PDE-PFN: Prior-Data Fitted Neural PDE Solver

Jaehyeon Park[1] **Mingu Kang**[2] **Dongseok Lee**[3] **Woojin Cho**[4]
Kookjin Lee[5] **Anthony Gruber**[6] **Youngjoon Hong**[3]* **Noseong Park**[2]*

[1]SolverX, [2]KAIST, [3]Seoul National University, [4]TelePIX
[5]Arizona State University, [6]Sandia National Laboratories

jaehyeon.park@solverx.ai, sgn08050@kaist.ac.kr, lorafa@snu.ac.kr,
woojin.py@gmail.com, kookjin.lee@asu.edu, adgrube@sandia.gov
hongyj@snu.ac.kr, noseong@kaist.ac.kr

Abstract

Motivated by the success of large language models (LLMs) with broad generalizability and robustness to noisy or unreliable pre-training data, we explore in-context prediction of PDE solution fields using approximate pre-training data. In addition, inspired by the posterior predictive mean-based inference mechanism of the in-context learning in prior-data fitted networks (PFNs), we propose PDE-PFN, a prior-data fitted neural solver that approximates the posterior predictive mean of PDE solutions via in-context learning. PDE-PFN builds on a PFN architecture with self- and cross-attention mechanisms of Transformer and is pre-trained on noisy approximate solutions generated by physics-informed neural networks, serving as diverse but not necessarily exact priors. Through experiments on a range of two-dimensional PDEs, we demonstrate that PDE-PFN achieves empirical generalization across heterogeneous equations, robustness under noisy priors, and zero-gradient-update in-context inference capability. PDE-PFN achieves competitive or improved predictive accuracy relative to the evaluated baselines while supporting flexible context and query representations.

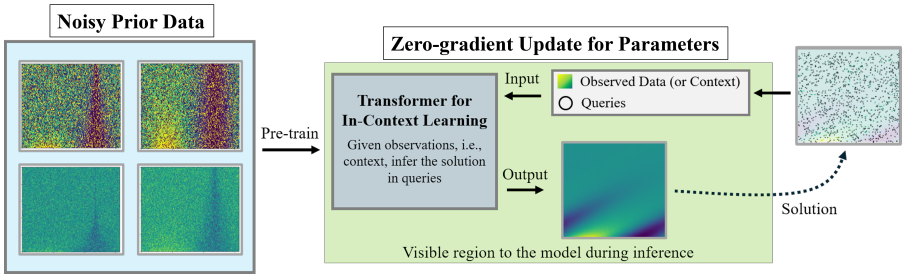

Figure 1: **The overall workflow of PDE-PFN.** Our model performs in-context learning (ICL) based on given observations (i.e., context) to infer solutions. At inference, PDE-PFN conditions on observations from the target instance without separate operator-level demonstrations or target-instance gradient updates. Even when trained with an approximated PINN prior, our method can improve over noisy approximate pretraining data.

1 Introduction

Partial differential equations (PDEs) have long been at the core of scientific and engineering research, as they provide a mathematical description of the fundamental laws governing diverse natural

*Corresponding Authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and engineered systems. With recent advances in machine learning, scientific machine learning (SciML) [42, 59, 46, 23, 9] has emerged as a promising paradigm for solving PDE-related problems.

Within the trend of SciML, there are two conventional approaches: physics-informed neural network (PINN)-based models [42, 45, 62, 31, 66] and operator-based models [27, 28, 51, 14]. PINNs embed a given physical law into a model by minimizing PDE residuals during training, without requiring large labeled datasets. Neural operator-based models learn mappings between infinite-dimensional function spaces given a target operator. While they are mostly general, some operator models require a fixed grid (or a pre-specified coordinate system).

In recent years, large language models (LLMs) have revolutionized natural language processing by introducing highly flexible and scalable architectures [5, 21, 50, 11, 10]. Motivated by the success of LLMs in enabling broad task generalization, scientific foundation models (SFMs) [30, 63, 61, 68] have recently been proposed. These models are typically pre-trained on a wide range of PDEs and subsequently fine-tuned for diverse downstream tasks, extending their applicability beyond the narrow scope of conventional neural operators.

Although SFMs represent a meaningful advance by enabling generalization across different PDEs, they continue to inherit other fundamental drawbacks from their backbones. In particular, they lack robustness to noisy input and impose input rigidity, often constraining data to uniform grids or fixed spatial locations. To address these limitations, we draw inspiration from prior-fitted networks (PFNs) [33], which implicitly utilizes the posterior predictive distribution (PPD) conditioned on prior information. Building on this idea, we propose **PDE-PFN**, a new SciML method that adopts a PFN architecture grounded in in-context learning. PDE-PFN uses PFN-style prior fitting to learn context-conditioned predictions, with experiments evaluating within-family task and coefficient generalization, cross-PDE transfer after fine-tuning, and flexible context/query inputs. The PPM perspective characterizes the population target of the MSE objective, while robustness to imperfect priors is evaluated empirically and supported theoretically only under the stated noise assumptions.

Within-family generalization and cross-PDE transfer We first evaluated the proposed method on the convection–diffusion–reaction (CDR) equations, a standard benchmark for parameterized PDEs. Our model exhibited strong task generalizability, successfully handling diverse tasks across both parameter and spatiotemporal domains without requiring task-specific fine-tuning. In addition, the CDR-pre-trained model demonstrated cross-PDE transfer after fine-tuning on downstream datasets using a reduced training schedule. Across the evaluated settings, PDE-PFN achieved competitive or improved predictive accuracy relative to task-specific baselines and fine-tuned scientific foundation models.

Flexibility on input shape without prior physical knowledge Our method advances SciML by removing two key constraints that limit existing approaches. First, it predicts solutions directly from observed quantities, such as velocity and pressure, without requiring access to symbolic equation information at inference time. This feature is particularly important in real-world applications where the underlying equations are incomplete or entirely unknown [6, 44, 3, 36, 25]. Excluding explicit equations from the input is thus an intentional design choice that broadens applicability across diverse domains. Second, in complex systems such as semiconductor manufacturing, observational data are often collected from irregularly distributed sensors rather than uniform grids [34, 41, 6]. In such settings, the ability to handle inputs of arbitrary shape is essential. Our model accepts variable-sized sets of coordinate–value observations and allows target coordinates to be specified as queries. Together, these properties support flexible prediction from pointwise observations without requiring a fixed observation grid. A detailed comparison with existing baselines is provided in Table 7 of Appendix D.

Robustness to noisy prior data (or pre-training data) For LLMs, one of the most challenging steps is collecting prior data, which typically involves crawling and cleaning sentences from the Internet. However, this process is far from perfect due to two key issues: (i) the Internet, as a data source, is inherently unreliable; and (ii) cleaning such vast amounts of data requires significant manual effort. Consequently, LLMs are often trained on incomplete or imperfect prior data but show surprising outcomes. This realistic yet critical issue has been largely overlooked in the literature on SFMs, despite their similarities to LLMs. In this work, we investigate PFN-style pre-training

on approximate PINN solutions and demonstrate strong predictive performance in the evaluated noisy-prior settings.

Zero-gradient-update in-context inference Within the CDR evaluations, a single pre-trained PDE-PFN checkpoint predicts query values from observed context without task-specific gradient updates or fine-tuning. The key property is that PDE-PFN adapts to the target PDE instance only through in-context conditioning. Compared with in-context operator-learning approaches such as ICON-LM [64], which require separate task/operator-level "demos"[1] before prediction, PDE-PFN uses only available observations as the conditioning context and does not require an additional demonstration set or parameter update. This makes the inference procedure suitable for settings where irregularly sampled observations are available but retraining or collecting separate demos for each target operator is impractical.

2 Background

Consider a sequence of pairs (c_1, y_1), (c_2, y_2), $\cdots$, (c_n, y_n), each within the measurable space $(\mathcal{C} \times \mathcal{Y}, \mathcal{A})$, where c_i represents the spatiotemporal coordinate, y_i denotes the corresponding solution, n is data size, and $\mathcal{A}$ denotes the Borel σ-algebra on the measurable space $\mathcal{C} \times \mathcal{Y}$. These pairs are drawn from a family of probability density distributions $\{p_q : q \in \mathcal{Q}\}$, where $\mathcal{Q}$ represents the *parameter space*, equipped with a σ-algebra $\mathcal{B}$, ensuring that the mappings $q \mapsto p_q(c, y)$ are measurable. The true parameter π is an element of $\mathcal{Q}$, and the pairs (c_i, y_i) are sampled according to p_π. Lacking the information on π, we adopt the following posterior predictive inference framework, which is also used to describe the in-context learning capability of LLMs [60], to establish a prior distribution Π which is defined as a probability measure on $(\mathcal{Q}, \mathcal{B})$. Then we have

$$\Pi(A \mid c, y) = \frac{\int_A p_q(c, y) \mathrm{d}\Pi(q)}{\int_{\mathcal{Q}} p_q(c, y) \mathrm{d}\Pi(q)}, \tag{1}$$

for any measurable set $A \in \mathcal{B}$. This prior is updated with the observed data to form the posterior distribution:

$$\Pi(A \mid D_n) = \frac{\int_A L_n(q) \mathrm{d}\Pi(q)}{\int_{\mathcal{Q}} L_n(q) \mathrm{d}\Pi(q)}, \tag{2}$$

where $L_n(q) = \prod_{i=1}^n \frac{p_q(c_i, y_i)}{p_\pi(c_i, y_i)}$ for $A \subset \mathcal{Q}$ and $D_n = \{(c_i, y_i)\}_{i=1}^n$. The resulting posterior is

$$q_n(c, y \mid D_n) = \int_{\mathcal{Q}} p_q(c, y) \mathrm{d}\Pi(q \mid D_n), \tag{3}$$

and the posterior predictive distribution (PPD) is written as

$$q_n(y \mid c, D_n) = \int_{\mathcal{Q}} p_q(y \mid c) \, \mathrm{d}\Pi(q \mid D_n). \tag{4}$$

As noted by [35, 54, 56, 55, 4], for a well-behaved prior, the PPD converges toward true π as the data size n increases. This aligns with findings by Blasi and Walker [4], demonstrating that in well-specified scenarios, strong consistency is achieved as

$$\Pi^n \left(\{q : H(p_\pi, p_q) > \epsilon\} \right) \to 0 \quad \text{almost surely}, \tag{5}$$

for any $\epsilon > 0$, where $\Pi^n(A) = \int_A d\Pi(q \mid D_n)$ is the posterior measure and H is the Hellinger distance:

$$H(p, q) = \left(\int_{\mathcal{C} \times \mathcal{Y}} (\sqrt{p} - \sqrt{q})^2 \right)^{1/2}. \tag{6}$$

Theorem 2.1. *There exists a neural network $p_{\hat{\theta}}$ that approximates the posterior predictive distribution $p_\pi(\cdot \mid c)$ (See Appendix A for the full Theorem claim and its proof).*

Theorem A.1 demonstrates the existence of the neural network $p_{\hat{\theta}}$ that approximates the PPD. While this result justifies learning the full PPD, modeling the entire distribution can be computationally intractable in practice. Since our goal is deterministic PDE prediction, we instead focus on learning the posterior predictive mean (PPM) in the next section.

[1] In ICON and ICON-LM, a demo refers to a set of (input, output) pairs of an operator to be inferred.

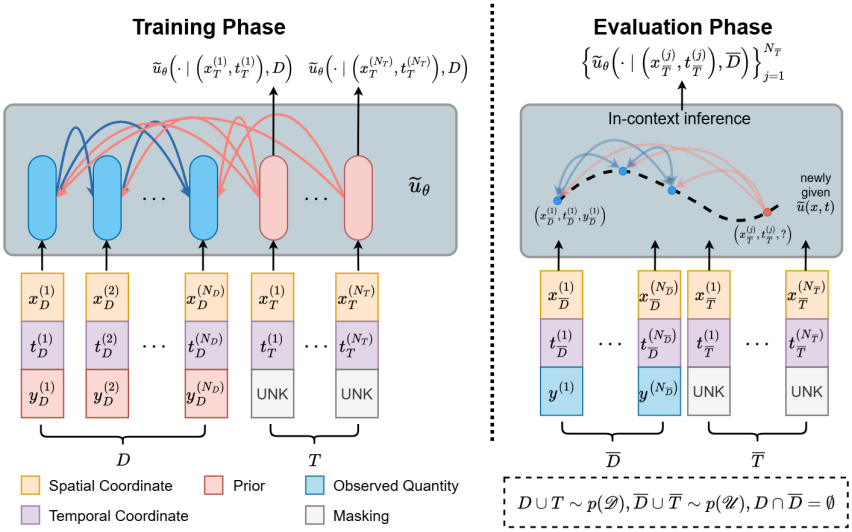

Figure 2: **A schematic diagram of our model.** *(Left)* Given the contexts D and queries T drawn from the prior distribution $\mathcal{D}$, our model $\tilde{u}_\theta$ is trained to infer the solutions of the queried points T in the training phase. ICL is leveraged with self-attention (blue) within D and cross-attention (red) from T to D. *(Right)* In the testing phase, the pre-trained $\tilde{u}_\theta$ takes contexts $\overline{D}$ and queries $\overline{T}$ drawn from the ground truth distribution $\mathcal{U}$, and predicts for the queried points $\overline{T}$.

3 Methods

In the Background section, we denoted the spatiotemporal coordinate as c for integration. However, for more clarity, as shown in Figure 2, we separate the spatiotemporal coordinate c into a spatial coordinate x and a temporal coordinate t, i.e., $c = (x, t)$. Let α denote a parameter vector representing the coefficients of the governing PDE dynamics. Figure 2 illustrates the schematic diagram of our model. As defined in Figure 2, $D = \{(x_D^{(i)}, t_D^{(i)}, y_D^{(i)})\}_{i=1}^{N_D}$ and $T = \{(x_T^{(i)}, t_T^{(i)}, y_T^{(i)})\}_{i=1}^{N_T}$ represent the context set and the query set for ICL, respectively. We assume these sets are independently and identically sampled from a family of probability density distributions $\{p_{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathcal{Q}\}$. Here, each y corresponds to the quantity at (x, t) for a PDE solution $u(\boldsymbol{\alpha})$ determined by the PDE parameter $\boldsymbol{\alpha}$. Depending on the problem setting, u can be either the ground-truth solution or an approximated solution generated by a PINN, since we consider both exact and approximated prior data. The PPD of the solutions given the context set is written as $\pi(y \mid x, t, D) = \int_{\mathcal{Q}} p_{\boldsymbol{\alpha}}(y \mid x, t)\, \mathrm{d}\Pi(\boldsymbol{\alpha} \mid D)$ and its posterior predictive mean (PPM), which is our training goal, as

$$\mathbb{E}[Y|x, t, D] = \int_Y y\, \pi(y \mid x, t, D)\, \mathrm{d}y. \tag{7}$$

For categorical outputs, PFNs can use cross-entropy to approximate posterior predictive class probabilities [60, 33]. In this work, because PDE-PFN is a deterministic regressor, we minimize the MSE between the predicted solution and the target solution.

Theorem 3.1. *Under squared loss, the population-optimal deterministic predictor is the posterior predictive mean conditioned on the available context. Even with the unbiased target noise, the same population minimizer equals the clean-solution PPM conditioned on the noisy context. (See Appendix A for the full Theorem claim and its proof).*

As shown in Theorem A.2, this objective corresponds to estimating the PPM induced by the prior over PDE solution fields, rather than the full PPD. The theorem also clarifies that, under unbiased target noise, the MSE-optimal deterministic predictor remains the clean-solution conditional mean given the noisy context.

Model architecture Since our model is essentially a prior-data fitted network (PFN), we construct its architecture by modifying certain components of the original PFN. To better handle the more complex PDE data, we first incorporate a Fourier feature embedding into the input data before it is

passed to the encoder, extending the original design that was limited to tabular data. We also introduce several modifications to the Transformer architecture. Specifically, to improve the model's ability to learn from complex data while reducing computational complexity and enhancing representational capacity, we adopt the attentive graph filter (AGF) [58] structure for Transformer layers. A more detailed description of the model architecture can be found in Appendix E.

Benchmark PDEs We consider convection-diffusion-reaction (CDR) equations as benchmark PDEs for diverse tasks[2]:

$$u_t = \mathcal{N}(\cdot), \quad \mathcal{N}(t, x, u, \boldsymbol{\beta}, \boldsymbol{\nu}, \rho_1, \cdots, \rho_J) = -\boldsymbol{\beta} \cdot \nabla u + \nabla \cdot (\boldsymbol{\nu} \nabla u) + \sum_{j=1}^{J} \rho_j f_j(u), \tag{8}$$

where u denotes the state variable; $\boldsymbol{\beta}$ is the convection (advection) velocity; $\boldsymbol{\nu}$ denotes the anisotropic diffusivity coefficient; and ρ_1, ρ_2, ρ_3 are scalar coefficients corresponding to the nonlinear reaction terms $f_1, f_2, f_3 : \mathbb{R} \to \mathbb{R}$, respectively. For clarity, we will refer to the two-dimensional CDR equations, including cases where certain coefficients are zero, as the family of CDR equations. This family encompasses not only generic parabolic equations but also extends to hyperbolic equations such as convection equations. Hence, the family of CDR equations contains three key terms—convection, diffusion, and reaction—each with distinct properties, making it an ideal benchmark problem. To the best of our knowledge, this work is the first to address a wide range of reaction terms within the family of CDR equations using a single unified model. We use a set of CDR-related terms and a linear combination of three nonlinear reaction terms to generate prior data. This formulation enables the incorporation of diverse reaction dynamics, making the benchmark problems more comprehensive.

PINN-based approximations of PDE solution space Although numerical solvers can solve the family of CDR equations, in this work, we use the predictions from PINNs as a representative example of noisy approximated data. This allows us to evaluate the robustness of our model to noise, which is commonly encountered in real-world scenarios. From this point, we refer to these PINN-based approximations as PINN priors. To approximate the solution space for the PDEs, we construct a practical parameter subspace, $\Omega \subseteq \mathcal{Q}$, which is the collection of coefficients in Eq. (8) and has a dictionary form $\Omega = \{\boldsymbol{\alpha} := (\boldsymbol{\beta}, \boldsymbol{\nu}, \rho_1, \cdots, \rho_J)\}$. Consequently, the target exact prior $\mathcal{U}$ defined below represents the collection of solutions $u(\boldsymbol{\alpha})$ at Eq. (8) for each parameter $\boldsymbol{\alpha} \in \Omega$, where $\mathcal{X}$ and $\mathcal{T}$ correspond to the spatial and temporal domains of interest, respectively.

$$\mathcal{U} = \bigcup_{\boldsymbol{\alpha} \in \Omega} \{u(\boldsymbol{\alpha}) \,|\, u_t = \mathcal{N}(t, x, u, \boldsymbol{\alpha})\}, \quad u : \mathcal{X} \times \mathcal{T} \to \mathbb{R}. \tag{9}$$

Since the target exact prior data $\mathcal{U}$ is hard to obtain, we instead use a PINN prior $\mathcal{D}$ that closely approximates $\mathcal{U}$ as follows. Suppose $\tilde{u}(\boldsymbol{\alpha})$ is the prediction by PINN, which is trained to predict the PDE $u_t = \mathcal{N}(\cdot)$ (see Appendix C for details). The PINN prior $\mathcal{D}$ is a collection of approximate solutions $\tilde{u}(\boldsymbol{\alpha})$ for each $\boldsymbol{\alpha} \in \Omega$,

$$\mathcal{D} = \bigcup_{\boldsymbol{\alpha} \in \Omega} \{\tilde{u}(\boldsymbol{\alpha})\}, \quad p(\mathcal{D}) \sim p(\mathcal{U}). \tag{10}$$

Subsequently, the model performs the posterior predictive mean inference conditioned on the generated prior $p(\mathcal{D})$ through ICL.

Objective function From a given parameter space Ω, the parameter $\boldsymbol{\alpha}$ is randomly drawn i.i.d. from Ω. After this, the corresponding prior $\tilde{u}(\boldsymbol{\alpha})$ is given as an input to our model $\tilde{u}_\theta$ to minimize the MSE at the queried points, as expressed in Eq. (11), which leads to the posterior predictive mean inference [33] — in other words, the MSE loss is enough to approximate the mean of Eq. (7) without knowing its uncertainty. The specific MSE loss chosen to regress the solution over the spatial and temporal domain for a given $\tilde{u}(\boldsymbol{\alpha})$ is defined as

$$\mathcal{L}_{\boldsymbol{\alpha}} = \frac{1}{N_T} \sum_{j=1}^{N_T} \left[\tilde{u}_\theta(x_T^{(j)}, t_T^{(j)} \mid D) - \tilde{u}(\boldsymbol{\alpha})(x_T^{(j)}, t_T^{(j)}) \right]^2. \tag{11}$$

[2]We note that in Eq. (8) J reaction terms are considered. In previous works [7, 8, 20], only one type of reaction term is considered. Eq. (8) therefore defines a parameterized CDR prior with three selected nonlinear reaction terms for PFN training. Our experiments differ from previous works in this regard.

In addition, we introduce a regularization term into the objective function to ensure orthogonality during the SVD process within the AGF layer (see Appendix E for detail). We denote this term as $\mathcal{L}_{AGF}$. To balance the trade-off between the loss and the regularization, we scale the latter by the number of context points N_D. Consequently, the final objective function $\mathcal{L}$ is expressed as

$$\mathcal{L} = \mathcal{L}_{\boldsymbol{\alpha}} + \frac{1}{N_D}\mathcal{L}_{AGF}. \tag{12}$$

The concrete flow of the training and the evaluation process is described in Appendix F and G.

4 Experiments

Our experimental section is divided into three phases. In the first phase, we introduce the dataset and establish the baselines for our experiments. In the second phase, we pre-train our model on the family of 2D CDR equations and evaluate it on a variety of tasks without any additional task-specific fine-tuning. This allows us to examine the model's ability to handle diverse tasks. In the third phase, we fine-tune our model on different PDEs, including the shallow water equations (SWE) and compressible Navier-Stokes equations (CNSE), to assess cross-PDE transfer after fine-tuning. Table 1 summarizes the experiments in this section.

Table 1: Roadmap of the experiment section. 'Inter' and 'Extra' denote spatiotemporal interpolation and temporal extrapolation tasks, respectively. The 'Coeff/dataset' column specifies either the coefficients of the equation or the dataset used in evaluation.

Section	Task	Coeff/Dataset	Equation	Objective
4.2.1	Inter & Extra	Seen	Family of 2D CDR equations	Within-family task generalization
4.2.2	Inter & Extra	Unseen		Within-family coefficient generalization
4.3.1	Inter & Extra	Seen & Unseen	SWE	Cross-PDE transfer after fine-tuning
4.3.2	Operator Learning	Unseen	CNSE	Cross-PDE transfer after fine-tuning

4.1 Experimental Setup

Dataset We consider a family of 2D CDR equations with three reaction terms as the pre-training dataset. This family is constructed from equations with coefficient dictionary $(\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\})$, resulting in a total of 2,187 unique PDEs. They are simulated from the same initial condition, which allows differences in the resulting dynamics to stem solely from the variations in coefficients. For PINN prior, each PINN is trained independently, and its predicted solutions are used as noisy priors. On average, the generated PINN priors exhibit an L_2 relative error of 0.02280. For cross-PDE transfer experiments, we employ the SWE and CNSE datasets from PDEBench [47]. These two equations are chosen because they are commonly used to validate PDE solvers in SciML. Both datasets consist of 1,000 samples generated from distinct initial conditions. The SWE dataset contains a single feature, the water height, whereas the CNSE dataset includes x- and y-velocities, pressure, and density. Detailed information on the datasets can be found in Appendix B.

Baseline methods We compare our model with six baseline models: deep operator network (DeepONet) [28], Fourier neural operator (FNO) [27], factorized Fourier neural operator (F-FNO) [51], adaptive Fourier neural operator (A-FNO) [14], latent neural operator (LNO) [57], Poseidon [17], and auto-regressive denoising pre-training operator Transformer (DPOT) [16]. These baselines are chosen for their scalability and ability to effectively handle complex 2D PDEs. For Poseidon and DPOT, scientific foundation models with publicly available pre-trained weights, we use the released weights and perform fine-tuning. The other baselines are trained and evaluated separately on each dataset and task. For the spatiotemporal interpolation task, DeepONet is the only baseline used, excluding those that require grid inputs. The best hyperparameters for all models are selected based on their best validation performance in each task. Detailed information on the baselines used in each experiment can be found in Appendix D.

4.2 Pre-training on the Family of 2D CDR Equations and Evaluation

The training dataset is constructed from 6 time steps within the range $t \in [0.0, 0.5]$ at intervals of 0.1. Our model takes as an input set of contexts, which include domain coordinates paired with their

solution values, along with queries, which consist of domain coordinates where the solution values are masked. We denote the version of our model pre-trained on numeric priors as **Ours** and the version pre-trained on PINN priors as **Ours (PINN)**. For the baselines, the training procedure differs by task. In the spatiotemporal interpolation task, since DeepONet requires fixed points as branch input, we predefined the fixed points in training data by sampling and the remaining points are used as queries to be predicted. In the temporal extrapolation task, the baselines are trained to take the grid points at a time t_n as input and predict the grid points at the subsequent time step t_{n+1}. Further information on the training procedure, such as the number of epochs, can be found in Appendix J.

4.2.1 Task Generalization for Seen coefficients

We first evaluate the task generalization ability in spatiotemporal interpolation and temporal extrapolation without additional task-specific fine-tuning. Both tasks are conducted within the coefficient dictionary used during pre-training, which we refer to as seen coefficients. For the spatiotemporal interpolation task, the baseline DeepONet requires fixed points as branch input. Therefore, our model is also provided with the same set of context points during evaluation to ensure a fair comparison. Specifically, we use the points at $t = 0.25$ as validation queries and the points at $t = 0.05, 0.15, 0.35,$ and 0.45 as test queries. For the temporal extrapolation task, the points at $t = 0.5$ are used as contexts, and those at $t = 0.6$ as queries for validation. At test time, the points at $t = 0.6$ serve as contexts, and those at $t = 0.7, 0.8, 0.9,$ and 1.0 are used as queries. The baselines predict a sequence of query points through rollout, whereas our model can directly predict the query points from their coordinates so that it is evaluated in this manner. The evaluation results are shown in Table 2.

Table 2: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to the family of 2D CDR equations. They are measured at the seen coefficients ($\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\}$). The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Spatiotemporal interpolation	L_1 Abs	**0.01218**	0.01507	0.01922	-	-	-	-	-	-
	L_2 Rel	**0.01643**	0.02057	0.02479						
	L_∞ Rel	**0.04600**	0.06584	0.05980						
Temporal extrapolation	L_1 Abs	**0.01474**	0.01940	0.06529	0.03302	0.02731	0.05800	0.02876	0.14252	0.04924
	L_2 Rel	**0.02261**	0.02705	0.08212	0.04058	0.03430	0.07915	0.03665	0.15580	0.08587
	L_∞ Rel	**0.08444**	0.08802	0.22802	0.09553	0.09877	0.28905	0.10948	0.28285	0.08587

The evaluation results demonstrate that both versions of our model outperform the baselines on both tasks. This indicates that once trained on a single dataset, our model can achieve strong performance without requiring task-specific fine-tuning. The PINN-prior results support robustness to the approximation errors present in the evaluated pre-training data. Additionally, we conduct experiments where the PINN prior is provided as a context set during evaluation in Appendix H.2.

4.2.2 Task Generalization for Unseen coefficients

We further evaluate the task generalization ability on both tasks in parameter space using the family of 2D CDR equations with unseen coefficients, without any additional fine-tuning. The unseen coefficients are categorized into two cases: coefficient interpolation and coefficient extrapolation. Coefficient interpolation experiments are performed on the equations with intermediate coefficient values not seen during pre-training, ($\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.25, 0.75\}$). Coefficient extrapolation experiments, on the other hand, are conducted on the equations with larger coefficient values than those used in pre-training, ($\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{1.25, 1.5\}$). For evaluation with unseen coefficients in the temporal extrapolation task, we modify the temporal extrapolation setup to remain within the temporal domain used during training: the points at $t = 0.1$ are provided as contexts, and models are asked to predict the values at $t = 0.2, 0.3, 0.4,$ and 0.5. The evaluation results are summarized in Table 3.

The evaluation results show that both versions of our model outperform the baselines on both tasks under coefficient interpolation and extrapolation. This shows that our models can make accurate predictions not only on PDEs with coefficients seen during training but also on PDEs with unseen coefficients, without requiring additional fine-tuning. In other words, the models can generalize across the parameter space, achieving both interpolation and extrapolation.

Table 3: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to the family of 2D CDR equations. They are measured at the unseen coefficients. The best performance is marked in **bold** and the second-best performance is marked with underline.

Coefficients	Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Interpolation	Spatiotemporal interpolation	L_1 Abs	**0.01241**	0.01277	0.01347	-	-	-	-	-	-
		L_2 Rel	**0.01658**	0.01761	0.01731						
		L_∞ Rel	0.04274	0.05503	**0.04114**						
	Temporal extrapolation	L_1 Abs	0.08957	**0.08709**	0.11918	0.11625	0.11194	0.12220	0.11085	0.17740	0.13879
		L_2 Rel	**0.11990**	0.12465	0.14903	0.14658	0.14348	0.18191	0.14913	0.21340	0.15085
		L_∞ Rel	0.29526	0.36472	0.35196	**0.28689**	0.30813	0.64429	0.33280	0.39890	0.15085
Extrapolation	Spatiotemporal interpolation	L_1 Abs	0.01661	**0.01582**	0.03576	-	-	-	-	-	-
		L_2 Rel	0.02398	**0.02038**	0.04669						
		L_∞ Rel	0.08324	**0.06890**	0.10392						
	Temporal extrapolation	L_1 Abs	**0.01027**	0.02063	0.05514	0.02652	0.02854	0.05013	0.05076	0.14661	0.05186
		L_2 Rel	**0.01068**	0.02169	0.06450	0.03043	0.03300	0.06351	0.06347	0.15650	0.05189
		L_∞ Rel	**0.01661**	0.03474	0.16555	0.06124	0.08343	0.20955	0.15091	0.27502	0.05189

4.3 Cross-PDE Transfer after Fine-tuning

To evaluate cross-PDE transfer after fine-tuning, we conduct experiments on heterogeneous PDEs beyond the 2D CDR family. On the shallow water equations (SWE) dataset, we assessed whether our model could adapt through fine-tuning to other PDEs on the same tasks as in the family, namely spatiotemporal interpolation and temporal extrapolation. The baselines for each task are trained in the same manner as in the family. On the CNSE dataset, we further examined whether the model could adapt to trajectory prediction for a different PDE.

4.3.1 Cross-PDE Transfer to Shallow Water Equations

To evaluate the PDE generalization ability of our model, we conduct experiments on the SWE, assessing whether the tasks defined in the family of 2D CDR equations can be successfully transferred to this heterogeneous PDE using half as many epochs as the task-specific baselines. Out of the total 1,000 samples, 700 are used for training, and the training, validation, and test datasets are constructed over the time interval in the same manner as for the family of 2D CDR equations. The test dataset generated from the initial condition samples used during training is referred to as the seen dataset, while the test dataset generated from the remaining 300 samples is referred to as the unseen dataset. For the baselines, training for each task is carried out in the same manner as in pre-training.

Table 4: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to SWE. They are measured at the seen/unseen datasets. The best performance is marked in **bold** and the second-best performance is marked with underline.

Evaluation dataset	Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Seen	Spatiotemporal interpolation	L_1 Abs	**0.00208**	0.00269	0.00271	-	-	-	-	-	-
		L_2 Rel	**0.00739**	0.00822	0.00997						
		L_∞ Rel	**0.05897**	0.06950	0.06951						
	Temporal extrapolation	L_1 Abs	0.01548	**0.01418**	0.06994	0.03271	0.02292	0.05036	0.02902	0.02964	0.03327
		L_2 Rel	0.03273	**0.03221**	0.08319	0.05429	0.04110	0.07436	0.05269	0.05208	0.05860
		L_∞ Rel	**0.20590**	0.24596	0.28736	0.24954	0.24478	0.38030	0.24832	0.25668	0.29727
Unseen	Spatiotemporal interpolation	L_1 Abs	**0.00213**	0.00272	0.00278	-	-	-	-	-	-
		L_2 Rel	**0.00754**	0.00828	0.01008						
		L_∞ Rel	**0.05986**	0.06981	0.07047						
	Temporal extrapolation	L_1 Abs	0.01571	**0.01448**	0.07047	0.03342	0.02371	0.05099	0.02877	0.03021	0.03372
		L_2 Rel	0.03323	**0.03288**	0.08347	0.05530	0.04218	0.07513	0.05225	0.05306	0.05914
		L_∞ Rel	**0.20686**	0.24686	0.28441	0.24980	0.24378	0.38196	0.24788	0.25710	0.29757

The experimental results in Table 4 show that both versions of our model outperform the baselines on both spatiotemporal interpolation and temporal extrapolation tasks, regardless of whether the dataset is seen or unseen. Specifically, Ours achieves the best performance on spatiotemporal interpolation, while Ours (PINN) performs best on temporal extrapolation. These results support transfer to SWE after fine-tuning and prediction on trajectories generated from initial conditions not used for fine-tuning.

4.3.2 Cross-PDE Transfer to Compressible Navier-Stokes Equations

To further evaluate cross-PDE transfer after fine-tuning, we conduct experiments on the compressible Navier–Stokes equations (CNSE), focusing in particular on learning the time trajectory predicting operator. This experiment is designed not only to evaluate adaptation to a different PDE family but also to assess the potential of our approach in learning an operator that was not included in previous experiments. The dataset consists of 1,000 samples, of which 700 are used for training, 100 for validation, and 200 for testing. The target operator is defined as follows: given the initial condition of four features, the models are required to predict the solution of them at later times $t = 0.25, 0.5, 0.75,$ and 1.0 (see Appendix B.3 for a formal description). Baseline models except DeepONet and DPOT are trained and evaluated using rollout predictions. In contrast, DeepONet and our model directly predict the solutions given the initial condition. Additionally, since DPOT allows the output timestep length to be adjusted, it also predicts the solutions in a single step.

Table 5: The evaluation results for the operator learning task applied to CNSE. The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Operator learning	L_1 Abs	0.09497	0.09429	0.17797	0.09581	**0.08734**	0.13637	0.10366	0.13313	0.09164
	L_2 Rel	0.54833	**0.54369**	1.09567	0.57905	0.55420	0.65798	0.62536	0.66904	0.67869
	L_∞ Rel	0.63533	**0.63374**	1.09165	0.70708	0.72876	0.85729	0.66936	0.92909	1.15028

The experimental results in Table 5 show that both versions of our model outperform the baselines on two relative error metrics. Notably, although DPOT's pre-training corpus already includes the CNSE dataset from PDEBench, our models still achieve superior performance over DPOT on both metrics. While DPOT was originally evaluated on predicting one time step ahead, in our setting the task was extended to predicting up to four time steps ahead. As a result, DPOT exhibited lower performance than reported in the original paper. These results support cross-PDE transfer after fine-tuning for multivariate trajectory prediction.

We further evaluate adaptation to additional PDE datasets and flexibility across input representations in supplementary experiments. First, we evaluate operator learning in the same form as in the CNSE experiments on the Airfoil dataset, which is constructed with an irregular mesh structure. In addition, we evaluate operator learning across different features, rather than within the same features, using the Darcy Flow dataset. Detailed information on these experiments is provided in Appendix H.

5 Related Works

In-context learning Transformers have shown remarkable ICL abilities in various studies. They generalize to unseen tasks by emulating Bayesian predictors [38] and linear models [67], while also efficiently performing Bayesian inference through prior-data fitted networks [33]. Their robustness extends to learning different function classes, such as linear and sparse linear functions, decision trees, and two-layer neural networks, even under distribution shifts [13]. Furthermore, Transformers can adaptively select algorithms based on input sequences, achieving near-optimal performance on tasks like noisy linear models [2]. They are also fast and effective for tabular data classification [18, 19]. Recent work has further connected ICL to richer Bayesian inference procedures [43] and implicit optimization dynamics such as gradient descent or Newton-like updates [53, 12].

Context-query neural operators. GNOT [15] and UPT [1] provide related architectures for prediction from flexible inputs and coordinate queries. GNOT supports multiple input functions and irregular meshes, while UPT supports queries of a latent representation at arbitrary spatiotemporal locations. PDE-PFN emphasizes PFN-style episodic prior fitting over PDE instances and context/query sets, enabling the same CDR checkpoint to support multiple prediction settings without task-specific updates. Cross-attention itself is not a unique contribution; a matched GNOT comparison and its attribution limitations are reported in Appendix H.17.

Scientific foundation models Recent studies have advanced in-context operator learning and PDE solving through Transformer-based models. The work Ye et al. [65] introduces PDEformer, a versatile model for solving 1D PDEs with high accuracy in inverse problems. In-context operator learning has also been extended to multi-modal frameworks, as seen in Yang et al. [64], where ICON-LM

integrates natural language with equations to outperform traditional models. Additionally, Yang and Osher [61] and Yang et al. [63] demonstrate the generalization capabilities of ICON in solving various PDE-related tasks, highlighting ICON's few-shot learning performance across various problems in differential equations. Several other studies have addressed the problem of solving diverse PDEs using a single trained model [68, 30, 17, 16]. However, many of these approaches rely on symbolic PDE information, require task/operator-level demonstrations, or adapt through fine-tuning, making their objectives different from ours.

6 Conclusion

We presented PDE-PFN, a new method for scientific machine learning to directly approximate the mean of posterior predictive distribution of PDE solutions via in-context learning. Our experiments demonstrate task and coefficient generalization within the CDR family without gradient updates, cross-PDE transfer after fine-tuning, flexible context/query inputs without symbolic equations at inference, and robustness in the evaluated noisy-prior settings. These findings support PFN-style prior fitting as a promising approach to equation-free in-context prediction over PDE solution fields. Future work will examine continuous coefficient priors, broader initial- and boundary-condition distributions, and systematic scaling with model size, prior size and diversity, context size, and distributional distance. Residual and conservation diagnostics, structured distribution shifts, and probabilistic prediction under ambiguous contexts will further assess physical consistency and the scope of generalization.

7 Acknowledgements

K. Lee acknowledges support from the U.S. National Science Foundation under grant IIS 2338909.

A. Gruber acknowledges support received through the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Multifaceted Integrated Capability Centers (MMICCS) program, under the Scalable, Efficient and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems (SEA-CROGS) project. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

The work of Y. Hong was supported by the Korea government (MSIT) through the NRF (RS-2023-00219980 & RS-2026-25491270); and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2021-II211343, Artificial Intelligence Graduate School Program, Seoul National University).

The work of N. Park was supported by Samsung Electronics Co., Ltd(IO250616-13070-01), Samsung Research Funding & Incubation Center for Future Technology of Samsung Electronics under Project Number SRFC-IT2402-08, and the Institute for Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korean government (MSIT) (No. RS-2025-25442149, LG AI STAR Talent Development Program for Leading Large-Scale Generative AI Models in the Physical AI Domain).

References

[1] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024.

[2] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36:57125–57211, 2023.

[3] Andrea Beck and Marius Kurz. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002, 2021.

[4] Pierpaolo De Blasi and Stephen G. Walker. Bayesian asymptotics with misspecified models. *Statistica Sinica*, pages 169–187, 2013.

[5] 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.

[6] Chen-Fu Chien, Chia-Yu Hsu, and Chih-Wei Hsiao. Manufacturing intelligence to forecast and reduce semiconductor cycle time. *Journal of Intelligent Manufacturing*, 23:2281–2294, 2012.

[7] Woojin Cho, Kookjin Lee, Donsub Rim, and Noseong Park. Hypernetwork-based meta-learning for low-rank physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36:11219–11231, 2023.

[8] Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized pdes. In *International Conference on Machine Learning*, pages 8510–8533, 2024.

[9] Junho Choi, Taehyun Yun, Namjung Kim, and Youngjoon Hong. Spectral operator learning for parametric pdes without data reliance. *Computer Methods in Applied Mechanics and Engineering*, 420:116678, 2024.

[10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of machine learning research*, 24(240): 1–113, 2023.

[11] Simon Frieder, Luca Pinchetti, Chevalier Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. Mathematical capabilities of chatgpt. *Advances in neural information processing systems*, 36:27699–27744, 2023.

[12] Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn to achieve second-order convergence rates for in-context linear regression. *Advances in Neural Information Processing Systems*, 37:98675–98716, 2024.

[13] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

[14] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient token mixing for transformers via adaptive fourier neural operators. In *International conference on learning representations*, 2021.

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

[16] 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. In *International Conference on Machine Learning*, pages 17616–17635. PMLR, 2024.

[17] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel De Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

[18] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023.

[19] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL `https://www.nature.com/articles/s41586-024-08328-6`.

[20] Mingu Kang, Dongseok Lee, Woojin Cho, Kookjin Lee, Anthony Gruber, Nathaniel Trask, Youngjoon Hong, and Noseong Park. Can we pre-train icl-based sfms for the zero-shot inference of the 1d cdr problem with noisy data? In *Neurips 2024 Workshop Foundation Models for Science: Progress, Opportunities, and Challenges*, 2024.

[21] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[22] David I Ketcheson, Kyle Mandli, Aron J Ahmadia, Amal Alghamdi, Manuel Quezada De Luna, Matteo Parsani, Matthew G Knepley, and Matthew Emmett. Pyclaw: Accessible, extensible, scalable tools for wave propagation problems. *SIAM Journal on Scientific Computing*, 34(4): C210–C231, 2012.

[23] Namjung Kim, Dongseok Lee, Chanyoung Kim, Dosung Lee, and Youngjoon Hong. Simple arithmetic operation in latent space can generate a novel three-dimensional graph metamaterials. *npj Computational Materials*, 10(1):236, 2024.

[24] Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. Apebench: A benchmark for autoregressive neural emulators of pdes. *Advances in Neural Information Processing Systems*, 37:120252–120310, 2024.

[25] Chin Yik Lee and Stewart Cant. A grid-induced and physics-informed machine learning cfd framework for turbulent flows. *Flow, Turbulence and Combustion*, 112(2):407–442, 2024.

[26] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[27] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

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

[29] Mingze Ma, Qingtian Zhu, Yifan Zhan, Zhengwei Yin, Hongjun Wang, and Yinqiang Zheng. Robustifying fourier features embeddings for implicit neural representations. *arXiv preprint arXiv:2502.05482*, 2025.

[30] Michael McCabe, Bruno Régaldo-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. In *NeurIPS 2023 AI for Science Workshop*, 2023.

[31] Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering*, 370:113250, 2020.

[32] Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *International Conference on Learning Representations*, 2019.

[33] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

[34] Stephen C Myers and Craig A Schultz. Improving sparse network seismic location with bayesian kriging and teleseismically constrained calibration events. *Bulletin of the Seismological Society of America*, 90(1):199–211, 2000.

[35] Thomas Nagler. Statistical foundations of prior-data fitted networks. In *International Conference on Machine Learning*, pages 25660–25676. PMLR, 2023.

[36] Zachary G Nicolaou, Guanyu Huo, Yihui Chen, Steven L Brunton, and J Nathan Kutz. Data-driven discovery and extrapolation of parameterized pattern-forming dynamics. *Physical Review Research*, 5(4):L042017, 2023.

[37] Francisco Palacios, Juan Alonso, Karthikeyan Duraisamy, Michael Colonno, Jason Hicken, Aniket Aranake, Alejandro Campos, Sean Copeland, Thomas Economon, Amrita Lonkar, et al. Stanford university unstructured (su 2): an open-source integrated computational environment for multi-physics simulation and design. In *51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, page 287, 2013.

[38] Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the bayesian prism. In *The Twelfth International Conference on Learning Representations*, 2024.

[39] John E Pearson. Complex patterns in a simple system. *Science*, 261(5118):189–192, 1993.

[40] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2020.

[41] Michael Quirk and Julian Serda. *Semiconductor manufacturing technology*, volume 1. Prentice Hall Upper Saddle River, NJ, 2001.

[42] 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.

[43] Arik Reuter, Tim GJ Rudner, Vincent Fortuin, and David Rügamer. Can transformers learn full bayesian inference in context? In *Forty-second International Conference on Machine Learning*, 2025.

[44] Nusrat Rouf, Majid Bashir Malik, Tasleem Arif, Sparsh Sharma, Saurabh Singh, Satyabrata Aich, and Hee-Cheol Kim. Stock market prediction using machine learning techniques: a decade survey on methodologies, recent developments, and future directions. *Electronics*, 10 (21):2717, 2021.

[45] Khemraj Shukla, Patricio Clark Di Leoni, James Blackshire, Daniel Sparkman, and George Em Karniadakis. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *Journal of Nondestructive Evaluation*, 39(3):61, 2020.

[46] 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. *Advances in Neural Information Processing Systems*, 36:71242–71262, 2023.

[47] 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.

[48] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[49] Eleuterio F Toro, Michael Spruce, and William Speares. Restoration of the contact surface in the hll-riemann solver. *Shock waves*, 4(1):25–34, 1994.

[50] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[51] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.

[52] Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

[53] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

[54] Stephen Walker. On sufficient conditions for bayesian consistency. *Biometrika*, 90(2):482–488, 2003.

[55] Stephen Walker. New approaches to bayesian consistency. *Annals of statistics*, 32(5):2028–2043, 2004.

[56] Stephen G Walker. Modern bayesian asymptotics. *Statistical Science*, pages 111–117, 2004.

[57] Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. *Advances in Neural Information Processing Systems*, 37:33085–33107, 2024.

[58] Hyowon Wi, Jeongwhan Choi, and Noseong Park. Learning advanced self-attention for linear transformers in the singular value domain. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 6561–6569, 2025.

[59] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys*, 55(4):1–37, 2022.

[60] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022.

[61] Liu Yang and Stanley J Osher. Pde generalization of in-context operator networks: A study on 1d scalar nonlinear conservation laws. *Journal of Computational Physics*, 519:113379, 2024.

[62] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

[63] 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.

[64] Liu Yang, Siting Liu, and Stanley J Osher. Fine-tune language models as multi-modal differential equation solvers. *Neural Networks*, page 107455, 2025.

[65] Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. Pdeformer: Towards a foundation model for one-dimensional partial differential equations. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024.

[66] Lei Yuan, Yi-Qing Ni, Xiang-Yun Deng, and Shuo Hao. A-pinn: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *Journal of Computational Physics*, 462:111260, 2022.

[67] Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024.

[68] Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: Pde-conditional transformers towards universal neural pde solvers. In *Forty-second International Conference on Machine Learning*, 2025.

A The Proof of Theorems

Theorem A.1 (Existence of the neural network approximating the posterior predictive distribution)**.** *Suppose that for any $\epsilon > 0$, there exists a Transformer parameterized by $\hat{\theta}$ such that*

$$\mathbb{E}_c\left[KL\left(p_{\hat{\theta}}(\cdot \mid c, D_n), q_n(\cdot \mid c, D_n)\right)\right] < \epsilon,$$

for any realization of D_n. If the posterior consistency condition Eq. (5) *holds, and for any $q \in \mathcal{Q}$, $p_q(c) = p_\pi(c)$ almost everywhere on $\mathcal{X}$, then the following holds almost surely:*

$$\mathbb{E}_c\left[H\left(p_{\hat{\theta}}(\cdot \mid c, D_n), p_\pi(\cdot \mid c)\right)\right] \xrightarrow{n\to\infty} 0.$$

Proof. For any two probability distributions p and q, recall that the Hellinger distance satisfies

$$H(p,q)^2 = \frac{1}{2}\int\left(\sqrt{p(\theta)} - \sqrt{q(\theta)}\right)^2 d\theta \le \frac{1}{2}KL(p\|q).$$

Thus, if we can show that the Kullback–Leibler divergence $KL\big(\hat{p}_{\hat{\theta}}(\cdot \mid c, \tilde{D}) \,\|\, p_\pi(\cdot \mid c)\big)$ vanishes asymptotically, then the convergence in Hellinger distance follows. For any n, ϵ, we derive that

$$\begin{aligned}
\mathbb{E}_c\left[H\left(p_{\hat{\theta}}(\cdot \mid c, D_n), p_\pi(\cdot \mid c)\right)\right] &\overset{(1)}{\le} \mathbb{E}_c\left[H\left(p_{\hat{\theta}}(\cdot \mid c, D_n), q_n(\cdot \mid c, D_n)\right)\right] && (13)\\
&\quad + \mathbb{E}_c\left[H\left(q_n(\cdot \mid c, D_n), p_\pi(\cdot \mid c)\right)\right] && (14)\\
&\overset{(2)}{\le} \sqrt{\frac{1}{2}\mathbb{E}_c\left[KL\left(p_{\hat{\theta}}(\cdot \mid c, D_n), q_n(\cdot \mid c, D_n)\right)\right]} && (15)\\
&\quad + \mathbb{E}_c\left[H\left(q_n(\cdot \mid c, D_n), p_\pi(\cdot \mid c)\right)\right] && (16)\\
&\overset{(3)}{\le} \sqrt{\frac{\epsilon}{2}} + \mathbb{E}_c\left[1 - \int_{\mathcal{Y}}\sqrt{\int p_q(y \mid c)p_\pi(y \mid c)\mathrm{d}\Pi^n(q)}\mathrm{d}y\right]^{1/2} && (17)\\
&\le \sqrt{\frac{\epsilon}{2}} + \left[1 - \int_{\mathcal{C}} p_\pi(c)\int_{\mathcal{Y}}\frac{1}{p_\pi(c)}\sqrt{\int q(y,c)p_\pi(y,c)\mathrm{d}\Pi^n(q)}\mathrm{d}y\mathrm{d}c\right]^{1/2} && (18)\\
&\le \sqrt{\frac{\epsilon}{2}} + \left[1 - \int_{\mathcal{C}}\int_{\mathcal{Y}}\int\sqrt{p_q(y,c)p_\pi(y,c)}\mathrm{d}\Pi^n(q)\mathrm{d}y\mathrm{d}c\right]^{1/2} && (19)\\
&= \sqrt{\frac{\epsilon}{2}} + \left[\int H\left(p_q, p_\pi\right)^2 \mathrm{d}\Pi^n(q)\right]^{1/2} && (20)\\
&\le \sqrt{\frac{\epsilon}{2}} + \left[\int H\left(p_q, p_\pi\right) d\Pi^n(q)\right]^{1/2} && (21)\\
&\overset{(4)}{=} \sqrt{\frac{\epsilon}{2}} + \left[\int_{\{q:H(p_\pi,p_q)>\epsilon\}} H\left(p_q, p_\pi\right)\mathrm{d}\Pi^n(q)\right]^{1/2} && (22)\\
&\quad + \left[\int_{\{q:H(p_\pi,p_q)\le\epsilon\}} H\left(p_q, p_\pi\right)\mathrm{d}\Pi^n(q)\right]^{1/2} && (23)\\
&\overset{(5)}{=} \sqrt{\frac{\epsilon}{2}} + \left(\Pi^n(\{q : H(p_\pi, p_q) > \epsilon\}) + \epsilon\right)^{1/2} \to \sqrt{\frac{\epsilon}{2}} + \sqrt{\epsilon} \quad \text{a.s.} && (24)
\end{aligned}$$

The first inequality ($\overset{(1)}{\le}$) is derived from the triangle inequality for the Hellinger distance, which states that for any intermediate distribution $q(\cdot \mid c, D_n)$, we have

$$H\left(p_{\hat{\theta}}(\cdot \mid c, D_n), p_\pi(\cdot \mid c)\right) \le H\left(p_{\hat{\theta}}(\cdot \mid c, D_n), q_n(\cdot \mid c, D_n)\right) + H\left(q_n(\cdot \mid c, D_n), p_\pi(\cdot \mid c)\right).$$

The second inequality ($\overset{(2)}{\leq}$) uses the fact that the Hellinger distance $H(p, q)$ is bounded above by the square root of the KL divergence $KL(p \parallel q)$, such that

$$H(p, q)^2 \leq \frac{1}{2} KL(p \parallel q).$$

Thus, we can bound the Hellinger distance by the KL divergence. In the third inequality ($\overset{(3)}{\leq}$), we make use of the assumption

$$\mathbb{E}_c \left[KL \left(p_{\hat{\theta}}(\cdot \mid c, D_n), q_n(\cdot \mid c, D_n) \right) \right] < \epsilon,$$

and utilize the definition of the Hellinger distance. In ($\overset{(4)}{=}$), we partition the domain into two regions– one where the Hellinger distance $H(p_\pi, p_q)$ exceeds ϵ and another where it is less than or equal to ϵ–and use this partitioning to demonstrate the inequality.

Finally, in ($\overset{(5)}{=}$), by posterior consistency, the region where the Hellinger distance is greater than ϵ vanishes as $n \to \infty$ such that

$$\Pi^n \{q : H(p_\pi, p_q) > \epsilon\} \to 0 \quad \text{almost surely.}$$

Since ϵ is arbitrary, we can conclude that

$$\mathbb{E}_c \left[H \left(p_{\hat{\theta}}(\cdot \mid c, D_n), p_\pi(\cdot \mid c) \right) \right] \xrightarrow{n \to \infty} 0 \quad \text{almost surely.}$$

□

Under its stated assumptions, Theorem A.1 describes approximation of the full posterior predictive distribution as context grows, rather than convergence of the deterministic PDE-PFN implementation. However, PDE-PFN is a deterministic predictor and is trained using the mean squared error loss. Therefore, it does not explicitly model the full posterior predictive distribution or its uncertainty. We now clarify how this deterministic training objective is connected to the posterior predictive mean (PPM): under the squared-loss objective, the population-optimal predictor is precisely the conditional mean of the target solution given the context.

Theorem A.2 (Posterior-predictive-mean inference under squared loss)**.** *Let $q \in \mathcal{Q}$ denote a latent PDE instance drawn from the prior distribution Π. For each q, let $u(q) : \mathcal{X} \times \mathcal{T} \to \mathcal{Y}$ denote the corresponding PDE solution field, where for a spatiotemporal coordinate $c = (x, t)$, the solution value is $y = u(q)(c)$. Let $\widetilde{D}_n = \{(c_i, \tilde{y}_i)\}_{i=1}^n$ be an observed context set, where $\tilde{y}_i = u(q)(c_i) + \eta_{c_i}$ and where η_{c_i} denotes zero-mean observation noise. We assume that the noise variables are independent across spatiotemporal coordinates and independent of the latent PDE instance q.*

For a query coordinate c, let the training target be $\tilde{Y}(c) = u(q)(c) + \eta_c$. Consider any deterministic predictor $\tilde{u}_\theta(c, \widetilde{D}_n)$, corresponding to $\tilde{u}_\theta(c \mid \widetilde{D}_n)$ in PDE-PFN, trained with the mean squared error loss

$$\mathcal{L}(\tilde{u}_\theta) = \mathbb{E} \left[\left\| \tilde{u}_\theta(c, \widetilde{D}_n) - \tilde{Y}(c) \right\|_2^2 \right],$$

where the expectation is taken over $q \sim \Pi$, the context and query. Then the minimizer satisfies

$$\tilde{u}_\theta^\star(c, \widetilde{D}_n) = \mathbb{E}\Big[\tilde{Y}(c) \mid c, \widetilde{D}_n \Big] = \mathbb{E}\Big[u(q)(c) \mid c, \widetilde{D}_n \Big].$$

Proof. For any measurable deterministic predictor $\tilde{u}_\theta$, we decompose the MSE loss by conditioning on $(c, \widetilde{D}_n)$:

$$\mathcal{L}(\tilde{u}_\theta) = \mathbb{E} \left[\mathbb{E} \left[\left\| \tilde{u}_\theta(c, \widetilde{D}_n) - \tilde{Y}(c) \right\|_2^2 \,\middle|\, c, \widetilde{D}_n \right] \right].$$

Using the standard bias-variance decomposition, the inner expectation becomes

$$\mathbb{E} \left[\left\| \tilde{u}_\theta(c, \widetilde{D}_n) - \tilde{Y}(c) \right\|_2^2 \,\middle|\, c, \widetilde{D}_n \right] = \mathrm{Var}\Big(\tilde{Y}(c) \mid c, \widetilde{D}_n \Big) + \left\| \tilde{u}_\theta(c, \widetilde{D}_n) - \mathbb{E}\Big[\tilde{Y}(c) \mid c, \widetilde{D}_n \Big] \right\|_2^2.$$

The variance term does not depend on $\tilde{u}_\theta$. Therefore, the loss is minimized pointwise by

$$\tilde{u}_\theta^\star(c, \widetilde{D}_n) = \mathbb{E}\Big[\tilde{Y}(c) \mid c, \widetilde{D}_n \Big].$$

Since the query noise η_c is zero-mean and independent of the context noise variables, the latent PDE instance, and the sampled coordinates, we have

$$\mathbb{E}[\eta_c \mid c, \widetilde{D}_n] = 0.$$

Therefore,

$$\mathbb{E}\Big[\tilde{Y}(c) \mid c, \widetilde{D}_n\Big] = \mathbb{E}\Big[u(q)(c) + \eta_c \mid c, \widetilde{D}_n\Big] = \mathbb{E}\Big[u(q)(c) \mid c, \widetilde{D}_n\Big].$$

This completes the proof. □

This result does not imply recovery of clean context observations or removal of arbitrary bias or correlated errors in PINN-generated priors.

Corollary A.3 (Effect of context size on the optimal MSE)**.** *Let $\widetilde{D}_n \subset \widetilde{D}_{n+1}$ be nested context sets, and define the population-optimal MSE with n context observations as*

$$\mathcal{L}_n^\star = \inf_{\tilde{u}_\theta} \mathbb{E}\left[\left\|\tilde{u}_\theta(c, \widetilde{D}_n) - \tilde{Y}(c)\right\|_2^2\right].$$

Then,

$$\mathcal{L}_{n+1}^\star \leq \mathcal{L}_n^\star.$$

In other words, the best achievable squared loss is non-increasing as more context observations are provided. Moreover, if the query target is noise-free or if the evaluation is performed against the clean solution $u(q)(c)$, and the latent PDE instance q becomes identifiable from increasingly many context observations, then the corresponding clean-solution prediction error can approach zero as n increases.

Proof. By Theorem A.2, the population-optimal predictor with n context observations is

$$\tilde{u}_{\theta,n}^\star(c, \widetilde{D}_n) = \mathbb{E}\Big[\tilde{Y}(c) \mid c, \widetilde{D}_n\Big],$$

and the corresponding optimal MSE is

$$\mathcal{L}_n^\star = \mathbb{E}\left[\mathrm{Var}\left(\tilde{Y}(c) \mid c, \widetilde{D}_n\right)\right].$$

Since $\widetilde{D}_n \subset \widetilde{D}_{n+1}$, conditioning on $\widetilde{D}_{n+1}$ provides at least as much information as conditioning on $\widetilde{D}_n$. By the law of total variance,

$$\mathbb{E}\left[\mathrm{Var}\left(\tilde{Y}(c) \mid c, \widetilde{D}_n\right)\right] = \mathbb{E}\left[\mathrm{Var}\left(\tilde{Y}(c) \mid c, \widetilde{D}_{n+1}\right)\right] + \mathbb{E}\left[\mathrm{Var}\left(\mathbb{E}[\tilde{Y}(c) \mid c, \widetilde{D}_{n+1}] \mid c, \widetilde{D}_n\right)\right].$$

The second term is non-negative, and therefore

$$\mathcal{L}_{n+1}^\star \leq \mathcal{L}_n^\star.$$

This shows that increasing the number of context observations cannot increase the population-optimal squared loss. □

B Datasets

B.1 Family of 2D Convection-Diffusion-Reaction Equations

The formulation of the family of 2D CDR equations with three reaction terms, used in Section 4.2, is given as follows.

$$\text{2D CDR: } u_t + \boldsymbol{\beta} \cdot \nabla u - \nabla \cdot (\boldsymbol{\nu} \nabla u) - \sum_{j=1}^{3} \rho_j f_j(u) = 0,\ x \in \mathcal{X},\ t \in \mathcal{T}, \tag{25}$$

where $\mathcal{X} = [0, \pi] \times [0, \pi]$ and $\mathcal{T} = [0, 1]$ are solution domain with a resolution of $32 \times 32 \times 21$. For the family of 2D CDR equations, we use analytic solutions whenever they are available. When an analytic solution is not available, the solution is generated using the spectral method, following the same approach as in [7]. The solutions are computed over the time interval $t \in [0.0, 1.0]$ with 1001 time steps at intervals of 0.001, from which 21 time steps at intervals of 0.05 (from $t = 0.0$ to $t = 1.0$) are used in the experiments. We pre-train our model on the family of 2D CDR equations with coefficients $(\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in 0.0, 0.5, 1.0)$. This results in a total of 2,187 unique PDEs. For PINN prior, because a PINN can be trained on only a single PDE at a time, we train an individual PINN for each of the 2,187 distinct 2D CDR equations.

B.2 2D Shallow Water Equation

The 2D shallow water equation (SWE) dataset used in Section 4.3.1 is obtained from PDEBench [47][3]. The SWE is widely used as a benchmark dataset and corresponds to a hyperbolic PDE of the following form:

$$\begin{aligned} \partial_t h + \partial_x(hu) + \partial_y(hv) &= 0, \\ \partial_t(hu) + \partial_x\left(u^2 h + \frac{1}{2} g_r h^2\right) + \partial_y(uvh) &= -g_r h \partial_x b, \\ \partial_t(hv) + \partial_y\left(v^2 h + \frac{1}{2} g_r h^2\right) + \partial_x(uvh) &= -g_r h \partial_y b, \end{aligned} \tag{26}$$

where u and v denote the horizontal and vertical velocities, h is the water depth, and g_r is the gravitational acceleration. The SWE is derived from the general Navier–Stokes equations and is well suited for modeling free-surface flow problems.

The dataset contains simulation results for the water height h and is generated using the PyClaw [22] Python package. It consists of samples produced from 1,000 different initial conditions. The simulation domain is $\mathcal{X} = [-2.5, 2.5]^2$ over the time interval $\mathcal{T} = [0.0, 1.0]$, with a resolution of $128 \times 128 \times 101$. From this, we subsample 21 time steps at intervals of 0.05 for use in our experiments.

B.3 2D Compressible Navier-Stokes Equation

The 2D compressible Navier–Stokes equations dataset, referred to as CNSE, used in Section 4.3.2 was obtained from PDEBench. The Navier–Stokes equation is one of the most widely adopted benchmarks for evaluating SciML PDE solvers, and in this work we focus on the compressible fluid case. The governing equations are given by

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{v}) &= 0, \\ \rho\left(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}\right) &= -\nabla p + \eta \Delta \mathbf{v} + \left(\zeta + \frac{\eta}{3}\right) \nabla(\nabla \cdot \mathbf{v}), \\ \partial_t\left(\epsilon + \frac{1}{2}\rho v^2\right) + \nabla \cdot \left[\left(\epsilon + p + \frac{1}{2}\rho v^2\right)\mathbf{v} - \mathbf{v} \cdot \boldsymbol{\sigma}'\right] &= 0, \end{aligned} \tag{27}$$

where ρ denotes the mass density, $\mathbf{v}$ the velocity, p the gas pressure, $\epsilon = p/(\Gamma - 1)$ the internal energy with $\Gamma = 5/3$, $\boldsymbol{\sigma}'$ the viscous stress tensor, and η, ζ the shear and bulk viscosity, respectively.

Numerical solutions were computed in the spatial domain $\mathcal{X} = [0.0, 1.0]^2$ and temporal domain $\mathcal{T} = [0.0, 1.0]$ using a second-order HLLC scheme [49] with the MUSCL method [52] for the inviscid part, and a central difference scheme for the viscous part. We use the solution computed by setting the Mach number as $M = |\mathbf{v}|/c_s = 1.0$, where $c_s = \Gamma p/\rho$ is the sound velocity, and both viscosity coefficients as $\eta = \zeta = 0.1$. Periodic boundary conditions were applied, and distinct initial conditions were generated from random fields. The dataset contains simulation results for x- and y-velocities $\mathbf{v}$, pressure p, and density ρ. The dataset consists of 1,000 samples simulated from different initial conditions, originally computed on a $128 \times 128 \times 21$ grid. Each feature in the dataset was normalized individually for the experiment. For our experiments, we employed a subsampled version of the dataset with resolution $64 \times 64 \times 5$. In the experiment we consider a field $\mathbf{u}(x,t) = [\mathbf{v}(x,t), p(x,t), \rho(x,t)] \in \mathbb{R}^4$ for $x \in \mathcal{X},\ t \in \mathcal{T}$. Here $\mathbf{u}(x,t)$ denotes a four-dimensional vector field representing four features of the system and the initial condition $\mathbf{u}(x, 0.0) = \mathbf{u}_0(x)$ for $x \in \mathcal{X}$. We are interested in learning the trajectory-predicting operator $\mathcal{G}^\dagger : \mathbf{u}_0(x) \mapsto \big(\mathbf{u}(x, 0.25),\ \mathbf{u}(x, 0.5),\ \mathbf{u}(x, 0.75),\ \mathbf{u}(x, 1.0)\big)$, which maps the initial state $\mathbf{u}_0 \in L^2\big((0.0, 1.0)^2; \mathbb{R}^4\big)$ to the solution at four future time instances.

B.4 Airfoil

The Airfoil dataset used in Section H.7 was obtained from the dataset provided by [26][4]. This dataset was originally generated in [40] by solving the Euler equations for compressible flow using the finite volume method built into the SU2 library [37][5]. The Euler equations have the following formulation:

[3]Dataset source: https://github.com/pdebench/PDEBench

[4]Dataset Source: https://github.com/BaratiLab/OFormer

[5]Dataset and numerical solver source: https://github.com/merantix-momentum/gnn-bvp-solver

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{v}) &= f_1, \\ \partial_t (\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p\mathbb{I}) &= \mathbf{f}_2, \\ \partial_t (\rho E) + \nabla \cdot (\rho E \mathbf{v} + p\mathbf{v}) &= f_3, \\ \rho := \rho(x,t),\ \mathbf{v} := \mathbf{v}(x,t),\ p := p(x,t), x \in \mathcal{X},\ t \in [0,T], \end{aligned} \tag{28}$$

where ρ denotes the density, $\mathbf{v}$ the velocity field, p the pressure, E the total energy per unit mass, and $f_1, \mathbf{f}_2, f_3$ are generic source terms. Li et al. [26] formulated this problem as a non-Markovian initial value problem and used a significantly larger time step size. The dataset spans $t \in \mathcal{T} = [0.0, 4.8]$ with 101 time steps, each spaced by a time interval of 0.24. At each timepoint, the dataset contains values for 5,233 irregular mesh nodes, including node position, node type, velocity, pressure, and density. Similar to CNSE, we consider a field $\mathbf{u}(x,t) = [\mathbf{v}(x,t), p(x,t), \rho(x,t)] \in \mathbb{R}^4$ for $x \in \mathcal{X},\ t \in \mathcal{T}$. Here $\mathbf{u}(x,t)$ denotes a four-dimensional vector field representing four features of the system. The initial condition is given by $\mathbf{u}(x,0) = \mathbf{u}_0(x)$ for $x \in \Omega$. We want to learn the trajectory-predicting operator $\mathcal{G}^\dagger : \mathbf{u}_0(x) \mapsto \big(\mathbf{u}(x,1.2), \mathbf{u}(x,2.4), \mathbf{u}(x,3.6), \mathbf{u}(x,4.8)\big)$, which maps the initial state $\mathbf{u}_0 \in L^2\big(\mathcal{X}; \mathbb{R}^4\big)$ to the solution at four future time instances.

B.5 Darcy Flow

The Darcy Flow dataset used in Section H.8 is also obtained from PDEBench. Darcy Flow is a widely adopted benchmark for evaluating operator learning models, formulated as a time-independent elliptic PDE. The governing equation is given by:

$$\begin{aligned} -\nabla(a(x)\nabla u(x)) &= f(x), \quad x \in \mathcal{X}, \\ u(x) &= 0, \quad x \in \partial\mathcal{X}, \end{aligned} \tag{29}$$

where the source term $f(x)$ is fixed to a constant value of 0.1. The dataset consists of steady-state solutions u corresponding to the viscosity fields a. To obtain these solutions, we simulate the following time-dependent formulation with random initial conditions until convergence to steady state:

$$\partial_t u(x,t) - \nabla\left(a(x)\nabla u(x,t)\right) = f(x), \quad x \in \mathcal{X} \tag{30}$$

Numerical solutions are simulated using a second-order central difference scheme in both time and space. Each data sample is represented as an input–output pair, where the input is a viscosity field $a(x)$ and the output is the corresponding steady-state solution $u(x)$. Both inputs and outputs are discretized on a 128×128 spatial grid. The dataset contains 2,000 samples, each generated from distinct random initial conditions within the simulation domain $\mathcal{X} = [0.0, 1.0]^2$. Unlike the CNSE setting, here we aim to learn the operator mapping a viscosity field $a(x) \in L^2\big((0.0, 1.0)^2; \mathbb{R}\big)$ to the corresponding steady-state solution $u(x)$ for $x \in \mathcal{X}$. Specifically, we want to learn the operator $\mathcal{G}^\dagger : a(x) \mapsto u(x)$.

B.6 3D Diffusion-Reaction Equations

The formulation of the 3D diffusion-reaction equations, used in Section H.10, is given as follows.

$$u_t - \nabla \cdot (\boldsymbol{\nu} \nabla u) - \rho u \cdot (1-u) = 0,\ x \in \mathcal{X},\ t \in \mathcal{T}, \tag{31}$$

where $\mathcal{X} = [0, \pi] \times [0, \pi] \times [0, \pi]$ and $\mathcal{T} = [0, 1]$ are solution domain with a resolution of $16 \times 16 \times 16 \times 21$. For generations, we used the same method and process with the family of 2D CDR equations. We conduct experiment on coefficients $(\nu_x, \nu_y, \nu_z, \rho \in 0.0, 0.5, 1.0, 1.5)$. This results in a total of 256 unique PDEs.

B.7 3D Gray-Scott

The 3D Gray-Scott equations dataset used in Appendix H.10 is obtained from APEBench [24], a benchmark suite for autoregressive neural emulators of PDEs. In APEBench, 3D Gray-Scott is categorized as a reaction-diffusion system and is provided as a multi-channel PDE. The governing equations are given by

$$\begin{aligned} \frac{\partial u_0}{\partial t} &= \nu_0 \nabla \cdot \nabla u_0 - u_0 u_1^2 + f(1-u_0), \\ \frac{\partial u_1}{\partial t} &= \nu_1 \nabla \cdot \nabla u_1 + u_0 u_1^2 - (f+k) u_1, \end{aligned} \tag{32}$$

where u_0 and u_1 denote the two chemical species, ν_0 and ν_1 are the diffusion coefficients, and f and k are the feed and kill rates, respectively. APEBench includes Gray-Scott among its supported semi-linear reaction-diffusion equations and uses a Fourier pseudo-spectral ETDRK solver for procedural trajectory generation. Following the APEBench setup, the Gray-Scott dynamics are generated in the physical parameterization. In particular, the default Gray-Scott configuration uses diffusion coefficients $\nu_0 = 2 \times 10^{-5}$ and $\nu_1 = 1 \times 10^{-5}$. For the Gray-Scott pattern type considered in APEBench, the feed and kill rates are selected according to the predefined pattern regimes introduced by Pearson [39]. Among those regimes, we only consider the case with $f = 0.04$ and $k = 0.06$. APEBench further notes that the 3D Gray-Scott experiments use a Gaussian-blob-type initialization for each channel. Both inputs and outputs are discretized on a $32 \times 32 \times 32$ spatial grid, and the dataset spans $t \in T = [0.0, 1000.0]$ with 51 time steps for training and $t \in T = [0.0, 4000.0]$ with 201 time steps for validation and testing, each spaced by a time interval of 20. We subsample the dataset with five time intervals, i.e., $\Delta t = 100$.

For our experiments, we use the 3D Gray-Scott dataset to evaluate operator learning in a higher-dimensional complex system. Let $u(x,t) = [u_0(x,t), u_1(x,t)] \in \mathbb{R}^2$ for $x \in \mathcal{X}$ and $t \in \mathcal{T}$ denote the two-channel state variable. Given the state at a current time step, the task is to predict the system state 5 time steps ahead. Thus, we learn the time-advancing operator $\mathcal{G}^\dagger : u(x,t) \mapsto u(x, t + \Delta t)$, which maps the current two-species concentration field to the future state after five time steps.

The specific information about PDE types used in each experiment is described in Table 6. Moreover, the detailed values regarding how many training, validation, and testing data points in each experiment are provided in Appendix J. For the spatiotemporal interpolation with unseen coefficients in the family of 2D CDR equation experiments, we use the same time range as in the corresponding task with seen coefficients.

Table 6: Time range of data points used for training, validation, and testing in all experiments.

PDE (Section)	**Task**	**Train**	**Valid**		**Test**	
			Context	**Query**	**Context**	**Query**
Family of 2D CDR equations (Section 4.2)	Spatiotemporal interpolation	$t \in [0.0, 0.5]$	predefined in $t \in [0.0, 0.5]$	grid at $t = 0.25$	predefined in $t \in [0.0, 0.5]$	grids at $t \in \{0.05, 0.15, 0.35, 0.45\}$
	Temporal extrapolation	$t \in [0.0, 0.5]$	grid at $t = 0.5$	grid at $t = 0.6$	grid at $t = 0.6$	grids at $t \in [0.6, 1.0]$
	Unseen coeff temporal extrapolation	-	-	-	grid at $t = 0.1$	grids at $t \in [0.2, 0.5]$
2D SWE (Section 4.3.1)	Spatiotemporal interpolation	$t \in [0.0, 0.5]$	predefined in $t \in [0.0, 0.5]$	grid at $t = 0.25$	predefined in $t \in [0.0, 0.5]$	grids at $t \in \{0.05, 0.15, 0.35, 0.45\}$
	Temporal extrapolation	$t \in [0.0, 0.5]$	grid at $t = 0.5$	grid at $t = 0.6$	grid at $t = 0.6$	grids at $t \in [0.6, 1.0]$
2D CNSE (Section 4.3.2)	Operator learning	$t \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$	grid at $t = 0.0$	grids at $t \in \{0.25, 0.5, 0.75, 1.0\}$	grid at $t = 0.0$	grids at $t \in \{0.25, 0.5, 0.75, 1.0\}$
Airfoil (Section H.7)	Operator learning	$t \in \{0.0, 1.2, 2.4, 3.6, 4.8\}$	grid at $t = 0.0$	grids at $t \in \{1.2, 2.4, 3.6, 4.8\}$	grid at $t = 0.0$	grids at $t \in \{1.2, 2.4, 3.6, 4.8\}$
Darcy Flow (Section H.8)	Operator learning	$a \mapsto u$				
3D diffusion-reaction equations (Section H.9)	Spatiotemporal interpolation	$t \in [0.0, 0.5]$	predefined in $t \in [0.0, 0.5]$	grid at $t = 0.25$	predefined in $t \in [0.0, 0.5]$	grids at $t \in \{0.05, 0.15, 0.35, 0.45\}$
	Temporal extrapolation	$t \in [0.0, 0.5]$	grid at $t = 0.5$	grid at $t = 0.6$	grid at $t = 0.6$	grids at $t \in [0.6, 1.0]$
3D Gray-Scott (Section H.10)	Operator learning	$t \in \{0, 100, \ldots, 1000\}$	grid at $t \in \{1000, \ldots, 3900\}$	grids at $t \in \{1100, \ldots, 4000\}$	grid at $t \in \{1000, \ldots, 3900\}$	grids at $t \in \{1100, \ldots, 4000\}$

C PINN Used in Prior Generation

In this study, we utilize the PINN introduced by Raissi et al. [42] to generate PINN priors. The loss function employed during the training of the PINN is as follows:

$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_f + \mathcal{L}_b, \tag{33}$$

where $\mathcal{L}_u, \mathcal{L}_f$ and $\mathcal{L}_b$ are defined as

$$\mathcal{L}_u = \frac{1}{N_u} \sum \left(\tilde{u}(x,0) - u(x,0)\right)^2, \quad \mathcal{L}_f = \frac{1}{N_f} \sum \left(\mathcal{N}(t,x,u,\alpha)\right)^2, \quad \mathcal{L}_b = \frac{1}{N_b} \sum \left(\tilde{u}(0,t) - \tilde{u}(2\pi,t)\right)^2, \tag{34}$$

for N_u points at initial condition, N_f collocation points, and N_b boundary points.

The generation time for the PINN prior varies depending on the number of training epochs and the PINN loss threshold. Based on the configuration used in our experiments, it took approximately 373 seconds to generate the prior for a single coefficient combination of the 2D CDR and utilized up to 4,552 MB of memory.

D Detailed Description of the Baselines

We provided descriptions of each baseline discussed in Section 4.1.

- DeepONet is a neural operator architecture designed to learn operators mapping input functions to output functions. It combines branch and trunk networks to predict values in a function space.
- FNO learns solution operators for partial differential equations (PDEs) using the Fourier transform. By mapping inputs to a frequency domain, FNO captures complex patterns and long-range dependencies and models complex systems.
- F-FNO extends the Fourier Neural Operator by factorizing its layers to reduce computational costs. This factorization enables efficient learning of solution operators for complex systems.
- A-FNO is a variant of the Fourier Neural Operator that dynamically adjusts the resolution of the frequency domain during training. This adaptation aims to capture relevant features across scales, enabling more flexible modeling of complex systems.
- Poseidon is a multiscale operator transformer architecture designed to model solution operators for PDEs with continuous-in-time capability. It introduces time-conditioned layer normalization and processes multi-resolution input features to capture complex spatiotemporal dynamics.
- DPOT is a scalable neural operator model designed for PDEs, featuring a Fourier attention mechanism that enables efficient learning of complex spatiotemporal dynamics. Its architecture incorporates an auto-regressive denoising pre-training strategy, facilitating stable and flexible modeling across diverse PDE datasets with varying resolutions and geometries.

In the spatiotemporal interpolation task, models other than DeepONet are excluded from comparison because they cannot process mesh-structured inputs. DeepONet requires input data from fixed positions for each dataset; therefore, for a fair comparison, our model is also provided with context data from the same positions during evaluation. For Poseidon and DPOT, which are scientific foundation models providing pre-trained weights, we utilized their released weights and performed fine-tuning. For a fair comparison, we trained them and our model using only half the number of epochs compared to the other models. Furthermore, computational costs, memory usage, experiment settings, and hyperparameters are comprehensively outlined in Appendix J.

Table 7: Major comparisons between baselines and our model on the shape of input and target data (context and query in our model). Neural operator baselines (excluding DeepONet) can only predict targets at predetermined locations and are limited to processing grid-structured input data. DeepONet, on the other hand, can handle inputs of various shapes but requires fixed-coordinate input values, though it can produce target solutions at desired coordinates. Our model accepts variable-sized coordinate–value context sets and supports coordinate queries, although accuracy depends on prior coverage and context informativeness.

Data	**FNO & F-FNO & A-FNO & Poseidon & DPOT**	**LNO**	**DeepONet**	**Ours**
Input data	Predefined grid	Predefined mesh	Predefined mesh	**Arbitrary Mesh**
Target data	Predefined grid	Predefined mesh	**Arbitrary Mesh**	**Arbitrary Mesh**

E Model Architecture

Our model is fundamentally based on the Prior-Fitted Network (PFN) architecture [33], and consists of three main components: an encoder, a Transformer block, and a decoder. In this section, we

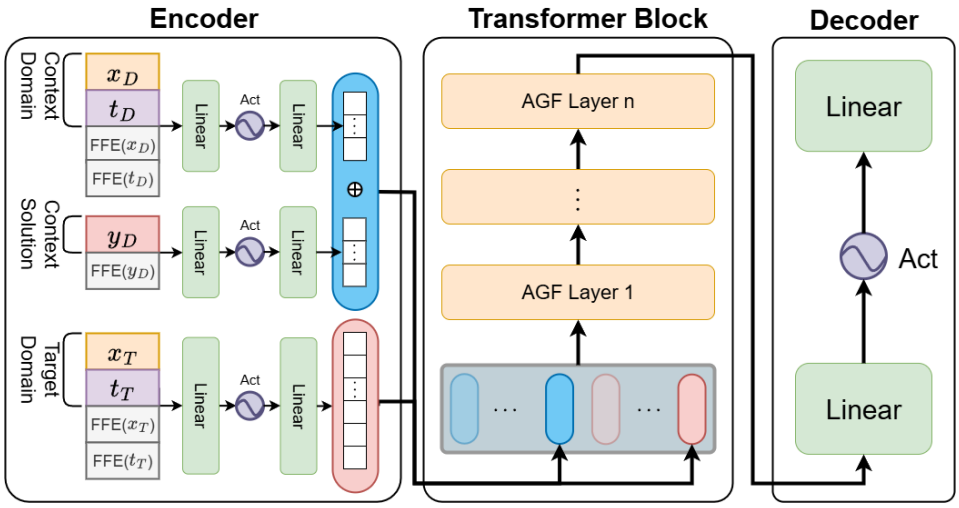

Figure 3: A diagram of PDE-PFN's architecture. The diagram is presented as an extension of Figure 2. The model consists of three main components: an encoder, a Transformer block, and a decoder. FFE denotes the Fourier feature embedding, and Act denotes the activation function.

describe the encoder. For clarity, we denote the training contexts and queries as D and T, respectively, consistent with the main text. Each input in D and T is composed of spatial coordinates x, temporal coordinates t, and solution values y; for queries T, the solution values are masked. The encoder optionally augments the raw inputs with Fourier features before MLP processing; FFE dimension zero retains only the raw inputs.

Encoder The role of FFE is to augment the raw inputs with high-frequency components, enabling the model to better capture complex solution patterns. This advantage of using FFE in conjunction with MLPs has been verified in [48, 29]. For grid-based inputs, we can employ a discrete Fourier transform via fast Fourier transform (FFT). For coordinate-based inputs, we use sinusoidal feature mappings with predetermined frequencies. Specifically, given an input $v \in \mathbb{R}$, we define the embedding as

$$\text{FFE}(v; n) = [sin(v \cdot \omega_1), cos(v \cdot \omega_1), \cdots, sin(v \cdot \omega_n), cos(v \cdot \omega_n)], \tag{35}$$

where the frequency coefficients are set as $\omega_i = 2\pi/i$. After applying FFE, the encoder constructs three types of feature vectors: $l_{CD} := [x_D, t_D, \text{FFE}(x_D), \text{FFE}(t_D)]$ for context domain information, $l_{CS} := [y_D, \text{FFE}(y_D)]$ for context solutions, and $l_{QD} := [x_T, t_T, \text{FFE}(x_T), \text{FFE}(t_T)]$ for query domain information, by concatenating the raw inputs and FFE results. Each of these vectors is passed through a separate MLP to learn PDE-specific embeddings. Each MLP consists of two linear layers and a single activation function between linear layers. For the activation, we employ a rational activation function [32], which has an adaptive nature and has been shown to provide greater flexibility compared to conventional nonlinearities. Finally, l_{CD} and l_{CS} are concatenated after passing each MLP to form the context representation used by the Transformer block.

Transformer block The encoder output is then processed through a Transformer block, which consists of a sequence of Transformer layers. Instead of the *vanilla* Transformer layer, we adopt the attentive graph filter (AGF) layer [58], which provides greater flexibility while reducing computational complexity. The AGF layer enhances standard linear attention by learning attention in the singular value domain. Specifically, it decomposes the attention operation into a set of spectral components and adaptively learns their importance, enabling the model to capture the dependencies more effectively. Specifically, the attention matrix $A \in \mathbb{R}^{n \times m}$ can be decomposed using singular value decomposition (SVD) as $A = U\Sigma V^T$, where U and V are orthogonal matrices, and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ contains the singular values. Therefore, as discussed in Section 3, when using the AGF layer, we include a regularization term to ensure the orthogonality of the matrices U and V. The regularization

term is defined as follows:

$$\mathcal{L}_{\text{AGF}} = \frac{1}{n^2}\left(\left\|U^\top U - \mathbf{I}\right\|\right) + \frac{1}{m^2}\left(\left\|VV^\top - \mathbf{I}\right\|\right),$$

where $\mathbf{I}$ is identity matrix. AGF learns to adaptively reweight these spectral components, effectively filtering the most informative modes while suppressing noise or redundant information. By learning attention in the singular value domain, AGF provides a more powerful representation of global dependencies compared to conventional linear attention, while avoiding the quadratic complexity of the *vanilla* Transformers.

To further enrich the model capacity, we separate the parameters used for self-attention and cross-attention within each layer. As illustrated in Figure 2, self-attention is applied among the contexts D, while cross-attention allows queries T to attend to contexts D. In the figure, self-attention and cross-attention are represented by blue and red arrows, respectively. The numbers of blue rods D and red rods T are described in Table 30 in Appendix J.

Decoder The output of the Transformer block is finally passed through a decoder, implemented as a simple MLP. Similar to the encoder, this MLP consists of two linear layers and a single activation function between linear layers. The decoder maps the latent representations produced by the Transformer into the predicted solution values at the queried coordinates.

F Training

Unlike other task-specific models, our model is trained in an integrated manner, enabling it to solve multiple tasks simultaneously in that dataset. For cases where both spatiotemporal interpolation and temporal extrapolation tasks must be evaluated, such as in the family of 2D CDR equations and SWE datasets, we perform training with temporal extrapolation as the primary objective. In this setup, other baselines treat grid data points at t_n and t_{n+1} as fixed input–output pairs. In contrast, our model constructs pairs by randomly mixing the data points within each (t_n, t_{n+1}) pair independently, rather than preserving their exact correspondence. This strategy enables the model to learn in a way that supports both spatiotemporal interpolation and temporal extrapolation within a unified framework. For the CNSE, Airfoil, and Darcy Flow datasets, where we conducted experiments only on operator learning, random mixing was not applied.

Training algorithm We train our model as follows:

Algorithm 1 Training our model

Input: contexts D and queries T from prior $p(\mathcal{D})$ in dataset
Output: Our model $\tilde{u}_\theta$ which can approximate the posterior predictive mean
Initialize the model $\tilde{u}_\theta$
for $i = 1$ to n **do**
  Sample $\boldsymbol{\alpha} \in \Omega$ and $D, T \subseteq \tilde{u}(\boldsymbol{\alpha}) \sim p(\mathcal{D})$ ($D := \{(x_D^{(i)}, t_D^{(i)})\}_{i=1}^{N_D}, T := \{(x_T^{(j)}, t_T^{(j)})\}_{j=1}^{N_T}$)
  if dataset $\in$ {Family of 2D CDR, SWE} **then**
    Regenerate D' and T' by shuffling data points in $D \cup T$ independently within each (t_n, t_{n+1}) pair, while preserving $|D'| = N_D$ and $|T'| = N_T$
  end if
  Compute MSE loss $\mathcal{L}_{\boldsymbol{\alpha}} = \frac{1}{N_T}\sum_{j=1}^{N_T}\left\{\tilde{u}_\theta(x_T^{(j)}, t_T^{(j)}|D'_n) - \tilde{u}(x_T^{(j)}, t_T^{(j)})\right\}^2$
  Compute AGF regularization term $\mathcal{L}_{AGF}$ and objective function $\mathcal{L} = \mathcal{L}_{\boldsymbol{\alpha}} + \frac{1}{N_D}\mathcal{L}_{AGF}$
  Update parameters θ with an AdamW optimizer
end for

G Evaluation

We employ L_1 absolute, L_2 relative, and L_∞ relative errors between the model's prediction for test queries and the ground truth. The errors are then averaged over the target parameter space or test

dataset. Given the true solution $y_{\alpha,k}$ and the corresponding prediction $\hat{y}_{\alpha,k}$ at the k-th time point out of a total of K evaluation time points, each metric is computed as follows:

$$L_p \text{ abs error} = \frac{1}{|\Omega| \cdot K} \sum_{\alpha \in \Omega} \sum_{k=1}^{K} ||y_{\alpha,k} - \hat{y}_{\alpha,k}||_p \ , \ L_p \text{ rel error} = \frac{1}{|\Omega| \cdot K} \sum_{\alpha \in \Omega} \sum_{k=1}^{K} \frac{||y_{\alpha,k} - \hat{y}_{\alpha,k}||_p}{||y_{\alpha,k}||_p}. \tag{36}$$

H Additional Experiments

H.1 2D CDR Equations with systematically Noisy Prior Data

In this experiment, we examine whether our model continues to perform well when the prior data is corrupted by systematic synthetic noise instead of being generated by PINNs. Specifically, we construct noisy prior data by injecting either Gaussian noise or uniform noise into the numerical solutions of the family of 2D CDR equations. For the Gaussian-noise setting, we add zero-mean Gaussian noise from $\mathcal{N}(0.0, 0.02)$, and for the uniform-noise setting, we add noise sampled from the uniform distribution $\mathcal{U}([-0.01, 0.01])$. We then train our model on these noisy priors and evaluate it using the same procedure as in the main experiments.

Table 8: The evaluation results of our model trained using systematically noisy data for the spatiotemporal interpolation and temporal extrapolation tasks applied to the family of 2D CDR equations. The text in parentheses following "Ours" indicates the noise type and its standard deviation or range. They are measured at the seen coefficients ($\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\}$). The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours (PINN)	Ours (Gaussian 0.02)	Ours (Uniform $[-0.01, 0.01]$)
Spatiotemporal interpolation	L_1 Abs	0.01218	0.01507	0.01244	**0.01060**
	L_2 Rel	0.01643	0.02057	0.01670	**0.01450**
	L_∞ Rel	0.04600	0.06584	0.04659	**0.04269**
Temporal extrapolation	L_1 Abs	**0.01474**	0.01940	0.01562	0.01513
	L_2 Rel	**0.02261**	0.02705	0.02383	0.02320
	L_∞ Rel	0.08444	0.08802	0.08750	**0.08213**

Table 8 shows that predictive accuracy is largely maintained under the evaluated zero-mean Gaussian and uniform prior corruptions. In the spatiotemporal interpolation task, the model trained with uniform noise achieves the best performance on all three metrics, even outperforming the version trained on exact numerical priors. In the temporal extrapolation task, the model trained with uniform noise achieves the best performance on L_∞ relative error and the second best performance on two metrics. The model trained with Gaussian noise also remains competitive across all metrics. These results support robustness in the tested noise regimes, without establishing robustness to arbitrary biased, correlated, or localized prior errors.

H.2 Test Time Evaluation Given PINN Prior in the Family of 2D CDR Equations

As an additional experiment on the family of 2D CDR equations, we modify the test procedure from Section 4.2.1. While keeping all other experimental settings the same, we compare the evaluation results when noisy PINN priors are provided as input during testing. This experiment is designed to assess the robustness of the models in producing accurate solutions despite noisy inputs.

As shown in the experimental results presented in Table 9, Ours achieves the best performance, demonstrating strong robustness. For Ours (PINN), however, the performance in terms of two relative errors on the spatiotemporal interpolation task is inferior to that of DeepONet. This can be attributed to the fact that Ours (PINN) is pre-trained to predict solutions from PINN priors as inputs, which may cause the model to generate outputs closer to the PINN prior rather than the analytic solution when the prior is provided at test time. Nevertheless, since Ours (PINN) still achieves better performance in terms of L_1 absolute error, we can conclude that it also exhibits a certain degree of robustness.

Table 9: The evaluation results for the spatiotemporal interpolation and temporal extrapolation task applied to the family of 2D CDR equations given a noisy PINN prior at evaluation. They are measured at the seen coefficients ($\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\}$). The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Spatiotemporal interpolation	L_1 Abs	**0.01953**	0.02445	0.02540	-	-	-	-	-	
	L_2 Rel	**0.02755**	0.03467	0.03413	-	-	-	-	-	-
	L_∞ Rel	0.10158	0.10716	**0.09286**	-	-	-	-	-	-
Temporal extrapolation	L_1 Abs	**0.01655**	0.02099	0.06888	0.03458	0.02816	0.05794	0.03048	0.14018	0.05051
	L_2 Rel	**0.02486**	0.02893	0.08591	0.04268	0.03582	0.07903	0.03907	0.15380	0.08624
	L_∞ Rel	0.08830	**0.09333**	0.23386	0.10203	0.10677	0.28805	0.11730	0.28481	0.08624

H.3 Test-Time Prediction with Biased Context Noise in the 2D CDR Family

Since Theorem A.2 assumes unbiased noise, we further investigate how robust our model remains when the inference stage includes biased Gaussian noise added to the analytic solution. We set the standard deviation of Gaussian noise to $\sigma = 0.01$ and vary only the magnitude of the bias. The evaluation is performed on the temporal extrapolation task of the 2D CDR equations, using the L_2 relative error for comparison.

Table 10: The evaluation results for the temporal extrapolation tasks applied to the family of 2D CDR equations. They are measured at the seen coefficients ($\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\}$).

Bias	No noise	0.1%	0.5%	1.0%	1.5%	2.0%	2.5%	3.0%	3.5%	4.0%
L_1 Abs	0.01474	0.01483	0.01472	0.01481	0.01514	0.01572	0.01664	0.01800	0.01985	0.02212
L_2 Rel	0.02261	0.02271	0.02246	0.02244	0.02271	0.02330	0.02255	0.02573	0.02772	0.03009
L_∞ Rel	0.08444	0.08556	0.08459	0.08426	0.08490	0.08646	0.08901	0.09265	0.09733	0.10276

The experimental results in Table 10 show that when the bias is below 3%, the performance degradation remains relatively minor compared to the bias level. However, once the bias reaches approximately 3% or higher, the error begins to increase more noticeably. This indicates that PDE-PFN maintains robustness against moderate systematic bias, but large bias eventually leads to a degradation in accuracy as expected.

H.4 Joint Coefficient and Temporal Extrapolation in the 2D CDR Family

In the experiments on unseen coefficients presented in the main text, the spatiotemporal domain was restricted to the temporal range used during training. This design choice ensured that only the parameter domain included unseen values, preventing both parameter and temporal extrapolation from occurring simultaneously. In this section, we extend the evaluation to a more challenging setting in which both the parameter domain and the temporal domain involve unseen ranges. The unseen coefficient ranges follow those used in Section 4.2.2, and for the temporal domain we adopt the evaluation range from the temporal extrapolation task in Section 4.2.1.

Table 11: The evaluation results for the temporal extrapolation tasks applied to the family of 2D CDR equations. They are measured at the unseen coefficients and unseen temporal domain. The best performance is marked in **bold** and the second-best performance is marked with underline.

Coefficients	Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Interpolation	Temporal extrapolation	L_1 Abs	**0.00736**	0.00939	0.05139	0.10863	0.01405	0.04695	0.01839	0.13852	0.04610
		L_2 Rel	**0.00954**	0.01178	0.06179	0.14380	0.01636	0.06227	0.02049	0.14993	0.04676
		L_∞ Rel	**0.03193**	0.03693	0.16008	0.34858	0.03847	0.21750	0.04512	0.26293	0.04676
Extrapolation	Temporal extrapolation	L_1 Abs	**0.01027**	0.02063	0.05514	0.03887	0.02312	0.05013	0.02560	0.14661	0.05186
		L_2 Rel	**0.01068**	0.02169	0.06450	0.05114	0.02470	0.06351	0.02677	0.15650	0.05189
		L_∞ Rel	**0.01661**	0.03474	0.16555	0.13713	0.04591	0.20955	0.04911	0.27502	0.05189

The experimental results in Table 11 show that our model continues to achieve the best performance. In contrast, the baselines experience greater difficulty when both types of extrapolation are required simultaneously, performing noticeably worse than our model in this setting.

H.5 2D CDR Equations with Different Initial Condition

In this experiment, we examine whether our model continues to perform well on the 2D CDR equations from the main paper when the dataset is generated using a different initial condition. Specifically, we replace the original initial condition, $1 + \sin(2x)\sin(2y)$, with $1 + \sin(3x)\sin(3y)$ to construct a new 2D CDR equation dataset. We then conduct training and evaluation following the same procedure used in the main experiments.

Table 12: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to the new family of 2D CDR equations. They are measured in the same way of Section 4.2.1. The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Spatiotemporal interpolation	L_1 Abs	**0.01339**	0.07394	-	-	-	-	-	-
	L_2 Rel	**0.01885**	0.09376						
	L_∞ Rel	**0.06606**	0.26320						
Temporal extrapolation	L_1 Abs	**0.02017**	0.03988	0.03184	0.03371	0.08342	0.04696	0.09397	0.12628
	L_2 Rel	**0.02929**	0.05282	0.04160	0.04192	0.09967	0.06282	0.10352	0.13633
	L_∞ Rel	**0.10793**	0.17487	0.12671	0.10980	0.27716	0.18313	0.23855	0.13633

The experimental results in Table 12 show that our model continues to achieve the best performance even when evaluated on datasets generated with a new initial condition. This confirms that our model does not merely perform well on a single fixed initial condition, but instead generalizes robustly across different initial conditions. Furthermore, the experiments on the SWE and CNSE datasets in the main paper also support this conclusion, as both datasets consist of samples generated from varying initial conditions under the same underlying PDE. Thus, our model is not limited to datasets constructed with a single, unified initial condition.

H.6 2D CDR Equations with Random Initial Condition Sampled from Uniform Distribution

In this experiment, we further examine whether our model continues to perform well on the family of 2D CDR equations when the initial condition is no longer fixed and instead varies randomly at the pixel level. Specifically, for each PDE instance, we generate the initial condition by sampling each spatial pixel independently from a uniform distribution. We then conduct training and evaluation following the same procedure used in the main experiments.

Table 13: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to the family of 2D CDR equations with random initial condition sampled from uniform distribution. They are measured at the seen coefficients ($\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\}$). The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
Spatiotemporal interpolation	L_1 Abs	0.1599	0.1513	**0.1401**	-	-	-	-	-	-
	L_2 Rel	0.2005	**0.1884**	0.2177						
	L_∞ Rel	0.4253	**0.3874**	0.6944						
Temporal extrapolation	L_1 Abs	**0.0432**	0.0553	0.0788	0.0492	0.0449	0.0710	0.0480	0.1581	0.0720
	L_2 Rel	**0.0578**	0.0719	0.0985	0.0641	**0.0578**	0.0940	0.0603	0.1766	0.2117
	L_∞ Rel	**0.1588**	0.1667	0.2706	0.1750	0.1591	0.3008	0.1631	0.3449	0.2117

The experimental results in Table 13 show that our model continues to achieve strong performance even when the initial condition is randomly sampled from a uniform distribution at the pixel level. In the spatiotemporal interpolation task, Ours (PINN) achieves the best performance on two metrics, while Ours achieves the second-best performance on two of them. In the temporal extrapolation task, Ours achieves the best performance on the three metrics. These results indicate that PDE-PFN remains robust even when the initial condition varies substantially across samples and that its performance is not dependent on a fixed or smooth initial-condition structure.

H.7 Operator Learning in Airfoil

To evaluate input flexibility in operator learning, we conduct experiments on the Airfoil dataset defined on an irregular mesh structure. This dataset consists of 2,000 samples, of which 700 are used

for training, 100 for validation, and 200 for testing. The target operator is formulated similarly to that in Section 4.3.2: given the initial condition, the model is required to predict the solutions at four time steps, $t = 1.2, 2.4, 3.6$, and 4.8 (see Appendix B.4 for a formal description). Most of the baselines used in Section 4 rely on data defined on a regular grid, so that cannot be applied here, since the Airfoil dataset is defined on an irregular mesh. Consequently, among the existing baselines, we only use DeepONet and additionally include Oformer [26] as a new baseline.

Table 14: The evaluation results for the operator learning task applied to the Airfoil dataset. The best performance is marked in **bold** and the second-best performance is marked with underline.

Metric	Ours	Ours (PINN)	DeepONet	Oformer
L_1 Abs	0.11047	**0.10755**	0.23160	0.12039
L_2 Rel	0.18831	**0.18357**	0.44101	0.19744
L_∞ Rel	0.33584	0.32887	0.73076	**0.32318**

The difference from the experimental results reported in the original Oformer paper arises because their setting involved using the first four time steps as input to predict subsequent time steps, whereas our experiment is formulated based on the initial condition only. The experimental results in Table 14 show that both versions of our model outperform the baselines on two metrics. These results demonstrate that our model's ICL capability effectively extends to irregular mesh data and can generalize well to new PDE problems. Detailed information on the best hyperparameter settings can be found in Appendix J.4.

H.8 Operator Learning in Darcy Flow

To further evaluate the PDE generalization ability of our model in operator learning, we conducted additional experiments on the Darcy Flow dataset. This dataset consists of 2,000 samples, with 1,400 used for training, 200 for validation, and 400 for testing. The operator learning task is defined as predicting the steady-state solution $u(x)$ from a given viscosity field $a(x)$ (see Appendix B.5 for a formal description). Unlike the CNSE dataset, where both context and query represent the same feature, the Darcy Flow dataset involves mapping between different quantities. This makes it a distinct form of operator learning task.

Table 15: The evaluation results for the operator learning task applied to Darcy Flow. The best performance is marked in **bold** and the second-best performance is marked with underline.

Metric	Ours	Ours (PINN)	DeepONet	FNO	F-FNO	A-FNO	LNO	Poseidon	DPOT
L_1 Abs	0.00796	0.00742	0.00996	0.00792	0.00806	0.00906	0.01413	0.00998	**0.00726**
L_2 Rel	0.28771	**0.28091**	0.40759	0.29292	0.34149	0.40357	0.51778	0.41001	0.51505
L_∞ Rel	**0.39694**	0.40124	0.59057	0.46398	0.59381	0.80674	0.61245	0.52933	0.51505

The experimental results in Table 15 show that both versions of our model outperform the baselines on two metrics. These findings further confirm that our model maintains strong PDE generalization ability even when the operator learning task requires transferring between different physical quantities. Detailed information on the best hyperparameter settings can be found in Appendix J.4.

H.9 Spatiotemporal Interpolation and Temporal extrapolation in 3D Diffusion-Reaction equations

To further explore the potential of our model for 3D PDEs, we conduct additional experiments on 3D diffusion–reaction equations. We construct a 3D diffusion–reaction equation dataset by extending the same procedure used for generating the 2D CDR equations. This formulation includes three diffusion coefficients (ν_x, ν_y, ν_z) and one reaction coefficient ρ, each sampled from $0.0, 0.5, 1.0, 1.5$, resulting in 256 distinct equations. We perform experiments on the same two tasks as in the 2D CDR equations—spatiotemporal interpolation and temporal extrapolation—using DeepONet, FNO, and DPOT as baselines, since only they can handle 3D datasets among baselines.

The experimental results in Table 16 show that our model outperforms the baselines in spatiotemporal interpolation in all metrics and in temporal extrapolation on two metrics, demonstrating its potential to generalize effectively to 3D datasets.

Table 16: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to the 3D diffusion reaction equations. They are measured at the seen coefficients ($\nu_x, \nu_y, \nu_z, \rho_1 \in \{0.0, 0.5, 1.0, 1.5\}$). The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours (PINN)	DeepONet	FNO	DPOT
Spatiotemporal interpolation	L_1 Abs	**0.02685**	0.02843	0.02816	-	-
	L_2 Rel	**0.03711**	0.04144	0.03981		
	L_∞ Rel	**0.10938**	0.12823	0.12724		
Temporal extrapolation	L_1 Abs	**0.01031**	0.01039	0.02421	0.01521	0.01485
	L_2 Rel	**0.01376**	0.01391	0.03033	0.01824	0.01638
	L_∞ Rel	0.04180	0.04545	0.11530	0.05545	**0.01638**

H.10 Operator Learning in 3D Gray-Scott equation

To further evaluate the PDE generalization ability of our model in a more challenging 3D setting, we conduct additional experiments on the 3D Gray-Scott equations from APEBench [24]. In this experiment, we focus on the operator learning task, where the models are required to learn the one-step time-advancing operator of the system. Specifically, let $u(x,t) = [u_0(x,t), u_1(x,t)] \in \mathbb{R}^2$ denote the two-channel state of the Gray-Scott system at spatial coordinate x. Given the state $u(x,t)$ at a current time step t, the model is asked to predict the state at the next time step $u(x, t+\Delta t)$. We compare our method with DeepONet, FNO, and DPOT, which are widely used baselines for operator learning, and additionally report the performance of our model trained with PINN priors.

Table 17: The evaluation results for the operator learning task in 3D Gray-Scott equation. The best performance is marked in **bold** and the second-best performance is marked with underline.

Metric	Ours	Ours (PINN)	DeepONet	FNO	DPOT
L_1 Abs	**0.00152**	0.00177	0.01622	0.00558	0.01458
L_2 Rel	**0.02626**	0.03004	0.24546	0.08019	0.22860
L_∞ Rel	**0.07482**	0.10394	0.57927	0.19076	0.26938

The experimental results in Table 17 show that both versions of our model substantially outperform all baselines across all evaluation metrics. In particular, Ours achieves the best performance on all three metrics, while Ours (PINN) consistently achieves the second-best performance. These results demonstrate that PDE-PFN can generalize effectively to a complex 3D system and can learn the corresponding operator with high accuracy even when trained with noisy PINN priors. The strong performance gap over the baselines further supports the effectiveness of our in-context learning framework for higher-dimensional PDE problems.

H.11 Sensitivity Analysis to the Number of Test Context Points in the Family of 2D CDR Equation

In this section, we empirically examine how PDE-PFN behaves as the number of test context $\overline{D}$ observations increases. Specifically, we evaluate the sensitivity of the neural network's posterior predictive mean to the number of the test context set $\overline{D}$ provided during the test process.

To this end, we conduct an evaluation of spatiotemporal interpolation on the family of 2D CDR equations experiment by varying the number of $\overline{D}$ while keeping other factors unchanged, including the best hyperparameter settings. The results, depicted in Figure 4, clearly demonstrate that as the size of $\overline{D}$ increases, the error consistently decreases across all three evaluation metrics. This trend is qualitatively consistent with Corollary A.3, which concerns non-increasing population-optimal MSE for nested context sets rather than guaranteed monotonic improvement of a finite trained model.

These empirical findings are consistent with Corollary A.3, which states that the population-optimal MSE is non-increasing as more context observations are provided. We emphasize that this experiment does not verify convergence of the full posterior distribution. Rather, it shows that PDE-PFN benefits from more informative context sets, as expected from the PPM interpretation under squared loss. Overall, the decreasing error trend suggests that PDE-PFN effectively benefits from more informative context sets, consistent with the PPM interpretation under squared loss.

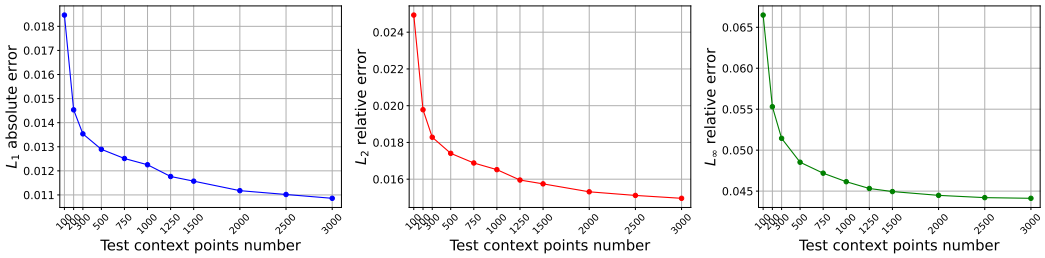

Figure 4: Sensitivity to the number of test context points $\overline{D}$ in the family of 2D CDR equations. Prediction errors decrease as more context observations are provided, which is consistent with the posterior-predictive-mean interpretation under squared loss.

H.12 Ablation Study

In this section, we conduct an ablation study to analyze the necessity of the key components in our model. Using the 2D CDR equation experiments, we replace AGF with a *vanilla* Transformer layer and substitute the rational activation with GELU to evaluate the impact of each component on performance.

Table 18: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to the family of 2D CDR equations. "Ours w.o. agf," "Ours w.o. act," and "Ours w.o. all" denote replacing AGF with a vanilla Transformer block, replacing the rational activation with GELU, and replacing both, respectively. They are measured at the seen coefficients $(\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\})$. The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours w.o. agf	Ours w.o. act	Ours w.o. all
Spatiotemporal interpolation	L_1 Abs	0.01218	0.01242	**0.01040**	0.01780
	L_2 Rel	0.01643	0.01625	**0.01495**	0.02281
	L_∞ Rel	0.04600	**0.04297**	0.04399	0.05736
Temporal extrapolation	L_1 Abs	**0.01474**	0.01776	0.01725	0.02454
	L_2 Rel	**0.02261**	0.02779	0.02665	0.03437
	L_∞ Rel	**0.08444**	0.09554	0.09376	0.10625

The experimental results in Table 18 show that although the ablated variants may occasionally perform slightly better on the interpolation task, our full model consistently achieves superior performance on the more challenging temporal extrapolation task. This supports our claim that the proposed architectural components enhance the model's ability to generalize across diverse tasks in a stable and efficient manner.

We additionally examine whether the robustness observed in Section 4.2 is preserved by conducting the same PINN-prior evaluation on the "Ours with sinusoidal activation" and "Ours w.o. AGF" variants.

Table 19: The evaluation results for the spatiotemporal interpolation and temporal extrapolation tasks applied to the family of 2D CDR equations. "Ours w.o. agf," "Ours w.o. act," and "Ours w.o. all" denote replacing AGF with a vanilla Transformer block, replacing the rational activation with GELU, and replacing both, respectively. They are measured at the seen coefficients $(\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3 \in \{0.0, 0.5, 1.0\})$. The best performance is marked in **bold** and the second-best performance is marked with underline.

Task	Metric	Ours	Ours w.o. agf	Ours with sinusoidal
Spatiotemporal interpolation	L_1 Abs	**0.01507**	0.01667	0.01482
	L_2 Rel	**0.02057**	0.02171	0.02096
	L_∞ Rel	0.06584	**0.05536**	0.06946
Temporal extrapolation	L_1 Abs	**0.01940**	0.02105	0.02671
	L_2 Rel	**0.02705**	0.03030	0.04159
	L_∞ Rel	**0.08802**	0.09518	0.14922

Table 19 shows that our model achieves the best results on both tasks, confirming strong robustness, whereas both ablations exhibit degraded performance. AGF and rational activation are most beneficial for temporal extrapolation and robustness under PINN-generated prior data, while interpolation performance can be achieved by simpler variants. We therefore view these components as stability-oriented rather than universally performance-dominating.

H.13 Computational Cost Analysis related to the Number of Context and Query Points

In this section, we analyze how the number of context and query points affects the training and inference time as well as memory cost. For both training and inference, we report the average per-iteration runtime measured over 10 iterations and record the peak GPU memory usage. When evaluating the effect of context size, we fix the number of query points to 4,096. Likewise, when evaluating the effect of query size, we fix the number of context points to 4,096. Training and inference are profiled under BF16 mixed precision in this section.

Table 20: Training and inference performance with different **context lengths**.

Context length	**1,024**	**2,048**	**4,096**	**8,192**	**12,288**	**16,384**
Training time (sec)	0.104	0.102	0.101	0.107	0.136	0.165
Training memory (MB)	280.0	329.8	420.7	617.2	800.4	987.4
Inference time (sec)	0.015	0.015	0.022	0.037	0.052	0.069
Inference memory (MB)	182.1	228.7	323.6	510.7	702.1	889.7

Table 21: Training and inference performance with different **query lengths**.

Query length	**1,024**	**2,048**	**4,096**	**8,192**	**12,288**	**16,384**
Training time (sec)	0.101	0.100	0.101	0.123	0.148	0.176
Training memory (MB)	872.7	1010.1	1265.4	1788.6	2303.0	2831.2
Inference time (sec)	0.02	0.02	0.021	0.024	0.027	0.03
Inference memory (MB)	242.3	270.1	323.1	433.6	540.7	648.1

As shown in Tables 20 and 21, runtime and memory show approximately linear scaling with context or query length. These empirical measurements characterize the evaluated configuration and do not by themselves establish asymptotic computational complexity.

H.14 Context Identifiability in the CDR Family

We examine whether the model can distinguish dynamics when the initial context is identical. We select two seen CDR instances,

$$q_1 = (0, 0, 0.5, 0.5, 0.5, 0.5, 0.5), \qquad q_2 = (1, 1, 0.5, 0.5, 0.5, 0.5, 0.5),$$

with coefficient order $(\beta_x, \beta_y, \nu_x, \nu_y, \rho_1, \rho_2, \rho_3)$. These instances have the same field at $t = 0$ but evolve differently. We fix $N_D = 1,024$ and start from the complete initial field. For each $m \in \{0, 4, 16, 32, 64\}$, we replace m randomly selected initial observations with observations at $t = 0.1$ at the same spatial locations. Both the time coordinate and the observed value are updated. We then predict the complete 32×32 field at $t = 0.2$ without gradient updates.

To quantify how strongly the predictions distinguish the instances, we define separation recovery as

$$R_{\text{sep}} = \frac{\|\hat{u}_{q_1} - \hat{u}_{q_2}\|_2}{\|u_{q_1} - u_{q_2}\|_2}.$$

Here, $R_{\text{sep}} = 0$ indicates identical predictions, whereas $R_{\text{sep}} = 1$ indicates recovery of the reference separation magnitude, not necessarily correct prediction fields. Results for $m > 0$ are reported as mean $\pm$ standard deviation over ten random observation selections; $m = 0$ has no selection variability.

With identical initial contexts, a deterministic predictor necessarily produces identical predictions. Introducing informative observations increases separation recovery and reduces both prediction errors, although substantial error remains. The variability across selections indicates that observation location matters. This experiment illustrates that prediction depends on distinctions present in the observed context; flexible context input does not resolve unobserved ambiguity.

Table 22: Context identifiability with fixed context cardinality. Replacing initial observations with later observations introduces information about the differing dynamics. Standard deviations reflect observation-selection variability, not training-seed variability.

m	**Separation recovery**	L_2 **Rel:** q_1	L_2 **Rel:** q_2
0	0.000	0.3059	0.2649
4	0.073 ± 0.195	0.3041 ± 0.0021	0.2638 ± 0.0012
16	0.270 ± 0.310	0.2970 ± 0.0057	0.2600 ± 0.0017
32	0.570 ± 0.337	0.2874 ± 0.0060	0.2550 ± 0.0013
64	0.887 ± 0.342	0.2692 ± 0.0109	0.2465 ± 0.0025

H.15 Sensitivity to Context Time

We evaluate context-time sensitivity for seen-coefficient CDR temporal prediction without retraining. The context contains the complete 32×32 spatial state, so $N_D = 1,024$ remains fixed. We vary the context time over $t_c \in \{0.45, 0.50, 0.60, 0.65\}$ while keeping the query fields fixed at $t \in \{0.7, 0.8, 0.9, 1.0\}$. All queried values are later than the context and remain masked in the model input.

Table 23: Sensitivity to the time of the observed context snapshot, with fixed context cardinality and future query times. The same trained checkpoint is used throughout.

Context time t_c	L_1 **Abs**	L_2 **Rel**	L_∞ **Rel**
0.45	0.01458	0.02199	0.08245
0.50	0.01437	0.02165	0.08024
0.60	0.01474	0.02261	0.08444
0.65	0.01604	0.02472	0.09530

Errors remain similar across the evaluated snapshots but do not decrease monotonically as the context approaches the query times. Contexts at $t_c = 0.45$ and 0.50 lie within the training-time domain, whereas those at 0.60 and 0.65 lie outside it; this distributional difference may contribute to the observed pattern. The results demonstrate flexibility across these complete snapshots, rather than invariance to context selection or robustness to all observation patterns.

H.16 Periodic Boundary Consistency

We evaluate periodic boundary consistency in the seen-coefficient CDR temporal extrapolation setting. Given the observed state $u(\cdot, 0.6)$, we independently query exact endpoint pairs at $t \in \{0.7, 0.8, 0.9, 1.0\}$ and examine the differences

$$\delta_x(y,t) = \hat{u}(0,y,t) - \hat{u}(\pi,y,t), \qquad \delta_y(x,t) = \hat{u}(x,0,t) - \hat{u}(x,\pi,t).$$

The endpoint π is absent from the stored grid, making this an off-grid query diagnostic. DeepONet is evaluated at the same endpoint pairs. Neither model is retrained or supplied with explicit boundary-condition constraints for this evaluation.

Table 24: Periodic endpoint mismatch for seen-coefficient CDR temporal extrapolation. Lower values indicate greater consistency of independently predicted periodic endpoint pairs.

Model	**Periodic BC RMSE**	**Relative periodic BC error**
DeepONet	0.07097	0.06898
PDE-PFN	**0.02081**	**0.02119**

PDE-PFN exhibits lower periodic mismatch in this evaluated setting. This provides evidence of approximate periodic consistency of its predictions, but not exact boundary-condition enforcement, PDE residual satisfaction, or conservation-law guarantees. The observed state is supplied as context rather than reconstructed, so this experiment does not evaluate initial-condition prediction error.

H.17 Comparison with GNOT

Experimental protocol. We compare PDE-PFN with GNOT [15], a Transformer-based neural operator that supports irregular meshes and pointwise prediction. The comparison matches datasets, splits, observed solution information, target coordinates, training-example counts, optimization budget, and evaluation metrics. We report the PDE-PFN configuration trained on numerical priors. The CDR experiments use the same pre-trained PDE-PFN checkpoint without task-specific gradient updates, whereas the SWE and CNSE experiments use downstream fine-tuning as in the main experiments. For unseen-coefficient CDR temporal extrapolation, the context is at $t = 0.1$ and queries are at $t \in \{0.2, 0.3, 0.4, 0.5\}$, within the training-time domain.

Table 25: Comparison with GNOT on the 2D CDR family. PDE-PFN uses numerical priors. temporal extrapolation with unseen coefficients remains within the training-time domain. Lower errors are better; the lower value for each metric in each setting is bold.

Task	Model	L_1 Abs	L_2 Rel	L_∞ Rel
Seen coefficients: spatiotemporal interpolation	PDE-PFN	**0.01218**	**0.01643**	**0.04600**
	GNOT	0.01477	0.01928	0.04619
Seen coefficients: temporal extrapolation	PDE-PFN	**0.01474**	**0.02261**	**0.08444**
	GNOT	0.02377	0.03126	0.08807
Coefficient interpolation: spatiotemporal interpolation	PDE-PFN	**0.01241**	0.01658	0.04274
	GNOT	0.01269	**0.01657**	**0.03930**
Coefficient interpolation: temporal extrapolation	PDE-PFN	**0.08957**	**0.11990**	**0.29526**
	GNOT	0.15797	0.22053	0.57981
Coefficient extrapolation: spatiotemporal interpolation	PDE-PFN	**0.01661**	**0.02398**	**0.08324**
	GNOT	0.04139	0.05339	0.12084
Coefficient extrapolation: temporal extrapolation	PDE-PFN	**0.01027**	**0.01068**	**0.01661**
	GNOT	0.06625	0.08667	0.22352

Table 26: Comparison with GNOT on SWE after downstream fine-tuning. PDE-PFN is initialized from pre-training on numerical CDR priors. Seen/unseen refers to initial-condition samples used/not used in downstream training. Lower errors are better; the lower value for each metric in each setting is bold.

Task	Model	L_1 Abs	L_2 Rel	L_∞ Rel
Seen initial conditions: spatiotemporal interpolation	PDE-PFN	**0.00208**	**0.00739**	**0.05897**
	GNOT	0.00317	0.00816	0.06265
Seen initial conditions: temporal extrapolation	PDE-PFN	**0.01548**	**0.03273**	**0.20590**
	GNOT	0.03076	0.07564	0.42872
Unseen initial conditions: spatiotemporal interpolation	PDE-PFN	**0.00213**	**0.00754**	**0.05986**
	GNOT	0.00317	0.00818	0.06233
Unseen initial conditions: temporal extrapolation	PDE-PFN	**0.01571**	**0.03323**	**0.20686**
	GNOT	0.03106	0.07636	0.43261

Table 27: Comparison with GNOT on CNSE trajectory prediction. PDE-PFN is fine-tuned from pre-training on numerical CDR priors. Lower errors are better; the lower value for each metric is bold.

Model	L_1 Abs	L_2 Rel	L_∞ Rel
PDE-PFN	**0.09497**	**0.54833**	**0.63533**
GNOT	0.09601	0.57140	0.65714

Results. Tables 25–27 report the CDR, SWE, and CNSE comparisons. PDE-PFN achieves lower errors on all three metrics in 10 of the 11 settings. The exception is CDR spatiotemporal interpolation at interpolated coefficients: PDE-PFN has lower L_1 error, whereas GNOT has slightly lower L_2 error (0.01657 versus 0.01658) and lower L_∞ error. PDE-PFN improves all three metrics in the other five CDR settings, all four SWE settings, and CNSE. These results extend the comparison to a related context-query neural operator. However, the models differ in both training formulation

and architecture, so this comparison does not isolate the contribution of PFN-style prior fitting from architectural design.

I Detailed Results

I.1 Detailed Experimental Results of PDE-PFN with Standard Deviation

In this section, we provide more detailed results for the family of 2D CDR equation experiments presented in Section 4.2. To further assess the stability of our model, we report the results over 5 random seeds in the form of mean $\pm$ standard deviation. As shown in Table 28, our model demonstrates consistently stable performance across all evaluation settings, including seen coefficients, coefficient interpolation, and coefficient extrapolation, for both spatiotemporal interpolation and temporal extrapolation tasks. These results further support that the strong performance of PDE-PFN is not due to a particular random seed, but is consistently observed across repeated runs.

Table 28: Detailed results of our model on the family of 2D CDR equations. All results are reported as mean $\pm$ standard deviation over 5 random seeds.

Metric	Seen Coeffcient		Coeff Interpolation		Coeff Extrapolation	
	Spatiotemporal Interpolation	Temporal Extrapolation	Spatiotemporal Interpolation	Temporal Extrapolation	Spatiotemporal Interpolation	Temporal Extrapolation
L_1 Abs	0.01218 ± 0.00218	0.01474 ± 0.00346	0.01241 ± 0.00170	0.08957 ± 0.01627	0.01661 ± 0.00132	0.01027 ± 0.00348
L_2 Rel	0.01643 ± 0.00250	0.02261 ± 0.00449	0.01658 ± 0.00176	0.11990 ± 0.02624	0.02398 ± 0.00202	0.01068 ± 0.00443
L_∞ Rel	0.04600 ± 0.00383	0.08444 ± 0.01088	0.04274 ± 0.00036	0.29526 ± 0.08702	0.08324 ± 0.00441	0.01661 ± 0.01137

I.2 Qualitative results

Figures 5 and 6 present qualitative comparisons for the 2D CDR and SWE experiments, respectively. For the CDR equation, PDE-PFN closely reconstructs the spatial patterns of the ground-truth solutions in both spatiotemporal interpolation and temporal extrapolation, while DeepONet exhibits larger deviations, particularly at later query times. For SWE, PDE-PFN better preserves the water-height profiles than DeepONet in the interpolation example and FNO in the temporal extrapolation example. These visual comparisons are consistent with the quantitative results reported in Tables 2 and 4.

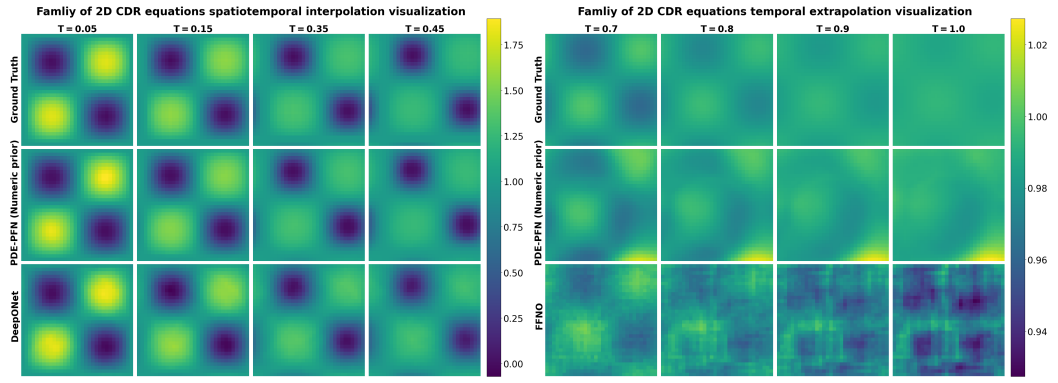

Figure 5: Visualization for the family of 2D CDR equation experiments. The left figure is for the spatiotemporal interpolation task and right figure is for the temporal extrapolation

J Experimental Details

J.1 Reproducibility

We are committed to ensuring the reproducibility of our research. All experimental procedures, data sources, and algorithms used in this study are clearly documented in the paper. The code will be provided as the supplementary material and be made publicly available upon publication, allowing others to validate our findings and build upon our work.

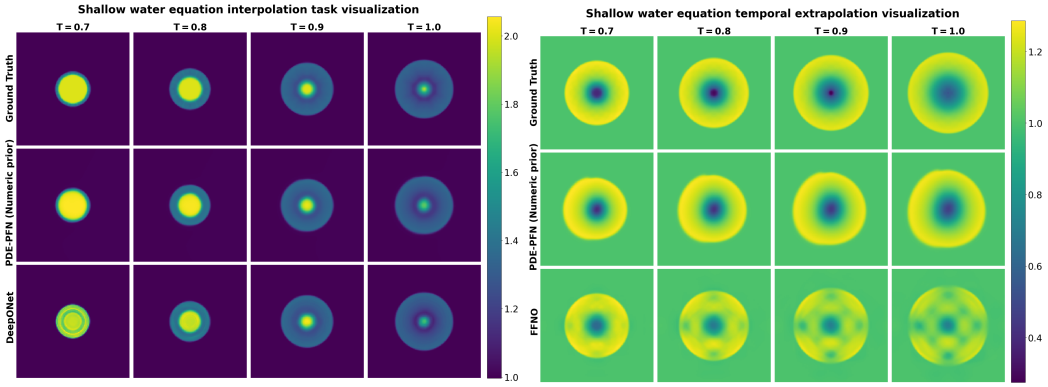

Figure 6: Visualization for the SWE experiments. The left figure is for the spatiotemporal interpolation task and right figure is for the temporal extrapolation

J.2 Environments

All the experiments were conducted using an NVIDIA RTX A6000. Details of the Python version and the packages used in the experiments can be found in the accompanying code's environment file.

J.3 Cost Comparison on the Main Experiments

In addition to the comparison between the baselines in Table 7, the additional comparison between the baselines is shown below in Table 29. In this tables, we compare the number of parameters, inference time, and the GPU memory usage for all models in their best settings. The inference time and GPU memory usage are measured in the test process on the family of 2D CDR equations temporal extrapolation task. In the experiment, both versions of our model are run with the same hyperparameters; therefore, we report the results under the single label Ours.

Table 29: Cost comparisons between baselines and our model in the family of 2D CDR equations temporal extrapolation test process.

Model	Number of parameters	inference time per sample(s)	GPU memory usage(MB)
Ours	4,488,873	0.24	26.64
DeepONet	723,969	0.01	11.06
FNO	143,268,097	0.10	1098.86
F-FNO	23,631,105	0.02	98.47
A-FNO	7,155,472	0.08	35.62
LNO	5,337,857	0.08	24.85
Poseidon	157,625,930	0.27	2408.40
DPOT	475,926,558	0.20	1825.72

J.4 Hyperparameters

In this section, we describe the number of data points for each equation and the hyperparameters used in the experiments for each model. In Table 30, the data points used in each experiment are shown separately for training, validation, and testing. For the training data, the context and query sets are constructed differently depending on the task. In the family of 2D CDR equations experiments with unseen coefficients, we only performed testing without training or validation; therefore, only the testing data points are reported.

In the family of 2D CDR equation experiments, the evaluations on unseen coefficients are conducted without additional training. Therefore, we report only the hyperparameters used for the spatiotemporal interpolation and temporal extrapolation tasks trained on seen coefficients. During fine-tuning after pre-training, the first linear layer of the encoder and the final linear layer of the decoder are re-initialized to deal with cases where the number of variables to be predicted differs. The selected

Table 30: Number of data points used for training, validation, and testing in all experiments.

PDE	Task	Train	Valid		Test	
			Context	Query	Context	Query
Family of 2D CDR equations	Spatiotemporal interpolation	6,144	1,024	1,024	1,024	4,096
	Temporal extrapolation	6,144	1,024	1,024	1,024	4,096
	Unseen coeff temporal extrapolation	-	-	-	1,024	4,096
SWE	Spatiotemporal interpolation	98,304	16,384	16,384	16,384	65,536
	Temporal extrapolation	98,304	16,384	16,384	16,384	65,536
CNSE	Time trajectory predicting operator learning	20,480	4,096	16,384	4,096	16,384
Airfoil	Time trajectory predicting operator learning	26,165	5,233	20,932	5,233	20,932
Darcy Flow	$a \mapsto u$ operator learning	a : 16,384 u : 16,384	a : 16,384 u : 16,384	a : 16,384 u : 16,384	a : 16,384 u : 16,384	a : 16,384 u : 16,384

CDR, SWE, and CNSE configurations use FFE dimension zero, while Airfoil uses sinusoidal FFE and Darcy Flow uses FFT-based embeddings. For fairness, all best hyperparameters were selected based on those that achieved the highest performance on the validation. The hyperparameters for our model and the baselines in each experiment can be found in Table from 31 to 33. In each experiment, both versions of our model are run with the same hyperparameters; therefore, we report the results under the single label Ours.

Table 31: Best hyperparameter for each model used in the family of 2D CDR equations and SWE experiments.

Model	Hyperparameter Name	Family of 2D CDR equations		SWE	
		Spatiotemporal interpolation	Temporal extrapolation	Spatiotemporal interpolation	Temporal extrapolation
Ours	Attention layers number	16			
	Attention hidden dimension	64			
	Attention head number	5			
	FFE dimension	0			
	AGF depth	15			
	En/Decoder hidden dimension	512			
DeepONet	Branch net depth	6	5	5	6
	Trunk net depth	5	5	6	6
	Hidden dimension	256	256	256	256
FNO	Layers number	-	4	-	3
	Hidden dimension	-	256	-	128
F-FNO	Layers number	-	5	-	3
	Hidden dimension	-	256	-	128
A-FNO	Layers number	-	12	-	3
	Hidden dimension	-	256	-	256
LNO	Layers number	-	3	-	3
	block number	-	8	-	8
Poseidon	Pre-trained weight	-	L	-	T
DPOT	Pre-trained weight	-	L	-	M
PINN prior	Training loss threshold	1×10^{-4}		-	
	Maximum training epoch	200		-	

K Impact Statement

This research adheres to the ethical standards required for scientific inquiry. We have considered the potential societal impacts of our work and have found no clear negative implications. We also see no direct path from our model to malicious uses such as surveillance, disinformation, or privacy

Table 32: Best hyperparameter for each model used in the CNSE and Darcy Flow experiments.

Model	**Hyperparameter Name**	CNSE	Darcy Flow
Ours	Attention layers number	16	
	Attention hidden dimension	64	
	Attention head number	5	
	FFE dimension	0	3
	AGF depth	15	
	En/Decoder hidden dimension	512	
DeepONet	Branch net depth	6	6
	Trunk net depth	5	5
	Hidden dimension	256	256
FNO	Layers number	3	6
	Hidden dimension	256	128
F-FNO	Layers number	4	3
	Hidden dimension	256	256
A-FNO	Layers number	8	16
	Hidden dimension	256	128
LNO	Layers number	3	4
	Block number	4	8
Poseidon	pre-trained weight	T	T
DPOT	pre-trained weight	L	S

Table 33: Best hyperparameter for each model used in the Airfoil experiments.

Model	**Hyperparameter Name**	Operator Learining
Ours	Attention layers number	16
	Attention hidden dimension	64
	Attention head number	5
	FFE dimension	2
	AGF depth	15
	En/Decoder hidden dimension	512
DeepONet	Branch net depth	6
	Trunk net depth	5
	Hidden dimension	256
Oformer	Layers number	3
	Hidden dimension	64

breaches. On the other hand, we recognize that improvements in PDE-solving capabilities could eventually be applied in real-world applications, such as climate modeling, biomedical simulations, or engineering systems.

All experiments were conducted in compliance with relevant laws and ethical guidelines, ensuring the integrity of our findings. We are committed to transparency and reproducibility in our research processes.

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: The findings in Section 4, Theorem 2.1 and A.2 ,and Appendix support and are consistent with the main claims articulated in the abstract and introduction.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 explicitly address the limitations of our work in Section 6, where we discuss potential weaknesses of our approach and experiments.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: This paper includes Theorem 2.1, Theorem A.2, and Corollary A.3. The complete proof of them are provided in Appendix A

Guidelines:

- The answer [N/A] 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 provide detailed information of hyperparameters for each model in Appendix J.4 and outline the full training procedure in the algorithm presented in Appendix F. We also discuss reproducibility in Appendix J.1

Guidelines:

- The answer [N/A] 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: To ensure reproducibility, we include the full experimental code in the supplementary material, accompanied by a README file detailing the data generation process and step-by-step instructions for reproducing all main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Comprehensive information about the datasets, including their sizes and the hyperparameters used in each experiment, is available in Appendix B and J.4.

   Guidelines:

   - The answer [N/A] 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: Table 28 reports detailed results for the experiment in Section 4.2, including standard deviations of each metric to capture variability across seeds.

   Guidelines:

   - The answer [N/A] 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 provide a comparison of computational costs in Section J.3 including runtime, and memory requirements for each experimental run. Appendix J.2 includes details on hardware used.

   Guidelines:

   - The answer [N/A] 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: We confirm that our work adheres to the NeurIPS Code of Ethics in all respects, including the preservation of anonymity and ethical considerations regarding data usage and experimental design.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: In Section K, we address both potential positive and negative societal impacts of our work. We commit to ongoing monitoring of broader impacts and support reproducibility through open code and evaluation protocols.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper poses no such risks.

Guidelines:

- The answer [N/A] 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: Among the datasets used in our experiments, SWE, CNSE, and Darcy flow datasets were provided by PDEbench [47]. The Airfoil dataset was previously utilized in the Oformer [26]. The 3D Gray-Scott dataset was introduced by APEbench [24]. In Appendix B, we provide the source from which we obtained these datasets, along with the original papers and corresponding URLs.

Guidelines:

- The answer [N/A] 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` 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: [Yes]

    Justification: In the supplementary material, we provide the code used for the experiments described in Section 4, along with documentation that guides readers through the experimental process. All code has been properly anonymized for submission.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.