Learning Data-Efficient and Generalizable Neural Operators via Fundamental Physics Knowledge

Siying Ma
Simon Fraser University
siying_ma@sfu.ca

Mehrdad M. Zadeh
Simon Fraser University

Mauricio Soroco
Simon Fraser University

Wuyang Chen
Simon Fraser University
wuyang_chen@sfu.ca

Jiguo Cao
Simon Fraser University

Vijay Ganesh
Georgia Institute of Technology

Abstract

Recent advances in scientific machine learning (SciML) have established neural operators (NOs) as powerful surrogates for partial differential equations (PDEs) governed dynamics. However, existing methods largely ignore the fundamental physical principles underlying these equations. We propose a multiphysics training framework that jointly learns from both full PDEs and their simplified basic forms. This approach improves data efficiency, reduces predictive error, and enhances out-of-distribution (OOD) generalization, including parameter shifts and synthetic-to-real transfer. Our method is architecture-agnostic and achieves over 11.5% and up to 67.5% improvement in nRMSE across diverse PDEs. Through extensive experiments, we show that explicitly incorporating fundamental physics knowledge substantially strengthens the generalization of neural operators. We will release models and data at https://sites.google.com/view/sciml-fundemental-pde/home.

1 Introduction

Recent advances in scientific machine learning (SciML) have broadened traditional machine learning (ML) for modeling physical systems, using deep neural networks (DNNs) especially neural operators (NOs) [20, 30, 19, 4] as fast, accurate surrogates for solving partial differential equations (PDEs) [33, 12, 18, 30]. However, compared to numerical methods, **a key disadvantage of recent data-driven SciML models is their limited integration of fundamental physical knowledge**. Numerical solvers, though tailored to specific PDEs or discretizations, inherently preserve physical laws (e.g., conservation, symmetry), ensuring consistent simulations across diverse conditions (physical parameters, boundaries, geometries, etc.) [17, 14, 26, 16]. In contrast, data-driven models, despite learning across PDE types (e.g., via multiphysics pretraining in SciML foundation models [23, 15]), remain sensitive to training distributions, degrading under distribution shifts [36, 3] and requiring large, diverse datasets. This gap introduces three key challenges: 1) **High data demands**; 2) **Physical inconsistency** in long-term predictions; 3) **Poor generalization** with unseen simulation settings.

Motivated by the above challenges, we asked two core scientific questions:

> ***Q1**: Can neural operators **understand both** original PDEs and fundamental physics knowledge?*
> ***Q2**: Can neural operators benefit from **explicit learning** of fundamental physics knowledge?*

By evaluating publicly released SciML foundation models (Fig. 1), we find a strong correlation (0.967) between errors on original PDEs and their basic terms. Yet, because these basic terms are absent from the training data, absolute errors remain much higher, indicating that although the models

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

transfer across multiple physics, they *fail* to capture the fundamental PDE components underlying more complex equations.

Our finding in (Fig. 1) motivates us to explicitly enforce the understanding of fundamental physical knowledge in neural operators. The key idea is to *identify physically plausible basic terms* that can be *decomposed from original PDEs*, and *explicitly incorporate their simulations during training*. Our experiments demonstrate that these fundamental physical terms encode rich physical knowledge, which can not only be utilized without incurring additional computational costs, but also *widely offer substantial and multifaceted benefits*, including 1) **data efficiency**, 2) **long-term physical consistency**, 3) **strong generalization in OOD and and real-world scenarios**.

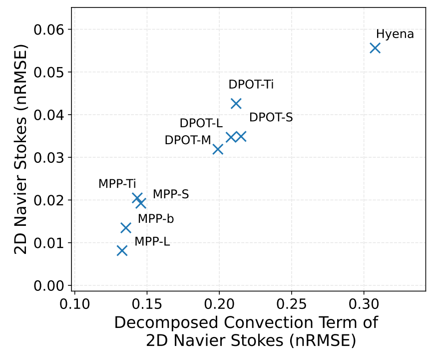

Figure 1: Neural operators and SciML foundation models (MPP [23], DPOT [15], Hyena [31]) exhibit correlated yet worse performance on fundamental physics (2D incompressible Navier Stokes).

Related Works. Prior work advanced PINNs [33] and NOs (DeepONet [22], FNO [20]), but these approaches struggled with optimization or required large datasets. Recent SciML foundation models [23, 15] extended to multiphysics training, yet primarily focused on generalization rather than data efficiency. Meanwhile, OOD studies showed that NOs remain fragile under parameter shifts [36, 3]. In contrast, we explicitly incorporate fundamental physical knowledge via decomposed PDE forms, leading to principled, architecture-agnostic improvements in both data efficiency and generalization.

2 Methods

For time-dependent PDEs, the solution is a vector-valued mapping $\boldsymbol{v} : \mathcal{T} \times \mathcal{S} \times \Theta \to \mathbb{R}^d$, defined on temporal domain $\mathcal{T}$, spatial domain $\mathcal{S}$, and parameter space Θ, with d the number of dependent variables. Numerical solvers compute $\boldsymbol{v}_\theta(t, \cdot)$ from $\ell \geq 1$ past steps, enabling finite-difference approximations: $\mathcal{N}_\theta = [\boldsymbol{v}_\theta(t - \ell \cdot \Delta t, \cdot), \ldots, \boldsymbol{v}_\theta(t - \Delta t, \cdot)] \mapsto \boldsymbol{v}_\theta(t, \cdot)$, where Δt is the temporal resolution. SciML aims to learn a surrogate operator $\widehat{\mathcal{N}}_{\theta,\phi}$, parameterized by physical parameters θ and learnable weights ϕ, that approximates this mapping. Given N simulations, $\mathcal{D} := \left\{\boldsymbol{v}^{(i)}([0 : t_{\max}], \cdot) \mid i = 1, \ldots, N\right\}$, the model is trained by minimizing a loss L, often the normalized root mean squared error: $\mathrm{nRMSE} = \frac{\|\boldsymbol{v}_{\text{pred}} - \boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2}$, where $\boldsymbol{v}_{\text{pred}}$ is the model prediction.

Defining Fundamental Physical Knowledge of PDEs. Let us consider the second-order PDE:

$$\sum_{i,j=1}^{n} a_{ij}(\boldsymbol{x}, \boldsymbol{u}, \nabla \boldsymbol{u}) \frac{\partial^2 \boldsymbol{u}}{\partial x_i \partial x_j} + \sum_{i=1}^{n} b_i(\boldsymbol{x}, \boldsymbol{u}, \nabla \boldsymbol{u}) \frac{\partial \boldsymbol{u}}{\partial x_i} + c(\boldsymbol{x}, \boldsymbol{u}, \nabla \boldsymbol{u}) = f(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{u}$ is the target solution, with $\boldsymbol{x} \in \mathbb{R}^n$ the physical space (e.g., $n = 3$ for 2D time-dependent PDEs). The coefficients a_{ij}, b_i, c ("physical parameters") govern the dynamics; mismatches between training and testing values cause domain shifts, leading to out-of-distribution (OOD) simulations. Finally, f denotes an external forcing function [27].

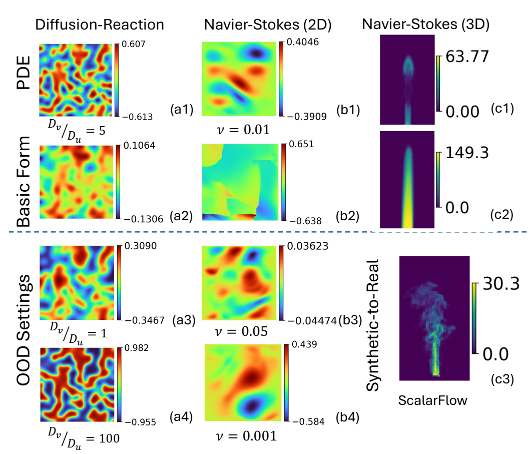

Figure 2: Visualizations of simulations of PDEs and their decomposed basic forms. From left to right: activator concentration, velocity, and density.

In our paper, based on Equation 1, we retain terms capturing dominant dynamics while discarding those causing stiffness, high cost, or limited impact on pattern formation. More generally, we **define fundamental physical knowledge as *any simplified variant of Equation 1 obtained by removing high-order, nonlinear, or forcing terms***. These basic forms simulate far more efficiently yet remain physically relevant, serving as cost-effective physics-based data augmentation. Unlike recent SciML foundation models that pretrain on unrelated PDEs [23, 15], our method exploits decomposed forms of the target PDE, ensuring task-aligned efficiency.

PDE 1: Diffusion-Reaction. The Diffusion–Reaction equation models activator–inhibitor systems that arise in chemistry, biology, and ecology, describing how species diffuse through a medium while undergoing local reactions. In our work, we adopt the FitzHugh–Nagumo variant:

$$\partial_t u = D_u \partial_{xx} u + D_u \partial_{yy} u + u - u^3 - k - v, \quad \partial_t v = D_v \partial_{xx} v + D_v \partial_{yy} v + u - v. \tag{2}$$

where u and v represent the concentrations of activator and inhibitor, respectively, D_u, D_v are diffusion coefficients and $k = 5 \times 10^{-3}$.

Decomposed Basic Form. We simplify Equation 2 by omitting nonlinear reaction terms, yielding pure diffusion: $\partial_t u = D_u(\partial_{xx} u + \partial_{yy} u), \quad \partial_t v = D_v(\partial_{xx} v + \partial_{yy} v)$. The diffusion term is the primary source of pattern formation, facilitating transport and stabilization. It encapsulates critical properties like isotropic spreading and mass conservation, providing inductive bias. Nonlinear reaction terms will introduce stiffness, resulting in increasing computational cost. By omitting nonlinear terms, the system becomes linear and more amenable to efficient numerical solutions.

Physical Scenarios. Pattern formation depends on the ratio of $\frac{D_v}{D_u}$ [29, 1, 13]. Classical Turing instability arises when $D_v \gg D_u$. Following prior work [24], we set $D_u = 10^{-3}$ and study $\frac{D_v}{D_u} = 5$ (in-distribution) and $\frac{D_v}{D_u} \in \{1, 100\}$ (OOD).

PDE 2: Incompressible Navier-Stokes. The Navier–Stokes equations govern the dynamics of fluid flow by enforcing mass conservation and the momentum of fluid parcels.

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nu \nabla^2 \boldsymbol{u} - \frac{1}{\rho} \nabla p + \boldsymbol{f}. \tag{3}$$

$\boldsymbol{u}$ is the velocity field, ν the dynamic viscosity, ρ the density, p the pressure, and $\boldsymbol{f}$ the external force.

Decomposed Basic Form. To isolate fundamental nonlinear transport mechanisms and reduce computational complexity, we simplify Equation 3 by omitting the pressure term (incompressibility via projection) $\frac{1}{\rho}\nabla p$ and diffusion term $\nu \nabla^2 \boldsymbol{u}$: $\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \boldsymbol{f}$. This form emphasizes inertial advection with external forcing, approximating high Reynolds number flows where viscosity is negligible. Pressure projection requires solving large linear systems, while diffusion introduces stiffness and substeps. Removing both accelerates simulation while retaining convection which is the main driver of nonlinear transport and turbulence.

Physical Scenarios. Dynamics are set by viscosity ν (or Reynolds number $Re = \frac{\rho u L}{\nu}$). Smaller ν yields more turbulence. Following prior work [35, 18, 28], we study $\nu = 0.01$ in-distribution and $\nu \in \{0.001, 0.05\}$ for OOD generalization. We also test a 3D case to assess robustness on complex spatiotemporal dynamics (Appendix D.2).

Joint Learning with Fundamental Physical Knowledge We adopt a multi-task setup, jointly training neural operators on simulations of both the original PDE and its decomposed basic form. Since basic forms are much cheaper to simulate (Table 1), we can trade expensive PDE samples for a larger number of basic-form samples under the same cost. We define the Sample Mixture Ratio as the ratio of their simulation costs, ensuring that replacing one original sample with the corresponding number of basic-form samples **maintains a comparable simulation budget relative to the baseline**.

Table 1: Summary of simulations of PDEs and their decomposed basic forms. GPU: NVIDIA RTX 6000 Ada.

PDE	Spatial Resolution	Temporal Steps	Target Variables	Simulation Costs (sec. per step)	Sample Mixture Ratio (PDE : Basic Form)
Diffusion-Reaction (Eq. 2) Basic Form	128×128	100	Activator u, Inhibitor v	1.864×10^{-2} 6.610×10^{-3}	1:3
Navier-Stokes (2D) (Eq. 3) Basic Form (2D)	256×256	1000	Velocity $\boldsymbol{u}$, Density s	2.775 0.113	1:24

We mainly consider Fourier Neural Operator (FNO) [20] and transformer as our neural architecture [10, 38, 23, 6]. We make our method agnostic to specific architectures of neural operators, with the structure of sharing the backbone of the model but the last prediction layer.

3 Experiments

Training Settings. We summarize our training details in Appendix C. In our method, we "exchange" half of the simulation budget of baseline model in the training to simulate basic forms with the sample mixture ratio defined in Table 1, and importantly, for both learning the original PDE and its decomposed basic form, we use the same hyperparameters and *keep their optimization costs (number of gradient descent steps) the same*, which both make their training fairly comparable. Since our goal is to evaluate performance on the original PDE, we use data from the basic form PDE only during training. All testing is conducted exclusively on the original PDE data.

Data Efficiency. We mainly consider the following methods: 1) *"Baseline"*: Neural operators that are only trained on simulations of the original PDE. 2) *Ours*: Replace simulations of original PDE with its decomposed basic form, allowing total simulation cost of training data can be comparable or even reduced. 3) *"Spatiotemporal Downsampling"*: Neural operators trained on a mix of full-resolution and low-resolution simulations (upsampled by interpolation), reducing simulation costs through coarser spatiotemporal data.

In Figure 3, we compare prediction errors against simulation costs. Across PDEs and architectures, our method (orange square) consistently lies below and to the left of the baseline (blue circle), achieving lower errors at reduced cost. Since decomposed basic forms are orthogonal to spatiotemporal downsampling, our method can serve as a complementary data augmentation, combined combined for further gains. On 2D Diffusion–Reaction, using basic forms at reduced resolution (green triangle) yields additional efficiency, surpassing the downsampled baseline (yellow diamond). For 2D Navier–Stokes with the Transformer, our method still outperforms both the baseline and spatiotemporal downsampling at comparable cost.

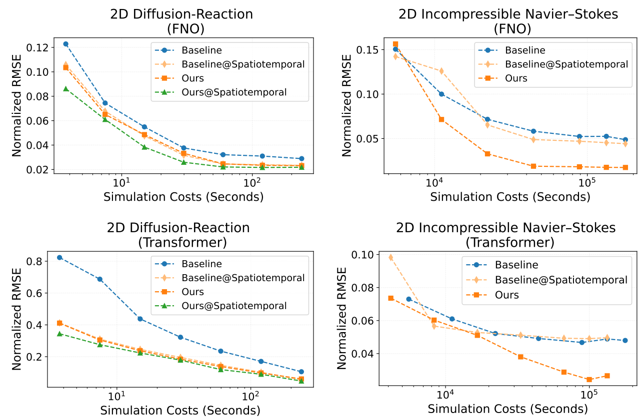

Figure 3: Joint training neural operators on data of the original PDE and the basic form improves performance and data efficiency.

Long-term Physical Consistency. Next-frame prediction [37, 23, 15] is a standard evaluation using ground-truth inputs at each step. In contrast, autoregressive inference where the model rolls out future steps using its own predictions is a more challenging test, as errors accumulate over time. As shown in Figure 4, with losses aggregated over five steps, our method maintains its improvements from Figure 3, demonstrating stronger long-term consistency.

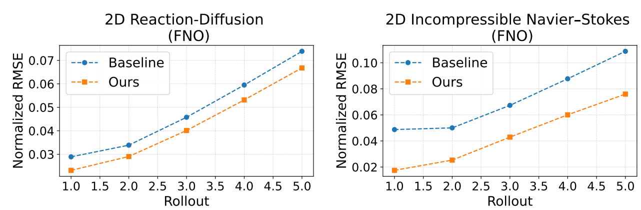

Figure 4: Joint training neural operators on data of the original PDE and the basic form improves performance with autoregressive inference at different unrolled steps.

Out-of-Distribution (OOD) Generalization. We next evaluate the generalization of our method in out-of-distribution (OOD) settings, where simulation parameters are significantly shifted. As shown in Table2, our approach improves both in-distribution accuracy and robustness to unseen physical dynamics, resulting in more reliable neural operators. Due to the page limit, we will show more results for long-term consistency and OOD in Appendix D.

Table 2: Comparisons of OOD generalization on 2D Diffusion-Reaction using FNO.

PDE	Model	Source		OOD Target 1		OOD Target 2	
		Setting	nRMSE	Setting	nRMSE	Setting	nRMSE
Diffusion-Reaction (2D)	Baseline	$\frac{D_v}{D_u} = 5$	0.0289	$\frac{D_v}{D_u} = 1$	0.0413	$\frac{D_v}{D_u} = 100$	0.0770
	Baseline@Spatiotemporal		0.0234		0.0303		0.0663
	Ours		0.0231		0.0331		**0.0538**
	Ours@Spatiotemporal		**0.0218**		**0.0298**		0.0596

Conclusion. We propose a principle- and architecture-agnostic approach to improve neural operators by explicitly incorporating fundamental physical knowledge during training. Experiments across diverse PDEs and architectures highlight the value of fundamental physics as an inductive bias for building reliable, cost-effective, and generalizable surrogate models in real-world simulations.

Acknowledgment

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231 using NERSC award NERSC DDR-ERCAP0034682, and also under Contract No. DE-AC02-05CH11231 using NERSC award ASCR-ERCAP0031463. J. Cao's research is partially supported by the Discovery Grants (RGPIN-2023-04057) of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada Research Chair program.

References

[1] Yazdan Asgari, Mehrdad Ghaemi, and Mohammad Ghasem Mahjani. Pattern formation of the fitzhugh-nagumo model: cellular automata approach. 2011.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[3] Jose Antonio Lara Benitez, Takashi Furuya, Florian Faucher, Anastasis Kratsios, Xavier Tricoche, and Maarten V de Hoop. Out-of-distributional risk bounds for neural operators with applications to the helmholtz equation. *Journal of Computational Physics*, page 113168, 2024.

[4] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

[5] Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, pages 2241–2256. PMLR, 2022.

[6] Wuyang Chen, Jialin Song, Pu Ren, Shashank Subramanian, Dmitriy Morozov, and Michael W Mahoney. Data-efficient operator learning via unsupervised pretraining and in-context learning. *arXiv preprint arXiv:2402.15734*, 2024.

[7] Wuyang Chen, Zhiding Yu, Shalini De Mello, Sifei Liu, Jose M Alvarez, Zhangyang Wang, and Anima Anandkumar. Contrastive syn-to-real generalization. *arXiv preprint arXiv:2104.02290*, 2021.

[8] Wuyang Chen, Zhiding Yu, Zhangyang Wang, and Animashree Anandkumar. Automated synthetic-to-real generalization. In *International conference on machine learning*, pages 1746–1756. PMLR, 2020.

[9] Michael Crawshaw. Multi-task learning with deep neural networks: A survey, 2020.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

[12] C. Edwards. Neural networks learn to speed up simulations. *Communications of the ACM*, 65(5):27–29, 2022.

[13] G Gambino, MC Lombardo, R Rizzo, and M Sammartino. Excitable fitzhugh-nagumo model with cross-diffusion: close and far-from-equilibrium coherent structures. *Ricerche di Matematica*, 73(Suppl 1):137–156, 2024.

[14] Derek Hansen, Danielle C Maddix, Shima Alizadeh, Gaurav Gupta, and Michael W Mahoney. Learning physical models that can respect conservation laws. In *International Conference on Machine Learning*, pages 12469–12510. PMLR, 2023.

[15] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*, 2024.

[16] Philipp Holl and Nils Thuerey. Differentiable simulations for pytorch, tensorflow and jax. In *Forty-first International Conference on Machine Learning*, 2024.

[17] 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.

[18] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[19] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

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

[21] Yifan Liu, Bohan Zhuang, Chunhua Shen, Hao Chen, and Wei Yin. Auxiliary learning for deep multi-task learning, 2019.

[22] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[23] 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. *arXiv preprint arXiv:2310.02994*, 2023.

[24] Lucas Menou, Chengjie Luo, and David Zwicker. Physical interactions promote turing patterns. *arXiv preprint arXiv:2302.12521*, 2023.

[25] Grégoire Mialon, Quentin Garrido, Hannah Lawrence, Danyal Rehman, Yann LeCun, and Bobak T Kiani. Self-supervised learning with lie symmetries for partial differential equations. *arXiv preprint arXiv:2307.05432*, 2023.

[26] S Chandra Mouli, Danielle C Maddix, Shima Alizadeh, Gaurav Gupta, Andrew Stuart, Michael W Mahoney, and Yuyang Wang. Using uncertainty quantification to characterize and improve out-of-domain learning for pdes. *arXiv preprint arXiv:2403.10642*, 2024.

[27] AK Nandakumaran and PS Datti. *Partial differential equations: classical theory with a modern touch*. Cambridge University Press, 2020.

[28] Jacob Page, Peter Norgaard, Michael P Brenner, and Rich R Kerswell. Recurrent flow patterns as a basis for two-dimensional turbulence: Predicting statistics from structures. *Proceedings of the National Academy of Sciences*, 121(23):e2320007121, 2024.

[29] Karen M Page, Philip K Maini, and Nicholas AM Monk. Complex pattern formation in reaction–diffusion systems with spatially varying parameters. *Physica D: Nonlinear Phenomena*, 202(1-2):95–115, 2005.

[30] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[31] Saurabh Patil, Zijie Li, and Amir Barati Farimani. Hyena neural operator for partial differential equations, 2023.

[32] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. Curriculum learning of multiple tasks, 2014.

[33] 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.

[34] Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017.

[35] Hermann Schlichting and Klaus Gersten. *Boundary-layer theory*. springer, 2016.

[36] Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *arXiv preprint arXiv:2306.00258*, 2023.

[37] 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.

[38] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

A More Simulation Settings

To ensure a fair comparison under equivalent simulation costs with the basic form of each PDE, we downsample the original PDE simulations both spatially and temporally. We also introduce the settings for the corresponding reduced spatiotemporal resolution simulations. Note that the simulation cost of these downsampled settings is matched to that of the basic form, which implies that their sample mixture ratios in joint training remain equivalent.

Diffusion–Reaction We follow [37] using PyClaw [17] (finite volume solver). The domain is $\Omega = [-1, 1]^2$ with resolution 128×128. We simulate for 5 seconds and recorde at 100 steps. Initial conditions are Gaussian noise with homogeneous Neumann boundaries.

We downsample to 96×96 spatial grids and 50 time steps, then upsample to $128 \times 128 \times 100$ via bilinear interpolation. The total simulation interval is maintained at 5 seconds, preserving the underlying physical dynamics.

Similarly, we can further simulate our decomposed basic form of Diffusion-Reaction at low spatiotemporal resolution (green curve in Figure 3), with a sample mixture ratio of 1:8 (Table 3).

Table 3: Summary of 2D Diffusion-Reaction simulations and its decomposed basic forms with reduced spatiotemporal resolution. GPU: NVIDIA RTX 6000 Ada.

PDE	Spatial Resolution	Temporal Steps	Target Variables	Simulation Costs (sec. per step)	Sample Mixture Ratio (PDE : Basic Form)
Diffusion-Reaction (Eq. 2)	128×128	100	Activator u, Inhibitor v	1.864×10^{-2}	1:8
Basic Form with Reduced Spatiotemporal Resolution	96×96	50		2.390×10^{-3}	

2D Incompressible Navier–Stokes We follow [37] using PhiFlow [16]. The domain is $\Omega = [0, 1]^2$ with resolution 256×256. We simulate for 5 seconds using $dt = 5 \times 10^{-5}$ and saved at 1,000 frames. Initial velocity $\boldsymbol{u}_0$ and forcing $\boldsymbol{f}$ are isotropic Gaussian random fields (low-frequency scale 0.15/0.4, smoothness 3.0/1.0), with Dirichlet boundaries.

We lower the spatial resolution to 100×100, then upsample to 256×256. For temporal downsampling, we increase the step size to $dt = 5 \times 10^{-4}$, reduce steps from 100,000 to 10,000, and decrease the frame interval from 100 to 10, preserving 5 seconds total duration and 1,000 outputs. This reduces computational cost by $\sim 10\times$.

B Model Structure

We mainly consider Fourier Neural Operator and transformer in this study. We apply a multi-task formulation, where the model learns from both the original PDE and its simplified basic form. The idea is inspired from curriculum learning [2, 32] and auxiliary task learning [21]. In our method, the basic forms act as a simpler, physically motivated auxiliary task that can facilitate more efficient representation learning and accelerate the convergence on the primary task. The model structures are shown in Figure 5. The basic form and the original PDE share all layers but the last prediction layer, which is agnostic to specific architectures of neural operators. This task-specific output layer is a very well-known and widely adopted configuration in multi-task learning. It has been highlighted in survey papers that hard parameter sharing (one input, shared hidden layers, multiple outputs) is the standard setup for multi-task learning due to its efficiency and representational benefits [34, 9].

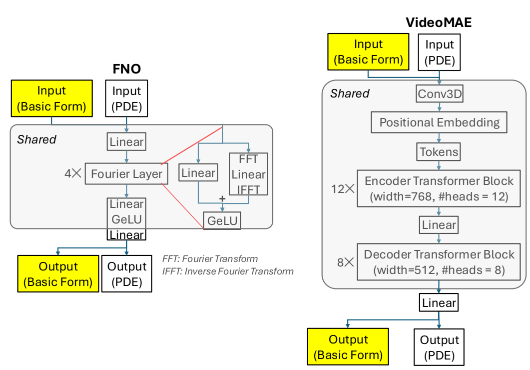

Figure 5: Our method is agnostic to specific architectures of neural operators: we always share the backbone of the model between learning of original PDE and its basic form, and decouple their predictions in the last layer.

C More Implementation Details

We summarize our training details in Table 4. We conducted our experiments on NVIDIA RTX 6000 Ada GPUs, each with 48 GB of memory.

The number of training samples in baseline ranges from 2 to 128 for the Diffusion-Reaction equation and from 2 to 64 for the Navier-Stokes equations. We use Adam optimizer and Cosine Annealing learning rate scheduler. We consistently use 100 samples for testing [37]. To train our neural operators, we use nRMSE defined in Section 2.

Table 4: Training details. “DR”: Diffusion-Reaction.“NS”: Navier Stokes.

	2D DR (FNO)	2D DR (Transformer)	2D NS (FNO)	2D NS (Transformer)	3D NS (FNO)	3D NS (Transformer)
Input Shape Format	H × W × T × C (C = 2)		H × W × T × C (C = 3)		X × Y × Z × T × C (C = 4)	
Number of Training Samples (PDE Simulations)	2, 4, 8, 16, 32, 64, 128		2, 4, 8, 16, 32, 48, 64		2, 4, 8, 16, 32, 48, 64	
Input Time Steps (ℓ in Section 2)	10	10	10	10	10	10
Sample Mixture Ratio	1:3	1:3	1:24	1:6	1:3	1:3
Learning Rate	0.001	0.0003	0.001	0.001	0.001	0.00015
Batch Size for Primary Data	4	8	16	16	8	8
Epochs	100	60	20	30	20	80
Auxiliary Task Loss Weight	0.7	0.7	0.7	0.7	0.7	0.7
Training Hours	$0.08 \sim 1.83$	$0.6 \sim 7$	$1 \sim 29$	$1.5 \sim 45$	$0.5 \sim 6.5$	$16 \sim 120$
Gradient Descent Steps Per Epoch (Baseline and Ours)	$46 \sim 2912$	$23 \sim 1456$	$124 \sim 3960$	$124 \sim 3960$	$70 \sim 2240$	$70 \sim 2240$

D More Results

D.1 Long-term Physical Consistency and Out-of-Distribution Generalization

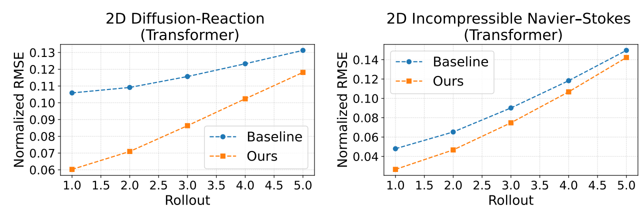

Figure 6: Joint training neural operators on data of the original PDE and the basic form improves performance with autoregressive inference at different unrolled steps.

Figure 6 shows the long-term consistency results from the Transformer, and Table 5 reports the out-of-distribution (OOD) generalization results across both the 2D Diffusion-Reaction (Transformer) and Navier-Stokes (FNO, Transformer) equations. Similar to the results in Figure 4 (FNO) and in Table 2 (2D Diffusion-Reaction (FNO)), here we can see that our approach not only achieves stronger long-term consistency but also consistently enhances generalization to simulations of unseen physical parameters. This robustness holds across both FNO and Transformer architectures and multiple PDEs, leading to more reliable and consistent neural operators under varying conditions.

Table 5: Comparisons of OOD generalization for different training methods with transformer. Models are evaluated using the best checkpoints from training in Figure 3, under comparable simulation cost settings. “Spatiotemporal“: short for “Spatiotemporal Downsampling”.

PDE	Model	Source Setting	Source nRMSE	Target 1 Setting	Target 1 nRMSE	Target 2 Setting	Target 2 nRMSE
Diffusion-Reaction (2D, Transformer)	Baseline	$\frac{D_v}{D_u} = 5$	0.1056	$\frac{D_v}{D_u} = 1$	0.1249	$\frac{D_v}{D_u} = 100$	0.1976
	Baseline@Spatiotemporal		0.0542		0.0698		0.0812
	Ours		0.0602		0.0782		0.0853
	Ours@Spatiotemporal		**0.0469**		**0.0489**		**0.0671**
Navier-Stokes (2D, FNO)	Baseline	$\nu = 0.01$	0.0487	$\nu = 0.05$	0.0825	$\nu = 0.0001$	0.0369
	Baseline@Spatiotemporal		0.0442		0.0743		0.0269
	Ours		**0.0175**		**0.0222**		**0.0125**
Navier-Stokes (2D, Transformer)	Baseline	$\nu = 0.01$	0.0479	$\nu = 0.05$	0.0853	$\nu = 0.0001$	0.0685
	Baseline@Spatiotemporal		0.0496		0.0568		0.0402
	Ours		**0.0265**		**0.0397**		**0.0256**

D.2 3D Navier-Stokes Extension

In real-world settings like atmospheric or smoke dynamics, buoyancy-driven flows add significant complexity [11]. To capture this, we extend to 3D incompressible Navier–Stokes in a rising plume scenario (see Figure 2 (c)), where smoke enters from a circular inflow at the bottom center (rate 0.2 units/timestep) and rises under buoyancy. This setup tests our method on challenging 3D spatiotemporal dynamics.

Simulation Settings Our solver is PhiFlow [16]. We simulate in a spatial domain of $\Omega = [0, 1]^3$, with resolution $50 \times 50 \times 89$. We simulate 150 steps with a $dt = 2 \times 10^{-4}$. We set the initial $\boldsymbol{u}_0$ as zero and upward buoyancy forcing term $\boldsymbol{f}_z = 5 \times 10^{-4}$. Unlike the 2D Navier-Stokes, we introduce randomness of the buoyancy forcing term on horizontal directions by uniformly drawing $\boldsymbol{f}_x$ $\boldsymbol{f}_x$ from $[-1, 1] \times 10^{-4}$.

Table 6: Summary of 3D Navier-Stokes simulation and its decomposed forms. GPU: NVIDIA RTX 6000 Ada.

PDE	Spatial Resolution	Temporal Steps	Target Variables	Simulation Costs (sec. per step)	Sample Mixture Ratio (PDE : Basic Form)
Navier-Stokes (3D) (Eq. 3) Basic Form (3D)	$50 \times 50 \times 89$	150	Velocity $\boldsymbol{u}$, Density s	1.047 0.300	1:3

Experiment Results We aim to also demonstrate three key benefits shown in Section 3 as well as the synthetic-to-real transfer.

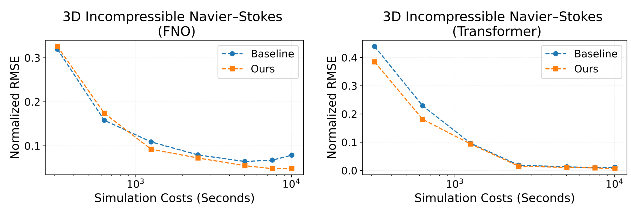

Figure 7: Joint training neural operators on data of the original 3D Navier-Stokes equation and the basic form improves performance and data efficiency.

Data Efficiency In Figure 7, we can see that joint training (orange square) on both the original and basic forms of the 3D Navier-Stokes equation consistently reduces normalized RMSE from baseline (blue circle) across varying simulation budgets. This improvement is observed for both FNO and Transformer architectures, highlighting enhanced data efficiency and generalization, which aligns with the results in Section 3.

Long-term Physical Consistency In our Figure 8, we show the rollout performance of the FNO model on the 3D incompressible Navier-Stokes equation. Here, we run the experiments with the best checkpoints from training in Figure 7. Losses will be aggregated for five consecutive time steps. We can see that our improvements in Figure 7 further persist across autoregressive steps, leading to improved long-term consistency.

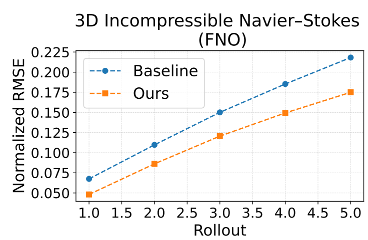

Figure 8: Joint training neural operators on data of the original 3D Navier-Stokes equation and the basic form improves performance with autoregressive inference at different unrolled steps.

Out-of-Distribution (OOD) Generalization In Table 7, we show that our joint training approach significantly improves out-of-distribution generalization on 3D Navier-Stokes across all test settings, outperforming the baseline for both FNO and Transformer models. Together with the results in Table 2 and 5, the consistent gains observed across all OOD setting results underscore the effectiveness and robustness of our method in generalizing to previously unseen physical regimes, particularly under significant shifts in simulation parameters.

Table 7: Comparisons of OOD generalization on 3D NS using different training methods.

Model	Model	Source		Target 1		Target 2	
		Setting	nRMSE	Setting	nRMSE	Setting	nRMSE
FNO	Baseline	$\nu = 0.01$	0.0675	$\nu = 0.1$	0.0393	$\nu = 0.0001$	0.0836
	Ours		**0.0481**		**0.0329**		**0.0602**
Transformer	Baseline	$\nu = 0.01$	0.0114	$\nu = 0.1$	0.0327	$\nu = 0.0001$	0.0816
	Ours		**0.0064**		**0.0124**		**0.0322**

Synthetic-to-Real Generalization Finally, we test neural operators trained on simulations of 3D Navier-Stokes in real-world scenarios. Essentially, transfering models trained on simulations to real observations is a synthetic-to-real generalization problem [8, 7], as domain gaps between numerical simulations and real-world measurements always persist.

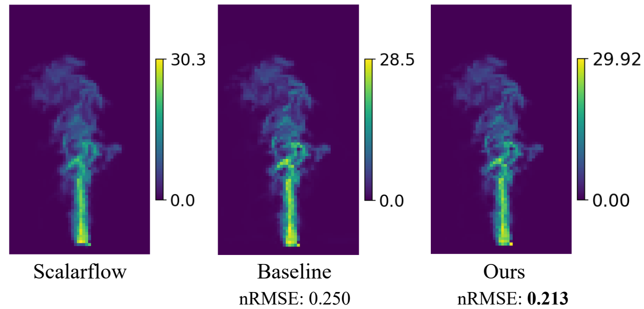

Figure 9: Visualizations of the last time step in the ScalarFlow and its predictions derived by baseline and our model.

We study the ScalarFlow dataset [11], which is a reconstruction of real-world smoke plumes and assembles the first large-scale dataset of realistic turbulent flows. For comparison, see our visualizations of synthetic 3D Navier-Stokes simulations in Figure 2(c1) and ScalarFlow data in Figure 2(c3).

We show the results and visualize the ground truth as well as the model predictions on smoke plumes from ScalarFlow in Figure 9. We can see that our method outperforms the baseline model and presents a qualitative comparison of scalar flow predictions on real data, illustrating that our jointly trained model exhibits improved synthetic-to-real generalization performance.

D.3 More Random Seeds

To ensure the statistical robustness of our findings, we now run FNO using three different random seeds during initialization and training. For each configuration, we report the average performance across the three runs, and include standard deviation as error bars in all plots in Figure 10. This enables a more rigorous evaluation of model performance, capturing the inherent variance and mitigating the risk of overinterpretation from single-seed outcomes. We can see that the results demonstrate that joint training of neural operators on data from both the original PDE and its decomposed basic form yields consistent improvements in predictive performance and data efficiency, highlighting the effectiveness of this multiphysics learning strategy.

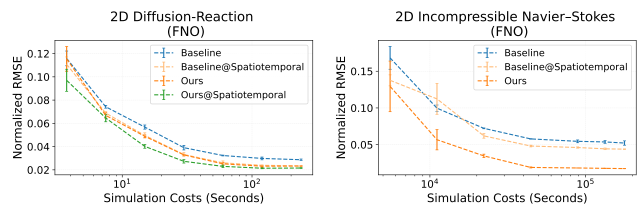

Figure 10: Model performance averaged over three random seeds. Dash lines indicate the mean performance, with error bars representing standard deviation. Legends align with the descriptions in Section 3.

D.4 Loss Reweighting

In Section 2, we define our total loss for joint learning of the original PDE (Loss_{PDE}) and its fundamental physical knowledge (decomposed basic form $\text{Loss}_{\text{Basic}}$) as

$$\text{Loss} \;=\; \text{Loss}_{\text{PDE}} \;+\; 0.7 \times \text{Loss}_{\text{Basic}}$$

To test sensitivity to the auxiliary loss weight, we conducted an ablation on FNO for 2D Diffusion–Reaction using weights 0.5, 0.7, and 1.0. As shown in Table 8, averaged nRMSE across training sample sizes varies only slightly. This result demonstrates that the model is largely insensitive to the specific choice of auxiliary weight and suggests that the improvements achieved through joint training are robust with respect to this hyperparameter. We therefore fix the auxiliary weight to 0.7 in all experiments.

Table 8: Ablation study on the auxiliary loss weight in joint training using FNO for the 2D Diffusion-Reaction across three settings: auxiliary weights of 0.5, 0.7, and 1.0.

Auxiliary Weights	0.5	0.7	1
Averaged nRMSE	0.0508	0.0491	0.0472

D.5 Choice of Fundamental Term

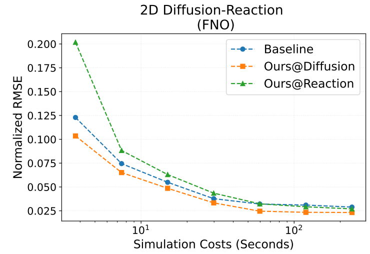

Figure 11: Joint training with both the fundamental Diffusion term and Reaction shows the importance of choice on fundamental term.

To demonstrate the importance of the choice of dropping terms, we conduct an ablation study on 2D Diffusion-Reaction, where we simulate reaction term instead of the fundamental diffusion term. The simulation cost for reaction-only term is 2.048×10^{-3} seconds per step, corresponding to a 1:9 sample mixture ratio when compared to the simulation cost of the original PDE data, making sure the comparable results.

From Figure 11, we can find that keeping the reaction term and removing the fundamental diffusion term will damage the accuracy with up to 64% increase of nRMSE, while applying the fundamental diffusion term keep boosting the model performance with 11% to 24% decrease of nRMSE. This ablation study confirms that **the correct fundamental basic term** can improve the data-efficiency when joint training with the original data, and proved that the source of improvement in Figure 3 is clearly from training with the fundamental term itself.

D.6 Lie Transform Argument on 2D Imcompressible Navier Stokes

Lie symmetries offer a way to generate new, physically valid training examples by exploiting the analytic group transformations that map one PDE solution to another. This enables the model to learn representations that are inherently equivariant to fundamental symmetries such as translation, rotation, and scaling. To further prove the strength of our model, we leverage the implementation of Lie point symmetry augmentation from [5, 25], which is orthogonal to our multiphysics joint training approach, to 2D incompressible Navier Strokes equation.

As our decomposed basic form is orthogonal to the Lie point symmetry augmentation, our method can serve as a complementary data augmentation. In Figure 12, we study prediction errors (nRMSE) of neural operators trained with different numbers of training samples (simulations). As we have already seen (Figure 3), our approach (orange square) significantly outperforms the baseline (blue circle). In contrast, the Lie-transform augmentation alone (yellow diamond) only marginally improves the baseline. As a result, combining our approach with Lie transformations (green triangle) yields strong performance, but is comparable with our approach alone, underscoring the orthogonal and complementary benefits of these two techniques.

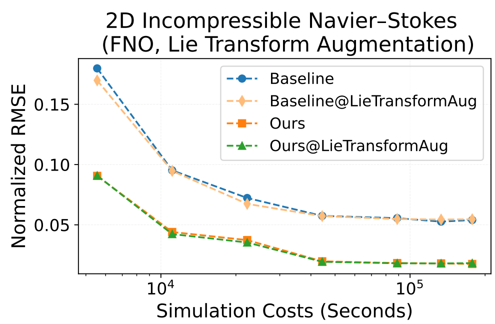

Figure 12: Joint training neural operators on data of the original PDE and the basic form, as a complementary data augmentation orthogonal to Lie-transform augumentation, can further improve performance and data efficiency. Y-axis: normalized RMSE. X-axis: simulation costs (seconds).