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

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent advances in scientific machine learning (SciML) have enabled neural operators (NOs) to serve as powerful surrogates for modeling the dynamic evolution of physical systems governed by partial differential equations (PDEs). While existing approaches focus primarily on learning simulations from the target PDE, they often overlook more fundamental physical principles underlying these equations. Inspired by how numerical solvers are compatible with simulations of different settings of PDEs, we propose a multiphysics training framework that jointly learns from both the original PDEs and their simplified basic forms. Our framework enhances data efficiency, reduces predictive errors, and improves out-of-distribution (OOD) generalization, particularly in scenarios involving shifts of physical parameters and synthetic-to-real transfer. Our method is architecture-agnostic and demonstrates over 11.5% improvement in normalized root mean square error (nRMSE) across a wide range of PDE problems. Through extensive experiments, we show that explicit incorporation of fundamental physics knowledge significantly strengthens the generalization ability of neural operators. We promise to release models and data upon acceptance.

1 Introduction

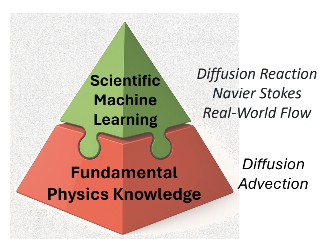

Figure 1: Can SciML models (e.g. neural operators trained on advanced PDEs) also understand fundamental physics knowledge (basic terms like diffusion, advection)?

Recent advances in scientific machine learning (SciML) have extended the scope of traditional machine learning (ML) to model the dynamic evolution of physical systems. Deep neural networks (DNNs) have been increasingly utilized to develop surrogate models that offer fast, approximate solutions to partial differential equations (PDEs), enabling significant computational speedups across a wide range of applications in the physical sciences [50, 14, 26, 46]. Among these approaches, neural operators (NOs) [33] have emerged as a powerful framework for learning mappings between infinite-dimensional function spaces, and have shown strong performance on both simulated data and real-world measurements. These data-driven models have been successfully applied in complex domains such as fluid dynamics and weather forecasting [46, 31, 4], where multiscale phenomena and sensitivity to small parameter changes present substantial modeling challenges.

Compared to numerical methods, **a key disadvantage of recent data-driven SciML models is their limited integration of fundamental physical knowledge**. Numerical solvers, though often tailored to specific PDEs or discretizations, inherently preserve physical laws (e.g., conservation, symmetry), enabling consistent and plausible simulations across diverse settings (physical parameters, boundary conditions, geometries, etc.). From a software verification perspective, numerical methods are rigorously tested through unit tests, and validated to obey governing laws across scenarios, which ensures reliability even in new simulation contexts [24, 21, 42, 23]. In contrast, data-driven SciML models like neural operators, despite their ability to learn across PDE types (e.g., via multiphysics training in SciML foundation models [39, 22]), are often sensitive to their training distributions, leading to performance degradation under distribution shifts [54, 2]. This fragility is worsened by the absence of rigorous verification: **unlike classical solvers, SciML models are rarely evaluated against decomposed PDE components**. This gap introduces three major challenges: 1) **High data demands**: Without physics priors, neural operators need large, diverse datasets for high precision. 2) **Physical inconsistency**: Lacking inductive biases, these models may violate conservation laws or produce unphysical outputs, particularly in long-term rollout predictions. 3) **Poor generalization**: Neural PDE solvers often struggle with unseen simulation settings and requires retraining.

Motivated by the above challenges, we ask two 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?*

In this paper, we highlight the importance of enforcing the learning 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 *incorporate their simulations during training*. Although often overlooked in SciML, our experiments demonstrate that these fundamental physical terms encode rich physical knowledge. Not only can they be utilized without incurring additional computational costs, but they also *widely offer substantial and multifaceted benefits*. This opens up a new door to improve the comprehensive generalization of neural operators with data efficiency.

We summarize our contributions below:

1. Through comprehensive evaluations of publicly released SciML models, we observe a strong correlation between their performance on original PDEs and basic PDE terms, despite the latter not being explicitly included in their training data (Section 2.2). This finding underscores the critical role of understanding fundamental physical knowledge in neural operators.
2. We propose to explicitly incorporate fundamental physical knowledge into neural operators. Specifically, we design a simple and principled multiphysics training strategy to train neural operators on simulations of both the original PDE and its basic form. (Section 3).
3. Our method exhibits three major benefits: 1) **data efficiency** (Section 4.2), 2) **long-term physical consistency** (Section 4.3), 3) **strong generalization in OOD (Section 4.4) and real-world (Section 4.5) scenarios**. We evaluate our method on a wide range of PDEs and neural architectures, achieving over 11.5% improvement in nRMSE (Section 4.2).

2 Background

2.1 Definition of PDE Learning in SciML

For time-dependent PDEs, the solution is a vector-valued mapping $\boldsymbol{v} : \mathcal{T} \times \mathcal{S} \times \Theta \to \mathbb{R}^d$ on a temporal domain $\mathcal{T}$, spatial domain $\mathcal{S}$, and some parameter space Θ. d indicates the number of dependent variables characterized by the PDE. In numerical solvers, the forward operator may depend on multiple previous $\ell \geq 1$ consecutive time steps, enabling finite-difference approximations of the temporal derivatives, i.e. $\mathcal{N}_\theta := [\boldsymbol{v}_\theta(t - \ell \cdot \Delta t, \cdot), \ldots, \boldsymbol{v}_\theta(t - \Delta t, \cdot)] \mapsto \boldsymbol{v}_\theta(t, \cdot)$, where Δt is the granularity of the temporal grid.

One objective of SciML is PDE learning, which is to find an ML-based parameterized surrogate model for forward modeling by learning an approximation $\widehat{\mathcal{N}}_{\theta,\phi} \simeq \mathcal{N}_\theta$ from data (θ for physical parameters of the PDE, ϕ for learnable parameters of surrogate models). To optimize $\widehat{\mathcal{N}}_{\theta,\phi}$, we take a dataset $\mathcal{D}$ comprising N discretized PDE simulations ("samples") $\mathcal{D} := \left\{\boldsymbol{v}^{(i)}\left([0 : t_{\max}], \cdot\right) \mid i = 1, \ldots, N\right\}$. Using a loss functional L, typically the normalized root of mean squared error ($\text{nRMSE} \equiv \frac{\|\boldsymbol{v}_{\text{pred}} - \boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2}$ where $\boldsymbol{v}_{\text{pred}}$ is the prediction from $\widehat{\mathcal{N}}_{\theta,\phi}$), we aim to minimize L with respect to ϕ on training data.

2.2 Motivation: Strong Neural Operators Also Outperform on Fundamental Physics

We first give a motivating example to illustrate the importance of incorporating fundamental physical knowledge into neural operators. We collect publicly released pretrained neural operators, especially SciML foundation models that are jointly pretrained on multiple PDEs ("multiphysics") [39, 22]. We evaluate their performance on fundamental physical knowledge (whose formal definition is provided in Section 3.1), and compare against their performance on the original PDE simulations.

We show the correlations based on 2D incompressible Navier-Stokes in Figure 2. Note that none of these models have explicitly learned the fundamental physical knowledge. We can clearly see that *any SciML foundation model that exhibits low error on the original PDE can also show outperforming performance on the fundamental physical knowledge*, and vice versa. With a Spearman correlation equal to 0.967, this observation indicates **strong positive correlations** between learning the original PDE and the corresponding fundamental physical knowledge, and motivates our method as explained below.

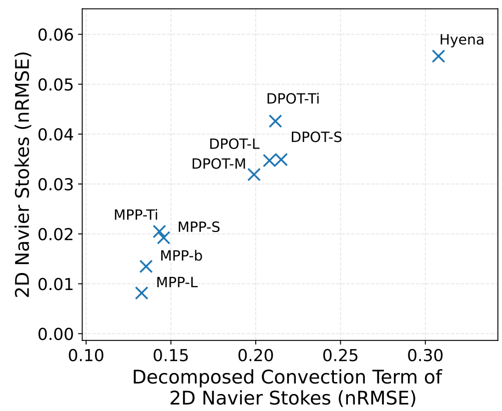

Figure 2: Strong neural operators also outperform on fundamental physics. We evaluate the latest SciML foundation models (MPP [39], DPOT [22] and Hyena [47]) on 2D incompressible Navier Stokes. Their performance on the original PDE (y-axis) and the decomposed basic form (x-axis) shows strong correlations, with the Spearman correlation equal to **0.967**.

3 Methods

Motivated by our observation in Section 2.2, we incorporate fundamental physics knowledge into learning neural operators via: 1) Defining and decomposing basic forms from the original PDE (Section 3.1); 2) Jointly training neural operators on simulations from both basic forms and the original PDE (Section 3.2). We provide an overview of our method in Figure 3.

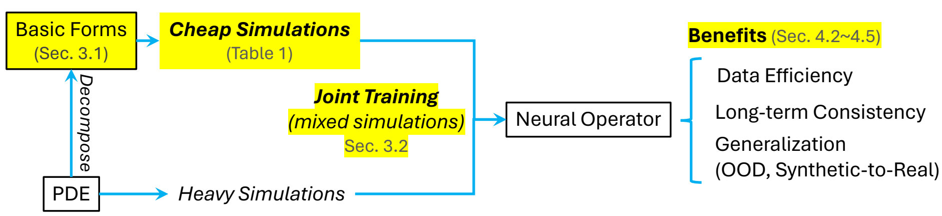

Figure 3: Overview of our method. Decomposed PDEs encode rich fundamental physical knowledge and introduce cheaper simulations. By jointly training on both the full PDE and its decomposed basic form, we bring multiple benefits to neural operators.

3.1 Fundamental Physical Knowledge via Decomposed Basic PDE Forms

3.1.1 Defining Fundamental Physical Knowledge of PDEs

Let us consider the second-order PDE as a general example:

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

where $\boldsymbol{u}$ is the target solution; $\boldsymbol{x} \in \mathbb{R}^n$ represents physical space that the target solution resides in, varying in different systems (e.g. $n = 3$ for 2D time-dependent PDEs); a_{ij}, b_i, c are coefficients (also known as "physical parameters" that determine behaviors of dynamics). If a_{ij}, b_i, c are different during training and testing, they will introduce domain shifts and essentially will lead of out-of-distribution (OOD) testing simulations. f denotes an external forcing function [43].

Our general definition of the fundamental physical knowledge of PDE is: Any variant of Equation 1 dropping either high-order terms, non-linear terms, or the forcing functions. Typically, this will lead to a more basic PDE form that can be simulated much faster. However, this basic form can still describe relevant physical dynamics of the original PDE. From a machine learning perspective, decomposing the original PDE into basic forms serves as a data augmentation strategy with lower data collection costs. Note that incorporating these basic PDE forms differs fundamentally from the multiphysics training across diverse PDE systems by recent SciML foundation models [39, 22]. While PDE solvers are not typically expected to handle a broad range of PDE types, they are expected to undergo rigorous tests against fundamental physical laws and to demonstrate reliability in new simulation contexts.

Next, we will present the specific PDEs and their corresponding decomposed basic forms that are the focus of our study. We provide visualizations of their simulations in Figure 4.

3.1.2 Diffusion-Reaction

The Diffusion-Reaction equation models an activator-inhibitor system, which typically happens in the dynamics of chemistry, biology, and ecology. The Diffusion-Reaction equation describes spatiotemporal dynamics where chemical species or biological agents diffuse through a medium and simultaneously undergo local reactions. These systems are often used to model pattern formation, such as Turing patterns, in domains ranging from chemistry to developmental biology.

$$\begin{aligned}\partial_t u &= D_u \partial_{xx} u + D_u \partial_{yy} u + R_u,\\ \partial_t v &= D_v \partial_{xx} v + D_v \partial_{yy} v + R_v.\end{aligned} \tag{2}$$

In Equation 2, u and v represent the concentrations of activator and inhibitor, respectively, and D_u, D_v are diffusion coefficients. The nonlinear reaction terms R_u, R_v model biochemical interactions. In our experiments, we adopt the FitzHugh–Nagumo variant:

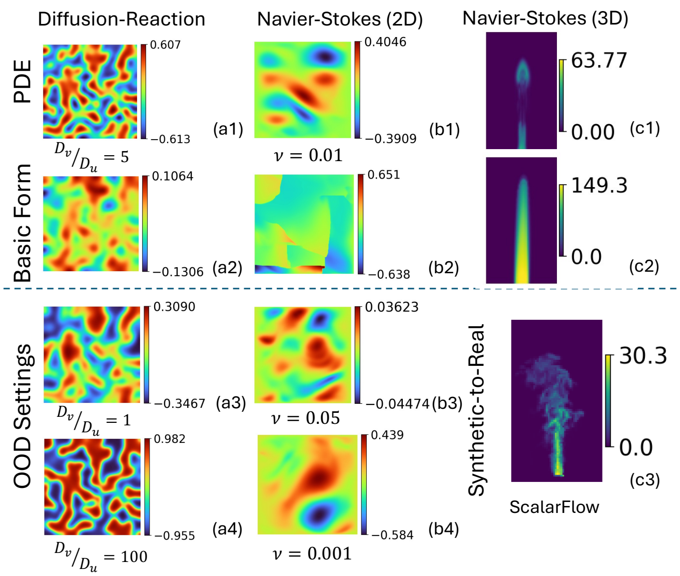

Figure 4: Visualizations of simulations of PDEs and their decomposed basic forms (Section 3.1). From left to right: activator concentration, velocity, and density. Each column shows a PDE system and its corresponding basic form and OOD setting. Basic forms are used for training neural operators with fundamental physics knowledge (Section 3.2), and the OOD settings are used for evaluating the generalization of neural operators (Section 4.4). D_v, D_u: diffusion coefficients (Equation 2). ν: viscosity (Equation 5).

$$R_u(u, v) = u - u^3 - k - v, \qquad R_v(u, v) = u - v, \tag{3}$$

where $k = 5 \times 10^{-3}$, consistent with values used in models of excitable media such as neurons or cardiac tissue.

Decomposed Basic Form. To isolate the fundamental transport behavior and reduce simulation cost, we consider a simplified form of Equation 2 by omitting the nonlinear reaction terms R_u and R_v, yielding pure diffusion equations:

$$\partial_t u = D_u \partial_{xx} u + D_u \partial_{yy} u, \quad \partial_t v = D_v \partial_{xx} v + D_v \partial_{yy} v. \tag{4}$$

- *Why Drop Reaction Terms?* This form retains the essential dispersal dynamics but eliminates the feedback coupling between u and v. Nonlinear reaction terms can vary rapidly, introducing stiffness into the PDE. This stiffness necessitates smaller time steps for stable numerical integration, increasing computational cost. By omitting these nonlinear terms, the system becomes linear and more amenable to efficient numerical solutions.
- *Why Prioritize the Diffusion Term?* Despite being simpler, pure diffusion encapsulates critical properties like isotropic spreading and mass conservation, which are relevant to many physical systems and provide inductive bias to the learning model. Unlike reaction terms, which act locally

to update activator and inhibitor concentrations, the diffusion term governs spatial coupling. It is the primary source of pattern formation and spatial dynamics, facilitating transport and stabilization across the domain. This also explains the visually similar patterns observed in Figure 4 a1 and a2.

Physical Scenarios. The emergence of spatial patterns in reaction-diffusion systems is governed by the ratio $\frac{D_v}{D_u}$, which affects the relative spreading rates of inhibitor and activator [45, 1, 15]. Classical turing instability arises when $D_v \gg D_u$, leading to diffusion-driven pattern formation, and the inhibitor spreads out while the activator stays localized. Following previous works [40], we set $D_u = 1 \times 10^{-3}$, and focus on learning simulations when $\frac{D_v}{D_u} = 5$, and possible OOD scenarios when $\frac{D_v}{D_u} \in \{1, 100\}$.

3.1.3 Incompressible Navier-Stokes

The Navier–Stokes equations govern the dynamics of fluid flow and serve as fundamentals for fluid simulations. It considers both the mass conservation and the momentum of fluid parcels.

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nu \nabla^2 \boldsymbol{u} - \frac{1}{\rho}\nabla p + \boldsymbol{f}, \tag{5}$$

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

Decomposed Basic Form. To isolate fundamental nonlinear transport mechanisms and reduce computational complexity, we simplify Equation 5 by omitting the pressure term (incompressibility via projection) $\frac{1}{\rho}\nabla p$ and diffusion term $\nu \nabla^2 \boldsymbol{u}$:

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \boldsymbol{f}, \tag{6}$$

This form captures inertial advection with external forcing and approximates high Reynolds number flows, where viscous effects are negligible. Such simplifications are analytically meaningful and are often used in turbulence modeling (e.g., inviscid Euler equations). Learning this reduced dynamics can help models internalize convection-dominant regimes.

- *Why Drop Pressure and Diffusion Terms?* The pressure term, which enforces fluid incompressibility, requires solving large linear systems and is difficult to parallelize, making it computationally expensive. Omitting it significantly accelerates the simulation. Similarly, the diffusion term in Navier-Stokes often uses explicit Euler integration with substeps, adding complexity. Removing it simplifies the simulation further. Moreover, for many visual effects like smoke or fire, viscosity is minimal, so the diffusion term has little visual impact and can often be omitted without noticeable loss in realism.
- *Why Prioritize the Convection Term?* Computationally, the convection term is cheap as it describes the local transport of fluid and no need to iterate across the spatial domain. Meanwhile, convection is the main driver of motion in most fluid flows, as it transports vorticity and mass. Without it, the fluid would just sit still or respond passively to forces. It captures nonlinear self-interaction, which is critical for dynamic, complex-looking behavior.

Physical Scenarios. In Navier-Stokes, the dynamic viscosity ν in Equation 5 (or the Reynolds number $Re = \frac{\rho u L}{\nu}$ where ρ is the density of the fluid, u is the flow speed, L is the characteristic linear dimension) controls the fluid dynamics. It measures the balance between inertial forces (pushes the fluid particles in different directions, leading to chaotic flow patterns) and viscous forces (resists motion and smoothes out differences in velocity, promoting an orderly flow) of a fluid. Following previous works [53, 26, 44], we will mainly focus on learning simulations when $\nu = 0.01$, and possible OOD scenarios when $\nu \in \{0.05, 0.001\}$, A smaller ν will lead to more turbulent flows.

3D Extension. In real-world scenarios such as atmospheric or smoke dynamics, buoyancy-driven flows provide additional complexity [13]. We extend our setting to simulate 3D incompressible Navier–Stokes in a rising plume scenario (see Figure 4 c1 and c3). We simulate how a plume of smoke rises and spreads in a 3D box-shaped environment. Smoke is introduced from a small circular inflow region located at the bottom center of the domain, at a steady inflow rate of 0.2 units per timestep. The smoke is carried upward due to buoyancy. This setting tests the method's robustness on complex spatiotemporal dynamics in three dimensions.

3.2 Joint Learning with Fundamental Physical Knowledge

After defining our fundamental physics knowledge, we now explain how to integrate it into learning neural operators in principle from two perspectives.

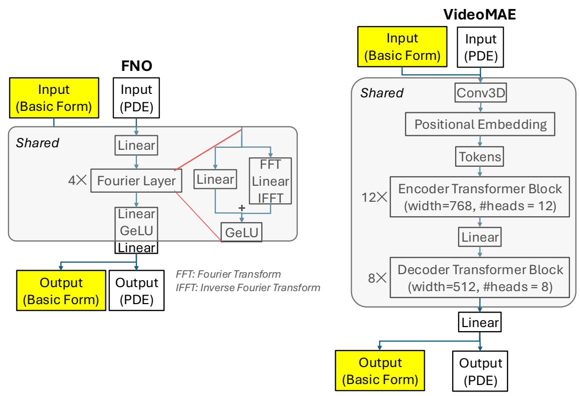

Figure 5: Our method is agnostic to specific architectures of neural operators: we always share the backbone of the model between the learning of the original PDE and its basic form, and decouple their predictions in the last layer.

1) Data Composition. First, from the data perspective, we jointly train neural operators on simulations of both our PDE and the decomposed basic form. Essentially, this will be a multiphysics training with a composite dataset. We summarize our simulations in Table 1. We can see that the simulation costs of decomposed basic forms are much cheaper than the original PDE. Therefore, we can "trade-in" simulations of the original PDE for more simulations of basic forms.

Table 1: Summary of simulations of PDEs and their decomposed basic forms. "Sample Mixture Rate": We replace simulations of the original PDE with its decomposed basic form and make sure the total simulation cost of the training data can be comparable or reduced. 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 (Eq. 4)	128×128	100	Activator u, Inhibitor v	1.864×10^{-2} 6.610×10^{-3}	1:3
Navier-Stokes (2D) (Eq. 5) Basic Form (2D) (Eq. 6)	256×256	1000	Velocity $\boldsymbol{u}$, Density s	2.775 0.113	1:24
Navier-Stokes (3D) (Eq. 5) Basic Form (3D) (Eq. 6)	$50 \times 50 \times 89$	150	Velocity $\boldsymbol{u}$, Density s	1.047 0.300	1:3

2) Neural Architecture. We mainly consider Fourier Neural Operator (FNO) [33] and transformer as our neural architecture [12, 56, 39, 9]. We make our method agnostic to specific architectures of neural operators. As shown in Figure 5, we always share the backbone of the model between the learning of the original PDE and its basic form, and decouple their predictions only in the last layer. We back-propagate losses from both PDE and its basic form, with the latter loss reweighted by 0.7.

4 Experiments

In our experiments, we aim to demonstrate three key benefits gained through explicit learning of fundamental physics knowledge: 1) data efficiency (Section 4.2); 2) long-term physical consistency (Section 4.3); 3) strong generalization in OOD (Section 4.4) and real-world (Section 4.5) scenarios.

4.1 Settings

We summarize our training hyperparameters in Appendix B. We use Adam optimizer and Cosine Annealing learning rate scheduler. We consistently use 100 samples for testing [55]. 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, making their training fairly comparable. To train our neural operators, we use nRMSE defined in Section 2.1.

4.2 Data Efficiency

We first study our method with different numbers of training samples, and demonstrate that neural operators trained with our method can achieve stronger performance with less training data. We mainly consider the following methods:

- "Baseline": Neural operators that are only trained on simulations of the original PDE.
- Ours: As described in Section 3.2, we can replace simulations of the original PDE with its decomposed basic form, allowing the total simulation cost of the training data can be comparable or even reduced. See Table 1 for the sample mixture rate for fair comparison.
- "Spatiotemporal Downsampling": Neural operators that are trained with a mixture of simulations of the original PDE and simulations at low spatial and temporal resolutions (then linearly interpolated to the original resolution). Similar to our method, we can also save simulation costs with reduced spatiotemporal resolutions. See Appendix A for rates of downsampling and more details).

In Figure 6, we study prediction errors of neural operators trained with different numbers of simulations (measured in their simulation costs). Based on our Table 1, the number of training samples ranges from 2 to 128 for the Diffusion-Reaction equation and from 2 to 64 for the Navier-Stokes equations. We can see that, across PDEs and neural architectures, our method (orange square) is to the lower left of the baseline (blue circle), which means that we can achieve *improved prediction errors with reduced simulation costs*. Meanwhile, as our decomposed basic form is orthogonal to spatiotemporal downsampling during simulations, our method can serve as a complementary data augmentation. On 2D Diffusion-Reaction, we can simulate our decomposed basic forms at lower spatiotemporal resolution, leading to further reduced simulation costs and improved data efficiency (green triangle), outperforming the baseline at the lower spatiotemporal resolution (yellow diamond).

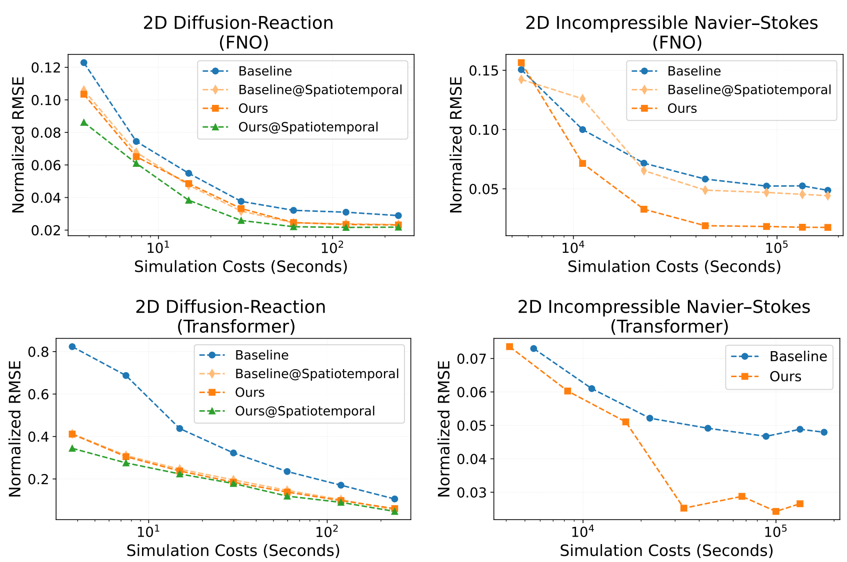

Figure 6: Joint training neural operators on data of the original PDE and the basic form improves performance and data efficiency. Rows: (top) FNO, (bottom) transformer. Columns: (left) 2D Diffusion-Reaction, (right) 2D Navier Stokes. "Spatiotemporal": short for "Spatiotemporal Downsampling". Y-axis: normalized RMSE. X-axis: simulation costs (seconds).

4.3 Long-term Physical Consistency

Next-frame prediction [55, 39, 22] is a widely adopted evaluation, where the input frames are always ground truth. Meanwhile, autoregressive inference, where the model keeps rolling out to further temporal steps with its own output as inputs, is also a meaningful and more challenging stress test. In autoregressive inference, a model forecasts futures with its own (noisy) output as inputs, and thus the prediction error will accumulate along the rollout steps.

In our Figure 7, losses will be aggregated for five consecutive time steps. We can see that our improvements in Figure 6 further persist across autoregressive steps, leading to improved long-term consistency.

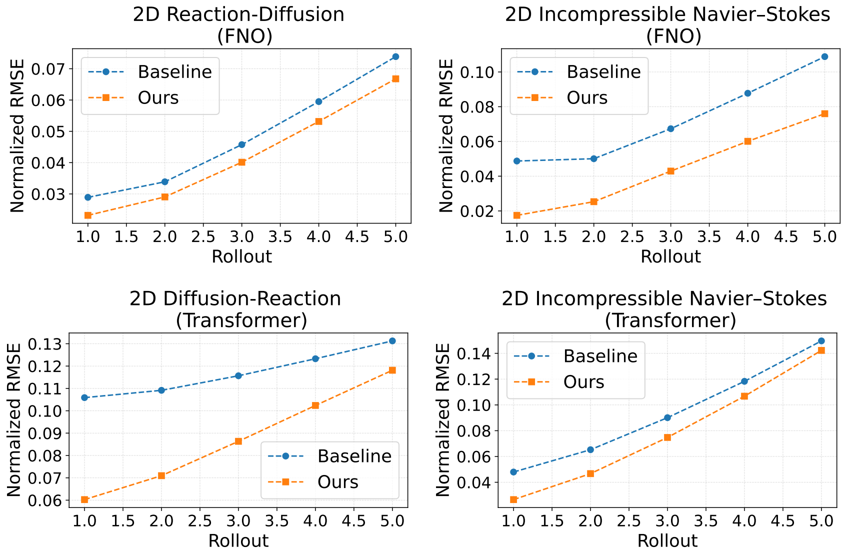

Figure 7: Joint training neural operators on data of the original PDE and the basic form improves performance with autoregressive inference at different unrolled steps. Rows: (top) FNO, (bottom) transformer. Columns: (left) 2D Diffusion-Reaction, (right) 2D Navier Stokes. Y-axis: nRMSE. X-axis: Rollout steps.

4.4 Out-of-Distribution (OOD) Generalization

We next show the benefits of our method towards the generalization of neural operators in out-of-distribution (OOD) settings, where the physical parameters used during simulation are significantly shifted. We consider physical scenarios described in Sectin 3.1, and show results in Table 2. We can see that our method not only improves in-distribution errors, but also generalizes better on unseen physical dynamics (simulations by unseen parameters), leading to more robust neural operators.

Table 2: Comparisons of OOD generalization for different training methods. Models are evaluated using the best-performing checkpoints from training, as shown in Figure 6, under comparable simulation cost settings. Due to page limits, we only show the results of the FNO here. Please refer to Appendix C.1 for results from transformer. “Spatiotemporal“: short for “Spatiotemporal Downsampling”.

PDE	Model	Source		Target 1		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
Navier-Stokes (2D)	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**

4.5 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 [11, 10], as domain gaps between numerical simulations and real-world measurements always persist.

We study the ScalarFlow dataset [13], 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 4(a) and ScalarFlow data in Figure 4(c).

We show the results and visualize the ground truth as well as the model predictions on smoke plumes from ScalarFlow in Figure 8. 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. Please read our Appendix C.2 for more results on 3D Navier Stokes.

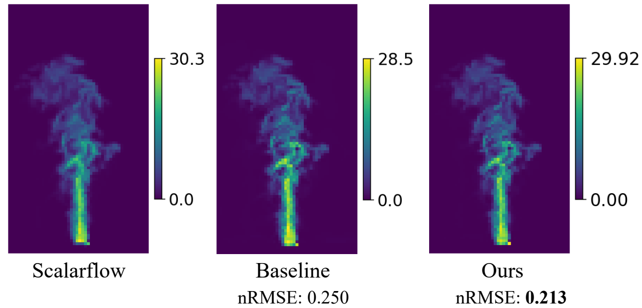

Figure 8: Visualizations of the last time step in the ScalarFlow and its predictions derived by baseline and our model. Joint training neural operators on data of the original PDE and the basic form improves performance on synthetic-to-real generalization.

5 Related Works

Machine Learning for Scientific Modeling Learning-based methods have long been used to model physical phenomena [29, 30, 8, 7]. Physics-informed neural networks (PINNs)[50, 65, 17, 16, 52] incorporate PDEs into loss functions to enforce physical laws, but often struggle with generalization and optimization issues[28, 14]. Operator learning methods like Fourier Neural Operators (FNO)[33, 32, 27] and DeepONet[38] offer greater flexibility by learning mappings between function spaces but require extensive labeled data [50, 5, 63]. We adopt a unique approach by evaluating and enhancing SciML through the lens of its compatibility with fundamental physical principles.

Data-Driven Neural PDE Solvers Machine learning has increasingly been used to approximate PDE solutions, with neural PDE solvers trained on diverse scenarios to mimic traditional simulators. Early models used CNNs [19, 64, 3], while DeepONet [38] introduced a neural operator (NO) framework separating input and query encodings, inspiring many extensions [58, 20, 59, 35, 57, 61, 39, 22]. Advances like FNO [33], LNO [6], CNO [51], and KNO [60] have expanded the field, with FNO particularly impactful across applications [34, 18, 62, 49, 48, 46, 36]. Compared with previous neural operator works, instead of naively swapping in cheaper simulations of simplified PDEs, the core merit of our work is to emphasize the multifaceted benefits of explicit learning of fundamental physics knowledge during operator learning.

Out-of-Distribution Generalization in SciML Interest in out-of-distribution (OOD) generalization for scientific machine learning (SciML) has grown recently. [54] showed that fine-tuning neural operators (NOs) on OOD PDEs often requires many OOD simulations, which may be impractical. [2] proposed a Helmholtz-specific FNO with strong OOD performance, supported by Rademacher complexity and a novel risk bound. Other work includes ensemble methods leveraging uncertainty [21, 42], loss functions informed by numerical schemes [25], and meta-learning for varied geometries [37]. However, varying PDE types and setups across studies hinder unified insights into OOD generalization for NOs. Our work demonstrates that neural operators explicitly trained with fundamental physics knowledge exhibit improved OOD and synthetic-to-real generalization.

6 Conclusion

We present a principle and architecture-agnostic approach to enhance neural operators by explicitly incorporating fundamental physical knowledge into their training. By decomposing complex PDEs into simpler, physically meaningful basic forms and using them as auxiliary training signals, our proposed method significantly improves data efficiency, long-term predictive consistency, and out-of-distribution generalization. These improvements are demonstrated across a variety of PDE systems and neural operator architectures. Our finding highlights the untapped potential of fundamental physics as an inductive bias in scientific machine learning, offering a robust and cost-effective pathway to more reliable and generalizable surrogate models in real-world physical simulations.

7 Limitations

Current limitations of our work: 1) We could design more decompositions of the original PDE; 2) We could study more PDE systems; 3) We could consider different neural operator architectures. We expect that addressing these limitations will lead to broader impacts in future works.

References

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

[2] 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.

[3] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64:525–545, 2019.

[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] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Lno: Laplace neural operator for solving differential equations. *arXiv preprint arXiv:2303.10528*, 2023.

[7] Tianping Chen and Hong Chen. Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks. *IEEE Transactions on Neural Networks*, 6(4):904–910, 1995.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.

[17] Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

[18] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

[19] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.

[20] Patrik Simon Hadorn. Shift-deeponet: Extending deep operator networks for discontinuous output functions. ETH Zurich, Seminar for Applied Mathematics, 2022.

[21] 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.

[22] 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.

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

[24] 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.

[25] Taeyoung Kim and Myungjoo Kang. Approximating numerical fluxes using fourier neural operators for hyperbolic conservation laws. *CoRR*, 2024.

[26] 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.

[27] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[28] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[29] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[30] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

[31] 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.

[32] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[33] 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.

[34] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[35] Guang Lin, Christian Moya, and Zecheng Zhang. B-deeponet: An enhanced bayesian deeponet for solving noisy parametric pdes using accelerated replica exchange sgld. *Journal of Computational Physics*, 473:111713, 2023.

[36] Burigede Liu, Nikola Kovachki, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, Andrew M Stuart, and Kaushik Bhattacharya. A learning-based multiscale method and its application to inelastic impact problems. *Journal of the Mechanics and Physics of Solids*, 158:104668, 2022.

[37] Wenzhuo Liu, Mouadh Yagoubi, and Marc Schoenauer. Meta-learning for airflow simulations with graph neural networks. *arXiv preprint arXiv:2306.10624*, 2023.

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

[46] 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.

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

[48] Md Ashiqur Rahman, Manuel A Florez, Anima Anandkumar, Zachary E Ross, and Kamyar Azizzadenesheli. Generative adversarial neural operators. *arXiv preprint arXiv:2205.03017*, 2022.

[49] Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

[50] 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.

[51] Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 36, 2024.

[52] Pu Ren, Chengping Rao, Yang Liu, Jian-Xun Wang, and Hao Sun. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *Computer Methods in Applied Mechanics and Engineering*, 389:114399, 2022.

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

[54] 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.

[55] 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.

[56] 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.

[57] Simone Venturi and Tiernan Casey. Svd perspectives for augmenting deeponet flexibility and interpretability. *Computer Methods in Applied Mechanics and Engineering*, 403:115718, 2023.

[58] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[59] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Improved architectures and training algorithms for deep operator networks. *Journal of Scientific Computing*, 92(2):35, 2022.

[60] Wei Xiong, Xiaomeng Huang, Ziyang Zhang, Ruixuan Deng, Pei Sun, and Yang Tian. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *Journal of Computational Physics*, page 113194, 2024.

[61] Wuzhe Xu, Yulong Lu, and Li Wang. Transfer learning enhanced deeponet for long-time prediction of evolution equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10629–10636, 2023.

[62] Yan Yang, Angela F Gao, Jorge C Castellanos, Zachary E Ross, Kamyar Azizzadenesheli, and Robert W Clayton. Seismic wave propagation and inversion with neural operators. *The Seismic Record*, 1(3):126–134, 2021.

[63] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*, 2023.

[64] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

[65] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

A Simulation Settings

In this section, we detail the simulation settings for the 2D Diffusion-Reaction, 2D Incompressible Navier-Stokes, and 3D Navier-Stokes equations (see Table 1 for the summary). We will also explain how we prepare simulations for the "Spatiotemporal Downsampling" in Section 4.2.

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.

To further explore the benefits of our multiphysics joint training approach with this reduced spatiotemporal resolution simulation strategies (refer to *Ours@Spatiotemporal* in Figure 6), we additionally introduce the simulation settings for the spatiotemporally downsampled basic form of the 2D Diffusion-Reaction equation. As discussed in Section 4.2, our framework is orthogonal to standard downsampling techniques, and combining the two can lead to further reductions in simulation cost. This reduction allows for an increased proportion of basic form samples in the training mixture under a fixed computational budget (see Table 3 for details).

A.1 Diffusion-Reaction

Our simulation setting for Diffusion-Reaction follows [55]. Our solver is PyClaw [24] that uses the finite volume method. We set the initial condition as standard normal random noise $u(t=0,x,y) \sim \mathcal{N}(0, 1.0)$. We use the homogeneous Neumann boundary condition. We simulate in a spatial domain of $\Omega = [-1,1]^2$, with resolution 128×128. We simulate 5 seconds and save into 100 temporal steps.

Reduced Spatiotemporal Resolution. When simulating the original Diffusion-Reaction equation at low spatiotemporal grids (yellow curves in Figure 6), we reduce the spatial resolution from 128×128 to 96×96, and reduce the number of temporal steps from 100 to 50. We then upsample to $128 \times 128 \times 100$ (steps) via bilinear interpolation to match the resolution of simulations of the original PDE. 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 6). Table 3 shows the simulation cost of decomposed basic forms with reduced spatiotemporal resolution and the sample mixture ratio.

Table 3: Summary of 2D Diffusion-Reaction simulations and its decomposed basic forms with reduced spatiotemporal resolution. "Sample Mixture Rate": We replace simulations of the original PDE with its decomposed basic form with reduced spatiotemporal resolution and make sure the total simulation cost of the training data can be comparable. 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 (Eq. 4) with Reduced Spatiotemporal Resolution	96×96	50		2.390×10^{-3}	

A.2 2D Incompressible Navier-Stokes

Our simulation setting for incompressible Navier-Stokes follows [55]. Our solver is PhiFlow [23]. We simulate in a spatial domain of $\Omega = [0,1]^2$, with resolution 256×256. We simulate 5 seconds with a $dt = 5 \times 10^{-5}$, and periodically save into 1000 temporal steps. Our initial conditions $\boldsymbol{u}_0$ and forcing term $\boldsymbol{f}$ are drawn from isotropic Gaussian random fields, where the low-frequency components of the spectral density is scaled with `scale` and high-frequency components are suppressed with power-law decay by `smoothness`. For $\boldsymbol{u}_0$, `scale` is 0.15 and `smoothness` is 3.0. For $\boldsymbol{f}$, `scale` is 0.4 and `smoothness` is 1.0. Boundary conditions are Dirichlet.

Reduced Spatiotemporal Resolution. When simulating the original 2D incompressible Navier-Stokes equation at low spatiotemporal grids (yellow curves in Figure 6), we reduce the spatial resolution from 256×256 to 100×100. We then spatially upsample to 256×256 via bilinear interpolation to match the resolution of simulations of the original PDE. To reduce the temporal resolution while maintaining the same total simulation time and number of recorded frames, we increase the time-step size and proportionally reduce the number of integration steps and output interval. Specifically, we change the time-step from $dt = 5 \times 10^{-5}$ to $dt = 5 \times 10^{-4}$, and reduce the total number of time steps from $n_{\text{steps}} = 100{,}000$ to $n_{\text{steps}} = 10{,}000$. To preserve the temporal spacing between output frames, we decrease the frame interval from 100 to 10. This ensures the same total simulation duration of 5 seconds and the same number of output frames (1,000). This modification reduces computational cost by roughly 10 times.

A.3 3D Incompressible Navier-Stokes

Our solver is PhiFlow [23]. 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 $[-1, 1] \times 10^{-4}$. We set Dirichlet zero boundary conditions.

B More Implementation Details

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

Table 4: Training hyperparameters. "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 Time Steps (ℓ in Section 2.1)	10	10	10	10	10	10
Learning Rate	0.001	0.0003	0.001	0.001	0.001	0.00015
Batch Size	4	8	16	16	8	8
Epochs	100	60	20	30	20	80
Training GPU Hours	0.08∼1.83	0.6hr∼7hr	1∼29	1.5∼45	0.5∼6.5	16∼120

C More Results

C.1 Out-of-Distribution Generalization of Transformer

Table 5 reports the out-of-distribution (OOD) generalization results from the Transformer across both the 2D Diffusion-Reaction and Navier-Stokes equations. Similar to the results of FNO in Table 2, here we can see that our approach not only improves in-distribution errors but also consistently enhances generalization to simulations of unseen physical parameters. This robustness holds across both FNO and Transformer architectures, 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 6, under comparable simulation cost settings. "Spatiotemporal": short for "Spatiotemporal Downsampling".

PDE	Model	Source		Target 1		Target 2	
		Setting	nRMSE	Setting	nRMSE	Setting	nRMSE
Diffusion-Reaction (2D)	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)	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**

C.2 Data Efficiency and Out-of-Distribution Generalization for 3D Navier Stokes

Similar as what we have studied in Section 4, we aim to also demonstrate three key benefits: data efficient, long-term physical consistency, and strong generalization in OOD simulations, for 3D Navier-Stokes as well.

In Figure 9, 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 4.2.

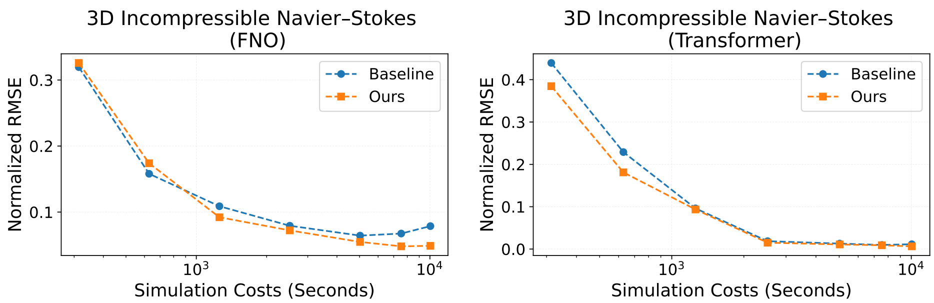

Figure 9: Joint training neural operators on data of the original 3D Navier-Stokes equation and the basic form improves performance and data efficiency. Columns: (left) FNO, (right) Transformer. Y-axis: nRMSE. X-axis: Simulation Costs (Seconds).

In our Figure 10, 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 9. Losses will be aggregated for five consecutive time steps. We can see that our improvements in Figure 9 further persist across autoregressive steps, leading to improved long-term consistency.

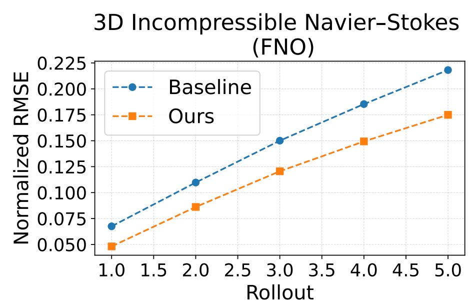

Figure 10: Joint training neural operators on data of the original 3D Navier-Stokes equation and the basic form improves performance with autoregressive inference at different unrolled steps. Y-axis: nRMSE. X-axis: Simulation Costs (Seconds).

In Table 6, 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 6: Comparisons of OOD generalization on 3D NS for different training methods across different models. Models are evaluated using the best checkpoints from training in Figure 9, under comparable simulation cost settings.

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**

C.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 11. 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.

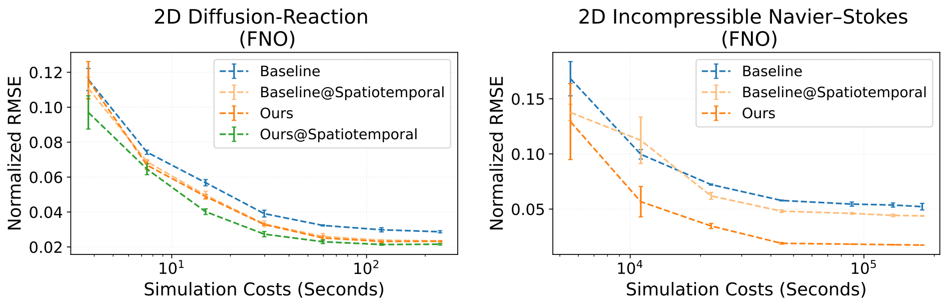

Figure 11: Model performance averaged over three random seeds. Joint training neural operators on data of the original PDE and the basic form consistently improves performance and data efficiency. Dash lines indicate the mean performance, with error bars representing standard deviation. Legends align with the descriptions in Section 4.2. Columns: (left) 2D Diffusion-Reaction, (right) 2D Navier Stokes. Y-axis: nRMSE. X-axis: Simulation Costs (seconds).

C.4 Loss Reweighting

In Section 3.2, 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 address the concern regarding the fixed auxiliary loss weight, we conducted an ablation study on the effect of auxiliary loss weighting in joint training using FNO for the 2D Diffusion-Reaction system. We evaluate model performance across three auxiliary weight settings, 0.5, 0.7, and 1.0, by averaging accuracy over the range of training sample sizes (Figure 6) used in the data efficiency experiments. In Table 7, we show the model averaged nRMSE, which is largely consistent with only minor variations in normalized RMSE. 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. Based on this analysis, we fix the auxiliary weight to 0.7 for all the experiments.

Table 7: 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. Results of averaged nRMSE demonstrate consistent performance, indicating robustness to the choice of auxiliary loss weighting.

Auxiliary Weights	0.5	0.7	1
Averaged nRMSE	0.0508	0.0491	0.0472

C.5 Visualization of Predictions

To show the predicted PDE solution from our jointly training neural operators on original PDE equation and its basic form aligns with the ground truth, we present qualitative visualizations of model predictions across three PDEs, 2D Diffusion-Reaction, 2D Incompressible Navier-Stokes, and 3D Incompressible Navier-Stokes, in Figure 12. For each PDE solution, we show the initial condition and predicted states at intermediate and final rollout times. The predictions are generated using the FNO model trained with our joint training framework. Across all systems and time points, the predictions closely align with the expected dynamics, accurately capturing both spatial patterns and temporal evolution. These visualizations highlight the model's capacity to generalize across scales and exhibit physically coherent behavior.

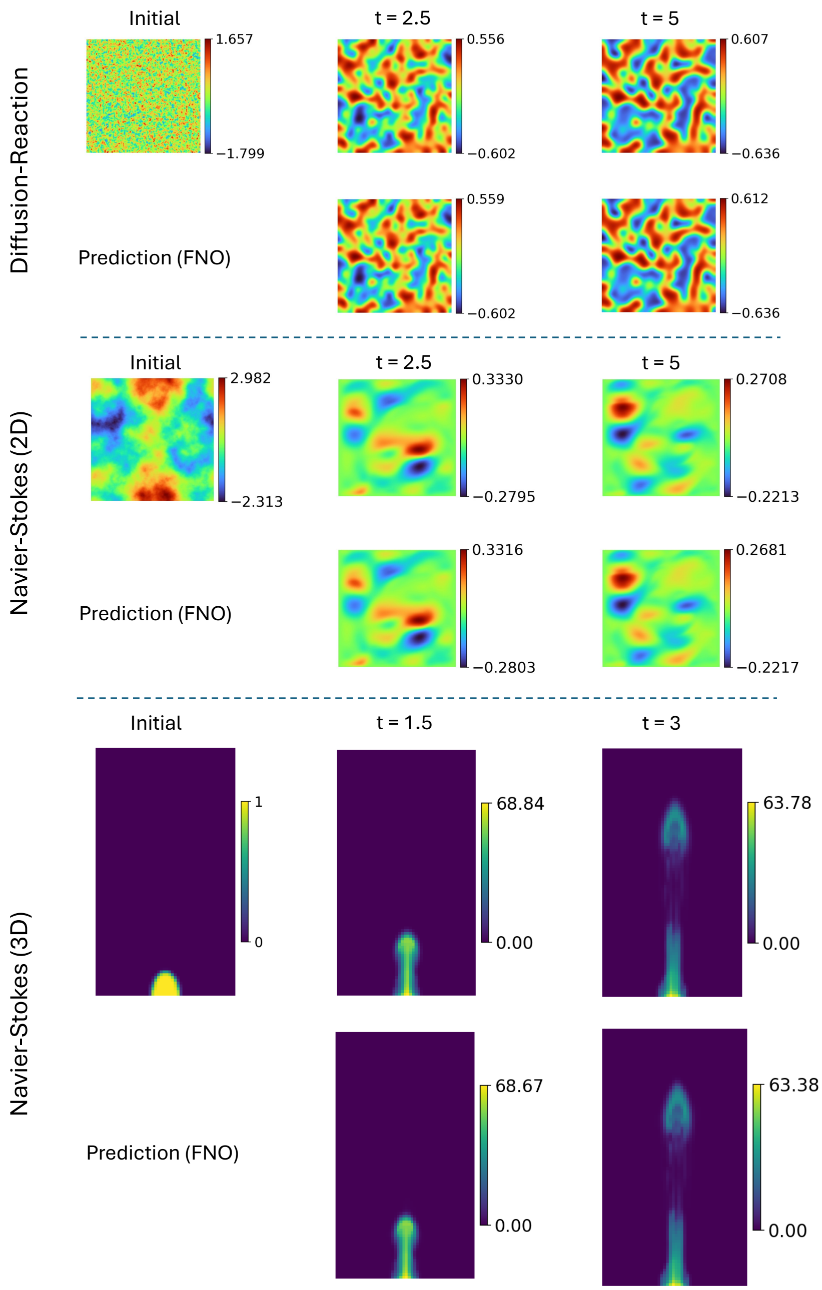

Figure 12: Qualitative visualization of model predictions for 2D Diffusion-Reaction, 2D Incompressible Navier-Stokes, and 3D Incompressible Navier-Stokes systems using FNO trained with our joint framework. For each case, the initial state and predicted states at intermediate and final rollout times are shown. The results demonstrate accurate temporal evolution and spatial coherence

C.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, 41], which is orthogonal to our multiphysics joint training approach, to 2D incompressible Navier Strokes equation.

We incorporate the augmentation process into our model. We only apply Lie-transform augmentations exclusively to the velocity ($\boldsymbol{u}$) of the original 2D incompressible Navier-Stokes, leaving the remaining density and all target variables from the decomposed basic forms unchanged. Following [41], the Lie transformation is implemented with a second-order Lie–Trotter splitting scheme with two steps, where the five fields $(x, y, t, \boldsymbol{u}_x, \boldsymbol{u}_y)$ were transformed in accordance with the sampled generator strengths as follows: a maximum time shift (g_1) strength of 0.1, maximum spatial translations (g_2, g_3) strength of 0.1 in x and y respectively, a maximum scaling (g_4) strength of 0.05, a maximum rotation (g_5) strength of $10°$, corresponding to $\pi/18$ radians, a maximum x-linear boost (g_6) and y-linear boost (g_7) strength of 0.2 and a maximum x- and y-quadratic boosts (g_8, g_9) strength of 0.05.

As our decomposed basic form is orthogonal to the Lie point symmetry augmentation, our method can serve as a complementary data augmentation. In Figure 13, we study prediction errors (nRMSE) of neural operators trained with different numbers of training samples (simulations). As we have already seen (Figure 6), 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.

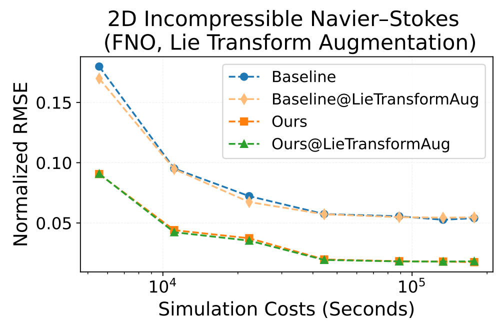

Figure 13: Joint training neural operators on data of the original PDE and the basic form, as a complementary data augmentation orthogonal to Lie-transform augumentation, can further improve performance and data efficiency. Y-axis: normalized RMSE. X-axis: simulation costs (seconds).

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Introduction explicitly state the main contribution (importance of fundamental physics knowledge, join training with simulations of the decomposed basic forms) and the results sections back each of these claims with empirical evidence.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

Justification: We include a “Limitation” section in the main paper.

Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [NA]

Justification: The paper is entirely empirical; it introduces no theorems or formal proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Section 4 details our training settings. We also attach our code in the supplement. Together these give enough information to rerun the experiments.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: We attach our code in the supplement. We will also release both our data, code, model upon acceptance.

   Guidelines:

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

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Dataset splits, particle counts, grid resolutions, dt, network architecture, and all training hyper-parameters are specified in Section 3, Section 4, and Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We present standard deviations of results in our Figure 6 over three random runs in Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In the Appendix we give our training GPU hours.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses only synthetic simulation data and involves no personal or sensitive information, so it aligns with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [NA]

Justification: Our paper poses no such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [No]

    Justification: The CC BY 4.0 license has been chosen on OpenReview.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We promise to release all our data upon acceptance.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

Answer: [NA]

Justification: No human-subject or crowdsourced data are used.

Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: Not applicable—no human-subject studies were conducted.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.