

Article

Physics-Informed Fine-Tuned Neural Operator for Flow Field Modeling

Haodong Feng ^{1,2} , Yuzhong Zhang ^{2,*}  and Dixia Fan ^{2,*} ¹ College of Environmental and Resource Sciences, Zhejiang University, Hangzhou 310027, China; h.feng@zju.edu.cn² Key Laboratory of Coastal Environment and Resources of Zhejiang Province, School of Engineering, Westlake University, Hangzhou 310030, China

* Correspondence: zhangyuzhong@westlake.edu.cn (Y.Z.); fandixia@westlake.edu.cn (D.F.)

Abstract

Modeling flow field evolution accurately is important for numerous natural and engineering applications, such as pollutant dispersion in the ocean and atmosphere, yet remains challenging because of the highly nonlinear, multi-physics, and high-dimensional features of flow systems. While traditional equation-based numerical methods suffer from high computational costs, data-driven neural networks struggle with insufficient data and lack physical explainability. The physics-informed neural operator (PINO) addresses this by combining physics and data losses but faces a fundamental gradient imbalance problem. This work proposes a physics-informed fine-tuned neural operator for high-dimensional flow field modeling that decouples the optimization of physics and data losses. Our method first trains the neural network using data loss and then fine-tunes it with physics loss before inference, enabling the model to adapt to evaluation data while respecting physical constraints. This strategy requires no additional training data and can be applied to fit out-of-distribution (OOD) inputs faced during inference. We validate our method using the shallow water equation and advection–diffusion equation using a convolutional neural operator (CNO) as the base architecture. Experimental results show a 26.4% improvement in single-step prediction accuracy and a reduction in error accumulation for multi-step predictions.

Keywords: physics-informed machine learning; neural operator; advection-diffusion equation; shallow water equation



Academic Editor: Dong-Sheng Jeng

Received: 13 December 2025

Revised: 6 January 2026

Accepted: 14 January 2026

Published: 19 January 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Flow fields represent fundamental physical phenomena that govern numerous natural and engineering processes [1–4]. Natural processes encompass critical environmental phenomena such as pollutant dispersion in oceans and the atmosphere [5], where understanding flow field evolution is essential for predicting contamination spread and assessing environmental impacts; ocean circulation systems, which transport heat, nutrients, and carbon across global scales and play a crucial role in climate regulation [6]; and atmospheric flows, including weather patterns and air quality dynamics that directly affect human health and agricultural productivity [7]. Engineering applications include chemical reactors, aerospace engineering, and energy systems [8]. The accurate modeling of flow field evolution is crucial for understanding, predicting, and controlling these complex systems [9]. However, flow fields exhibit inherent challenges: they are nonlinear,

multi-physics coupled, and high-dimensional systems, making their accurate and efficient modeling a significant challenge [9].

Traditional approaches to flow field modeling rely on numerical methods based on Partial Differential Equations (PDEs). The types of numerical methods mainly include finite difference methods, finite element methods, and finite volume methods [10,11]. While these methods have been extensively developed and validated, they suffer from substantial computational costs, particularly for high-resolution simulations and multi-physics coupling. Moreover, these numerical schemes may face stability issues and divergence failures, especially in complex flow regimes [12].

The development of deep learning has opened significant potential avenues for flow field modeling through data-driven neural network approaches [13–15]. These methods can learn complex nonlinear mappings from input initial conditions, boundary conditions, and physical parameters to future states directly from data, potentially offering significant computational advantages over traditional numerical methods [16]. However, data-driven approaches face critical limitations: they exhibit poor accuracy when training data is insufficient, and their black-box nature leads to violations of physical principles and a lack of explainability [17,18].

To address these limitations, researchers have developed the physics-informed neural operator (PINO), physics-informed operator learning, and their variants. They incorporate physics by integrating physics loss terms with the data loss [19–21]. These methods jointly optimize neural network parameters using a combined loss function: $\mathcal{L}_{total} = \mathcal{L}_{data} + \lambda \cdot \mathcal{L}_{physics}$, where λ is a weighting parameter. While promising, this approach faces a fundamental challenge: the gradients of physics loss and data loss differ significantly in magnitude and direction. When λ is small, the physics loss provides minimal constraint, resulting in small accuracy improvements. Conversely, when λ is large, the data loss becomes dominated, causing the optimization to violate the data distribution and potentially increase prediction errors. This gradient imbalance makes it difficult to optimize neural networks in many systems and experimental settings [22,23].

In this work, we propose a physics-informed fine-tuned neural operator for high-dimensional flow field modeling that addresses the gradient imbalance problem and effectively utilizes the physical information by decoupling the optimization of physics loss and data loss. Our approach consists of two distinct phases: (1) **Data-Driven Pretraining phase**: The deep model is first trained using the training set with only data loss, allowing it to learn the data distribution effectively. (2) **Physics-Informed Fine-tuning phase**: Before inference on evaluation data, we compute physics loss from the model's input of evaluation data and output predictions and perform additional optimization steps. This enables the model to transfer to the distribution of specific evaluation data using physical information. There are two advantages of the proposed method. First, it requires no additional training data beyond the original dataset yet improves accuracy through the incorporation of physical information. Second, by separating the optimization phases, we avoid the gradient imbalance problem that plagues joint optimization approaches in physics-informed machine learning for flow field modeling.

To validate our method, we conduct experiments on two distinct flow field systems: (1) **Shallow water equation**, which models the height field and two-directional velocity fields. This equation is fundamental for understanding wave propagation, tsunami forecasting, and coastal hydrodynamics, playing a critical role in oceanography and atmospheric science for predicting storm surges and flood dynamics [24]. (2) **Advection–diffusion equations**, which describe the evolution of the concentration field, two-directional velocity fields, and pressure fields. These equations are essential for modeling pollutant transport in environmental systems, chemical reaction processes in industrial applications, and heat and mass transfer phenomena, making them crucial for environmental protection and process optimization [25]. For both

systems, we employ a convolutional neural operator (CNO) [26] as the base model architecture, which is suited for 2D spatial-temporal fields. We integrate our physics-informed fine-tuning method with the CNO (PFT-CNO) and evaluate its performance against baseline techniques.

Results of our experiments show the effectiveness of the proposed PFT-CNO. After physics-informed fine-tuning, the model achieves an 8.3% and 26.4% improvement in single-step prediction accuracy compared to the baseline for two systems, respectively. Moreover, in long-term multi-step predictions, the method reduces error accumulation while maintaining relatively better accuracy throughout the prediction horizon.

The main contributions of this work are summarized as follows:

1. We present a physics-informed fine-tuning method for high-dimensional flow field modeling, which separates the optimization of physics loss and data loss into distinct phases. By fine-tuning using inference data input via physics loss, it addresses the gradient imbalance problem that plagues joint optimization approaches in physics-informed neural operators.
2. We apply the physics-informed fine-tuning method to the CNO and propose PFT-CNO, which combines the ability of the CNO to handle high-dimensional spatio-temporal flow fields with the advantages of physics-informed fine-tuning, requiring no additional training data yet significantly improving prediction accuracy through physics-informed fine-tuning during inference.
3. We demonstrate the effectiveness of our method on two distinct flow field systems (shallow water equation and advection-diffusion equation), achieving substantial improvements in both single-step and multi-step predictions, validating the practical utility of PFT-CNO.

2. Related Works

The emergence of deep learning has revolutionized flow field modeling through data-driven approaches. Convolutional neural networks (CNNs) and U-Net have been applied to learn spatial patterns in flow fields [27], while Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks capture temporal dependencies [28,29]. Graph Neural Networks (GNNs) have been employed for irregular mesh-based flow simulations [30,31]. However, these methods discretize the variable-to-variable mapping of the flow field, ignoring the continuity characteristics of the flow field.

Neural operators represent a significant advancement over traditional neural networks by learning mappings between function spaces, enabling generalization to different resolutions and domains. The Fourier Neural Operator (FNO) leverages spectral methods to efficiently process spatial information [32]. DeepONet uses branch and trunk networks to approximate nonlinear operators [33]. The Convolutional Neural Operator (CNO) employs filtered convolutions to maintain spatial locality while learning operator mappings [26]. Transolver introduces attention-based learning of flow fields, enabling the neural network to achieve intrinsic geometry-independent ability [34]. The Wavelet Diffusion Neural Operator (WDNO) introduces the wavelet transform into the diffusion model, enabling more accurate predictions for complex high-dimensional fluid dynamics [35]. These methods have shown promise in learning PDE solution operators, but they still heavily rely on data, have poor accuracy with limited training data, and violate physical laws.

Physics-Informed Neural Networks (PINNs) incorporate physical knowledge by adding physics loss terms to the training objective [36]. The physics loss is computed by evaluating the residual of the governing PDEs at collocation points, ensuring the neural network solution satisfies the underlying physical laws. PINNs have been extended to various applications, including heat transfer [37] and wave propagation [38]. However, PINNs typically solve specific instances of PDEs rather than learning general solution operators.

To combine the operator learning capability of neural operators with the physics constraints of PINNs, researchers have developed physics-informed neural operators (PINOs), physics-informed operating learning (PI-DeepONet) [19,39], and others [40,41]. These methods incorporate physics loss into the training of neural operators, enabling them to learn solution operators while respecting physical laws. However, these methods face a fundamental challenge: the joint optimization of data loss and physics loss suffers from gradient imbalance, where the gradients of these two loss components differ significantly in magnitude and direction. This imbalance makes it difficult to leverage physics loss in many cases, often resulting in either insufficient physics constraints or violation of data distributions [22]. Although the physics-informed fine-tuning method has also been introduced in previous works [15,42–44], these only studied modeling on partial observations or simple 1D equations [42] or studied transfer learning of PINNs instead of neural operators with generalization ability. We propose PFT-CNO for the first time in this work, thus extending the physics-informed fine-tuned neural operator to 2D high-dimensional flow field modeling.

3. Materials and Methods

3.1. Problem Setup

We consider the problem of learning the solution operator for a family of parametric PDEs. Neural operators extend traditional neural networks from finite-dimensional vector mappings to infinite-dimensional functional mappings, naturally accommodating different mesh resolutions and discretization schemes. To formalize this problem, we consider a parametric PDE defined on a spatial domain D and temporal domain $(0, \infty)$:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \mathcal{R}(u), \quad \text{in } D \times (0, \infty), \\ u &= g, \quad \text{in } \partial D \times (0, \infty), \\ u &= u(0), \quad \text{in } D \times \{0\},\end{aligned}\tag{1}$$

where $\mathcal{R}$ is a possibly nonlinear partial differential operator associated with Banach spaces $\mathcal{U}$ and $\mathcal{A}$, g is the known boundary conditions, and $u(0)$ denotes the initial condition. The unknown function $u(t) \in \mathcal{U}$, where $t > 0$, represents the solution we aim to approximate. We assume that u exists and is bounded for all time and for every initial condition in the appropriate function space.

The definition leads to an operator of solution $\mathcal{G} : \mathcal{A} \rightarrow \mathcal{U}$ that maps input functions $a \in \mathcal{A}$ (initial conditions, boundary conditions, and coefficient functions) to output solution functions $u \in \mathcal{U}$ that satisfy the governing PDE. Prototypical examples of such problems include the shallow water equation, advection–diffusion equation, and Navier–Stokes equation, which are fundamental in fluid dynamics and exhibit complex nonlinear behavior.

3.2. Neural Operator

To approximate the solution operator $\mathcal{G}$, we employ neural operators, which are generalizations of standard deep neural networks to the operator setting. Neural operators compose linear integral operators with nonlinear activation functions to approximate highly nonlinear operators. Formally, we define the neural operator model as follows:

$$\mathcal{G}_\theta := Q \circ \sigma(W_L + K_L) \circ \cdots \circ \sigma(W_1 + K_1) \circ P,\tag{2}$$

where θ denotes all learnable parameters. This architecture consists of three key components:

- **Lifting operator** P : $P : \mathbb{R}^{d_a} \rightarrow \mathbb{R}^{d_1}$ lifts the lower-dimensional input function $a \in \mathcal{A}$ (with co-dimension d_a) into a higher-dimensional latent space.

- **Iterative layers:** The model stacks L layers of $\sigma(W_l + K_l)$, where $W_l \in \mathbb{R}^{d_{l+1} \times d_l}$ are local linear operators, $K_l : \{D \rightarrow \mathbb{R}^{d_l}\} \rightarrow \{D \rightarrow \mathbb{R}^{d_{l+1}}\}$ are integral kernel operators that capture non-local dependencies in function spaces, and σ denotes pointwise activation functions.
- **Projection operator Q :** $Q : \mathbb{R}^{d_L} \rightarrow \mathbb{R}^{d_u}$ projects the higher-dimensional function back to the output space (with co-dimension d_u).

The neural operator $\mathcal{G}_\theta$ learns to approximate the above-mentioned operator $\mathcal{G}$ by combining local operations with global integral operations, enabling it to capture both fine-scale local features and long-range dependencies in the solution space. This framework provides the foundation for learning mappings between function spaces in operator learning problems.

3.3. Convolutional Neural Operator

As a specific instantiation of the neural operator, we introduce the convolutional neural operator (CNO) [26], which specializes the integral kernel operators K_l to convolutional operators. The CNO is designed to operate on the space of bandlimited functions $\mathcal{B}_w(D)$, where D is the spatial domain and w denotes the bandwidth. Under this framework, the CNO maintains the general form $\mathcal{G}_\theta = Q \circ \sigma(W_L + K_L) \circ \dots \circ \sigma(W_1 + K_1) \circ P$, with the following specializations:

- **Lifting operator P :** The lifting operator is instantiated as a convolutional layer that maps the input function $u \in \mathcal{B}_w(D, \mathbb{R}^{d_x})$ to the latent space $v_0 \in \mathcal{B}_w(D, \mathbb{R}^{d_0})$, where $d_0 > d_x$ is the number of latent channels.
- **Iterative layers:** Each layer l implements the transformation $v_{l+1} = \sigma(W_l(v_l) + K_l(v_l))$, where
 - K_l is a convolutional operator with learnable kernels that captures spatial dependencies;
 - W_l represents skip connections or residual mappings;
 - σ applies pointwise nonlinear activations.

Additionally, resolution adjustment operators (upsampling or downsampling) may be incorporated within layers to enable multi-scale feature extraction, inspired by architectures for image generation.

- **Projection operator Q :** The projection operator is realized as a convolutional layer that maps the final latent function $v_L \in \mathcal{B}_w(D, \mathbb{R}^{d_L})$ to the output space $u' \in \mathcal{B}_w(D, \mathbb{R}^{d_y})$.

The core computational unit of the CNO is the convolution operator K , which employs learnable kernels of size $k \times k$:

$$K(i, j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} w(m, n) \cdot v(i + m, j + n), \quad (3)$$

where $w(m, n)$ represents trainable kernel weights. A distinguishing feature of the CNO compared to spectral methods such as the FNO is its direct operation in the spatial domain through “real-space” transformations, rather than relying on frequency-domain convolutions via Fourier transforms. This spatial parameterization enhances locality, naturally accommodates non-periodic boundary conditions, and extends readily to irregular geometries. To enable multi-scale feature extraction, it introduces resolution-changing operators, including a downsampling block, upsampling block, residual block, etc. [26].

To effectively capture multi-scale features inherent in flow fields, a modified U-Net architecture is employed for the CNO implementation. The U-Net [45], originally developed for image segmentation tasks, features a symmetric encoder–decoder structure with skip connections that directly link corresponding resolution levels. This design is particularly well-suited for the CNO in several aspects. First, the encoder progressively

reduces spatial resolution by downsampling while expanding channel capacity, enabling the extraction of increasingly abstract and global features. Second, the decoder reconstructs fine-scale details by gradually upsampling while reducing channel dimensions. Third, the skip connections through residual blocks bypass the bottleneck to transfer high-frequency information directly from encoder to decoder at matching resolutions, preventing the loss of fine-scale features during the encoding process.

A key property of the CNO is its structure-preserving nature: bandlimited input functions are mapped to bandlimited output functions, respecting the continuous–discrete equivalence. This property enables the operator to generalize across various discretizations while maintaining underlying continuous structures of solution spaces. Moreover, the universal approximation of the CNO is proved by [26]. By leveraging efficient convolutional operations combined with multi-scale processing, the CNO is particularly well-suited for learning operators on high-dimensional spatio-temporal flow fields.

3.4. Physics-Informed Fine-Tuned CNO

In this work, we utilize the CNO as the base model and integrate the physics-informed fine-tuning strategy and then propose the physics-informed fine-tuned convolutional neural operator (PFT-CNO) to adapt to high-dimensional flow field modeling. While the neural network architecture of PFT-CNO is identical to the standard CNO model described in Section 3.3, we employ a novel two-phase training strategy as illustrated in Figure 1 to enhance generalization and mitigate out-of-distribution (OOD) performance degradation. The pseudocode and hyperparameters are shown in Appendixes A and B.

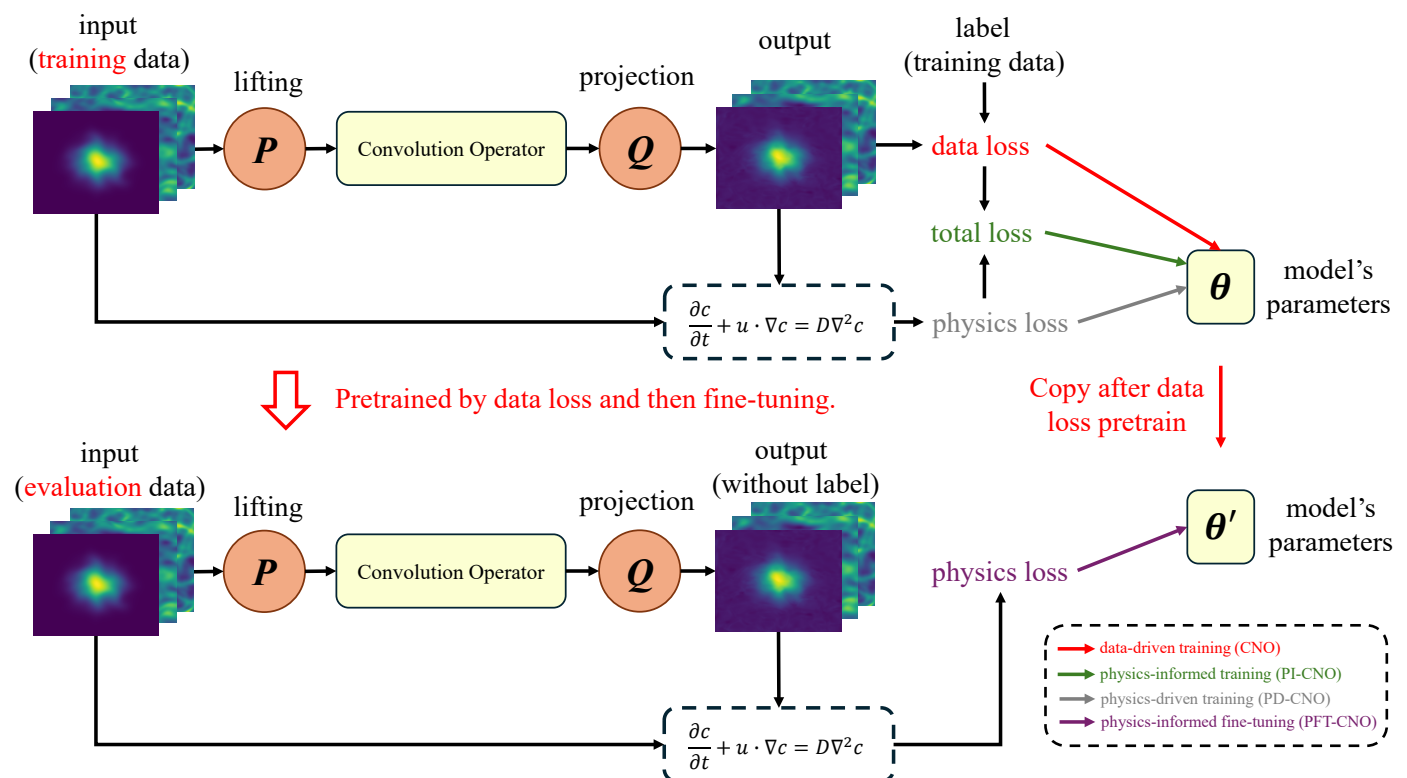


Figure 1. Illustration of the training pipeline of the physics-informed fine-tuned operator. The figure shows the calculation logic of data loss, physics-informed loss, and physics loss. PFT-CNO is first pretrained using data loss and then fine-tuned using physics loss.

Two-Phase Training Strategy

Phase I: Data-Driven Pretraining. The first phase follows the conventional supervised learning paradigm. We construct the training dataset $\mathbb{D} = \{(u_t^{(i)}, u_{t+1}^{(i)})\}_{i=1}^{N_{train}}$ by solving

Equation (1) using numerical methods on discrete temporal and spatial domains. Each sample consists of an input–output pair (u_t, u_{t+1}) , where u_t represents the system state at time t and u_{t+1} represents the corresponding state at time $t + 1$. The pretraining objective minimizes the data loss, defined as follows:

$$\mathcal{L}_{data} = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \frac{\|\hat{u}_{t+1}^{(i)} - u_{t+1}^{(i)}\|_2}{\|u_{t+1}^{(i)}\|_2}, \quad (4)$$

where $\hat{u}_{t+1}^{(i)} = \mathcal{G}_\theta(u_t^{(i)})$ and $\|\cdot\|_2$ represents the L^2 norm. This phase enables the model to learn the general mapping from the training data distribution.

Phase II: Physics-Informed Fine-Tuning. After pretraining, conventional approaches would directly test the model on the evaluation dataset. However, we introduce an intermediate physics-informed fine-tuning phase to adapt the model to the distribution of evaluation data. The evaluation dataset $\mathbb{B} = \{u_t^{(j)}, u_{t+1}^{(j)}\}_{j=1}^{N_{eval}}$ is generated using the same numerical solver as $\mathbb{D}$, but with different initial conditions and physical parameters (e.g., viscosity coefficient, diffusion coefficient, and Reynolds number).

Crucially, during fine-tuning, we only have access to the input states $u_t^{(j)}$ from $\mathbb{B}$, not the ground truth outputs $u_{t+1}^{(j)}$, as these represent the quantities to be predicted. To overcome this challenge, we leverage the underlying physical information to construct a physics loss function. After the pretraining, data loss has initialized the model parameters to a constrained range. Here, the goal of using physics loss is to start from the point after data loss optimization and further constrain the model parameters to the optimal solution that fits the distribution of the evaluation dataset, avoiding the imbalance problem caused by synchronous competition between the two loss functions. Specifically, the physics loss is computed using the predicted output $\hat{u}_{t+1}^{(j)} = \mathcal{G}_\theta(u_t^{(j)})$:

$$\mathcal{L}_{physics} = \frac{1}{N_{eval}} \sum_{j=1}^{N_{eval}} \left| \frac{\partial \hat{u}_{t+1}^{(j)}}{\partial t} - \mathcal{R}(u) \right|^2. \quad (5)$$

Unlike methods that employ automatic differentiation (autograd) to compute spatial and temporal derivatives, we utilize finite difference approximations following [15,46], which significantly improves computational efficiency and reduces memory consumption. This design choice is particularly advantageous for high-resolution spatial domains where autograd-based approaches become prohibitively expensive.

It is worth noting that although in our work the CNO is employed as the base architecture and relative L_2 error as the data loss, with the finite difference method and MSE as the physics loss, our method can also be extended to other base models, and data loss and physics loss functions.

4. Results

We validate the performance of our proposed PFT-CNO method on two distinct PDE systems: the shallow water equation (SWE) and the advection–diffusion equation (ADE). Our method is compared against several baseline approaches to demonstrate the improvements achieved through physics-informed fine-tuning. The experimental results show that PFT-CNO consistently outperforms existing methods in both single-step and multi-step predictions, demonstrating the effectiveness of decoupling physics loss and data loss optimization.

4.1. Baseline Methods

To examine the modeling accuracy of PFT-CNO, we compare PFT-CNO with the following baseline approaches.

- **DeepONet** [33]: This is a data-driven neural operator that uses branch and trunk networks to approximate nonlinear operators. This method serves as a representative baseline for purely data-driven operator learning approaches.
- **PFT-DeepONet** [42]: The physics-informed fine-tuning method is applied to DeepONet and validated on simple 1D problems. It serves as a baseline to compare the advantages of our proposed PFT-CNO over existing models for more complex systems.
- **Transolver** [34]: This is a data-driven transformer-based neural operator that leverages attention mechanisms to achieve geometry-independent learning of flow fields. This method demonstrates the capability of transformer architectures in operator learning.
- **CNO** [26]: It serves as our base model architecture. This purely data-driven method provides the foundation for our physics-informed fine-tuning approach and allows us to isolate the contribution of the fine-tuning strategy.
- **PD-CNO**: The Physics-Driven CNO is a variant of the CNO trained exclusively using physics loss without any data loss. This baseline demonstrates the performance when relying solely on physical constraints, highlighting the importance of data-driven initialization.
- **PI-CNO** [40]: This is the Physics-Informed CNO, which jointly optimizes both data loss and physics loss during training using the combined objective $\mathcal{L}_{total} = \mathcal{L}_{data} + \lambda \cdot \mathcal{L}_{physics}$. This method represents the conventional approach to incorporating physical information into neural operators and allows us to compare against joint optimization strategies.

4.2. Evaluation Metrics

We employ five metrics to evaluate the performance of all methods.

- **Relative L^2 Error**: This metric measures the normalized prediction error relative to the ground truth magnitude, defined as follows:

$$\text{Rel } L^2 \text{ Error} = \frac{1}{N} \sum_{i=1}^N \frac{\|\hat{u} - u\|_2}{\|u\|_2}, \quad (6)$$

where N is the total number of spatial points, u represents the ground truth, and $\hat{u}$ denotes the predicted solution, which is the output of neural operators $\mathcal{G}_\theta$ defined in Section 3.2. As we mentioned in Section 3.1, $\mathcal{G}_\theta$ maps input functions $a \in \mathcal{A}$ (e.g., initial conditions, boundary conditions, or coefficient functions) to output solution functions $u \in \mathcal{U}$ that satisfy the governing PDE. This metric provides a scale-invariant measure of prediction accuracy.

- **Root Mean Square Error (RMSE)**: The RMSE can be expressed in terms of the standard deviations and correlation coefficient:

$$\text{RMSE} = \sqrt{\sigma_o^2 + \sigma_p^2 - 2\sigma_o\sigma_p R}, \quad (7)$$

where σ_o and σ_p represent ground truth and predicted standard deviations, respectively, and R represents the Pearson correlation coefficient between them. This metric penalizes variance differences and insufficient correlation.

- **Mean Absolute Error (MAE)**: The MAE quantifies the average absolute deviation between predictions and the ground truth:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |\hat{u}_i - u_i|. \quad (8)$$

This metric offers an intuitive interpretation of prediction errors in the original units.

- **Coefficient of Determination (R^2):** The R^2 score measures the proportion of variance in the ground truth explained by the predictions:

$$R^2 = 1 - \frac{\sum_{i=1}^N (u_i - \hat{u}_i)^2}{\sum_{i=1}^N (u_i - \bar{u})^2}, \quad (9)$$

where $\bar{u} = \frac{1}{N} \sum_{i=1}^N u_i$ is the mean of the ground truth. An R^2 value close to 1 indicates excellent predictive performance, while negative values suggest predictions worse than the mean.

- **Inference Time:** This metric measures the computational efficiency by recording the average time required to perform a single forward prediction. Lower inference times indicate better computational efficiency for real-time applications. We measure the inference time of all methods using an NVIDIA H800 GPU, 64 cores, and a batch size of 64.

We also use additional metrics to evaluate the statistical significance of model performance differences following the previous work [47].

- **ANOVA:** This assesses whether significant differences exist among multiple models using the F-statistic ($p < 0.05$ indicates significance).

$$F = \frac{\sum_{i=1}^k n_i (\bar{u}_i - \bar{u})^2 / (k - 1)}{\sum_{i=1}^k \sum_{j=1}^{n_i} (u_{ij} - \bar{u}_i)^2 / (N - k)}, \quad (10)$$

where k is the number of models, n_i is the sample size of the i -th model, N is the total sample size, $\bar{u}_i$ is the mean of the i -th model, and $\bar{u}$ is the overall mean.

- **T-test:** The test compares performance differences between two models on the same data.

$$t = \frac{\bar{d}}{\sigma_d / \sqrt{n}}, \quad (11)$$

where $\bar{d}$ is the mean of paired differences ($\text{MSE}_1 - \text{MSE}_2$), σ_d is the standard deviation of paired differences, and n is the sample size.

- **Cohen's d :** It quantifies the magnitude of the results between two models. For paired samples, it is calculated as follows:

$$d = \frac{\bar{d}}{\sigma_d}. \quad (12)$$

d indicates which model performs better: $d > 0$ suggests that Model 2 has better performance, while $d < 0$ indicates that Model 1 has better performance.

4.3. Shallow Water Equation

The SWE is a fundamental system in geophysical fluid dynamics [24], modeling the evolution of fluid height and horizontal velocity fields. This system provides a benchmark for evaluating our method on multi-field, nonlinear PDE systems.

4.3.1. Data Collection

The SWE in two dimensions is given by

$$\begin{aligned} \frac{\partial h}{\partial t} + \frac{\partial(hu)}{\partial x} + \frac{\partial(hv)}{\partial y} &= 0, \\ \frac{\partial(hu)}{\partial t} + \frac{\partial}{\partial x} \left(hu^2 + \frac{1}{2}gh^2 \right) + \frac{\partial(huv)}{\partial y} - fhu &= \nu \left(\frac{\partial^2(hu)}{\partial x^2} + \frac{\partial^2(hu)}{\partial y^2} \right), \\ \frac{\partial(hv)}{\partial t} + \frac{\partial(huv)}{\partial x} + \frac{\partial}{\partial y} \left(hv^2 + \frac{1}{2}gh^2 \right) + fhv &= \nu \left(\frac{\partial^2(hv)}{\partial x^2} + \frac{\partial^2(hv)}{\partial y^2} \right), \end{aligned} \quad (13)$$

where $h(x, y, t)$ represents the fluid height, $u(x, y, t)$ and $v(x, y, t)$ denote the velocity field components in the x and y directions, respectively, $g = 9.81 \text{ m/s}^2$ is the gravitational acceleration, f is the Coriolis parameter, and ν is the viscosity coefficient.

We solve these equations numerically using central difference schemes for spatial derivatives and the explicit fourth-order Runge–Kutta integrator for temporal discretization. The simulations are conducted on a spatial domain discretized with a resolution of 128×128 grid points, employing periodic boundary conditions. The temporal discretization uses a time step of $\Delta t = 0.001 \text{ s}$, and we integrate the system for a total duration of $T = 1 \text{ s}$, resulting in 1000 time steps per numerical simulation. Then, we downsample 1000 time steps to 100 time steps for training and evaluation to ensure distinctions between adjacent steps.

To ensure clear evaluation across different flow regimes, we generate 10 distinct trajectories by varying initial conditions and physical parameters. We initialize all three state variables (fluid height h and velocity components u and v) using the Gaussian Random Field (GRF) generated via a spectral method in Fourier space following previous works [19,21]. For each numerical simulation, we sample the GRF parameters from uniform distributions to ensure diverse initial conditions that capture a wide range of flow regimes while maintaining physically reasonable perturbation amplitudes. Moreover, the Coriolis parameter f is randomly sampled from a uniform distribution over the range $[0, 1.5 \times 10^{-4}]$ to control rotational effects, where this range is selected based on realistic Earth Coriolis parameter values. The viscosity coefficient ν is randomly sampled from a uniform distribution over the range $[10^{-5}, 10^{-3}]$ to govern the dissipative behavior. Among these 10 trajectories, 5 are allocated for training the neural operators, while the remaining 5 are reserved for evaluation, enabling assessment of generalization to unseen parameter configurations.

4.3.2. Single-Step Prediction Performance

Table 1 presents the quantitative results for single-step prediction on the evaluation set of the SWE. The results demonstrate the superior performance of our proposed PFT-CNO across all evaluation metrics.

Table 1. Performance comparison of different methods on SWE single-step prediction. Best in **bold** and second best underlined.

Method	Rel L^2 Error ↓	RMSE ↓	MAE ↓	R^2 ↑	Eval Time (s) ↓
DeepONet	0.16331	0.09720	0.07124	−0.05412	0.00727
PFT-DeepONet	0.15846	0.09475	0.06924	−0.00190	0.00737
Transolver	0.16255	0.09682	0.07093	−0.04465	0.00156
CNO	0.01284	0.00790	0.00389	0.99304	0.00326
PD-CNO	0.01693	0.01031	0.00538	0.98813	0.00334
PI-CNO	<u>0.01256</u>	<u>0.00774</u>	<u>0.00384</u>	<u>0.99331</u>	0.00352
PFT-CNO	0.01178	0.00737	0.00372	0.99394	<u>0.00314</u>

The results reveal several key observations. First, the purely data-driven methods DeepONet and Transolver exhibit higher errors (relative L^2 errors of 0.16331 and 0.16255, respectively) and negative R^2 scores. This demonstrates the challenge of learning high-dimensional flow field dynamics with limited training data when relying solely on data-driven approaches. Moreover, the base model CNO training in a data-driven manner achieves better performance with a relative L^2 error of 0.01284 and an R^2 score of 0.99304, demonstrating the effectiveness of the convolutional architecture for the 2D flow field.

Second, PI-CNO, which jointly optimizes data and physics losses, achieves a slight improvement over CNO (2.18% improvement), demonstrating the potential benefits of incorporating physical information. However, the improvement is modest, likely due to the gradient imbalance problem discussed in the Introduction.

Most importantly, our proposed PFT-CNO shows the best accuracy, with a relative L^2 error of 0.01178 (8.26% improvement over CNO and 6.21% improvement over PI-CNO) and an R^2 score of 0.99394. These results validate the effectiveness of decoupling physics loss and data loss optimization, allowing the model to leverage physical constraints during inference, avoiding the gradient imbalance issues that plague joint optimization approaches. Compared with the existing PFT-DeepONet method, PFT-CNO has significant advantages in spatial feature extraction ability due to its convolutional neural network architecture, which can better capture local and global correlations in high-dimensional space. The inference time of PFT-CNO is also comparable with that of CNO because they have the same architectures. Compared to the computation time of numerical solvers (0.32 s), the speed of neural operators has been greatly improved (at least 43 times), which is an important advantage over numerical solvers. This is also the core motivation for developing fluid solvers based on neural operators, which is to use the high efficiency of neural operators to accelerate simulations [48]. On the other hand, using physics loss directly to train the model instead of fine-tuning (PD-CNO) results in greater errors, as it ignores the data distribution and overfits on the physical equation. This highlights the importance of initialization via a data-driven method and the limitations of physics-only training.

Figure 2 provides visual comparisons of the predictions from all methods, further illustrating that PFT-CNO captures the complex flow fields more accurately. More visualizations are shown in Appendix C.

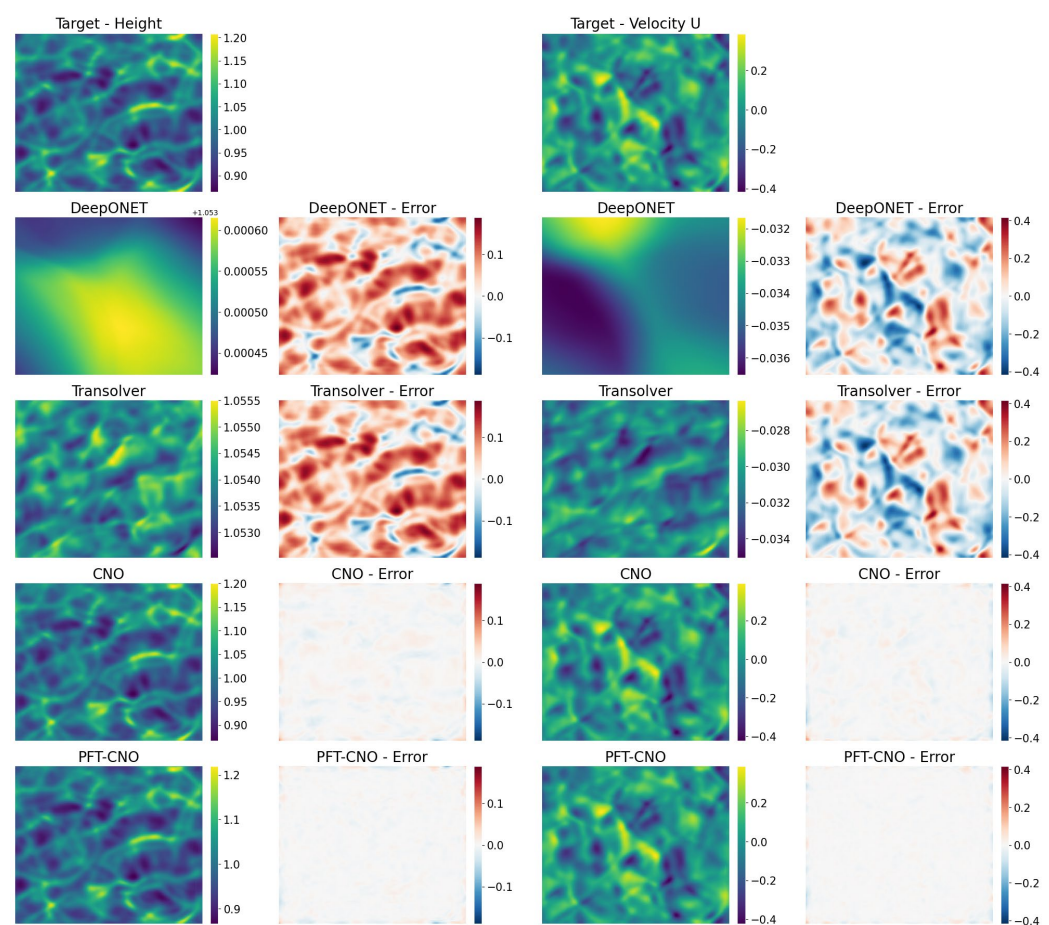


Figure 2. Visualizations of predictions on the shallow water equation. The **left** side shows the prediction and error of height h , while the **right** side shows the prediction and error of velocity field u . The first line shows the ground true values of both, and each line below is the prediction of the method. It can be seen that our method PFT-CNO has the smallest error in the last line.

4.3.3. Multi-Step Prediction Performance

To evaluate the long-term predictive capability and error accumulation behavior, we perform autoregressive multi-step predictions by iteratively applying each model to predict 10 consecutive time steps. At each step, the model takes the previous prediction as input to generate the next state, simulating real-world forecasting scenarios where the ground truth is unavailable.

Figure 3a shows the evolution of the relative L^2 error over the 10-step prediction horizon for all methods. The average relative L_2 error is shown in Table 2. The results reveal critical differences in error accumulation behavior.

The purely data-driven CNO method exhibits substantial error accumulation, with the relative L^2 error increasing rapidly over the prediction steps. This behavior is characteristic of autoregressive models where small errors at each step compound over time, leading to significant deviations from the true solution of the simulation.

In contrast, PFT-CNO demonstrates stability, maintaining consistently low errors throughout the prediction horizon. The physics-informed fine-tuning process enables the model to adapt to the evolving prediction distribution at each step, constraining the predictions to remain physically consistent even as the system state deviates from the training data. The superior multi-step performance of PFT-CNO further validates the effectiveness of our decoupled optimization strategy, demonstrating that physics-informed fine-tuning mitigates error accumulation and maintains prediction accuracy over extended time horizons.

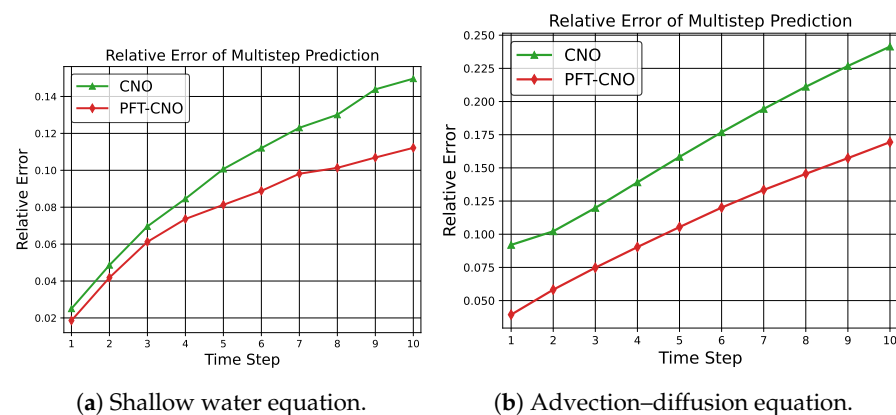


Figure 3. Multi-step prediction performance. PFT-CNO maintains consistently lower error accumulation throughout the prediction horizon.

Table 2. Average relative error of CNO and PFT-CNO on multi-step prediction. Best in **bold**.

Method	SWE ↓	ADE ↓
CNO	0.09250	0.19034
PFT-CNO	0.06551	0.15664

4.4. Advection–Diffusion Equation

The ADE is essential for modeling pollutant transport both in the ocean and atmosphere [25]. The ADE represents a more challenging benchmark due to its inherent complexity, coupling diffusion dynamics with the Navier–Stokes equations to simultaneously govern concentration transport, velocity fields, and pressure distributions. This multi-physics system poses substantial difficulties for learning-based methods, particularly in low-data regimes.

4.4.1. Data Collection

The ADE in two dimensions, coupled with the incompressible Navier–Stokes equations for the velocity and pressure fields, is given by

$$\begin{aligned}
\frac{\partial c}{\partial t} + u \frac{\partial c}{\partial x} + v \frac{\partial c}{\partial y} &= D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) + S, \\
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} &= -\frac{1}{\rho} \frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \\
\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} &= -\frac{1}{\rho} \frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right), \\
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0,
\end{aligned} \tag{14}$$

where $c(x, y, t)$ represents the concentration field, $u(x, y, t)$ and $v(x, y, t)$ denote the velocity components, $p(x, y, t)$ is the pressure field, $S(x, y, t)$ is the source term, D is the diffusion coefficient, ν represents the viscosity coefficient, and ρ denotes the fluid density.

We employ the same numerical discretization strategy as for the SWE, where the central difference scheme is used for spatial derivatives and the RK4 method is used for temporal integration. The simulations are conducted on a 128×128 spatial grid with periodic boundary conditions, using a time step of $\Delta t = 0.0005$ s and integrating for a total duration of $T = 0.6$ s, resulting in 1200 time steps per numerical simulation. To ensure that the system has achieved a developed flow state, we utilize the last 1000 time steps from each numerical simulation and downsample to 100 steps for training and evaluation. This is due to the small range of point source diffusion in the initial time step, leading to little difference between trajectories.

The initial conditions employ a two-stage sampling. The velocity fields are generated using the GRF, while the concentration field starts from zero and evolves through a continuous Gaussian point source. First, we generate the initial velocity field (u_0, v_0) using the same Fourier-based spectral method as described above, but with the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$ strictly enforced. Second, the concentration field c_0 is initialized as zero throughout the domain. The initial concentration distribution emerges through a continuous Gaussian point source term $S(x, y)$ in the ADE:

$$S(x, y) = \frac{Q}{2\pi\sigma^2} \exp\left(-\frac{(x - x_s)^2 + (y - y_s)^2}{2\sigma^2}\right), \tag{15}$$

where $Q = 50.0$ is the source strength, $\sigma = 0.1$ is the standard deviation controlling the spatial extent of the source, and (x_s, y_s) is the source location (set at the domain center). The concentration field evolves from this source term through the coupled advection–diffusion dynamics, naturally generating diverse plume patterns that depend on the underlying velocity field. Moreover, the diffusion coefficient D is randomly sampled from a uniform distribution over the range $[10^{-3}, 10^{-2}]$ to control the rate of concentration spreading. The viscosity coefficient ν is randomly sampled from a uniform distribution over the range $[10^{-5}, 10^{-3}]$ to govern the momentum diffusion. The dataset is similarly split into five training trajectories and five evaluation trajectories.

4.4.2. Single-Step Prediction Performance

From Table 3, we observe that purely data-driven approaches struggle considerably. DeepONet achieves only a relative L^2 error of 0.25895 with an R^2 of 0.46684, indicating poor predictive capability, while Transolver, despite its attention mechanism, still yields a relatively high error of 0.15784. Even the CNO baseline, which incorporates convolutional neural operators, achieves a relative L^2 error of 0.06477, suggesting that the limited training data are insufficient for the model to fully learn the intricate interactions between flow and diffusion. The same observation as for the SWE is that when only training the model with physics loss (PD-CNO), the accuracy decreases due to the inability to obtain information

about the data distribution. PI-CNO, optimized by combining data and physics loss, only brings a small improvement (about 2.8%) compared to the data-driven CNO.

Table 3. Performance comparison of different methods on ADE single-step prediction. Best in **bold** and second best underlined.

Method	Rel L^2 Error ↓	RMSE ↓	MAE ↓	R^2 ↑	Eval Time (s) ↓
DeepONet	0.25895	6.94667	1.51632	0.46684	0.00719
PFT-DeepONet	0.19702	5.35981	1.16720	0.68260	0.00736
Transolver	0.15784	3.34494	1.27491	0.87638	0.00224
CNO	0.06477	1.19996	0.41310	0.98409	0.00365
PD-CNO	0.06986	1.30461	0.44717	0.98120	0.00360
PI-CNO	<u>0.06297</u>	<u>1.16599</u>	<u>0.39620</u>	<u>0.98498</u>	<u>0.00344</u>
PFT-CNO	0.04767	0.90596	0.31588	0.99093	0.00358

In stark contrast, our proposed PFT-CNO method demonstrates the significant improvement of physics-informed fine-tuning for this coupled system, achieving a relative L^2 error of 0.04767 and R^2 score of 0.99093. This represents a 26.4% improvement over the CNO baseline and a 24.3% improvement over PI-CNO, significantly outperforming the 8.3% gain observed for the SWE. This improvement can be attributed to the physics and data decoupled optimization strategy, which allows the model to effectively leverage physical constraints from both the advection–diffusion equation and the Navier–Stokes equations during fine-tuning without suffering from gradient conflicts. By separating data-driven pretraining from physics-based refinement, PFT-CNO successfully mitigates the data scarcity issue while guiding consistency with the governing equations. Similarly to the SWE experiment, PFT-CNO learned spatial and multi-physics field information better than PFT-DeepONet, resulting in the smaller error. Compared to the computation time of numerical solvers (0.89 s), the speed of neural operators has been significantly improved (at least 120 times).

Figure 4 provides visual comparisons of the predictions from all methods, further illustrating the relatively better accuracy of PFT-CNO in capturing the complex flow field dynamics. More visualizations are shown in Appendix C.

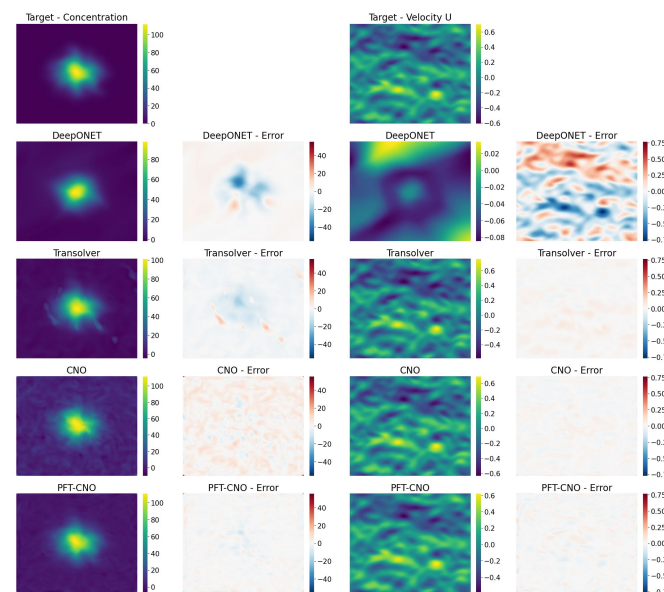


Figure 4. Visualizations of predictions on the advection–diffusion equation. The **left** side shows the prediction and error of concentration field c , while the **right** side shows the prediction and error of velocity field u . The first line shows the ground true values of both, and each line below is the prediction of the method. It can be seen that our method PFT-CNO has the smallest error in the last line.

4.4.3. Multi-Step Prediction Performance

We also perform autoregressive multi-step predictions to predict 10 time steps for evaluating the long-term prediction and error accumulation. The average relative L_2 error is shown in Table 2.

Figure 3b demonstrates how prediction errors evolve during the rollout for both methods. By incorporating physical constraints through fine-tuning, PFT-CNO achieves superior long-term accuracy with slower error accumulation. The relative L^2 error of PFT-CNO consistently outperforms that of CNO by a significant margin throughout the prediction sequence. This robust behavior stems from the physics-constrained refinement process, which guides compliance with governing equations and prevents predictions from violating fundamental conservation laws.

4.5. ANOVA and T-Test

Table 4 shows the statistical comparison of different models for the SWE and ADE. Through a paired T-test, ANOVA, and Cohen's d effect size, the results consistently indicate that PFT-CNO performs well in all comparisons.

For the SWE benchmark, PFT-CNO significantly outperforms the data-driven CNO with a large effect size (Cohen's $d = 5.04$, p -Value < 0.001), demonstrating that physics-informed fine-tuning substantially enhances predictive accuracy. Compared to PI-CNO, PFT-CNO shows more effective performance than direct physics-informed training. Against PFT-DeepONet, PFT-CNO shows a large effect size ($d = 2.59$, p -Value < 0.001), confirming that the convolutional architecture captures spatial features more effectively, which is suitable for high-dimensional flow field modeling.

Table 4. Comprehensive statistical comparison of different models.

	Model Comparison	Mean Difference	ANOVA (F-Statistic)	T-Test (T-Statistic)	Cohen's d
SWE	CNO vs. PFT-CNO	0.0277	1234.20	106.84	5.04
	CNO vs. PI-CNO	0.0101	3.42	14.50	0.68
	PFT-DeepONet vs. PFT-CNO	0.0681	2147.95	54.93	2.59
ADE	CNO vs. PFT-CNO	1.6386	266.51	33.56	1.58
	CNO vs. PI-CNO	1.2240	81.40	19.69	0.93
	PFT-DeepONet vs. PFT-CNO	2.8370	2329.92	46.60	2.20

Similar conclusions emerge for the ADE benchmark. PFT-CNO achieves large effect sizes compared to data-driven CNO (Cohen's $d = 1.58$), PI-CNO (Cohen's $d = 0.93$), and PFT-DeepONet (Cohen's $d = 2.20$), with all comparisons showing p -Value < 0.001 . The consistent improvement in these two tests demonstrates the robustness and generalizability of PFT-CNO to spatio-temporal flow field problems.

5. Conclusions

In this work, we propose PFT-CNO with physics-informed fine-tuning for high-dimensional flow field modeling, which addresses the gradient imbalance problem in physics-informed neural operators through a decoupled optimization strategy. By separating the data-driven pretraining phase from the physics-informed fine-tuning phase, our method enables effective integration of physical information without the competing gradient conflicts that plague joint optimization approaches. We validate PFT-CNO on two distinct flow systems: the shallow water equation and the advection–diffusion equation coupled with the Navier–Stokes equation. The experimental results demonstrate improvements over the baseline methods, achieving accuracy gains of 8.3% and 26.4% for the two systems, respectively. More importantly, PFT-CNO exhibits better stability in

long-term multi-step predictions, effectively mitigating error accumulation. The greater improvement on the advection–diffusion equation highlights the particular effectiveness of our method for multi-physics systems, where limited training data poses significant challenges for purely data-driven approaches. In addition, PFT-CNO demonstrates significant performance in statistical validation. A key advantage of our method is that it requires no additional training data beyond the original dataset yet improves prediction accuracy by leveraging governing equations and the inputs of evaluation data. This capability makes PFT-CNO valuable in real-world cases where data acquisition is expensive or limited but the underlying physical laws are established.

6. Limitations and Future Perspectives

While our proposed PFT-CNO method demonstrates significant improvements, several limitations warrant discussion and suggest directions for future research. First, the current experimental setup employs training and evaluation datasets generated from the same numerical solver with variations only in initial conditions and physical parameters (e.g., viscosity and diffusion coefficients). Although these variations introduce distinctions, the training and evaluation distributions remain relatively similar, which may not fully demonstrate the potential advantages of physics-informed fine-tuning. We anticipate that when the model encounters more substantial distribution shifts, such as flow fields governed by different but related PDEs or significantly different parameter regimes, the benefits of physics-informed fine-tuning will be more pronounced. Moreover, a particularly promising application lies in bridging the simulation-to-reality gap. Numerical simulations inherently contain discretization errors and modeling approximations that deviate from real-world measurements. In such scenarios, physics-informed fine-tuning can constrain predictions to remain within physically plausible ranges when adapting from simulation-trained models to real-world modeling, potentially offering substantial accuracy improvements for practical deployment.

Second, the current implementation performs physics-informed fine-tuning as a separate phase before inference, rather than integrating it directly into the inference process. This offline fine-tuning strategy, while effective, does not fully exploit the potential for online adaptation. Future work can explore online physics-informed fine-tuning, where the model dynamically updates its parameters upon receiving each new input during inference, using both the input state and governing equations to perform real-time adaptation. This capability would be particularly critical for control applications and feedback decision-making systems, where the model must continuously adapt to evolving system states and operating conditions. Additionally, the current method employs full-parameter fine-tuning, which incurs considerable computational costs and time. Inspired by recent advances in test-time training and low-rank adaptations, future research can investigate strategies that update only a small subset of model parameters during inference. Such approaches can dramatically reduce computational cost while maintaining the benefits of physics-informed adaptation, making the method more practical for large-scale models and real-time applications.

Moreover, the core of our work is to propose a novel methodology. We validated the method on two systems in Section 4 and cylinder wake flow in Appendix D. In the future, this method can be applied to and developed for more complex flow field modeling, such as turbulence and multi-phase flow modeling [49,50], to improve the accuracy of existing neural operator models and reduce computational time compared to classical numerical solvers.

Author Contributions: H.F.: conceptualization (equal); data curation; formal analysis (equal); investigation; methodology; project administration (equal); software; validation; visualization; and writing—original draft. Y.Z.: conceptualization (equal); funding acquisition (equal); formal analysis

(equal); project administration (equal); and writing—review and editing (equal). D.F.: conceptualization (equal); funding acquisition (equal); formal analysis (equal); project administration (equal); and writing—review and editing (equal). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Zhejiang Province Leading Goose Plan Project (Grant No. 2025C02017) and the Research Center for Industries of the Future at Westlake University under Grant No. WU2024C001.

Data Availability Statement: The data that support this work will be made available upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Pseudocode

The pseudocode of the proposed physics-informed fine-tuned neural operator is shown in Algorithm A1.

Algorithm A1 Physics-informed fine-tuned neural operator

Require: Training dataset $\mathbb{D} = \{(u_t^{(i)}, u_{t+1}^{(i)})\}_{i=1}^{N_{train}}$

Require: Evaluation dataset $\mathbb{B} = \{u_t^{(j)}\}_{j=1}^{N_{eval}}$ (only inputs)

Require: Neural operator $\mathcal{G}_\theta$, learning rates η_{pre}, η_{ft}

Require: Optimizers Opt_{pre} (for pretraining), Opt_{ft} (for fine-tuning)

Require: Number of pretraining epochs E_{pre} , fine-tuning epochs E_{ft}

1: **Phase I: Data-Driven Pretraining**

2: **for** $epoch = 1$ to E_{pre} **do**

3: **for** each batch $(u_t^{(i)}, u_{t+1}^{(i)}) \in \mathbb{D}$ **do**

4: $\hat{u}_{t+1}^{(i)} \leftarrow \mathcal{G}_\theta(u_t^{(i)})$ ▷ Forward

5: $\mathcal{L}_{data} \leftarrow \frac{1}{N_{train}} \sum_i \frac{\|\hat{u}_{t+1}^{(i)} - u_{t+1}^{(i)}\|^2}{\|u_{t+1}^{(i)}\|^2}$ ▷ Compute data loss

6: $\theta \leftarrow \text{Opt}_{pre}.\text{step}(\nabla_\theta \mathcal{L}_{data})$ ▷ Update parameters

7: **end for**

8: **end for**

9: **Phase II: Physics-Informed Fine-Tuning**

10: $\theta_{ft} \leftarrow \theta$ ▷ Initialize from pretrained model

11: **for** $epoch = 1$ to E_{ft} **do**

12: **for** each evaluation input batch $u_t^{(j)} \in \mathbb{B}$ **do**

13: $\hat{u}_{t+1}^{(j)} \leftarrow \mathcal{G}_{\theta_{ft}}(u_t^{(j)})$ ▷ Forward

14: $\mathcal{L}_{physics} \leftarrow \left\| \frac{\partial \hat{u}_{t+1}^{(j)}}{\partial t} - \mathcal{F}(u_t^{(j)}, \nabla u_t^{(j)}, \nabla^2 u_t^{(j)}) \right\|^2$ ▷ Compute physics loss

15: $\theta_{ft} \leftarrow \text{Opt}_{ft}.\text{step}(\nabla_{\theta_{ft}} \mathcal{L}_{physics})$ ▷ Fine-tune parameters

16: **end for**

17: **end for**

Appendix B. Model Parameters

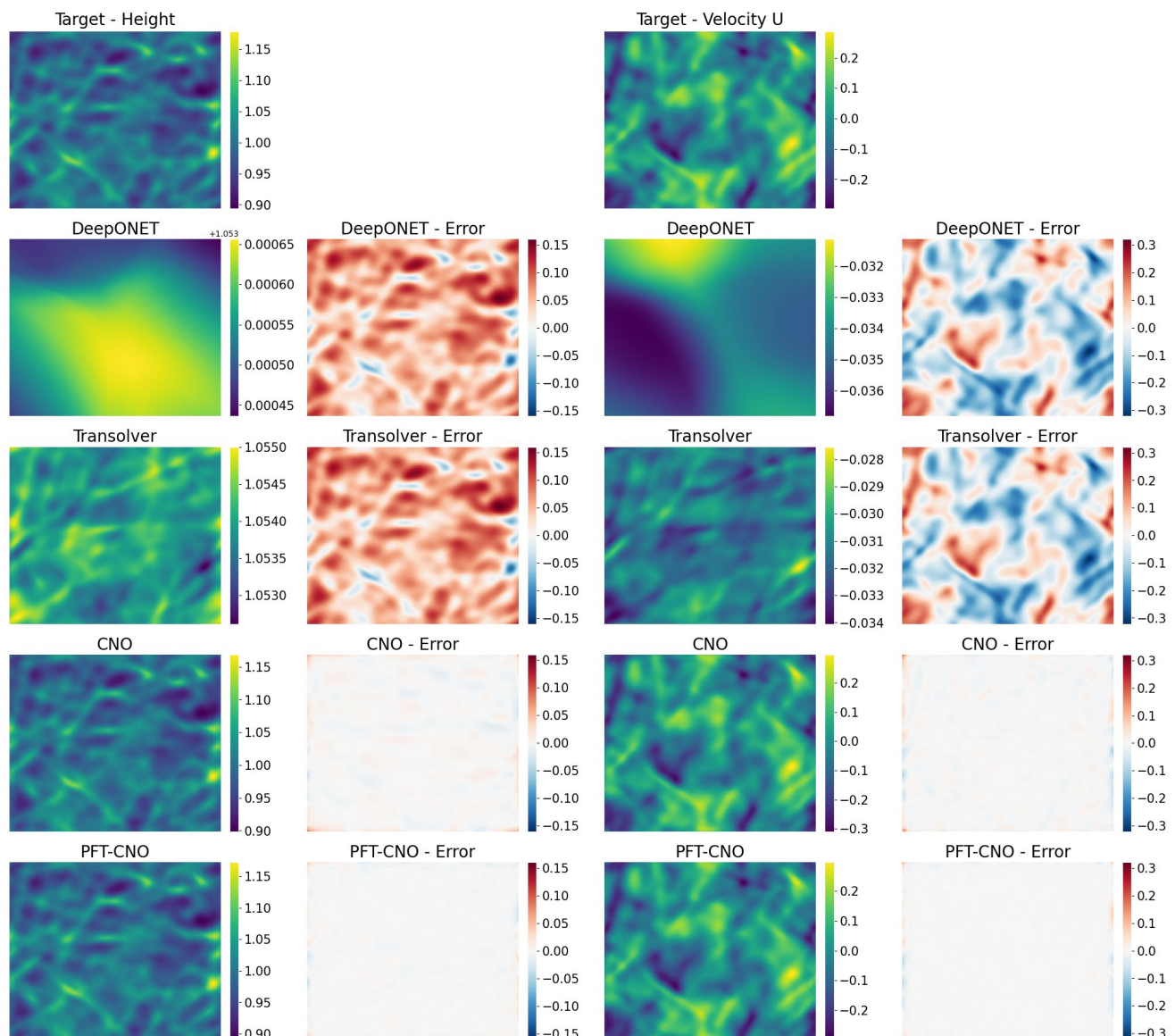
The hyperparameters of PFT-CNO both for training/fine-tuning and the model architecture are listed in Table A1.

Table A1. Hyperparameters for PFT-CNO training (fine-tuning) and model architecture.

Training Hyperparameters		Model Hyperparameters	
Parameter	Value	Parameter	Value
Optimizer	Adam	N_{block}	3
Learning rate	0.001	$N_{residual_layer}$	3
LR scheduler	Step	N_{neck_layer}	5
Scheduler step size	100	Channel multiplier	32
Gamma	0.5	Activation	LeakyReLU
Train batch size	16	Kernel size	3
Eval batch size	64	Cutoff denominator	2.0001
Training updates	1000	Filter size	6
Physics loss weight (λ)	1	LReLU upsampling	2
		Half-width mult.	0.8
		Batch norm	True
		Latent dim	128

Appendix C. Visualizations of Results

We present more visualizations of prediction results here.

**Figure A1.** Visualizations of predictions on the shallow water equation.

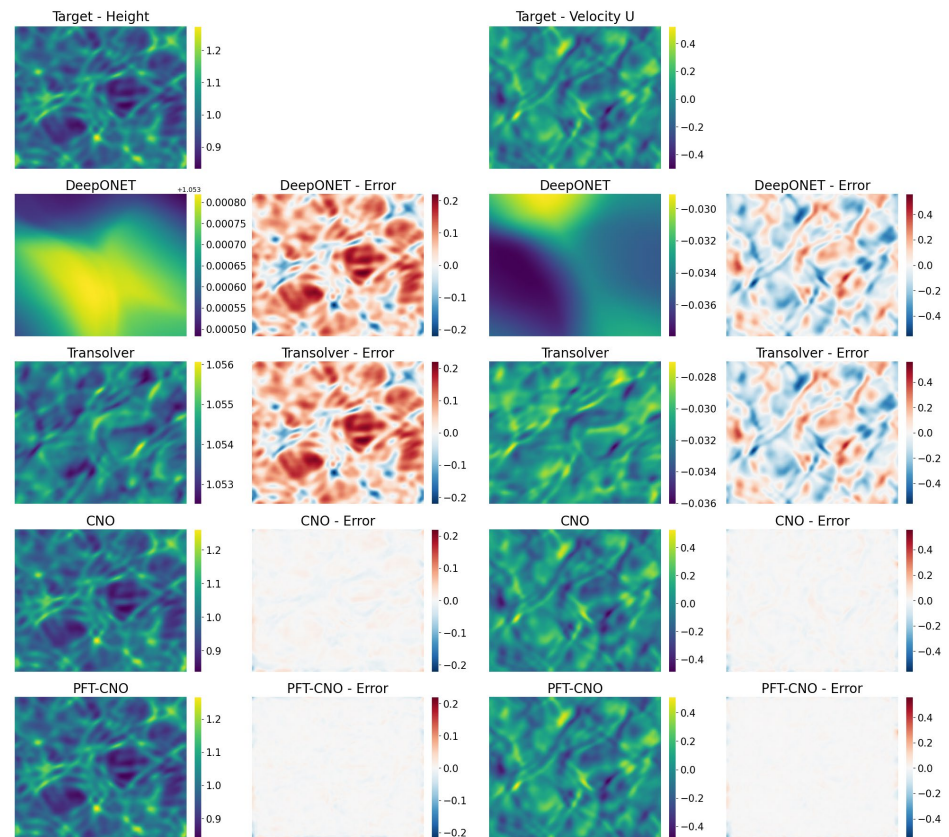


Figure A2. Visualizations of predictions on the shallow water equation.

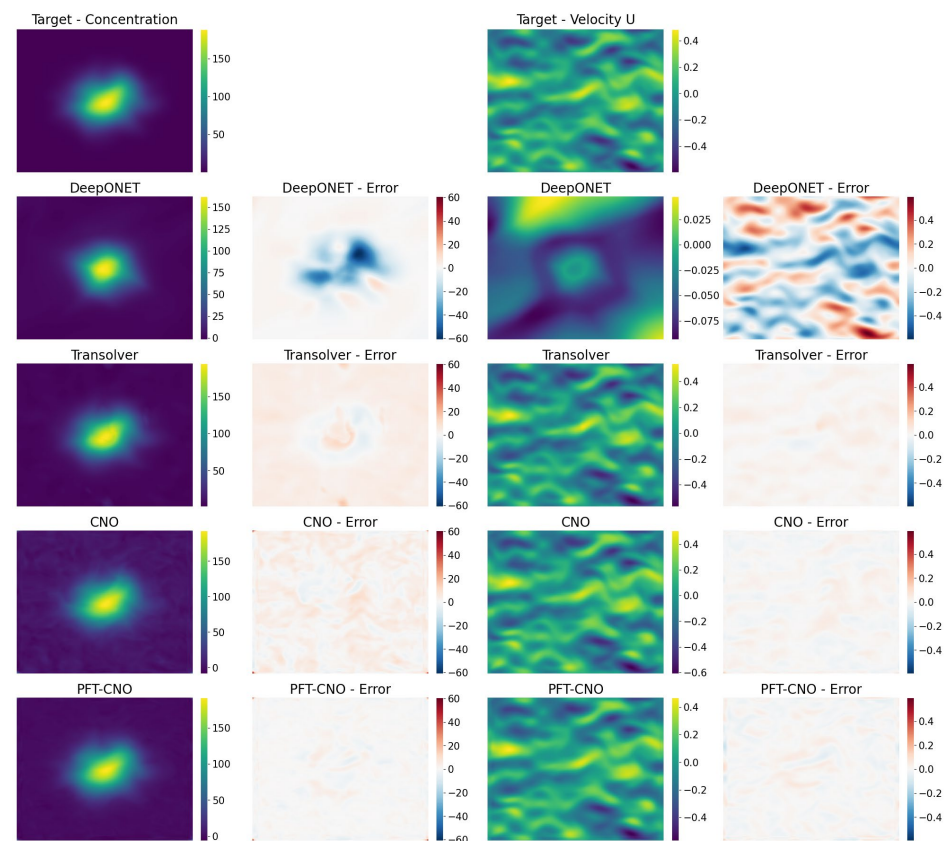


Figure A3. Visualizations of predictions on the advection–diffusion equation.

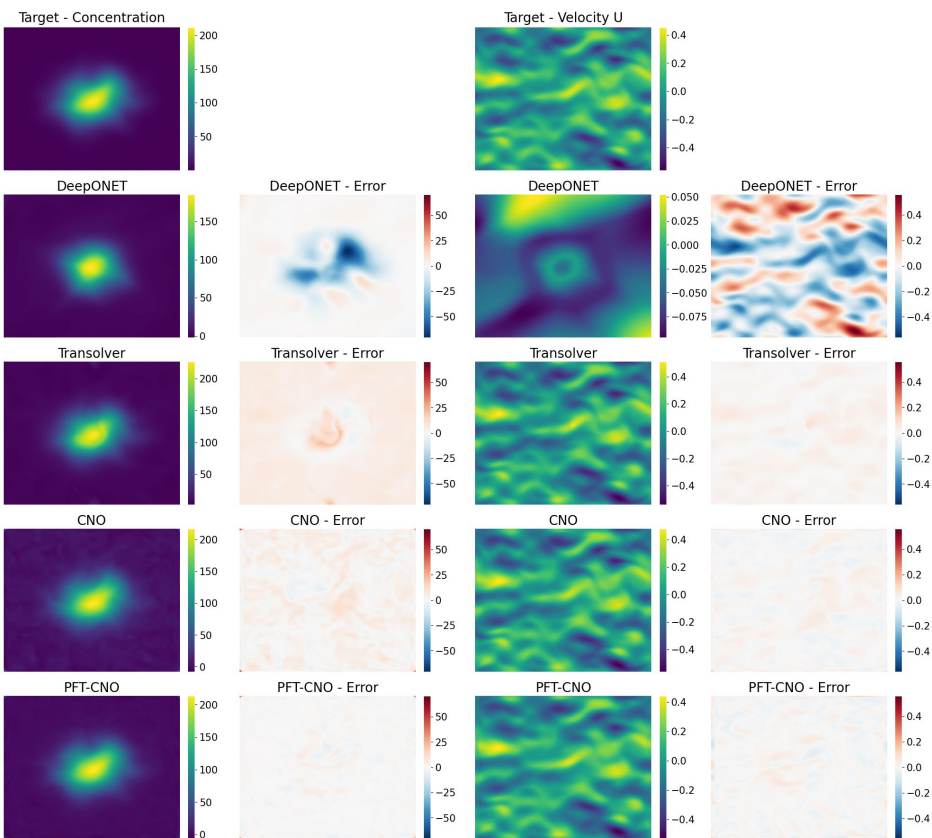


Figure A4. Visualizations of predictions on the advection–diffusion equation.

Appendix D. Experiment on Cylinder Wake Flow

In addition to the experiments on the two systems mentioned in the main text, we also conduct flow field modeling on more complex cylinder wake flow data to further demonstrate the advantages of the proposed method. Here, we use the BDIM solver [51] to calculate 10 numerical simulations of the u and v velocity fields and pressure p field for flow around a cylinder, with Reynolds numbers including [1781, 1875, 2062, 2156, 2250, 2343, 2531, 2718, 2812, 2906]. The first five configurations are used as the training dataset, and the last five configurations are used as the testing dataset. The method includes the model and the calculation method of physics loss, which are consistent with those described in Section 3.4.

The experimental results are shown in Table A2. We can see that PFT-CNO reduces the error by 49.8% compared to data-driven CNO. Therefore, the conclusion is consistent with the previous system modelings. Our proposed physics-informed fine-tuning strategy for flow field modeling can improve the accuracy of neural operators.

Table A2. Performance comparison of different methods on cylinder wake flow prediction. Best in bold.

Method	Rel L^2 Error ↓	RMSE ↓	MAE ↓	R^2 ↑	Eval Time (s) ↓
DeepONet	0.30349	0.05495	0.02918	0.62043	0.00192
CNO	0.08664	0.01586	0.00817	0.96837	0.00178
PFT-CNO	0.04348	0.00785	0.00387	0.99225	0.00171

References

1. Uchida, T.; Ohya, Y. Numerical simulation of atmospheric flow over complex terrain. *J. Wind. Eng. Ind. Aerodyn.* **1999**, *81*, 283–293. [CrossRef]

2. Klein, R. Scale-dependent models for atmospheric flows. *Annu. Rev. Fluid Mech.* **2010**, *42*, 249–274. [CrossRef]

3. Liu, X.; Zhang, Y.; Song, D.; Yang, H.; Li, X. A multifractal cascade model for energy evolution and dissipation in ocean turbulence. *J. Mar. Sci. Eng.* **2023**, *11*, 1768. [\[CrossRef\]](#)
4. Xu, H.; Wang, M.; Zhou, Z.; Sun, T.; Zhang, G. Unsteady Flow and Loading Characteristics of Rotating Spheres During Underwater Ejection. *J. Mar. Sci. Eng.* **2025**, *13*, 2331. [\[CrossRef\]](#)
5. Maximillian, J.; Brusseau, M.L.; Glenn, E.P.; Matthias, A.D. Pollution and environmental perturbations in the global system. In *Environmental and Pollution Science*, 3rd ed.; Academic Press: Cambridge, MA, USA, 2019; pp. 457–476.
6. Van Gennip, S.J.; Popova, E.E.; Yool, A.; Pecl, G.T.; Hobday, A.J.; Sorte, C.J.B. Going with the flow: The role of ocean circulation in global marine ecosystems under a changing climate. *Glob. Change Biol.* **2017**, *23*, 2602–2617. [\[CrossRef\]](#)
7. Bodnar, C.; Bruinsma, W.P.; Lucic, A.; Stanley, M.; Allen, A.; Brandstetter, J.; Garvan, P.; Riechert, M.; Weyn, J.A.; Dong, H.; et al. A foundation model for the Earth system. *Nature* **2025**, *641*, 1180–1187. [\[CrossRef\]](#)
8. Liu, P.; Georgiadis, M.C.; Pistikopoulos, E.N. Advances in energy systems engineering. *Ind. Eng. Chem. Res.* **2011**, *50*, 4915–4926. [\[CrossRef\]](#)
9. Brunton, S.L.; Noack, B.R.; Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **2020**, *52*, 477–508. [\[CrossRef\]](#)
10. Cai, Z. On the finite volume element method. *Numer. Math.* **1990**, *58*, 713–735. [\[CrossRef\]](#)
11. Teixeira, F.L.; Sarris, C.; Zhang, Y.; Na, D.-Y.; Berenger, J.-P.; Su, Y.; Okoniewski, M.; Chew, W.C.; Backman, V.; Simpson, J.J. Finite-difference time-domain methods. *Nat. Rev. Methods Prim.* **2023**, *3*, 75. [\[CrossRef\]](#)
12. Pirozzoli, S. Numerical methods for high-speed flows. *Annu. Rev. Fluid Mech.* **2011**, *43*, 163–194. [\[CrossRef\]](#)
13. Cheng, M.; Fang, F.; Pain, C.C.; Navon, I.M. Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network. *Comput. Methods Appl. Mech. Eng.* **2020**, *365*, 113000. [\[CrossRef\]](#)
14. Vinuesa, R.; Brunton, S.L.; McKeon, B.J. The transformative potential of machine learning for experiments in fluid mechanics. *Nat. Rev. Phys.* **2023**, *5*, 536–545. [\[CrossRef\]](#)
15. Feng, H.; Wang, Y.; Fan, D. How to re-enable PDE loss for physical systems modeling under partial observation. *Proc. Aaaai Conf. Artif. Intell.* **2025**, *39*, 182–190. [\[CrossRef\]](#)
16. Gilpin, W. Generative learning for nonlinear dynamics. *Nat. Rev. Phys.* **2024**, *6*, 194–206. [\[CrossRef\]](#)
17. Solomatine, D.P.; Ostfeld, A. Data-driven modelling: Some past experiences and new approaches. *J. Hydroinform.* **2008**, *10*, 3–22. [\[CrossRef\]](#)
18. Marcus, E.; Teuwen, J. Artificial intelligence and explanation: How, why, and when to explain black boxes. *Eur. J. Radiol.* **2024**, *173*, 111393. [\[CrossRef\]](#)
19. Li, Z.; Zheng, H.; Kovachki, N.; Jin, D.; Chen, H.; Liu, B.; Azizzadenesheli, K.; Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM/IMS J. Data Sci.* **2024**, *1*, 1–27. [\[CrossRef\]](#)
20. Goswami, S.; Bora, A.; Yu, Y.; Karniadakis, G.E. Physics-informed deep neural operator networks. In *Machine Learning in Modeling and Simulation: Methods and Applications*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 219–254.
21. Rosofsky, S.G.; Al Majed, H.; Huerta, E.A. Applications of physics informed neural operators. *Mach. Learn. Sci. Technol.* **2023**, *4*, 025022. [\[CrossRef\]](#)
22. Wang, S.; Teng, Y.; Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* **2021**, *43*, A3055–A3081. [\[CrossRef\]](#)
23. Wang, S.; Yu, X.; Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *J. Comput. Phys.* **2022**, *449*, 110768. [\[CrossRef\]](#)
24. Sahoo, M.; Chakraverty, S. Dynamics of tsunami wave propagation in uncertain environment. *Comput. Appl. Math.* **2024**, *43*, 266. [\[CrossRef\]](#)
25. Feng, H.; Hu, P.; Wang, Y.; Fan, D.; Wu, T.; Zhang, Y. Physics-informed super-resolution and forecasting method based on inaccurate partial differential equations and partial observation. *Phys. Fluids* **2025**, *37*, 066625. [\[CrossRef\]](#)
26. Raonic, B.; Molinaro, R.; De Ryck, T.; Rohner, T.; Bartolucci, F.; Alaifari, R.; Mishra, S.; de Bézenac, E. Convolutional neural operators for robust and accurate learning of PDEs. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 77187–77200.
27. Garcia-Fernandez, R.; Portal-Porras, K.; Irigaray, O.; Ansa, Z.; Fernandez-Gamiz, U. CNN-based flow field prediction for bus aerodynamics analysis. *Sci. Rep.* **2023**, *13*, 21213. [\[CrossRef\]](#)
28. Deng, Z.; Chen, Y.; Liu, Y.; Kim, K.C. Time-resolved turbulent velocity field reconstruction using a long short-term memory (LSTM)-based artificial intelligence framework. *Phys. Fluids* **2019**, *31*, 075108. [\[CrossRef\]](#)
29. Zhuang, D.L.; Wu, Y.; Wang, Q. A recurrent neural network for particle trajectory prediction in complex flow fields. *Phys. Fluids* **2025**, *37*, 105101. [\[CrossRef\]](#)
30. Li, J.; Li, Y.; Liu, T.; Zhang, D.; Xie, Y. Multi-fidelity graph neural network for flow field data fusion of turbomachinery. *Energy* **2023**, *285*, 129405. [\[CrossRef\]](#)

31. Wu, T.; Wang, Q.; Zhang, Y.; Ying, R.; Cao, K.; Susic, R.; Jalali, R.; Hamam, H.; Maucec, M.; Leskovec, J. Learning large-scale subsurface simulations with a hybrid graph network simulator. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 4184–4194.
32. Li, Z.; Huang, D.Z.; Liu, B.; Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries. *J. Mach. Learn. Res.* **2023**, *24*, 1–26.
33. Lu, L.; Jin, P.; Pang, G.; Zhang, Z.; Karniadakis, G.E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **2021**, *3*, 218–229. [[CrossRef](#)]
34. Wu, H.; Luo, H.; Wang, H.; Wang, J.; Long, M. Transolver: A fast transformer solver for PDEs on general geometries. In Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024; pp. 53681–53705.
35. Hu, P.; Wang, R.; Zheng, X.; Zhang, T.; Feng, H.; Feng, R.; Wei, L.; Wang, Y.; Ma, Z.-M.; Wu, T. Wavelet Diffusion Neural Operator. In Proceedings of the Thirteenth International Conference on Learning Representations, Singapore, 24–28 April 2025.
36. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [[CrossRef](#)]
37. Cai, S.; Wang, Z.; Wang, S.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks for heat transfer problems. *J. Heat Transf.* **2021**, *143*, 060801. [[CrossRef](#)]
38. Rasht-Behesht, M.; Huber, C.; Shukla, K.; Karniadakis, G.E. Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions. *J. Geophys. Res. Solid Earth* **2022**, *127*, e2021JB023120. [[CrossRef](#)]
39. Wang, S.; Wang, H.; Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Sci. Adv.* **2021**, *7*, eabi8605. [[CrossRef](#)] [[PubMed](#)]
40. Ma, X.; Alkhalifah, T. An effective physics-informed neural operator framework for predicting wavefields. *arXiv* **2025**, arXiv:2507.16431. [[CrossRef](#)]
41. Yamazaki, Y.; Harandi, A.; Muramatsu, M.; Viardin, A.; Apel, M.; Brepols, T.; Reese, S.; Rezaei, S. A finite element-based physics-informed operator learning framework for spatiotemporal partial differential equations on arbitrary domains. *Eng. Comput.* **2025**, *41*, 1–29. [[CrossRef](#)]
42. Zhang, Z.; Moya, C.; Lu, L.; Lin, G.; Schaeffer, H. Deepnet as a multi-operator extrapolation model: Distributed pretraining with physics-informed fine-tuning. *J. Comput. Phys.* **2025**, *547*, 114537. [[CrossRef](#)]
43. Kang, S.; Park, G.; Kum, D. A Transfer Learning Physics-Informed Neural Network Framework for Solving Multi-Coupled Partial Differential Equations with Complex Geometries: Fuel Cell Case Study. *SSRN* **2025**. [[CrossRef](#)]
44. Wang, Y.; Bai, J.; Eshaghi, M.S.; Anitescu, C.; Zhuang, X.; Rabczuk, T.; Liu, Y. Transfer Learning in Physics-Informed Neural Networks: Full Fine-Tuning, Lightweight Fine-Tuning, and Low-Rank Adaptation. *Int. J. Mech. Syst. Dyn.* **2025**, *5*, 212. [[CrossRef](#)]
45. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
46. Ren, P.; Rao, C.; Liu, Y.; Ma, Z.; Wang, Q.; Wang, J.X.; Sun, H. PhySR: Physics-informed deep super-resolution for spatiotemporal data. *J. Comput. Phys.* **2023**, *492*, 112438. [[CrossRef](#)]
47. Akbar, Z.; Murtaza, N.; Pasha, G.A.; Iqbal, S.; Ghumman, A.R.; Abbas, F.M. Predicting scour depth in a meandering channel with spur dike: A comparative analysis of machine learning techniques. *Phys. Fluids* **2025**, *37*, 045158. [[CrossRef](#)]
48. Azizzadenesheli, K.; Kovachki, N.; Li, Z.; Liu-Schiaffini, M.; Kossai, J.; Anandkumar, A. Neural operators for accelerating scientific simulations and design. *Nat. Rev. Phys.* **2024**, *6*, 320–328. [[CrossRef](#)]
49. Wang, Y.; Li, Z.; Yuan, Z.; Peng, W.; Liu, T.; Wang, J. Prediction of turbulent channel flow using Fourier neural operator-based machine-learning strategy. *Phys. Rev. Fluids* **2024**, *9*, 084604. [[CrossRef](#)]
50. Zhang, K.; Zuo, Y.; Zhao, H.; Ma, X.; Gu, J.; Wang, J.; Yang, Y.; Yao, C.; Yao, J. Fourier neural operator for solving subsurface oil/water two-phase flow partial differential equation. *SPE J.* **2022**, *27*, 1815–1830. [[CrossRef](#)]
51. Weymouth, G.D.; Yue, D.K. Boundary data immersion method for Cartesian-grid simulations of fluid-body interaction problems. *J. Comput. Phys.* **2011**, *230*, 6233–6247. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.