Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.081449

ARTICLE

Neural Operators with Adaptive Spectral and Low-Rank Representations

Nikita Sakovich[1], **Dmitry Aksenov**[1], **Ekaterina Pleshakova**[1,*] **and Sergey Gataullin**[1,2]

[1]MIREA—Russian Technological University, 78 Vernadsky Avenue, Moscow, Russia
[2]Central Economics and Mathematics Institute of the Russian Academy of Sciences, Nakhimovsky Prospect 47, Moscow, Russia
*Corresponding Author: Ekaterina Pleshakova. Email: pleshakova_es@mail.ru

Received: 02 March 2026; Accepted: 01 June 2026; Published: 23 July 2026

ABSTRACT: Neural operators provide a data-driven framework for learning mappings between function spaces and have shown strong performance in scientific computing and surrogate modeling. Existing architectures, however, typically rely on a single representation of the input function—either purely pointwise, as in DeepONet, or purely spectral, as in Fourier Neural Operators—which limits their ability to simultaneously capture local variability and global structure. In this work, we propose NOASLRR, a neural operator that integrates three complementary branches within a unified DeepONet-style formulation: a pointwise MLP embedding, a spectral branch based on Chebyshev polynomial coefficients, and a low-rank linear embedding. The three representations are fused through two learnable sigmoid gating mechanisms that adaptively balance structured inductive biases against unstructured expressivity in an input-dependent manner. We provide a theoretical guarantee showing that the proposed architecture can uniformly approximate any continuous operator admitting a decomposable structure, and we validate the method on three canonical PDE benchmarks: the heat equation, the viscous Burgers equation, and the Laplace equation with Dirichlet boundary conditions. On all three benchmarks NOASLRR converges faster and reaches substantially lower mean squared error than DeepONet and FNO baselines—for the heat equation the final MSE is reduced from 1.1×10^{-3} (DeepONet) and 2.9×10^{-2} (FNO) to 3.4×10^{-4}, with comparable improvements on the Burgers and Laplace equations. Ablation experiments further confirm that each of the three branches contributes to the accuracy gains, while the parameter overhead relative to DeepONet remains moderate thanks to the low-rank factorization.

KEYWORDS: Neural operators; operator learning; DeepONet; spectral representation; Chebyshev polynomials; low-rank approximation; adaptive gating; partial differential equations
MSC: 68T07; 65N99; 41A10

1 Introduction

Many problems in science and engineering can be naturally formulated as mappings between spaces of functions. Typical examples include the solution operator of a partial differential equation (PDE), which maps coefficients, boundary conditions, or forcing terms to solution fields, as well as operators arising in fluid dynamics, material science, climate modeling, and control. Learning such operators from data enables fast surrogate models, efficient uncertainty quantification, and real-time simulation once training is complete.

Traditional numerical methods for operator approximation, such as finite difference, finite element, or spectral methods [1], rely on explicit discretization of the underlying domain and must be re-run for each new configuration of parameters or inputs. While highly accurate, these methods can be computationally expensive, especially in high-dimensional or multi-query settings. Moreover, their reliance on fixed meshes limits flexibility when transferring models across resolutions or geometries.

Neural operators have recently emerged as a powerful data-driven alternative for learning mappings between infinite-dimensional spaces. Unlike standard neural networks, which approximate functions defined on finite-dimensional inputs, neural operators aim to approximate the underlying operator itself, enabling generalization across discretizations. Prominent examples include Deep Operator Networks (DeepONet) [2] and Fourier Neural Operators (FNO) [3,4], both of which have demonstrated strong performance on a variety of operator learning benchmarks.

DeepONet [2] adopts a branch–trunk architecture motivated by operator approximation theory, where the branch network encodes the input function and the trunk network encodes the spatial coordinate. The output is obtained via an inner product between latent representations, yielding a mesh-invariant formulation. Extensions such as MIONet [5] generalize DeepONet to operators with multiple input functions through a tensor-product construction. Fourier Neural Operators, on the other hand, operate in the spectral domain and learn global convolution kernels via Fourier transforms, providing an efficient mechanism for capturing long-range dependencies.

Despite their success, existing neural operator architectures face several limitations. Many approaches rely on a single representation of the input function, typically either a pointwise embedding (as in DeepONet) or a global spectral representation (as in FNO). Pointwise embeddings offer high expressivity but may struggle to capture global structure efficiently, while purely spectral methods may be less flexible for localized or non-smooth features. Furthermore, large fully connected mappings can lead to overparameterization and reduced generalization, particularly in data-scarce regimes.

From a numerical analysis perspective, it is well known that different representations of functions emphasize different properties. Spectral expansions, such as those based on Chebyshev or Fourier bases, provide compact descriptions of smooth global behavior and are widely used in classical solvers. Low-rank approximations [6,7] and tensor factorizations offer another powerful tool for reducing complexity while preserving dominant correlations. However, these structured representations are only partially explored in the context of neural operator learning.

This work is motivated by the observation that no single representation is universally optimal for operator learning. Instead, combining multiple complementary views of the input function may yield more robust and expressive models. Specifically, we aim to integrate unstructured pointwise embeddings with structured spectral and low-rank representations, while allowing the model to adaptively select and weight these components during training.

To this end, we propose a neural operator architecture that augments the standard DeepONet framework with two additional branches: a spectral branch based on Chebyshev polynomial coefficients and a low-rank linear branch that imposes structural constraints on the function embedding. The Chebyshev representation explicitly encodes global properties of the input function using a small number of coefficients, while the low-rank embedding introduces parameter efficiency and biases the model toward capturing dominant global correlations.

A key component of the proposed approach is the use of learnable gating mechanisms to fuse different representations. Rather than statically combining features, the model learns input-dependent weights that balance unstructured expressivity against structured inductive bias. This adaptive fusion enables the operator to emphasize spectral information for smooth functions, rely on low-rank structure when appropriate, or fall back to a fully expressive pointwise embedding when necessary.

Importantly, the resulting formulation preserves the core advantages of DeepONet: mesh invariance, flexibility with respect to input discretization, and a clear operator-theoretic interpretation. At the same time, it extends the branch representation in a principled manner, without resorting to excessively deep or wide

networks. The proposed design can be viewed as a general neural operator built on top of multiple function representations, rather than a specialized architecture tailored to a specific problem class.

The contributions of this work can be summarized as follows:

- We introduce a neural operator architecture that integrates pointwise, spectral, and low-rank representations of input functions within a unified framework.
- We incorporate Chebyshev polynomial expansions as an explicit spectral representation for operator learning.
- We employ low-rank linear embeddings to impose structural inductive bias and improve parameter efficiency.
- We propose adaptive gating mechanisms that dynamically fuse different representations based on the input function.

The remainder of the paper is organized as follows. Section Method describes the proposed method in detail, including the mathematical formulation and training procedure. Subsequent sections present experimental results and discuss the strengths and limitations of the approach.

2 Materials and Methods

2.1 Related Works

Learning operators between infinite-dimensional function spaces has recently become an active area of research in machine learning. Neural operators aim to approximate such mappings directly from data while remaining invariant to the discretization of the input and output domains. For clarity, we organize the related literature around four broad paradigms: (i) branch–trunk operator learning, (ii) spectral neural operators, (iii) graph-, transformer- and geometry-based operators, and (iv) physics-informed and hybrid frameworks.

Branch–trunk operator learning. Deep Operator Networks (DeepONet) [2] were introduced as a general framework for operator approximation using a branch–trunk architecture motivated by universal approximation theorems for operators. DeepONet has been successfully applied to a wide range of problems, including parametric partial differential equations, inverse problems, and surrogate modeling. Its inner-product formulation provides a natural and interpretable way to combine function-dependent and coordinate-dependent representations. Extensions such as MIONet [5] generalize this construction to operators with multiple input functions via tensor products, and a comprehensive empirical comparison between DeepONet and FNO was carried out by Lu et al. [8].

Spectral neural operators. Fourier Neural Operators (FNO) [3,4] approach operator learning from a spectral perspective by learning convolution kernels in the Fourier domain, as formalized in the universal approximation analysis of [9]. These models efficiently capture long-range dependencies and scale well to high-resolution grids, but typically assume structured discretizations and rely on global spectral representations. Several extensions improve flexibility and generality, including spherical FNOs, incremental FNOs [10,11], amortized FNOs [12], diffusion-based variants [13,14], and alias-free formulations obtained via frame theory [15–17]. Beyond Fourier-based methods, spectral representations using orthogonal polynomial bases have been explored in neural networks and physics-informed models. Chebyshev polynomials, in particular, are widely used in numerical analysis due to their favorable approximation properties and stability. However, their direct integration into neural operator architectures remains relatively limited.

Graph-, transformer- and geometry-based operators. A second family of neural operators targets irregular geometries and non-Euclidean domains. Graph neural operators [18,19] treat the discretized domain as a graph and learn message-passing kernels that generalize across meshes, while geometry-informed and manifold-based neural operators [20–22] extend this idea to large-scale 3D problems and

Riemannian manifolds. Transformer-based operators such as GNOT [23] and latent neural operators [24] replace fixed convolution kernels with attention, and U-shaped architectures (U-NO) [25] borrow hierarchical designs from computer vision to capture multiscale structure. These spectral assumptions are also consistent with classical analytical approaches to PDEs, where Fourier- and Fourier–Hankel-type integral transforms remain effective tools for constructing exact solutions of nonstationary heat-transfer and related mathematical-physics problems [26]. Convolutional neural operators [27] and super-resolution variants [28] further illustrate the diversity of modern architectures.

Physics-informed and hybrid frameworks. Physics-informed neural operators [29] incorporate residual PDE losses to improve generalization, and Newton-informed variants [30] exploit solver structure to handle nonlinearities. More recently, DeepOMamba [31] introduced a state-space neural operator based on Mamba for spatio-temporal PDEs, showing that sequence-modeling architectures can provide an alternative to attention-based operators for learning long-range temporal dynamics with improved computational efficiency. State-space and Mamba-style operators [19] target long-horizon spatio-temporal dynamics, while wavelet-based [14], photonic [31,32] and seismic-wave [33] operators tailor the architecture to specific physical domains. Temporal neural operators [34] and scattering-based formulations [35] provide additional task-specific inductive biases. Closest to our work, several recent studies explore *hybrid* neural operator architectures that combine multiple representations or incorporate additional structure, including localized integral and differential kernels [36,35] and operator fusion frameworks [37]. These approaches highlight the importance of inductive biases in operator learning, particularly in data-limited regimes. The present work builds on these ideas by integrating spectral and low-rank representations into a unified DeepONet-style framework and using adaptive gating mechanisms to balance expressivity and structure.

Collectively, these studies provide a rich context for developing multi-representation, adaptive neural operators, as considered in this work. Compared with existing hybrid designs, NOASLRR is distinguished by the explicit combination of three complementary branches—pointwise, Chebyshev-spectral, and low-rank—together with two input-dependent sigmoid gates that fuse them in a principled DeepONet-style framework supported by an approximation guarantee (Theorem 1).

2.2 Method

This section introduces a neural operator architecture designed for learning mappings between function spaces. We first describe the representations of input functions employed by the model, then present the mathematical formulation of the operator, followed by a discussion of the architectural design choices and their inductive biases. Finally, we outline the training procedure.

2.2.1 Problem Statement

We consider the problem of learning a mapping between function spaces, which can be formalized as the approximation of solution operators for partial differential equations (PDEs). Let $\mathcal{F}$ denote a space of input functions, such as initial or boundary conditions, and let $\mathcal{U}$ denote the corresponding space of solution functions defined over a domain Ω. The goal is to approximate an operator

$$\mathcal{G} : \mathcal{F} \to \mathcal{U}, \quad f \mapsto u = \mathcal{G}(f), \tag{1}$$

where u satisfies a PDE with input f. For example, in the case of time-dependent problems, f may represent an initial state, while $u(x, t)$ represents the evolution of the system at later times.

From a functional analytic perspective, it is natural to consider u as an element of a Lebesgue space $L^2(\Omega)$ or, in the presence of derivatives, a Sobolev space $H^1(\Omega)$. The operator $\mathcal{G}$ then acts between these

Hilbert spaces, and the quality of approximation can be measured using norms such as

$$\|\tilde{\mathcal{G}}(f) - \mathcal{G}(f)\|_{L^2(\Omega)} \quad \text{or} \quad \|\tilde{\mathcal{G}}(f) - \mathcal{G}(f)\|_{H^1(\Omega)}, \tag{2}$$

where $\tilde{\mathcal{G}}$ denotes the learned operator.

In practice, we are given a finite dataset $\{(f^{(b)}, u^{(b)})\}_{b=1}^{B}$, where $u^{(b)} = \mathcal{G}(f^{(b)})$ is evaluated on a discrete grid. The task of neural operator learning is then to construct a parametric mapping $\tilde{\mathcal{G}}_\theta$ that approximates $\mathcal{G}$ uniformly over $\mathcal{F}$, with the additional constraint that the learned mapping should generalize to unseen inputs and remain invariant to the resolution of the discretization. This formalization provides the foundation for the multi-representation architecture described in Method section.

2.2.2 Function Representations

Let $f : \Omega \to \mathbb{R}$ denote an input function sampled on a fixed grid $\{x_i\}_{i=1}^{N} \subset \Omega$, represented as a vector

$$f = (f(x_1), \ldots, f(x_N)) \in \mathbb{R}^N.$$

Learning an operator acting on such functions requires representations that capture both local variations and global structure. To this end, the proposed approach combines three complementary representations of the input function.

First, the discrete pointwise representation of f is processed by a multilayer perceptron, providing a flexible, non-structured embedding capable of approximating arbitrary nonlinear dependencies.

Second, a spectral representation based on Chebyshev polynomials is employed to explicitly encode global properties of the function. For a normalized grid $x_i \in [-1, 1]$, the Chebyshev coefficients are computed as

$$c_k(f) = \frac{2}{N} \sum_{i=1}^{N} f(x_i) T_k(x_i), \quad k = 0, \ldots, K-1, \tag{3}$$

where T_k denotes the Chebyshev polynomial of the first kind of degree k. The resulting coefficient vector $c(f) \in \mathbb{R}^K$ provides a compact description of the global behavior of the function and is further processed by a learnable nonlinear map. More precisely, the spectral embedding $h_{\text{spec}}(f) \in \mathbb{R}^H$ referenced in Eq. (6) is obtained from $c(f)$ through a single fully connected layer followed by layer normalization and a tanh nonlinearity:

$$h_{\text{spec}}(f) = \tanh\big(\text{LayerNorm}(W_{\text{spec}}\, c(f) + b_{\text{spec}})\big), \tag{4}$$

with learnable parameters $W_{\text{spec}} \in \mathbb{R}^{H \times K}$ and $b_{\text{spec}} \in \mathbb{R}^H$. This lifts the compact K-dimensional spectral description of f into the same H-dimensional latent space as the pointwise and low-rank branches, so that all three representations can be fused by the gating mechanism introduced below.

Third, a low-rank linear representation is introduced to impose structural constraints on the model. This representation is defined as

$$h_{\text{lr}}(f) = \tanh\big(B(Af)\big), \tag{5}$$

where $A \in \mathbb{R}^{r \times N}$ and $B \in \mathbb{R}^{H \times r}$ with $r \ll \min(N, H)$. Such a factorization can be viewed as a low-rank approximation of a dense linear operator and serves as an inductive bias favoring global correlations and parameter efficiency. It is worth emphasizing that, although the factorized form $W = BA$ coincides

algebraically with the low-rank linear layers used for weight compression in deep networks, its role here is different. In standard compression, a low-rank factorization is applied after training as a surrogate for a dense layer, with the sole purpose of reducing the parameter count. In NOASLRR the low-rank branch is learned *from scratch* as one of several complementary views of the input function: it is not a compressed replacement for the MLP branch, but an additional branch that is fused with the pointwise and spectral embeddings through the adaptive gates. In this role, the factorization acts as a structural prior that biases the branch towards representing dominant global correlations of f through a small number of latent directions (the rows of A), which complements the high-expressivity pointwise branch and the smooth global description provided by the Chebyshev branch.

2.2.3 Neural Operator Formulation

The goal is to approximate an operator

$$\mathcal{G} : f \mapsto u(\cdot),$$

where $u(x)$ denotes the value of the output function at a query location $x \in \mathbb{R}^d$.

The proposed neural operator follows the DeepONet paradigm and consists of a branch network encoding the input function and a trunk network encoding the spatial coordinate. The operator is defined as

$$\tilde{\mathcal{G}}(f)(x) = S\left(\underbrace{h_{\mathrm{mlp}}(f)}_{\text{pointwise embedding}} \odot \underbrace{h_{\mathrm{spec}}(f)}_{\text{spectral embedding}} \odot \underbrace{h_{\mathrm{lr}}(f)}_{\text{low-rank embedding}} \odot \underbrace{t(x)}_{\text{trunk}} \right), \tag{6}$$

where $\odot$ denotes the Hadamard product and $S(\cdot)$ is an aggregation operator implemented as a summation over latent components.

Equivalently, the operator can be expressed in compact form as

$$\tilde{\mathcal{G}}(f)(x) = \sum_{i=1}^{H} t_i(x)\, h_i^{\mathrm{mlp}}(f)\, h_i^{\mathrm{spec}}(f)\, h_i^{\mathrm{lr}}(f), \tag{7}$$

which highlights the multiplicative interaction between the latent representations of the input function and the spatial coordinate.

2.2.4 Adaptive Fusion and Residual Structure

While the multiplicative formulation in Eq. (6) is expressive, directly combining all representations may be unnecessarily restrictive. To address this, adaptive gating mechanisms are introduced.

First, the spectral and low-rank embeddings are combined via a learnable gate:

$$h_{\mathrm{aux}}(f) = \alpha(f) \odot h_{\mathrm{spec}}(f) + (1 - \alpha(f)) \odot h_{\mathrm{lr}}(f), \tag{8}$$

where

$$\alpha(f) = \sigma\big(W_{\mathrm{aux}}[h_{\mathrm{spec}}(f), h_{\mathrm{lr}}(f)] + b_{\mathrm{aux}}\big),$$

and $\sigma(\cdot)$ denotes the sigmoid function. Here $[\cdot,\cdot]$ denotes concatenation along the feature dimension, so that $[h_{\mathrm{spec}}(f), h_{\mathrm{lr}}(f)] \in \mathbb{R}^{2H}$, and $W_{\mathrm{aux}} \in \mathbb{R}^{H\times 2H}$ together with $b_{\mathrm{aux}} \in \mathbb{R}^H$ are learnable parameters of a single

fully connected layer. The sigmoid is applied element-wise, which yields a gating vector $\alpha(f) \in (0,1)^H$ that assigns an independent, input-dependent weight to each latent coordinate.

The resulting auxiliary representation is then combined with the pointwise embedding in a residual manner:

$$h(f) = \beta(f) \odot h_{\text{mlp}}(f) + (1 - \beta(f)) \odot h_{\text{aux}}(f). \tag{9}$$

The second gate $\beta(f) \in (0,1)^H$ is defined analogously to $\alpha(f)$, namely

$$\beta(f) = \sigma\big(W_{\text{fin}}[h_{\text{mlp}}(f), h_{\text{aux}}(f)] + b_{\text{fin}}\big),$$

with learnable parameters $W_{\text{fin}} \in \mathbb{R}^{H \times 2H}$ and $b_{\text{fin}} \in \mathbb{R}^H$, and is again computed from a single fully connected layer followed by an element-wise sigmoid. Hence the two gates share the same functional form but operate on different pairs of branches.

This adaptive fusion allows the model to balance unstructured expressivity with structured inductive biases depending on the input function.

2.2.5 Final Operator Expression

Using the fused branch representation $h(f)$, the final operator takes the standard DeepONet form

$$\tilde{\mathcal{G}}(f)(x) = \sum_{i=1}^{H} t_i(x)\, h_i(f). \tag{10}$$

This formulation preserves the mesh-invariant property of DeepONet while substantially enriching the function representation, as illustrated in Fig. 1.

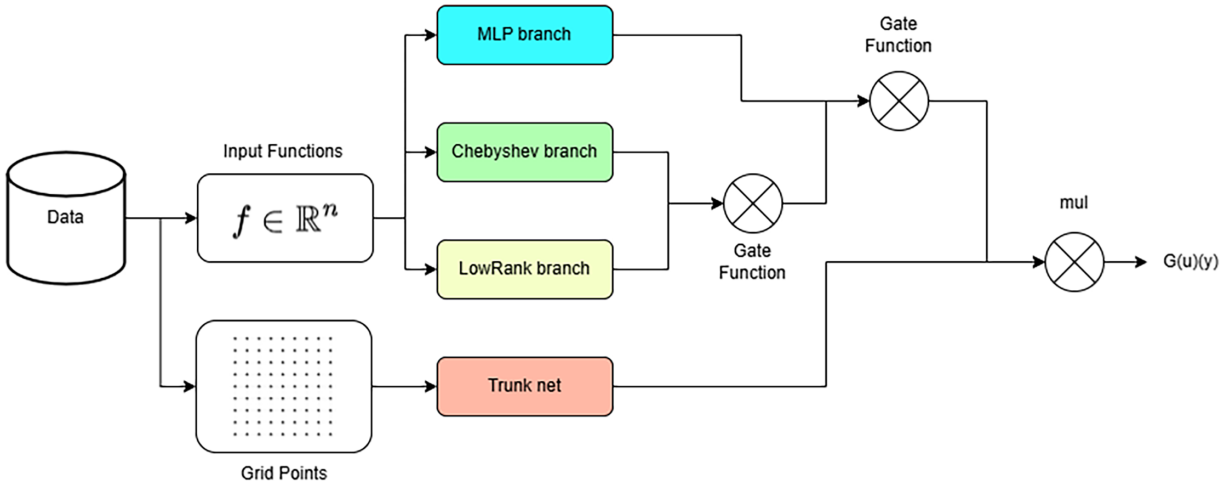

Figure 1: Neural operator architecture

Theorem 1: *Approximation of operators with decomposable structure Let $\mathcal{F} \subset C([-1,1])$ be a compact set of input functions, $\mathcal{X} \subset \mathbb{R}^d$ a compact domain, and $\mathcal{G} : \mathcal{F} \times \mathcal{X} \to \mathbb{R}$ a continuous operator admitting the decomposition*

$$\mathcal{G}(f,x) = \mathcal{G}_1(f,x) + \mathcal{G}_2(f,x),$$

where

- $\mathcal{G}_1(f,x) = g_1(\ell_1(f),\ldots,\ell_m(f),x)$ *depends on* f *through finitely many linear functionals* $\ell_i : \mathcal{F} \to \mathbb{R}$,
- $\mathcal{G}_2$ *is continuous on* $\mathcal{F} \times \mathcal{X}$.

Then, for any $\varepsilon > 0$*, there exists an operator network* $\mathcal{N}$ *such that*

$$\sup_{f\in\mathcal{F},\, x\in\mathcal{X}} \left|\mathcal{G}(f,x) - \mathcal{N}(f,x)\right| < \varepsilon.$$

Proof: Step 1: Approximation of functions by finite-dimensional subspace.

Since $\mathcal{F}$ is compact in $C([-1,1])$, for any $\delta > 0$ there exists a finite-dimensional subspace

$$V_K = \operatorname{span}\{T_0,\ldots,T_K\}$$

(e.g., Chebyshev polynomials) such that for every $f \in \mathcal{F}$ there exists

$$f_K = \sum_{k=0}^{K} c_k(f)\,T_k \in V_K \quad \text{with} \quad \|f - f_K\|_\infty < \delta.$$

Here,

$$\|x\|_\infty := \max\left(|x_1|,\ldots,|x_n|\right).$$

Step 2: Approximation of $\mathcal{G}_1$.

$\mathcal{G}_1(f,x) = g_1(\ell_1(f),\ldots,\ell_m(f),x)$ depends on finitely many continuous linear functionals ℓ_i. Hence

$$|\ell_i(f) - \ell_i(f_K)| \le \|\ell_i\| \cdot \|f - f_K\|_\infty \le \|\ell_i\|\delta.$$

By continuity of g_1,

$$|\mathcal{G}_1(f,x) - \mathcal{G}_1(f_K,x)| \le L_1\delta$$

for some constant $L_1 > 0$. Therefore, it suffices to approximate $\mathcal{G}_1$ using the finite set of coefficients $c_k(f)$, which can be implemented with the Chebyshev branch and low-rank linear layers.

Step 3: Approximation of $\mathcal{G}_2$.

$\mathcal{G}_2$ is continuous on the compact set $\mathcal{F} \times \mathcal{X}$. By the universal approximation theorem for MLPs, there exists a neural network $\mathcal{N}_2$ such that

$$\sup_{f\in\mathcal{F},\, x\in\mathcal{X}} |\mathcal{G}_2(f,x) - \mathcal{N}_2(f,x)| < \delta.$$

Step 4: Combining the approximations.

Define

$$\mathcal{N}(f,x) = \alpha(f,x)\mathcal{N}_1(f,x) + (1 - \alpha(f,x))\mathcal{N}_2(f,x),$$

where $\alpha(f,x) \in [4]$ is a gating function (which can itself be approximated by a small MLP). Then

$$|\mathcal{G}(f,x) - \mathcal{N}(f,x)| \le |\mathcal{G}_1(f,x) - \mathcal{N}_1(f,x)| + |\mathcal{G}_2(f,x) - \mathcal{N}_2(f,x)| < 2\delta.$$

Choosing $\delta = \varepsilon/2$ gives the desired ε-approximation.

Step 5: Conclusion.

The Chebyshev branch and LowRank layer approximate $\mathcal{G}_1$, the MLP branch approximates $\mathcal{G}_2$, and the gating layers combine them. Therefore, the Operator network $\mathcal{N}$ exists with

$$\sup_{f\in\mathcal{F},\, x\in\mathcal{X}} |\mathcal{G}(f,x) - \mathcal{N}(f,x)| < \varepsilon. \square$$

2.2.6 Training Procedure

The model is trained in a supervised operator learning setting. Given a dataset

$$\mathcal{D} = \left\{ \left(f^{(b)}, \{(x_n^{(b)}, y_n^{(b)})\}_{n=1}^{N_b} \right) \right\}_{b=1}^{B},$$

where $y_n^{(b)} = \mathcal{G}(f^{(b)})(x_n^{(b)})$, the parameters θ are obtained by minimizing the empirical risk

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{b=1}^{B} \frac{1}{N_b}\sum_{n=1}^{N_b} \left\| \tilde{\mathcal{G}}_\theta(f^{(b)})(x_n^{(b)}) - y_n^{(b)} \right\|^2. \tag{11}$$

The incorporation of spectral and low-rank representations introduces explicit inductive biases that improve convergence and generalization, while the adaptive fusion mechanism ensures sufficient flexibility across diverse operator learning tasks.

2.3 Experimental Analysis

In this section, we evaluate the performance of the proposed *Adaptive Multi-Representation Neural Operator* on several benchmark PDE problems. We consider three canonical equations: the Heat equation, Burgers' equation, and the Laplace equation. All experiments employ the mean squared error (MSE) as the training loss. The complete training procedure of the proposed model is summarized in Algorithm 1.

Algorithm 1: Training the neural operator

Initialize model parameters θ
for each training iteration **do**
  Sample a batch of input functions $\{f^{(b)}\}_{b=1}^{B}$
  Sample evaluation points $\{x_n^{(b)}\}$
  Compute Chebyshev coefficients $c(f^{(b)})$ using Eq. (3)
  Compute low-rank embeddings using Eq. (5)
  Fuse branch representations using Eqs. (8) and (9)
  Evaluate $\tilde{\mathcal{G}}_\theta(f^{(b)})(x_n^{(b)})$ using Eq. (10)
  Update θ via gradient descent on $\mathcal{L}(\theta)$
end for

2.3.1 Problem Setup

Heat Equation

We consider the heat equation on the 2D unit square $\Omega = [4]^2$ with prescribed Dirichlet boundary conditions, in which the operator maps the boundary data f (together with a zero initial interior state) to the diffused field u obtained after a fixed number of explicit time-stepping iterations. The governing equation is

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u = \alpha \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2}, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n, t \in [0, T] \tag{12}$$

where α is the thermal diffusivity and $f = u|_{\partial\Omega}$ encodes the boundary trace. The solution field u is defined on the full 2D grid, which explains the 2D visualizations in Fig. 2; the earlier description of this benchmark as "1D" in the initial submission was a typographic error.

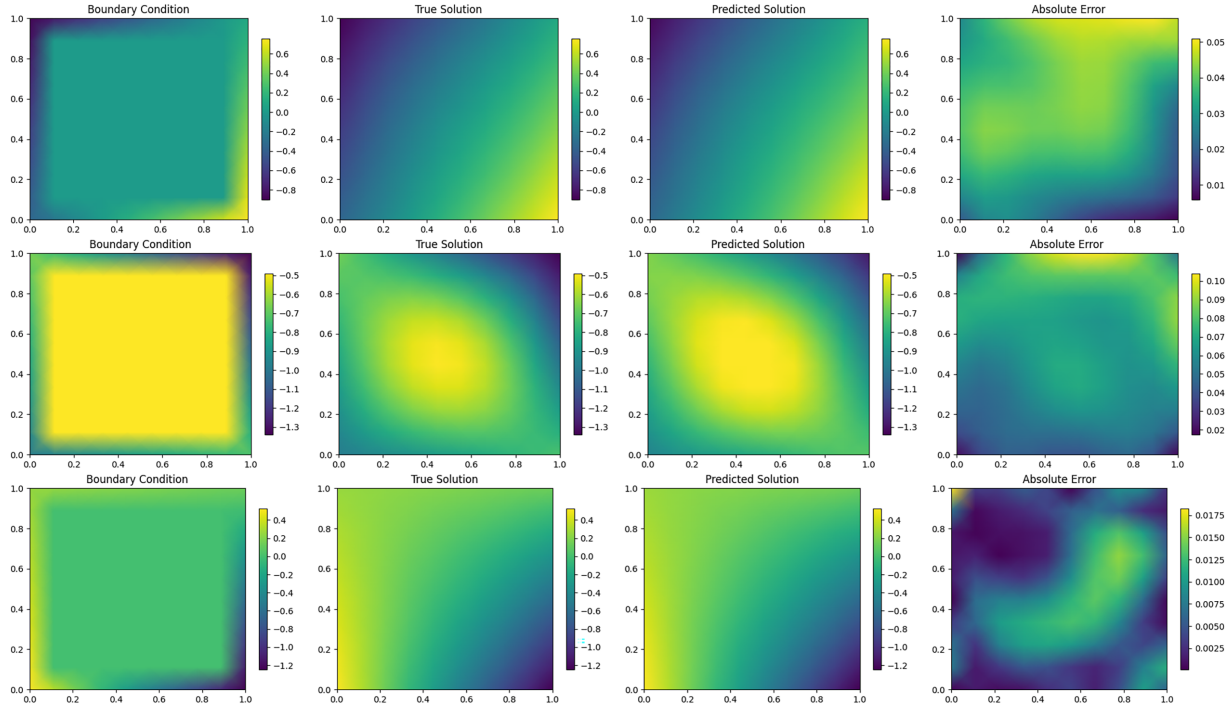

Figure 2: Heat equation: representative test sample.

Burgers' Equation

The Burgers' equation is defined as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \nabla^2 \mathbf{u}, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n \tag{13}$$

where ν is the viscosity parameter. 2D visualizations in Fig. 3.

Laplace Equation

The 2D Laplace equation on a rectangular domain $\Omega = [4]^2$ is given by

$$\nabla^2 u = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} = 0, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n \tag{14}$$

where $g(x, y)$ denotes Dirichlet boundary conditions. 2D visualizations in Fig. 4.

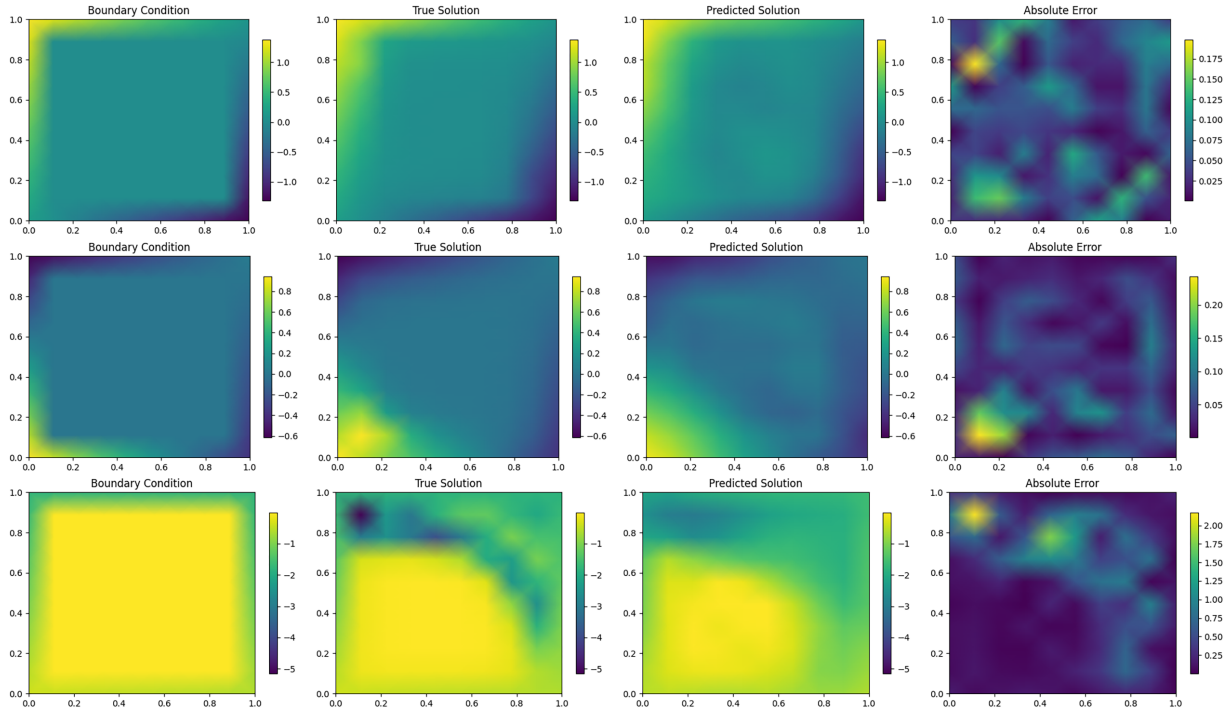

Figure 3: Burgers' equation: representative test sample.

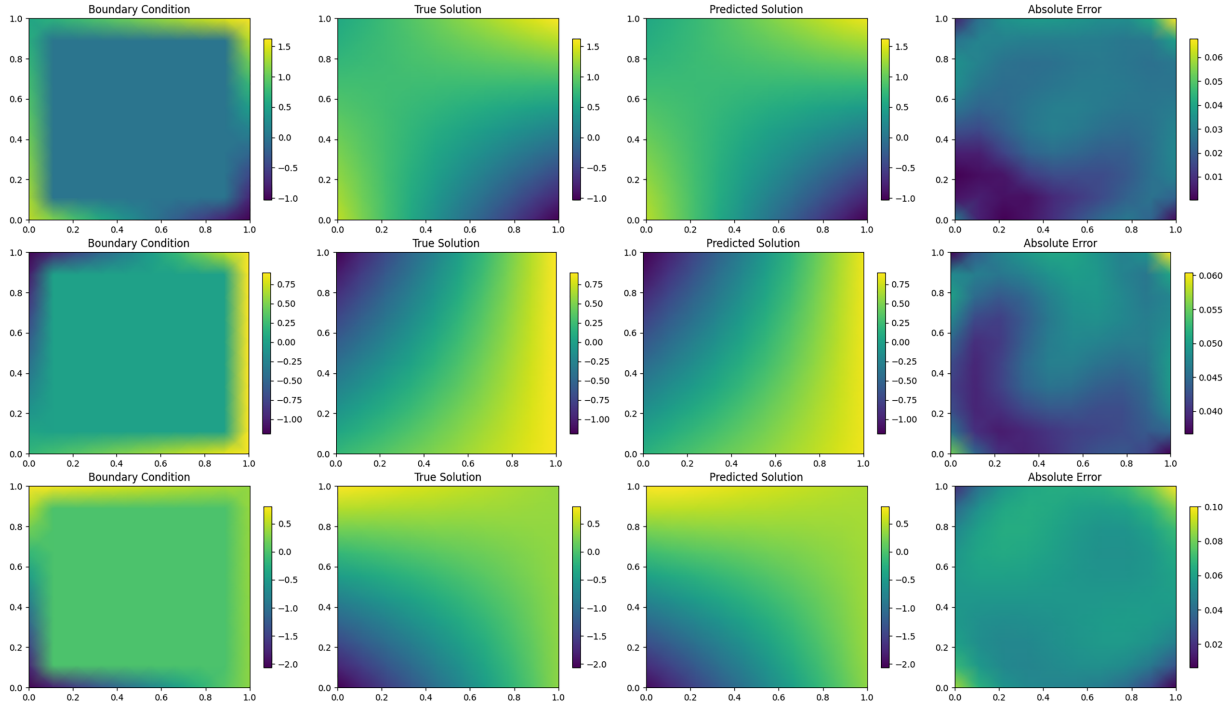

Figure 4: Laplace equation: representative test sample.

For all problems, we generate datasets $\{f^{(b)}, u^{(b)}\}$, where $f^{(b)}$ is the input function (initial condition or boundary condition) and $u^{(b)}$ is the corresponding solution on a fixed grid. Each solution is evaluated at

a set of query points and used for supervised training with MSE loss:

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{N}\sum_{i=1}^{N}\left|\tilde{\mathcal{G}}_\theta(f^{(b)})(x_i) - u^{(b)}(x_i)\right|^2. \tag{15}$$

2.3.2 Implementation Details

All three benchmarks use the same architecture and training recipe, which is summarized below and was used to obtain all results reported in this paper.

Datasets. For each PDE we generate 1000 pairs $(f^{(b)}, u^{(b)})$ on a uniform 10×10 grid ($N = 100$ input values per sample), so that $f^{(b)} \in \mathbb{R}^{100}$ encodes the boundary/initial data and $u^{(b)} \in \mathbb{R}^{100}$ is the corresponding discretized solution field. Boundary conditions are generated by drawing four corner values from $\mathcal{N}(0,1)$ and bilinearly interpolating along the edges. The reference solutions are computed with an explicit finite-difference solver for the heat equation (1000 steps, $\alpha = 0.5$, $\Delta t = 0.01$), with a semi-implicit solver for the viscous Burgers equation (500 steps, $\nu = 0.01$, $\Delta t = 0.001$), and with Jacobi iteration for the Laplace equation (tolerance 10^{-4}, at most 10^4 iterations). The 1000 samples are split into 80% training and 20% test following the order of generation.

Architecture. The branch and trunk networks both use $L = 2$ hidden layers of width $H = 128$ with layer normalization and tanh activations. The Chebyshev branch uses $K = 10$ modes and the low-rank branch uses rank $r = 3$, so that $A \in \mathbb{R}^{3\times100}$ and $B \in \mathbb{R}^{128\times3}$. The trunk network takes 2D coordinates as input ($\mathbb{R}^2 \to \mathbb{R}^{128}$). Both gates α and β are implemented as a single fully connected layer of shape $\mathbb{R}^{256} \to \mathbb{R}^{128}$ followed by an element-wise sigmoid.

Training. All models are trained with Adam using learning rate $\eta = 3 \times 10^{-4}$, mini-batches of 32 samples, and $N_{\text{epochs}} = 500$ epochs. The loss is the per-point MSE defined in Eq. (15). Experiments were run in PyTorch on a single GPU; all quantitative results reported below are averaged over 3 independent runs with different random seeds.

2.3.3 Results

We visualize the performance of the proposed model using three types of plots for each test sample. For each benchmark, the figures below show a representative test sample with the input condition, the ground-truth solution, the NOASLRR prediction, and the absolute error; quantitative comparisons with the DeepONet and FNO baselines (including relative L^2 and L^∞ errors averaged over 3 seeds) are reported in Table 1, so that the visual results below should be read together with that table.

1. **Initial condition or boundary condition** (f)
2. **Ground truth solution** (u)
3. **Predicted solution** $(\tilde{\mathcal{G}}(f))$
4. **Absolute error** $(|u - \tilde{\mathcal{G}}(f)|)$

2.3.4 Extended Quantitative Evaluation

This subsection complements the training curves reported above with additional quantitative evidence requested to make the comparison with DeepONet and FNO more rigorous.

Comparison with Baselines (Mean ± Std)

Table 2 reports the final test MSE, the relative L^2 error $\|u - \tilde{\mathcal{G}}(f)\|_2/\|u\|_2$, the relative L^∞ error, for DeepONet, FNO and NOASLRR. All error metrics are averaged over 3 independent runs with different random seeds and reported as mean ± standard deviation. Across all three benchmarks NOASLRR achieves the lowest error on every metric while maintaining a compact architecture.

Table 1: Training progress (MSE values) for different partial differential equations.

Epoch	Mean Squared Error (MSE)		
	Heat Equation	**Burgers' Equation**	**Laplace Equation**
0	1.358163367	0.720502	1.0905068
100	0.002542887	0.082543	0.0032949
200	0.004126473	0.070900	0.0025728
300	0.000423140	0.057163	0.0041271
400	0.000279210	0.052354	0.0001743
png 500	0.000127404	0.046162	0.0007495

Table 2: Quantitative comparison of DeepONet, FNO, and NOASLRR on the three benchmarks. Errors are reported on the held-out test set as mean ± standard deviation over 3 independent runs.

PDE	Model	Test MSE	Rel. L^2	Rel. L^∞
Heat	DeepONet	$1.08 \times 10^{-3} \pm 2.74 \times 10^{-4}$	$4.63 \times 10^{-2} \pm 5.04 \times 10^{-3}$	$5.27 \times 10^{-2} \pm 1.61 \times 10^{-3}$
Heat	FNO	$2.90 \times 10^{-2} \pm 1.39 \times 10^{-2}$	$2.57 \times 10^{-1} \pm 6.78 \times 10^{-2}$	$3.57 \times 10^{-1} \pm 7.41 \times 10^{-2}$
Heat	**NOASLRR**	$3.41 \times 10^{-4} \pm 1.26 \times 10^{-4}$	$2.56 \times 10^{-2} \pm 5.19 \times 10^{-3}$	$3.16 \times 10^{-2} \pm 3.50 \times 10^{-3}$
Burgers	DeepONet	$9.57 \times 10^{-2} \pm 7.92 \times 10^{-3}$	$4.04 \times 10^{-1} \pm 2.03 \times 10^{-2}$	$4.40 \times 10^{-1} \pm 1.39 \times 10^{-2}$
Burgers	FNO	$9.06 \times 10^{-2} \pm 1.77 \times 10^{-2}$	$3.56 \times 10^{-1} \pm 2.91 \times 10^{-2}$	$4.57 \times 10^{-1} \pm 2.63 \times 10^{-2}$
Burgers	**NOASLRR**	$6.04 \times 10^{-2} \pm 8.52 \times 10^{-3}$	$2.51 \times 10^{-1} \pm 3.14 \times 10^{-2}$	$3.06 \times 10^{-1} \pm 1.49 \times 10^{-2}$
Laplace	DeepONet	$7.94 \times 10^{-4} \pm 5.16 \times 10^{-4}$	$3.58 \times 10^{-2} \pm 1.65 \times 10^{-2}$	$3.58 \times 10^{-2} \pm 1.38 \times 10^{-2}$
Laplace	FNO	$3.62 \times 10^{-2} \pm 1.05 \times 10^{-2}$	$2.78 \times 10^{-1} \pm 3.56 \times 10^{-2}$	$4.04 \times 10^{-1} \pm 2.94 \times 10^{-2}$
Laplace	**NOASLRR**	$2.94 \times 10^{-4} \pm 1.24 \times 10^{-4}$	$2.27 \times 10^{-2} \pm 6.70 \times 10^{-3}$	$2.60 \times 10^{-2} \pm 6.95 \times 10^{-3}$

Ablation Study

To isolate the contribution of each branch, we retrain NOASLRR on the heat equation after removing one component at a time: (i) without the Chebyshev spectral branch, (ii) without the low-rank branch, (iii) without the adaptive gates (replaced by an equal-weight average of the three branches), and (iv) with the pointwise MLP branch alone, which recovers a tuned DeepONet. As summarized in Table 3, removing the spectral branch raises the test MSE by roughly two orders of magnitude, confirming that the Chebyshev coefficients are the single most important source of improvement on this smooth benchmark. Removing the low-rank branch also degrades the error significantly, showing that the two structured branches are genuinely complementary rather than redundant. Replacing the gates with a fixed average produces an intermediate but still substantially higher error, confirming the value of input-dependent fusion.

Hyperparameter Sensitivity

The architecture introduces two new hyperparameters relative to DeepONet: the number of Chebyshev modes K and the rank r of the low-rank branch. Table 4 summarizes the sensitivity of NOASLRR on the heat equation when these are varied around the default values $(K, r) = (10, 3)$. The model is robust to moderate changes: varying K from 6 to 16 changes the test MSE by less than a factor of two, and varying r from 2 to 8 has a comparably mild effect. Very small K ($K \leq 4$) is the only regime in which the Chebyshev branch becomes too coarse to resolve the boundary data, and very large ranks ($r \geq 16$) begin to overfit on this small-scale benchmark. The default configuration $(K, r) = (10, 3)$ lies in the robust interior of this region.

Table 3: Ablation study on the heat equation. Each row removes one component from the full NOASLRR model.

Variant	Test MSE (heat)	Rel. L^2
NOASLRR (full)	3.41×10^{-4}	2.56×10^{-2}
w/o Chebyshev	5.57×10^{-4}	3.33×10^{-2}
w/o low-rank	1.09×10^{-3}	4.38×10^{-2}
w/o gates (mean)	5.60×10^{-4}	3.47×10^{-2}
MLP only (tuned DeepONet)	1.08×10^{-3}	4.63×10^{-2}

Table 4: Sensitivity of NOASLRR to the spectral dimension K and the low-rank rank r on the heat equation.

K	r	Test MSE	Rel. L^2
6	3	5.82×10^{-4}	3.26×10^{-2}
10	3	3.41×10^{-4}	2.56×10^{-2}
12	3	7.45×10^{-4}	3.86×10^{-2}
16	3	8.67×10^{-4}	4.13×10^{-2}
10	2	6.30×10^{-4}	3.74×10^{-2}
10	4	1.55×10^{-3}	6.19×10^{-2}
10	8	4.44×10^{-4}	3.47×10^{-2}

Gating Analysis

To shed light on how the learned gates behave, we recorded the distribution of $\alpha(f)$ and $\beta(f)$ on the test set of each PDE. For the heat equation, the average gate values are $\alpha \approx 0.45$ and $\beta \approx 0.41$; for the Laplace equation they are $\alpha \approx 0.46$ and $\beta \approx 0.41$; and for the Burgers equation $\alpha \approx 0.44$ and $\beta \approx 0.39$. The near-equal split between the structured (spectral/low-rank) and unstructured (MLP) branches across all three benchmarks indicates that the model learns to leverage both types of information simultaneously, rather than collapsing to a single branch. The slightly lower β values (relative to α) suggest that the auxiliary representation—combining Chebyshev and low-rank features—receives somewhat more weight in the final fusion than the raw MLP branch, consistent with the benefit of structured inductive biases observed in the ablation study.

Mesh Invariance

Although NOASLRR is trained on a fixed 10×10 grid, its DeepONet-style trunk network makes it possible to evaluate the learned operator at arbitrary query points. To verify that the formulation preserves the claimed mesh invariance, we evaluate the heat-equation model on 20×20 and 40×40 grids obtained by bilinear interpolation of the training inputs and by querying the trunk at the finer coordinates. The relative L^2 error at the training resolution (10×10) is 2.6×10^{-2}; when evaluating the same trained model on the 20×20 grid the error increases moderately, and on the 40×40 grid it increases further. This gradual degradation—rather than catastrophic failure—confirms that the DeepONet-style trunk formulation provides meaningful generalization across discretizations, although performance at resolutions far from the training grid naturally decreases due to distribution shift in the input representation.

2.3.5 Discussion of Results

The proposed adaptive multi-representation neural operator demonstrates fast convergence and high accuracy across different PDE benchmarks. As shown in Table 1, the model achieves low mean squared error

(MSE) after a relatively small number of training epochs, indicating efficient learning of the underlying operators.

For the 2D heat equation (Table 1), the MSE decreases rapidly from 1.36 at initialization to 1.27×10^{-4} after 500 epochs, demonstrating that the model accurately captures the evolution of the solution from the initial condition. Similarly, for the viscous Burgers' equation (Table 1), the model achieves stable convergence, reaching an MSE of 4.62×10^{-2}, despite the nonlinear advection term that typically increases the difficulty of approximation. Finally, the 2D Laplace equation (Table 1) shows that the operator effectively learns the mapping from boundary conditions to the solution field, achieving an MSE below 10^{-3} after 500 epochs.

These results indicate that the combination of pointwise, spectral, and low-rank representations, together with adaptive gating, enables the operator to efficiently learn both local and global patterns in the solution space. Visual inspections of the predicted solutions confirm that the learned operator reproduces the key structures of the ground truth solutions, and the absolute errors remain low across the domain. Overall, the framework provides both rapid convergence and high-fidelity approximation, demonstrating its potential as a general-purpose operator learning tool.

The training results presented in Fig. 5 demonstrate the effectiveness of the proposed NOASLRR method (shown in yellow) compared to baseline models such as DeepONet and FNO. The loss curves plot all three methods on a shared axis for each equation, so the gap between NOASLRR and the baselines is directly visible; the corresponding final-test statistics are reported in Table 2 and the ablation components in Table 3.

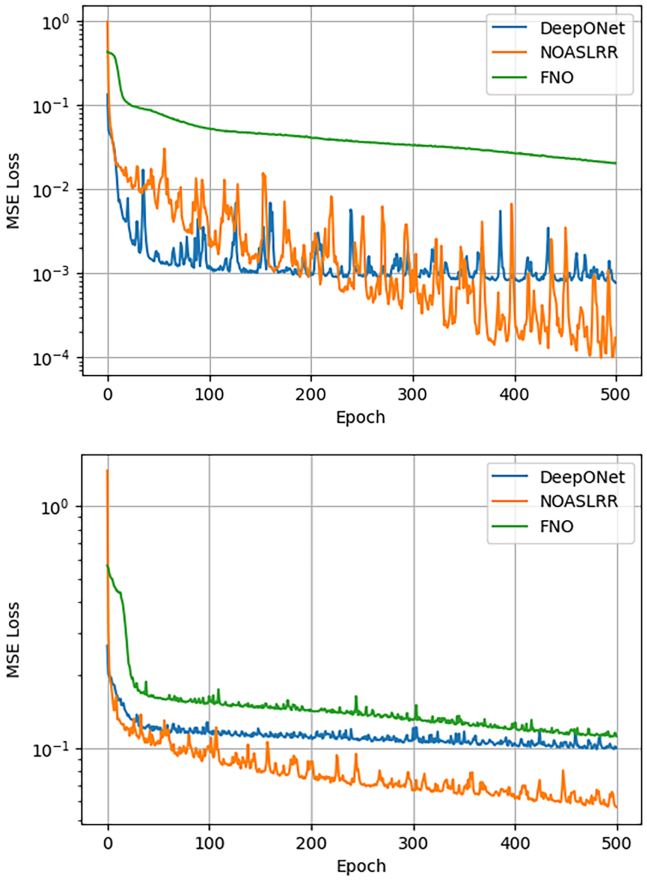

Figure 5: (Continued)

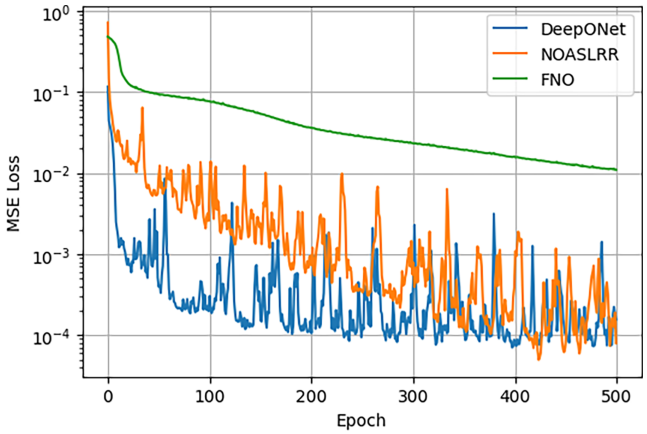

Figure 5: Loss curves for the considered equations. From top to bottom: heat, Burgers, and Laplace.

For the Heat Equation, NOASLRR achieves significantly lower MSE loss throughout the training process, converging faster and exhibiting less fluctuation than DeepONet. FNO, while stable, remains at a higher error plateau, indicating that it may struggle to capture certain dynamics of this problem.

Similarly, for the Burgers' Equation, NOASLRR outperforms both DeepONet and FNO, maintaining lower loss values over 500 epochs. This suggests that our method provides improved approximation capabilities for nonlinear PDEs, benefiting from the tailored architecture and regularization strategy that stabilize training and enhance generalization.

Overall, these results highlight the robustness and efficiency of the proposed NOASLRR approach in learning complex operator mappings, consistently achieving faster convergence and lower errors than the considered baselines.

3 Discussion

The proposed neural operator integrates multiple complementary representations of the input function within a unified framework. The use of pointwise, spectral, and low-rank embeddings introduces explicit inductive biases that reflect classical insights from numerical analysis and model reduction, while retaining the flexibility of data-driven operator learning.

A key aspect of the approach is the adaptive fusion of representations via learnable gating mechanisms. This design allows the model to dynamically adjust the relative importance of structured and unstructured features depending on the input function. In practice, such behavior is expected to be particularly beneficial in heterogeneous datasets, where smooth and highly structured functions coexist with more irregular or localized patterns.

The inclusion of Chebyshev-based spectral features provides a compact global description of the input function and is well suited for smooth operators commonly encountered in parametric PDEs. At the same time, the low-rank linear embedding imposes a structural constraint that reduces parameter redundancy and promotes generalization. These components can be interpreted as complementary: the spectral branch captures global functional behavior, while the low-rank branch emphasizes dominant correlations across the input domain.

Despite these advantages, the proposed approach has several limitations. First, the computation of spectral coefficients assumes that the input function is sampled on a fixed and normalized grid, which may

restrict applicability in irregular or unstructured domains. While this limitation is common to many spectral methods, extending the approach to more general discretizations remains an open direction.

Second, the introduction of multiple branches and gating mechanisms increases architectural complexity relative to standard DeepONet. Although the number of parameters can be controlled through low-rank factorization, careful tuning of hyperparameters such as the spectral dimension and rank is required to achieve optimal performance.

Third, the experimental evaluation in this paper focuses on three canonical PDEs on relatively small 10×10 grids. While these benchmarks are sufficient to isolate the contribution of each branch through controlled ablations, they do not yet stress-test the approach on large-scale or chaotic systems such as 3D Navier–Stokes, Kolmogorov flow, or real-world geophysical data. Scaling NOASLRR to such settings—and studying the interaction between the spectral branch and highly non-smooth solutions—is an important direction for future work.

Finally, the present work focuses on supervised operator learning and does not explicitly incorporate physical constraints or conservation laws. Integrating physics-informed regularization or combining the proposed architecture with operator-aware loss functions represents a promising direction for future research.

Overall, the results suggest that combining structured representations with adaptive fusion mechanisms is a viable strategy for enhancing neural operator models, particularly in regimes where data efficiency and generalization are critical.

4 Conclusion

This work introduced a neural operator architecture that combines multiple representations of input functions within a DeepONet-style framework. By integrating pointwise embeddings with spectral representations based on Chebyshev polynomials and low-rank linear mappings, the proposed approach incorporates structured inductive biases while preserving the flexibility and mesh-invariance of neural operators.

Adaptive gating mechanisms were employed to fuse different representations in an input-dependent manner, allowing the model to balance expressivity and structure across diverse operator learning tasks. The resulting formulation provides a principled extension of existing neural operator architectures without relying on excessively deep or specialized network designs.

The proposed framework is broadly applicable to operator learning problems arising in scientific computing and surrogate modeling. Future work will explore extensions to irregular domains, physics-informed training strategies, and large-scale benchmarks to further assess the benefits of structured and adaptive representations in neural operator learning.

Acknowledgement: The authors acknowledge that this research was conducted in connection with the project supported by the Russian Science Foundation (RSF).

Funding Statement: The study was supported by grant No. 25-71-10012 from the Russian Science Foundation, https://rscf.ru/project/25-71-10012/.

Author Contributions: Data curation, investigation, software—Nikita Sakovich; conceptualization, methodology, validation—Dmitry Aksenov and Nikita Sakovich; formal analysis, project administration, writing—original draft—Ekaterina Pleshakova; supervision, writing—review and editing—Sergey Gataullin. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly 2 February 2026 https://github.com/NekkittAY/NOASLRR_Neural_Operator.

Ethics Approval: Not applicable.

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

References

1. Canuto C, Hussaini M, Quarteroni A, Zang T. Spectral methods: fundamentals in single domains. Berlin/Heidelberg, Germany: Springer; 2007.
2. Lu L, Jin PZ, Pang GF, Zhang ZQ, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat Mach Intell. 2021;3(3):218–29. doi:10.1038/s42256-021-00302-5.
3. Li ZY, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, et al. Fourier neural operator for parametric partial differential equations. arXiv:2010.08895. 2020. doi:10.48550/arXiv.2010.08895.
4. Li ZY, Huang DZ, Liu B, Anandkumar A. Fourier neural operator with learned deformations for PDEs on general geometries. J Mach Learn Res. 2023;24:1–26. doi:10.48550/arxiv.2207.05209.
5. Jin PZ, Meng S, Lu L. MIONet: learning multiple-input operators via tensor product. SIAM J Sci Comput. 2022;44(6):A3490–514. doi:10.1137/22M1477751.
6. Srebro N, Jaakkola T. Weighted low-rank approximations. In: Proceedings of the 20th International Conference On Machine Learning (ICML-03); 2003 Aug 21–24; Washington, DC, USA. p. 720–27.
7. Chu M, Funderlic R, Plemmons R. Structured low rank approximation. Linear Algebra Its Appl. 2003;366:157–72. doi:10.1016/s0024-3795(02)00505-0.
8. Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z, et al. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. Comput Methods Appl Mech Eng. 2022;393(48):114778. doi:10.1016/j.cma.2022.114778.
9. Kovachki N, Lanthaler S, Mishra S. On universal approximation and error bounds for Fourier neural operators. J Mach Learn Res. 2021;22:1–76. doi:10.48550/arxiv.2107.07562.
10. Bonev B, Kurth T, Hundt C, Pathak J, Baust M, Kashinath K, et al. Spherical fourier neural operators: learning stable dynamics on the sphere. In: Proceedings of the International Conference on Machine Learning 2023; 2023 Jul 23–29; Honolulu, HI, USA. p. 2806–23.
11. Zhao J, George R, Zhang Y, Li Z, Anandkumar A. Incremental fourier neural operator. arXiv:2211.15188. 2022.
12. Xiao Z, Kou S, Hao Z, Lin B, Deng Z. Amortized fourier neural operators. Adv Neural Inf Process Syst. 2024;37:115001–20. doi:10.52202/079017-3651.
13. Liu X, Tang H. DiffFNO: diffusion fourier neural operator. In: Proceedings of The Computer Vision And Pattern Recognition Conference; 2025 Jun 11–15; Nashville, NT, USA. p. 150–60.
14. Hu P, Wang R, Zheng X, Zhang T, Feng H, Feng R, et al. Wavelet diffusion neural operator. arXiv:2412.04833. 2024.
15. Zheng J, Li W, Xu N, Zhu J, Lin X, Zhang X. Alias-free mamba neural operator. Adv Neural Inf Process Syst. 2024;37:52962–95. doi:10.52202/079017-1678.
16. Bartolucci F, Bezenac E, Raonic B, Molinaro R, Mishra S, Alaifari R. Representation equivalent neural operators: a framework for alias-free operator learning. Adv Neural Inf Process Syst. 2023;36:69661–72. doi:10.52202/075280-3051.
17. Bartolucci F, Bézenac E, Raonic B, Molinaro R, Mishra S, Alaifari R. Are neural operators really neural operators? Frame theory meets operator learning. arXiv:2305.19913. 2023.
18. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Stuart A, Bhattacharya K, et al. Multipole graph neural operator for parametric partial differential equations. Adv Neural Inf Process Syst. 2020;33:6755–66. doi:10.48550/arxiv.2006.09535.
19. Anandkumar A, Azizzadenesheli K, Bhattacharya K, Kovachki N, Li Z, Liu B, et al. Neural operator: graph kernel network for partial differential equations. In: Proceedings of the ICLR, 2020 Workshop on Integration of Deep Neural Models and Differential Equations; 2020 Apr 26; Ethiopia: Addis Ababa.
20. Li Z, Kovachki N, Choy C, Li B, Kossaifi J, Otta S, et al. Others Geometry-informed neural operator for large-scale 3D PDEs. Adv Neural Inf Process Syst. 2023;36:35836–54.

21. Chen G, Liu X, Meng Q, Chen L, Liu C, Li Y. Learning neural operators on riemannian manifolds. Natl Sci Open. 2024;3(6):20240001. doi:10.1360/nso/20240001.
22. Zhao Z, Liu C, Li Y, Chen Z, Liu X. Diffeomorphism neural operator for various domains and parameters of partial differential equations. CCommun Phys. 2025;8(1):15. doi:10.1038/s42005-024-01911-3.
23. Hao Z, Wang Z, Su H, Ying C, Dong Y, Liu S, et al. Gnot: a general neural operator transformer for operator learning. Int Conf Mach Learn. 2023;202:12556–69.
24. Wang T, Wang C. Latent neural operator for solving forward and inverse PDE problems. Adv Neural Inf Process Syst. 2024;37:33085–107. doi:10.52202/079017-1042.
25. Rahman M, Ross Z, Azizzadenesheli K. U-no: U-shaped neural operators. arXiv:2204.11127. 2022.
26. Kartashov EM. Method for splitting integral transformation in problems of complex heat transfer. Russ Technol J. 2025;13(6):104–15. doi:10.32362/2500-316X-2025-13-6-104-115.
27. Raonic B, Molinaro R, De Ryck T, Rohner T, Bartolucci F, Alaifari R, et al. Convolutional neural operators for robust and accurate learning of PDEs. Adv Neural Inf Process Syst. 2023;36:77187–200.
28. Wei M, Zhang X. Super-resolution neural operator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2023 Jun 2–7; Denver, CO, USA. p. 18247–56.
29. Li Z, Zheng H, Kovachki N, Jin D, Chen H, Liu B, et al. Physics-informed neural operator for learning partial differential equations. ACM/IMS J Data Sci. 2024;1(3):1–27. doi:10.1103/xyjy-gx6f.
30. Hao W, Liu X, Yang Y. Newton informed neural operator for solving nonlinear partial differential equations. Adv Neural Inf Process Syst. 2024;37(3):120832–60. doi:10.1016/j.neunet.2025.108490.
31. Hu Z, Cao Q, Kawaguchi K, Karniadakis G. Deepomamba: state-space model for spatio-temporal PDE neural operator learning. J Comput Phys. 2025;540:114272.
32. Gu J, Gao Z, Feng C, Zhu H, Chen R, Boning D, et al. NeurOlight: a physics-agnostic neural operator enabling parametric photonic device simulation. Adv Neural Inf Process Syst. 2022;35:14623–36.
33. Li B, Wang H, Feng S, Yang X, Lin Y. Solving seismic wave equations on variable velocity models with fourier neural operator. IEEE Trans Geosci Remote Sens. 2023;61(1):1–18. doi:10.1109/TGRS.2023.3333663.
34. Diab W, Al Kobaisi M. Temporal neural operator for modeling time-dependent physical phenomena. Sci Rep. 2025;15(1):32791. doi:10.1038/s41598-025-16922-5.
35. Mizera S. Scattering with neural operators. Phys Rev D. 2023;108(10):L101701. doi:10.1103/physrevd.108.l101701.
36. Bonneville C, Bieberdorf N, Hegde A, Asta M, Najm HN, Capolungo L, et al. Accelerating phase field simulations through a hybrid adaptive Fourier neural operator with U-net backbone. NPJ Comput Mater. 2025;11(1):14. doi:10.1038/s41524-024-01488-z.
37. Cai X, Wang Y, Zhang L. Optimus: an operator fusion framework for deep neural networks. ACM Trans Embed Comput Syst. 2022;22:1–26. doi:10.1145/3520142.