Joint unsupervised pretraining on multiple PDEs (green solid curve) further improves the data efficiency of neural operators** when fine-tuning on Poisson (left), Helmholtz (middle), Reaction-Diffusion (right). “random init.”: models are trained from scratch with random initialization. “unsupervised”: models are pretrained on a *single* unsupervised PDE data. “unsupervised joint”: models are pretrained on a *joint* of multiple unsupervised PDE datasets. “NS”: Navier Stokes. “RD”: Reaction-Diffusion.

## H Fine-tuning on Unseen PDEs is Challenging

We also try to fine-tune neural operators on unseen PDEs (i.e. PDEs different from pretraining). Mismatched channels are padded with zeros. We find this will lead to worse performance compared with models pretrained on the same PDE, as shown in Figure 11.

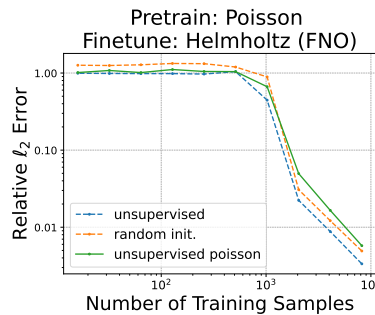


Figure 11: Fine-tuning FNO (pretrained on Poisson) on unseen samples from Helmholtz.

## I Consistently Improved Generalization

Beyond the generation gap we have shown in Figure 4 (left) for the Navier Stokes equation from PDEBench, we further collect generation gaps of our models learned on other PDEs. As shown in Figure 12, on diverse PDE systems, our method can contribute to universally reduced overfitting.

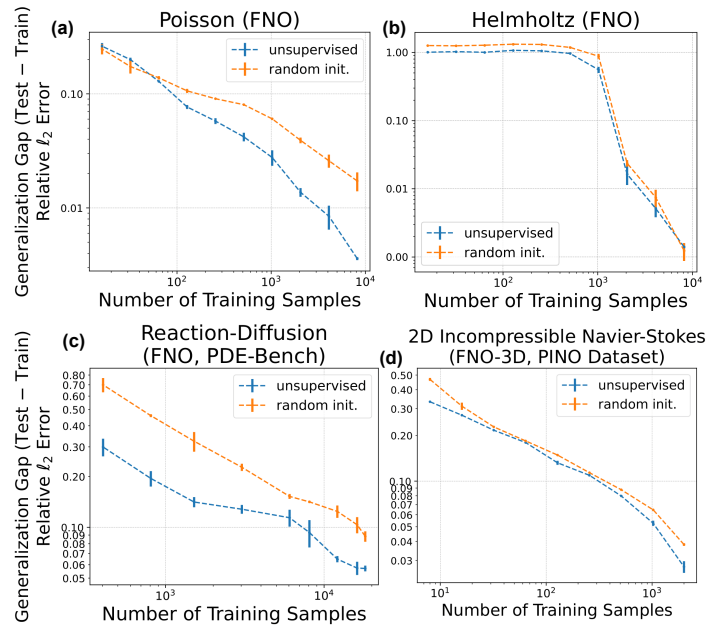


Figure 12: Universally reduced overfitting (i.e. smaller generalization gaps) on diverse PDEs (a to d).

## J More Ablation Studies

### J.1 Magnitude of Perturbations during Pretraining.

We study the optimal magnitude of perturbations during pretraining: “Mask Ratio” (1 indicates no visible grids, and 0 means no masks); and “Blur Sigma” for the variance of Gaussian kernel for blur (larger the more degradation). We show our results in Table 5, 6, 7, 8. For example, on 2D incompressible Navier-Stokes with FNO, as shown in Table 8, we can find that when training with a low volume of data, we should use much stronger perturbations (high masking ratios and strong blur), whereas a high volume of data only requires mild perturbations.

Table 5: Best choice of mask ratio and blur sigma for pretraining on Poisson equation.

| #Samples | Mask Ratio | Blur Sigma |
|----------|------------|------------|
| 16       | 0          | 0          |
| 32       | 0          | 0~1        |
| 64       | 0          | 0~1        |
| 128      | 0          | 0~1        |
| 256      | 0          | 0~1        |
| 512      | 0          | 0~1        |
| 1024     | 0          | 0~1        |
| 2048     | 0          | 0~1        |
| 4096     | 0          | 0~1        |
| 8192     | 0          | 0~1        |

Table 6: Best choice of mask ratio and blur sigma for pretraining on Helmholtz equation.

| #Samples | Mask Ratio | Blur Sigma |
|----------|------------|------------|
| 16       | 0.2        | 0~1        |
| 32       | 0.2        | 0~1        |
| 64       | 0.2        | 0~1        |
| 128      | 0.2        | 0~1        |
| 256      | 0.2        | 0~1        |
| 512      | 0.6        | 0~2        |
| 1024     | 0.6        | 0~2        |
| 2048     | 0          | 0~1        |
| 4096     | 0          | 0~1        |
| 8192     | 0          | 0~0.5      |

### J.2 Ablation of the Number of Pretraining Samples.

As shown in Table 9, the more unlabeled PDE data we use for pretraining, the better quality the pretrained model will be.

Table 7: Best choice of mask ratio and blur sigma for pretraining on 2D Diffusion-Reaction equation.

| #Samples | Mask Ratio | Blur Sigma |
|----------|------------|------------|
| 404      | 0          | 0~1        |
| 808      | 0          | 0~1        |
| 1515     | 0          | 0~1        |
| 3030     | 0.7        | 0~2        |
| 6060     | 0.9        | 0~1        |
| 8080     | 0.3        | 0~1        |
| 12120    | 0          | 0~1        |
| 16160    | 0          | 0~1        |
| 18180    | 0          | 0~1        |

Table 8: Best choice of mask ratio and blur sigma for pretraining on 2D incompressible Navier-Stokes.

| #Samples | Mask Ratio | Blur Sigma |
|----------|------------|------------|
| 8        | 0.7        | 0~4        |
| 16       | 0.7        | 0~4        |
| 32       | 0.7        | 0~4        |
| 64       | 0          | 0~1        |
| 128      | 0          | 0~1        |
| 256      | 0.05       | 0          |
| 512      | 0.05       | 0          |
| 1024     | 0.05       | 0          |
| 2000     | 0.05       | 0          |

Table 9: More unlabeled PDE data improve the quality of pretraining. FNO on 2D incompressible Navier-Stokes, pretrained with mask ratio as 0.7.

| #Pretraining Samples    | 2000   | 57545  |
|-------------------------|--------|--------|
| Relative $\ell_2$ Error | 0.3594 | 0.3246 |

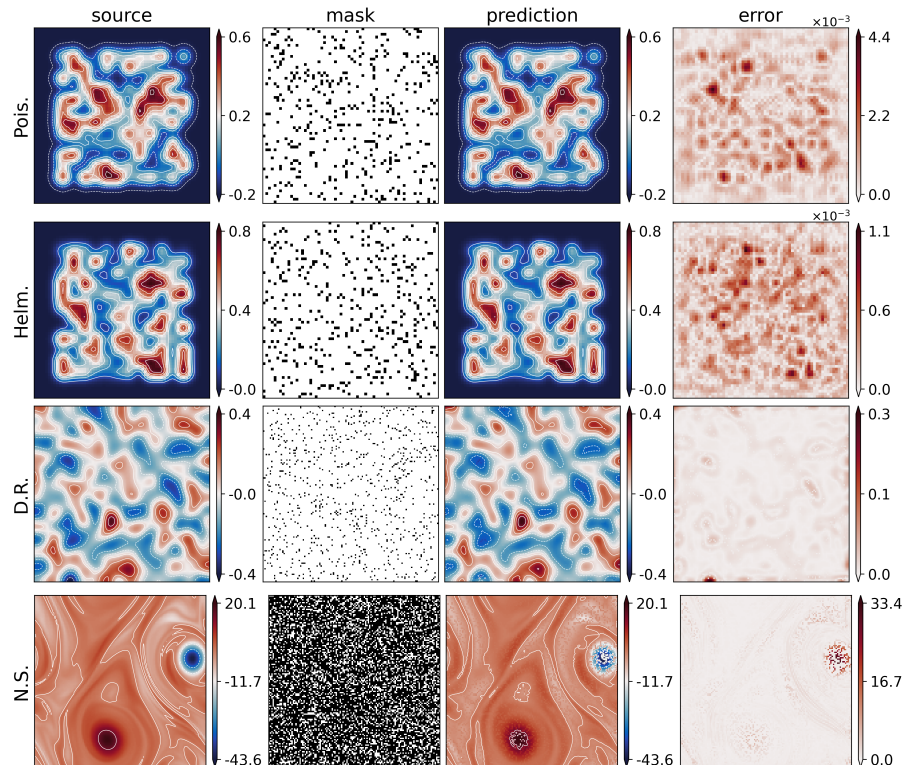


Figure 13: Visualization of FNO reconstructions of unlabeled PDE data on the Poisson (“Pois.”), Helmholtz (“Helm.”), 2D Diffusion-Reaction (“D.R.”), and 2D incompressible Navier-Stokes (“N.S.”) equations during MAE pretraining. (Mask ratio: 0.1 for Poisson, Helmholtz, and 2D Diffusion-Reaction equations; 0.7 for incompressible Navier-Stokes.) In masks, only white areas are visible to the model during pretraining.

## K Visualization of MAE Pretraining

To demonstrate the efficacy of our MAE-based pretraining, we show the unlabeled PDE data and its reconstructed version in Figure 13 (MAE pretraining on different PDEs) and Figure 14 (MAE pretraining on 2D incompressible Navier-Stokes with varying mask ratios). We can see that all inputs are accurately reconstructed with low errors and similar patterns.

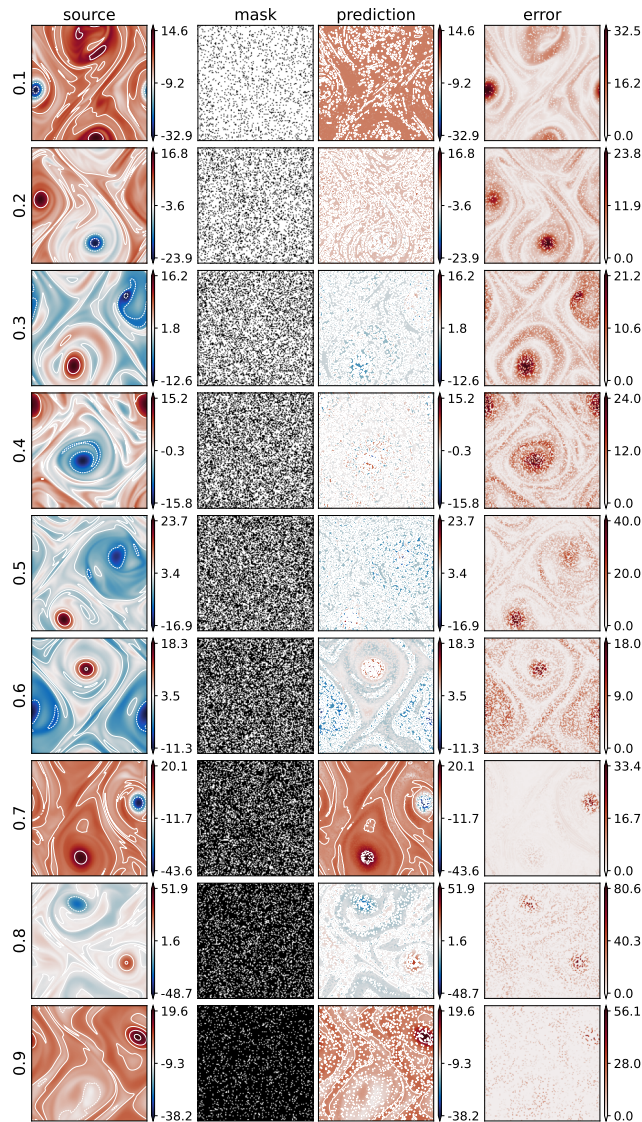


Figure 14: Visualization of FNO reconstructions of unlabeled PDE data on the 2D incompressible Navier-Stokes equations during MAE pretraining with mask ratio from 0.1 to 0.9.

## L Details and Visualizations of Real-World Data

### L.1 ECMWF Reanalysis v5 (ERA5)

We utilize data from 2006 to 2015, with the snapshots taken every 6 hours and a spatial resolution of  $360 \times 360$ . The total number of snapshots is 14600. We apply the mean-standard deviation normalization to the data and downsample the snapshots to a spatial resolution of  $180 \times 180$ . We split 75% of the data for pretrain and 25% for finetune. For each split, we further separate 80% of the data for training, 10% for validation, and 10% for testing.

## L.2 ScalarFlow

The original spatial resolution is  $1062 \times 600$ . We crop the snapshots to  $900 \times 600$  to remove the padding and background. We remove the first 15 timeframes for each simulation to avoid the initial transient phase at the beginning of the smoke flow generation. We then downscale the snapshots to  $180 \times 120$  and apply mean-standard deviation normalization to the data. With a total of 70200 snapshots, we split 80% of the data for pretrain and 20% for finetune. For each split, we further separate 80% of the data for training, 10% for validation, and 10% for testing.

## L.3 Airfoil

We split 80% of the training data for pretrain and 20% for finetune. The original spatial resolution of each sample is  $128 \times 128$ . We crop the snapshots to  $96 \times 45$  to remove the background. We then apply min-max normalization channel-wise to the sample.

We show visualizations of real-world scientific data in Figure 15.

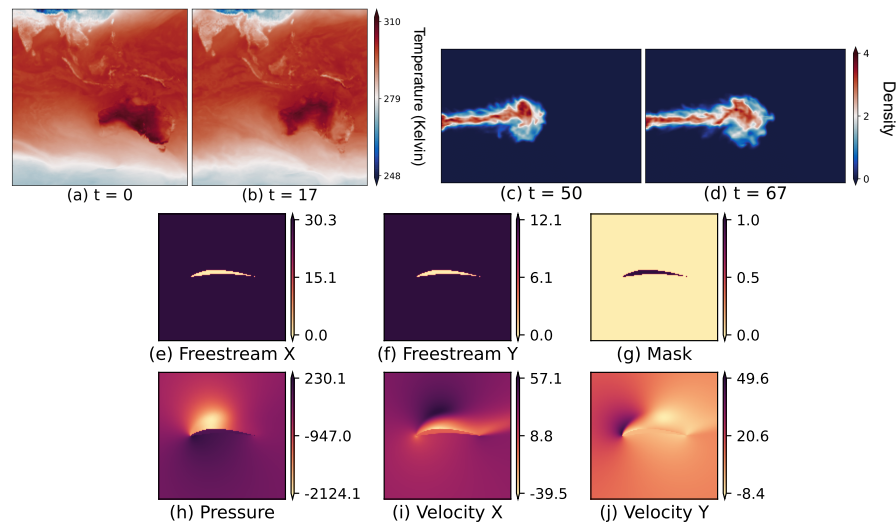


Figure 15: We show snapshot examples from ERA5 temperature [30] (a, b) and ScalarFlow [14] (c, d) at different temporal steps; and also an example of Airfoil mask, velocities, and pressure [75] (e-j).

## M Benefits of In-context Examples

We try to understand the benefit of leveraging in-context examples by decomposing the relative MSE error into “scale” and “shape.” “Scale” is the slope of a linear regression between targets and predictions (closer to 1 the better), indicating the alignment of the range of model outputs with targets. “Shape” is the normalized relative MSE (i.e., model outputs or targets are normalized by their own largest magnitude before MSE), indicating the alignment of scale-invariant spatial/temporal structures. We show results in Figure 16. We find that the benefit of in-context examples lies in that the scale of the model’s output keeps being better calibrated (“scale” being closer to 1) when adding more demos.

In numerical simulations or predictions of PDEs, there are settings where the scale or magnitude of solutions is more important than the exact shape/pattern:

1. Heat Transfer: In large-scale systems, the focus might be on overall temperature and extreme values. For instance, in evaluating a cooling system, the key concern might be the peak temperature rather than the detailed temperature distribution.
2. Fluid Dynamics: For applications like aerodynamics, the overall drag or lift force on an object is often more critical than capturing every detail of the flow pattern, such as in airfoil design.
3. Environmental Modeling: The concentration of pollutants at specific locations or total pollutant transport is often more crucial than the exact distribution, such as in groundwater flow studies.

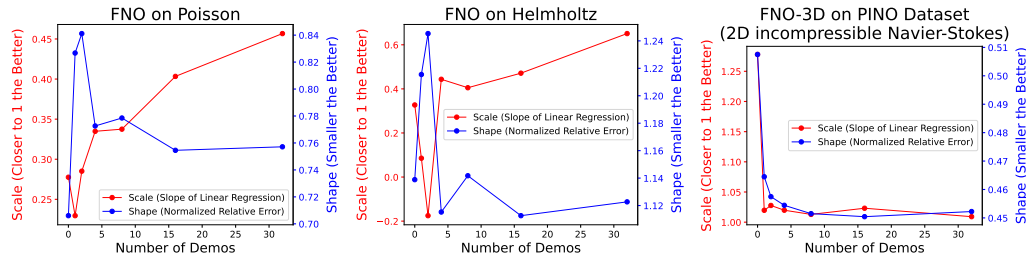


Figure 16: Benefits of in-context examples. To analyze the benefit of in-context examples for complicated PDE systems, we decompose the relative MSE error into “Scale” and “Shape”. “Scale” indicates the alignment of the range of model outputs with targets (closer to 1 the better), via the slope of a linear regression. “Shape” indicates the alignment of scale-invariant spatial/temporal structures via normalized relative MSE (i.e. model outputs or targets are normalized by their own largest magnitude before MSE). We find that the benefit of in-context examples lies in that the scale of the model’s output keeps being calibrated (red line being closer to 1) when adding more demos.

## N Visualizations with In-Context Examples

We show visualizations of our similarity-based mining of in-context examples in Figure 17. In this visualization, we find that the range of numerical solutions (e.g., values in colorbars) predicted with in-context examples becomes closer to the target. Meanwhile, based on this visualization, we conclude that OOD generalization is challenging for neural operators because of: 1) Significantly different patterns of solutions under different physical parameters; 2) Different value ranges of solutions under different physical parameters.

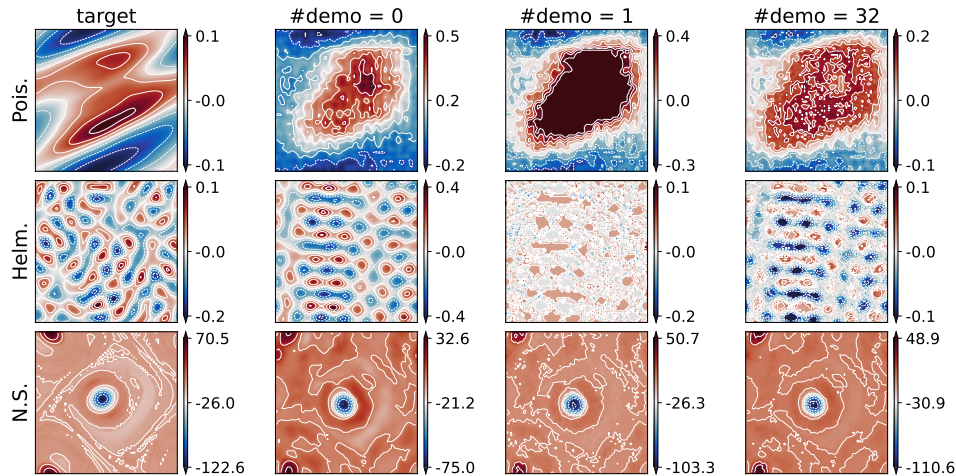


Figure 17: Visualizations of mining in-context examples for FNO in OOD testing. Ranges of solutions predicted with in-context examples (min/max of each snapshot, reflected in colorbars) become closer to the target.

## NeurIPS Paper Checklist

### 1. Claims

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

Answer: [Yes] .

Justification: All claims stated in the abstract and introduction are evaluated in our experiments (Sec. 4).

Guidelines:

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

### 2. Limitations

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

Answer: [Yes] .

Justification: We discussed our limitations in a separate Section 6.

Guidelines:

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

### 3. Theory Assumptions and Proofs

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

Answer: [NA] .

Justification: This paper does not include theoretical results.

Guidelines:

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

#### 4. Experimental Result Reproducibility

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

Answer: [\[Yes\]](#) .

Justification: We include instructions for reproducibility in the Appendix, including choices of unlabeled PDE data (Appendix A), data generation and training hyperparameters B, and model structures D. We also attached our code in supplement.

Guidelines:

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

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

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

Answer: [\[Yes\]](#) .

Justification: We attached our code in the supplement.

Guidelines:

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

## 6. Experimental Setting/Details

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

Answer: [\[Yes\]](#) .

Justification: We include our experimental settings in Appendix B.

Guidelines:

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

## 7. Experiment Statistical Significance

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

Answer: [\[Yes\]](#) .

Justification: All our operator pretraining experiments are reproduced for three times.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments Compute Resources

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

Answer: [Yes] .

Justification: We discussed the compute resources we used in Appendix B.3.

Guidelines:

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

## 9. Code Of Ethics

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

Answer: [Yes] .

Justification: Our work does not involve any potential harm, societal impact, or violation.

Guidelines:

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

## 10. Broader Impacts

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

Answer: [NA] .

Justification: This work is about foundational research on data augmentations. It does not lead to any societal impact because: 1) our motivations, goals, and results are too distant from social applications to make any social impact; 2) we mainly use public data and benchmarks; and 3) we use public model architectures and follow previous works which have open access.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

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

Answer: [NA] .

Justification: The paper poses no such risks because all datasets studied in our work are from public benchmarks, and we have clearly cited original papers.

Guidelines:

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

## 12. Licenses for existing assets

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

Answer: [Yes] .

Justification: All datasets studied in our work are from public benchmarks, and we have clearly cited original papers.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

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

### 13. **New Assets**

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

Answer: [NA] .

Justification: The paper does not release new assets.

Guidelines:

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

### 14. **Crowdsourcing and Research with Human Subjects**

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

Answer: [NA] .

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

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

### 15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

Answer: [NA] .

Justification: The paper does not involve any potential risk regarding research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.