Towards Universal Neural Operators through Multiphysics Pretraining

Mikhail Masliaev
ITMO University
St. Petersburg, Russia, 197101
maslyaitis@gmail.com

Dmitry A. Gusarov
ITMO University
St. Petersburg, Russia, 197101
gusdmitr@itmo.ru

Ilya Markov
ITMO University
St. Petersburg, Russia, 197101
iomarkov@itmo.ru

Alexander Hvatov
ITMO University
St. Petersburg, Russia, 197101
alex_hvatov@itmo.ru

Abstract

Although neural operators are widely used in data-driven physical simulations, their training remains computationally expensive. Recent advances address this issue via downstream learning, where a model pretrained on simpler problems is fine-tuned on more complex ones. In this research, we investigate transformer-based neural operators, which have previously been applied only to specific problems, in a more general transfer learning setting. We evaluate their performance across diverse PDE problems, including extrapolation to unseen parameters, incorporation of new variables, and transfer from multi-equation datasets. Our results demonstrate that advanced neural operator architectures can effectively transfer knowledge across PDE problems.

1 Introduction

Contemporary science commonly uses partial differential equations (PDEs) and systems of partial differential equations to model spatio-temporal processes. For instance, reaction-diffusion equations describe, how mass moves and disperses within a fluid system, and gas and liquid dynamics are commonly modeled with variants of Euler/Navier-Stokes equations. While the analytical solutions are applicable in idealized problem statement, it is challenging to construct them in realistic scenarios. Thus, numerical simulation techniques, such as finite-element or spectral methods have been developed, yet in many cases such solutions can be computationally costly. However, in many cases, such as meteorological forecasting or multi-physics simulation in the engineering, the numerical solution of differential equations tends to be a computationally costly procedure.

With the development of scientific machine learning, greater emphasis in physics systems simulations is placed on data-driven methods. Physics-informed neural networks (PINN) (1) extend the loss with PDE-based terms, so the training has an objective of matching network's output with governing PDE. However, PINNs require explicit formulations and guarantee accuracy only at training mesh nodes. Operator learning, realized through Deep Operator Networks (DeepONet) (2) and kernel-based neural operators (NO) (3), offers a faster alternative to classical solvers, approximating mappings between functional spaces rather than dynamics at discrete nodes, providing discretization invariance and efficient inference.

A recent direction in NO research is the design of foundation models. Originating in NLP (4) or vision-language models (5), such models contain billions of parameters (e.g., GPT-4 exceeds one

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

trillion (6)) and are pretrained on large-scale datasets. They can then be fine-tuned for downstream tasks at reduced cost. In this work, we aim to develop a standardized approach to applying large NO-based models for generalized dynamics and transfer learning across PDE problems. Thus, with the neural operator the model is pre-trained on a simplified problem statement, which is later transferred to another (typically, more complex) problem in the fine-tuning phase.

Our contributions are as follows: the proposed method enables neural operator learning on diverse multi-physics datasets by introducing an adapter-based approach for simultaneous training on PDE-based problems with different sets of input functions. The results demonstrate that the transfer learning approach can significantly enhance model quality and reduce fine-tuning costs.

2 Related work

Pretraining of neural operators has thus far mainly been case-specific, with limited generalization. The concept of transfer learning using DeepONet operator models are employed in conditional shift scenarios in study (7). Also, issues of solving transfer learning problems on multi-scale data with convolutional neural networks were discussed in (8).

Several foundational models for PDE systems have been proposed beyond classical NO approaches: a foundational model tranining framework for equations of fixed types (in the study, steady-state equations were considered) is proposed in the research (9). Boundary-Embedded Neural Operators (BENO) (10) solve elliptic PDEs using graph neural networks, where boundary geometry is encoded via a transformer block into latent vectors guiding message passing. Other works leverage transformer-based architectures to encode PDE structure (11).

Transformer-based approaches proved to be capable of modeling complex interactions between selected token functions. POSEIDON, a hierarchical vision transformer with shifted windows, applicable to transfer knowledge across Euler/Navier–Stokes cases (12). Codomain Attention Neural Operator (CoDA-NO) (13), designed for multiphysics PDE transfer learning, employs codomain attention with function space dot product.

Transformers have also been explored within neural operator learning (14). In contrast, we deliberately avoid physics-informed approaches and focus on assessing the capacity of neural operators to learn generalized dynamics purely from data samples.

3 Method

An operator learning framework has been developed for data-driven modeling of dynamical systems, governed by parametric partial differential equations $L_p\mathbf{u}(t,\mathbf{x}) = f(t,\mathbf{x})$ on the bounded domain $\mathcal{D}$ in a mesh-agnostic (albeit with some limitations, as examined in (15)) approach.

Neural operator learning: In this research, we have focused on the kernel-integral neural operators, mainly Fourier Neural Operators. Neural Operators are designed to learn mappings between the input space $\mathcal{U}$, representing sets of input functions $\mathbf{a} = \{a_1,\ ...\ a_{n\_in}\},\ a_i : \mathcal{D} \longrightarrow \mathbb{R}$ and the output functions $\mathbf{u} = \{u_1,\ ...\ u_{n\_out}\},\ u_i : \mathcal{D} \longrightarrow \mathbb{R}$. While the outputs are typically fixed to the dependent variables of the PDE (or system of PDEs), the choice of input functions is guided by the equation structure and includes meshes, initial conditions, forcing terms, or equation coefficients.

To respect the non-localities of the model, NO design adds integral kernel operators $(\mathcal{K}(v))(x)$ to the arguments (A_t, and b_t for weights and biases) of activation functions σ. Here the $\kappa_t \in C(D_{t+1} \times D_t|\theta_{k,t})$; $\theta_{k,t} \in \mathbb{R}^{n_{v_{t+1}} \times n_{v_t}}$ is the kernel function, parameterized (with $\theta_{k,t}$) by the method of choice (FNO, GNO, etc.), and $v_t(y)$ - hidden representation of input functions, obtained from the t-th layer. D_t is the hidden dimensionality, which is linked to the number of modes in FNO. Thus, the parameters of NO main part include weights, biases and kernel parameters $\theta_{\mathcal{F}} = \{A_t, b_t, \theta_{k,t} : t = 1,\ ...\ , n_{\text{layers}}\}$.

$$\mathcal{F}_t(x) = \sigma\left(A_t v_t(x) + \int_{D_i} \kappa_t(x,y)v_t(y)dy + b_t(x)\right),\ \forall x \in D_t,\ t = 1,\ ...\ , n_{\text{layers}} \qquad (1)$$

The architecture of layers sequence in NO goes as follows: the inputs are transformed to their higher-dimensional hidden representation by lifting layers (typically, a feed-forward neural network) $\mathcal{L} : \mathcal{L}(\mathbf{a}) = \sigma(A_{\mathcal{L}}\mathbf{a} + b_{\mathcal{L}})$, where parameters of the lifting include weights and biases $\theta_{\mathcal{L}} = \{A_{\mathcal{L}}, b_{\mathcal{L}}\}$. The hidden representations are sequentially mapped with the integral-operator blocks (1). In contrast, the output of the last block is projected to the space of outputs by the point-wise function $\mathcal{P}$ with parameters $\theta_{\mathcal{L}} = \{A_{\mathcal{P}}, b_{\mathcal{P}}\}$. The operator approximation takes form of model $\mathcal{G}_\theta$: $\tilde{\mathbf{u}}(\mathbf{x}) = \mathcal{G}_\theta(\mathbf{a}) = \mathcal{P} \circ \mathcal{F} \circ \mathcal{L}(\mathbf{a}) = \mathcal{P} \circ \mathcal{F}_{n\_layers} \circ ... \circ \mathcal{F}_1 \circ \mathcal{L}(\mathbf{a})$.

Improving the generalization ability of neural operators: In this research, we employed two types of NO modifications: state-space models and transformer-based models. In the first approach, inserting a Mamba-SSM module (16) $\mathcal{M}_\phi$ after the lifting map $\mathcal{L}$ allows the model to encode long-range temporal and spatial dependencies directly in the hidden representation. For lifted features $v_0(x) = \mathcal{L}(\mathbf{a})(x)$, the Mamba module computes

$$\widetilde{v}_0(x,t) = (\mathcal{M}_\phi v_0)(x,t) = \sum_{\tau \le t} K_\tau \, v_0(x, t-\tau), \tag{2}$$

with learnable convolution kernels K_τ defining the causal recurrence. This step acts as a latent preconditioner: embeddings are aligned with dominant dynamical motifs (transport, diffusion, oscillation) common across PDEs, so that when passed into the Fourier integral layers, the effective operator acts on inputs of reduced variability and lower spectral rank. Consequently, the composition $\mathcal{F}_t \circ \mathcal{M}_\phi$ yields more stable training and improves efficiency in transferring pre-trained representations to new PDEs during fine-tuning.

The next approach, examined in the study, involved the attention method & transformer-based blocks. The introduction of Perceiver (17) enabled the encoding of information with a smaller number of latent feature arrays, internal to operator blocks, thereby operating with more abstract feature arrays and maintaining a limited number of parameters. As the operators we employ, we use blocks based on the Perceiver IO (18), where the mapping is performed with a symmetrical cross-attention mechanism for outputs, which mirrors the cross-attention block for constructing representations of latent arrays from the input process. While the previously used self-attention blocks can discover dependencies between hidden features, obtained from lifting or previous layers, Perceivers are able to constructs additional latent process representation.

The latent variables and input state are combined first with the cross-attention block, where keys and values are obtained from FNO-based mapping from the inputs $K_1 = FNO_{K_1}(X)$, $V_1 = FNO_{V_1}(X)$, and latent variables are taken as queries $Q_1 = L$. The cross-attention block is followed by self-attention between latent representation. The output of the block is constructed with the cross-attention, matching the queries from the inputs with the keys and values, taken from the transformed latent representations.

Commonly used self-attention mechanism involves similarity function $\mathrm{sim}(q_m, k_j)$ between given sets of finite-dimensional vectors of queries $\{\mathbf{q}_i\},\ i = 0,\ ...\ , N_q$ and keys $\{\mathbf{k}_i\},\ i = 0,\ ...\ , N_k$, which is used to obtain the output from the m-th query with the set of value vectors $\{\mathbf{v}_i\},\ i = 0,\ ...\ , N_v$ with the relation. The similarity is commonly obtained using *Softmax* function of the dot products between corresponding queries and keys. Codomain attention mechanisms, introduced in (13), are advantageous to the conventional transformers in the neural-operator based problems: the dot product detecting similarity not between samples, but between features, mapped with neural operators.

Pre-training and fine-tuning: One of the benefits of using lifting-operator-projection architecture is the simplicity of decoupling adapters from the main model, streamlining model storage and extension for novel fine-tuning problems. The `lift` and `proj` blocks are considered as the adapters, representing the mappings, associated with the problem-specific part of dynamics: they are introduced to contain different cardinality input sets, projecting into the fixed number of hidden features and contain small number of parameters to represent limited part of the total model variance, as it is common in the adapter design for large language models (19).

In the pre-training phase the entire parameters set $(\theta_{\mathcal{P}_1},\ ...\ , \theta_{\mathcal{P}_N}, \theta_{\mathcal{F}}, \theta_{\mathcal{L}_1},\ ...\ , \theta_{\mathcal{L}_N})$ is subject to optimization. By problems 1 to N, we present separate physical processes, demanding different (but, probably, overlapping) sets of input functions. Previously, such problems were solved with liftings with extensive inputs. For example, in training on the set of steady-state problems, coefficients before

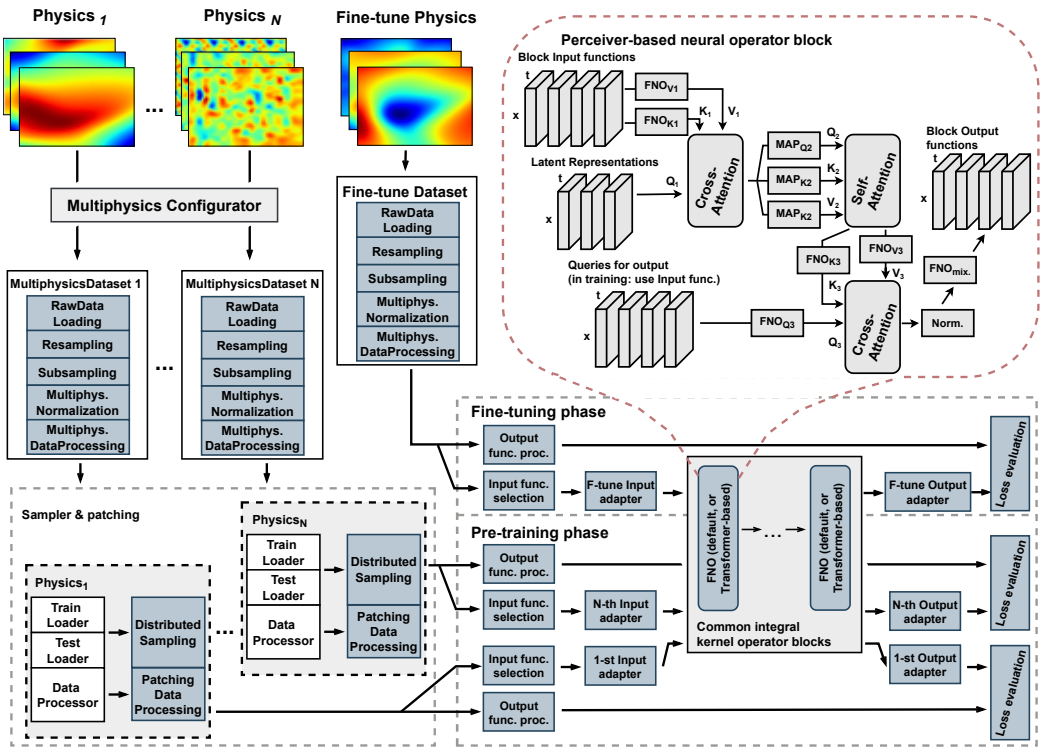

Figure 1: Scheme of the neural operator pre-training and fine-tuning stages. In this scheme, FNO blocks denote arbitrary kernel integral operators, including the transformer-based architectures. The separate physics 1 to N may vary from the different manifestations of the same system to multiple different physics (but with the same problem dimensionality) in the dataset collection.

terms with all spatial derivatives, which occur in the training set, are used as inputs. In the fine-tuning stage we fix the parameters $\theta_{\mathcal{F}}$ both to highlight the generalizing properties of the operator and to reduce training costs: only the new adapter parameters $(\theta_{\mathcal{P}_{ft}}, \theta_{\mathcal{L}_{ft}})$ are trained.

4 Experiments

We aim on validating our approach on three distinct types of problems without modifications in the modeling approach. The first scenario involves cases, when the pre-training and fine-tuning processes are governed by the equations with same input functions, but with different parameters. In addition to parametric variations within a single physical law, we further extend our investigation to cross-domain transfer learning between classes of differential equations representing distinct physical phenomena. We have selected datasets to represent diverse physical phenomena, including advective transport processes, nonlinear wave dynamics governed by Burgers equation, reaction-diffusion systems exhibiting pattern formation, and other fundamental physical processes described by partial differential equations.

$$\mathrm{NMAE}(\theta) = \frac{1}{|\mathcal{D}_{\mathrm{RD}}^{\mathrm{test}}|} \sum_{(\mathbf{a},u)\in\mathcal{D}_{\mathrm{RD}}^{\mathrm{test}}} \frac{\|\mathcal{G}_\theta(\mathbf{a}) - u\|_{1,G}}{\max_G u - \min_G u + \varepsilon} \tag{3}$$

In this comparison, we employ developed post-lifting (PL) MambaFNO models, post-lifting (PL) LocalAttnFNO, and Perceiver IO-based NO, as well as Swin-v2 transformers and CodaNO models. As the baseline, we employed the default FNO. We used the range-normalized mean absolute error (NMAE) (3) as a quality metric. The code and experiments are available in the repository `https://anonymous.4open.science/r/multiphysics_neurop-F385/`.

Out-of-sample parameter values scenario First, we conducted several experiments on cases where the pretraining equations and fine-tuning ones differed only in the coefficient values. The experiments were conducted using Burgers' equation , the Gray-Scott model of the reaction-diffusion process , and the Navier-Stokes equations for an incompressible flow. The results of the comparison are presented in Tab. 1.

Table 1: Average metric values for out-of-sample parameter values across all conducted experiments. Methods were compared with training from scratch on the examined datasets scenario and using the pre-trained model to fine-tune on the new dynamics.

Model	MSE	NMAE (%)	Avg. epoch (s)	Param.
Mamba FNO (pretr.)	1.009×10^{-7}	0.0120	21.91	$\approx 10^7$
Mamba FNO (scratch)	1.193×10^{-7}	0.0213	40.14	$\approx 10^7$
Perc. (pretr.)	1.425×10^{-7}	0.0169	**3.21**	$\approx 10^8$
Perc. (scratch)	1.981×10^{-7}	0.0219	204.73	$\approx 10^8$
FNO (scratch)	1.774×10^{-7}	0.0204	7.44	$\approx 10^6$
Swin-v2 (p.+s.)	4.391×10^{-8}	**0.0092**	101.3	$\approx 10^9$
CoDA-NO (pretr.)	2.881×10^{-7}	0.0343	62.91	$\approx 10^8$
CoDA-NO (scratch)	4.912×10^{-7}	0.0712	63.29	$\approx 10^8$

Input function set extension scenario & General multi-physics learning To assess the applicability of the adapter-based approach, several experiments were conducted on scenarios where the equations were extended with additional terms. Here, for fine-tuning, we added convection to the heat equation and extended reaction-diffusion equations with advection.

In the final stage, we evaluated the capabilities of the developed methods to transfer knowledge from the dynamics of advection and Burgers' equation to reaction–diffusion, based on the PDEBench dataset (20). The combined results of the experiments are presented in Tab. 2. The significant speedup achieved with the pre-trained models can be attributed to the optimization of just a subset of parameters, whereas "from scratch" involved a full parameter search.

Table 2: Results of experiments with Heat & Reaction–Diffusion equation extension, and multi-physics pre-training with fine-tuning on different dynamics.

Model	MSE	NMAE (%)	Avg. epoch (s)
Mamba FNO (pretr.)	3.91×10^{-6}	**0.0041**	131.2
Mamba FNO (scratch)	4.291×10^{-6}	0.0054	261.1
Perc. (pretr.)	4.107×10^{-6}	0.0051	**20.4**
Perc. (scratch)	6.315×10^{-6}	0.0074	804.0
FNO (scratch)	7.286×10^{-6}	0.0121	41.3
Swin-v2 (p.+s.)	6.276×10^{-6}	0.009	301.1
CoDA-NO (pretr.)	1.043×10^{-5}	0.013	185.1
CoDA-NO (scratch)	1.239×10^{-5}	0.018	181.9

5 Conclusion

In this research, we have examined the performance of NO architectures, based on transformers and structured state space models, on a set of transfer learning problems involving changes to the parameters of equations, the inclusion of additional physics, and, finally, pre-training on multiphysics datasets. In contrast to the default NO approach, developed models have greater generalizing power. Use of parameter sets and problem-specific adapters enables the more effortless transfer of knowledge, reducing the cost of obtaining a decent model, even for novel PDE-based problems.

This research was conducted primarily as the first stage of work towards a foundational model, pre-trained on vast and heterogeneous multiphysics dataset collections. The follow-up work shall be directed towards improving model performance and training, implementing data augmentation tools (both generic and PDE-specific, e.g., based on Lie symmetries), and further examining neural operator generalizations.

References

[1] Raissi, M., P. Perdikaris, G. 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.

[2] Lu, L., P. Jin, G. Pang, et al. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[3] Kovachki, N., Z. Li, B. Liu, et al. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[4] Bommasani, R. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[5] Awais, M., M. Naseer, S. Khan, et al. Foundation models defining a new era in vision: a survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[6] Achiam, J., S. Adler, S. Agarwal, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[7] Goswami, S., K. Kontolati, M. D. Shields, et al. Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 4(12):1155–1164, 2022.

[8] Subel, A., Y. Guan, A. Chattopadhyay, et al. Explaining the physics of transfer learning in data-driven turbulence modeling. *PNAS nexus*, 2(3):pgad015, 2023.

[9] Subramanian, S., P. Harrington, K. Keutzer, et al. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *Advances in Neural Information Processing Systems*, 36:71242–71262, 2023.

[10] Wang, H., J. Li, A. Dwivedi, et al. Beno: Boundary-embedded neural operators for elliptic pdes. *arXiv preprint arXiv:2401.09323*, 2024.

[11] Zhang, R., Q. Meng, Z.-M. Ma. Deciphering and integrating invariants for neural operator learning with various physical mechanisms. *National Science Review*, 11(4):nwad336, 2024.

[12] Herde, M., B. Raonic, T. Rohner, et al. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

[13] Rahman, M. A., R. J. George, M. Elleithy, et al. Pretraining codomain attention neural operators for solving multiphysics pdes. *Advances in Neural Information Processing Systems*, 37:104035–104064, 2024.

[14] Boya, S. K., D. Subramani. A physics-informed transformer neural operator for learning generalized solutions of initial boundary value problems. *arXiv preprint arXiv:2412.09009*, 2024.

[15] Fanaskov, V. S., I. V. Oseledets. Spectral neural operators. In *Doklady Mathematics*, vol. 108, pages S226–S232. Springer, 2023.

[16] Gu, A., T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[17] Jaegle, A., F. Gimeno, A. Brock, et al. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[18] Jaegle, A., S. Borgeaud, J.-B. Alayrac, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[19] Hu, E. J., Y. Shen, P. Wallis, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[20] Takamoto, M., T. Praditia, R. Leiteritz, et al. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.