Improved Operator Learning by Orthogonal Attention

Zipeng Xiao [1] **Zhongkai Hao** [2] **Bokai Lin** [1] **Zhijie Deng** [1] **Hang Su** [2]

Abstract

This work presents orthogonal attention for constructing neural operators to serve as surrogates to model the solutions of a family of Partial Differential Equations (PDEs). The motivation is that the kernel integral operator, which is usually at the core of neural operators, can be reformulated with orthonormal eigenfunctions. Inspired by the success of the neural approximation of eigenfunctions (Deng et al., 2022b), we opt to directly parameterize the involved eigenfunctions with flexible neural networks (NNs), based on which the input function is then transformed by the rule of kernel integral. Surprisingly, the resulting NN module bears a striking resemblance to regular attention mechanisms, albeit without softmax. Instead, it incorporates an orthogonalization operation that provides regularization during model training and helps mitigate overfitting, particularly in scenarios with limited data availability. In practice, the orthogonalization operation can be implemented with minimal additional overheads. Experiments on six standard neural operator benchmark datasets comprising both regular and irregular geometries show that our method can outperform competing baselines with decent margins.

1. Introduction

Partial Differential Equations (PDEs) are essential tools for modeling and describing intricate dynamics in scientific and engineering domains (Zachmanoglou & Thoe, 1986). Solving the PDEs routinely rely on well-established numerical approaches such as finite element methods (FEM) (Zienkiewicz et al., 2005), finite difference methods (FDM) (Thomas, 2013), spectral methods (Ciarlet, 2002; Courant et al., 1967), etc. Due to the infinite-dimensional nature of the function space, traditional numerical solvers often rely on discretizing the data domain. However, this introduces a balance between efficiency and accuracy: finer discretization offers higher precision but at the expense of greater computational complexity.

Deep learning methods have shown promise in lifting such a trade-off (Li et al., 2020) thanks to their high inference speed and expressiveness. Specifically, physics-informed neural networks (PINNs) (Raissi et al., 2019) first combine neural networks (NNs) with physical principles for PDE solving. Yet, PINNs approximate the solution associated with a certain PDE instance and hence cannot readily adapt to problems with different yet similar setups. By learning a map between the input condition and the PDE solution in a data-driven manner, neural operators manage to solve a family of PDEs, with the DeepONet (Lu et al., 2019) as a representative example. Fourier Neural Operators (FNOs) (Li et al., 2020; Tran et al., 2023; Li et al., 2022; Wen et al., 2022; Grady et al., 2022; Gupta et al., 2021; Xiong et al., 2023) shift the learning to Fourier space to enhance speed while maintaining efficacy through the utilization of the Fast Fourier Transform (FFT). Since the development of attention mechanism (Vaswani et al., 2017), considerable effort has been devoted to developing attention-based neural operators to improve expressiveness and address irregular mesh (Cao, 2021; Li et al., 2023a; Ovadia et al., 2023; Fonseca et al., 2023; Hao et al., 2023; Li et al., 2023b).

Despite the considerable progress made in neural operators, there remain non-trivial challenges in its practical applications. On the one hand, the training targets of neural operators are usually acquired from classical PDE solvers, which can be computationally demanding. For instance, simulations for tasks like airfoils can require about 1 CPU-hour per sample (Li et al., 2022). On the other hand, complex deep models are prone to deteriorate when confronted with limited training data.

This work aims to develop a novel neural operator that inherently accommodates proper regularization to cope with the challenges in the processing of PDE data. We start from the observations that the kernel integral operator, a core module of the solving operator of PDEs, can be rewritten with orthonormal eigenfunctions. Such an expansion substan-

[1]Qing Yuan Research Institute, SEIEE, Shanghai Jiao Tong University [2]Dept. of Comp. Sci. & Tech., Tsinghua University. Correspondence to: Zhijie Deng <zhijied@sjtu.edu.cn>, Hang Su <suhangss@tsinghua.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

tially resembles the attention mechanism without softmax while incorporating the orthogonal regularization (detailed in Section 3.3). Empowered by this, we follow the notion of neural eigenfunctions (Deng et al., 2022b;a) to implement an orthogonal attention module and stack it repeatedly to construct *orthogonal neural operator (ONO)*. As shown in Figure 1, ONO is structured with two disentangled pathways. The bottom one approximates the eigenfunctions through expressive NNs, while the top one specifies the evolvement of the PDE solution. In practice, the orthogonalization operation can be implemented by cheap manipulation of the exponential moving average (EMA) of the feature covariance matrix. It is empirically proven that ONO can generalize substantially better than competitive baselines across both spatial and temporal axes. To summarize, our contributions are:

- We introduce the novel orthogonal attention, which is inherently integrated with orthogonal regularization while maintaining moderate complexity, and detail the theoretical insights.
- We introduce ONO, a neural operator built upon orthogonal attention. ONO employs two disentangled pathways for approximating the eigenfunctions and PDE solutions separately.
- We conduct comprehensive experiments on six challenging operator learning benchmarks and achieve satisfactory results: ONO reduces prediction errors by up to 30% compared to baselines and achieves 80% reduction of test error for zero-shot super-resolution on Darcy.

2. Related Work

2.1. Neural Operators

Neural operators map infinite-dimensional input and solution function spaces, allowing them to handle multiple PDE instances without retraining. Following the advent of DeepONet (Lu et al., 2019), the domain of operator learning has recently gained much attention. Specifically, DeepONet employs a branch network and a trunk network to separately encode input functions and location variables, subsequently merging them for output computation. Numerous alternative variants have been proposed from various perspectives thus far (Grady et al., 2022; Wen et al., 2022; Xiong et al., 2023). FNO (Li et al., 2020) learns the integral operator in the spectral domain to conjoin accuracy and inference speed. Geo-FNO (Li et al., 2022) employs a map connecting irregular domains and uniform latent meshes to address arbitrary geometries effectively. F-FNO (Tran et al., 2023) improves FNO by integrating dimension-separable Fourier layers and residual connections. However, FNOs are grid-based, leading to increased computational demands for both training and inference as PDE dimensions expand.

Considering the input sequence as a function evaluation within a specific domain, attention operators can be seen as learnable projection or kernel integral operators. These operators have gained substantial research attention due to their scalability and effectiveness in addressing PDEs on irregular meshes. Kovachki et al. (2021) demonstrates that the standard attention mechanism can be considered as a neural operator layer. Galerkin Transformer (Cao, 2021) proposes two self-attention operators without softmax and provides theoretical interpretations for them. HT-Net (Liu et al., 2022) proposes a hierarchical attention operator to solve multi-scale PDEs. GNOT (Hao et al., 2023) proposes a linear cross-attention block to facilitate the encoding of diverse input types. However, despite their promising potential, attention operators are susceptible to overfitting, especially when the available training data are rare.

2.2. Efficient Attention Mechanisms

The Transformer model (Vaswani et al., 2017) has gained popularity in diverse domains (Chen et al., 2018; Parmar et al., 2018; Rives et al., 2021). However, the vanilla softmax attention encounters scalability issues due to its quadratic space and time complexity. To tackle this, several methods with reduced complexity have been proposed (Child et al., 2019; Zaheer et al., 2020; Wang et al., 2020; Katharopoulos et al., 2020; Xiong et al., 2021). Concretely, Sparse Transformer (Child et al., 2019) reduces complexity by sparsifying the attention matrix. Linear Transformer (Katharopoulos et al., 2020) achieves complexity by replacing softmax with a kernel function. Nyströmformer (Xiong et al., 2021) employs the Nyström method to approximate standard attention, maintaining linear complexity.

In the context of PDE solving, Galerkin Transformer (Cao, 2021) proposes the linear Galerkin-type attention mechanism, which can be regarded as a trainable Petrov–Galerkin-type projection. OFormer (Li et al., 2023a) develops a linear cross-attention module for disentangling the output and input domains. FactFormer (Li et al., 2023b) employs axial computation in the attention operator to reduce computational costs. Compared to them, we not only introduce an attention mechanism without softmax at linear complexity but also include an inherent regularization mechanism.

3. Methodology

This section begins with an overview of the orthogonal neural operator and subsequently delves into the orthogonal attention mechanism and its theoretical foundations.

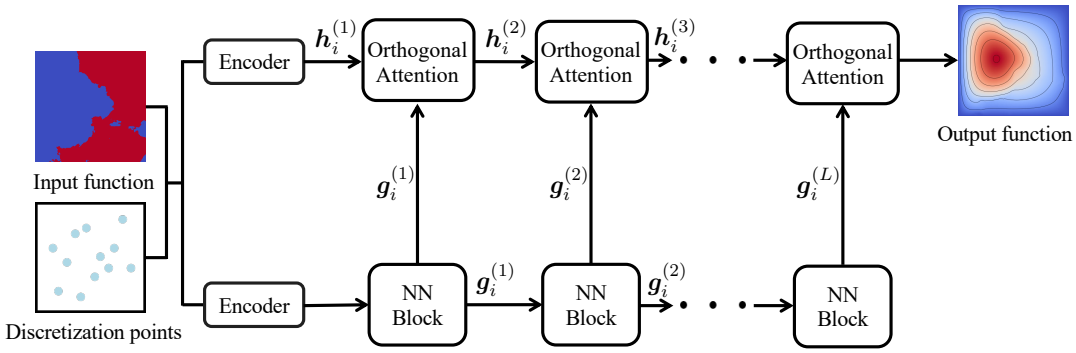

Figure 1. Model overview. There are two flows in ONO. The bottom one extracts expressive features for input data, forming an approximation to the eigenfunctions associated with the kernel integral operators for defining ONO. The top one updates the PDE solutions based on orthogonal attention, which involves linear attention and orthogonal regularization.

3.1. Problem Setup

Operator learning involves learning the mapping from the space of input functions $f : D \to \mathbb{R}^{d_f} \in \mathcal{F}$ to the space of PDE solutions $u : D \to \mathbb{R}^{d_u} \in \mathcal{U}$, where D is a bounded open set. Let $\mathcal{G} : \mathcal{F} \to \mathcal{U}$ denotes the ground-truth solution operator. Our objective is to train a $\boldsymbol{\theta}$-parameterized neural operator $\mathcal{G}_{\boldsymbol{\theta}}$ to approximate $\mathcal{G}$. The training is driven by a collection of function pairs $\{f_i, u_i\}_{i=1}^N$. Deep models routinely cannot accept an infinite-dimensional function as input or output, so we discretize f_i and u_i on mesh $\mathbf{X} := \{\boldsymbol{x}_j \in D\}_{1\le j\le M}$, yielding $\boldsymbol{f}_i := \{(\boldsymbol{x}_j, f_i(\boldsymbol{x}_j))\}_{1\le j\le M}$ and $\boldsymbol{u}_i := \{(\boldsymbol{x}_j, u_i(\boldsymbol{x}_j))\}_{1\le j\le M}$. We use $\boldsymbol{f}_{i,j}$ to denote the element in $\boldsymbol{f}_i$ that corresponds to $\boldsymbol{x}_j$. The data fitting is usually achieved by optimizing the following problem:

$$\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N} \frac{\|\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{f}_i) - \boldsymbol{u}_i\|_2}{\|\boldsymbol{u}_i\|_2}, \tag{1}$$

where the regular mean-squared error (MSE) is augmented with a normalizer $\|\boldsymbol{u}_i\|_2$ to account for variations in absolute scale across benchmarks. We refer to this error as l_2 relative error in the subsequent sections.

3.2. Orthogonal Neural Operator

Overview. Basically, an L-stage ONO takes the form of

$$\mathcal{G}_{\boldsymbol{\theta}} := \mathcal{P} \circ \mathcal{K}^{(L)} \circ \sigma \circ \mathcal{K}^{(L-1)} \circ \cdots \circ \sigma \circ \mathcal{K}^{(1)} \circ \mathcal{E}, \tag{2}$$

where $\mathcal{E}$ maps $\boldsymbol{f}_i$ to hidden states $\boldsymbol{h}_i^{(1)} \in \mathbb{R}^{M\times d}$, $\mathcal{P}$ projects the states to solutions, and σ denotes the non-linear transformation. $\mathcal{K}^{(l)}$ refer to parameterized kernel integral operators following the prior arts in neural operator (Kovachki et al., 2021), which is motivated by the link between kernel integral operator and Green's function for solving linear PDEs.

Note that $\mathcal{K}^{(l)}$ accepts hidden states $\boldsymbol{h}_i^{(l)} \in \mathbb{R}^{M\times d}$ as input instead of infinite-dimensional functions as in the traditional kernel integral operator. It should also rely on some parameterized configuration of a kernel. FNO addresses this by employing linear transformations on truncated frequency modes in the Fourier domain, albeit with potential limitations in effectively handling high-frequency information (Li et al., 2020). Instead, we advocate directly parameterizing the kernel in the original space with the help of neural eigenfunctions (Deng et al., 2022b;a). Specifically, we leverage an additional NN to extract hierarchical features from $\boldsymbol{f}_i$, which, after orthogonalization and normalization, suffice to define $\mathcal{K}^{(l)}$. The orthogonalization serves as a regularization to enhance the model generalization ability.

We outline the overview of ONO in Figure 1, where the two-flow structure is clearly displayed. We pack the orthonormalization step and eigenfunctions-based kernel integral into a module named *orthogonal attention*. The decoupled architecture offers significant flexibility in specifying the NN blocks within the bottom flow.

Encoder. The encoder is multi-layer perceptrons (MLPs) that accept $\boldsymbol{f}_i$ as input for dimension lifting. Features at every position $\boldsymbol{x}_j$ are extracted separately.

NN Block. In the bottom flow, the NN blocks are responsible for extracting features, which subsequently specify the kernel integral operators in the orthogonal attention modules. We can leverage any existing architecture here but focus on transformers due to their great expressiveness. In detail, we formulate the NN block as follow:

$$\begin{aligned} \tilde{\boldsymbol{g}}_i^{(l)} &= \boldsymbol{g}_i^{(l)} + \mathrm{Attn}(\mathrm{LN}(\boldsymbol{g}_i^{(l)})), \\ \boldsymbol{g}_i^{(l+1)} &= \tilde{\boldsymbol{g}}_i^{(l)} + \mathrm{FFN}(\mathrm{LN}(\tilde{\boldsymbol{g}}_i^{(l)})), \end{aligned} \tag{3}$$

where $\boldsymbol{g}_i^{(l)} \in \mathbb{R}^{M\times d'}$ denotes the output of l-th NN block for the data $\boldsymbol{f}_i$. $\mathrm{Attn}(\cdot)$ represents a self-attention module applied over the M positions. $\mathrm{LN}(\cdot)$ indicates layer normalization (Ba et al., 2016). $\mathrm{FFN}(\cdot)$ refers to a two-layer feed forward network. Here, we can freely choose well-studied

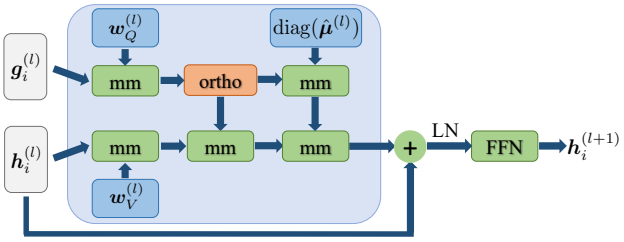

Figure 2. Orthogonal attention: the module incorporates matrix multiplications ("mm") and an orthogonalization process ("ortho"). The output of the NN block, denoted as $\boldsymbol{g}_i^{(l)}$, and the hidden state of the input function, represented as $\boldsymbol{h}_i^{(l)}$, undergo processing as shown in Equation 5. Following this, the module includes a residual connection, layer normalization, and a two-layer FFN.

self-attention mechanisms, e.g., standard attention (Vaswani et al., 2017) and other variants that enjoy higher efficiency to suit specific requirements.

Orthogonal Attention. We introduce the orthogonal attention module with orthogonal regularization to remediate the potential overfitting in the context of operator learning. This module characterizes the evolution of PDE solutions. It transforms the deep features from the NN blocks to orthogonal eigenmaps, based on which the kernel integral operators are constructed and the hidden states of PDE solutions are updated. Concretely, we first project the NN features $\boldsymbol{g}_i^{(l)} \in \mathbb{R}^{M\times d'}$ to:

$$\hat{\boldsymbol{\psi}}_i^{(l)} = \mathrm{ort}(\hat{\boldsymbol{g}}_i^{(l)}) = \mathrm{ort}(\boldsymbol{g}_i^{(l)}\boldsymbol{w}_Q^{(l)}) \in \mathbb{R}^{M\times k}, \tag{4}$$

where $\boldsymbol{w}_Q^{(l)} \in \mathbb{R}^{d'\times k}$ is a trainable weight. $\mathrm{ort}(\cdot)$ is the orthonormalization operation which renders each column of $\hat{\boldsymbol{\psi}}_i^{(l)}$ correspond to the evaluation of a specific neural eigenfunction on $\boldsymbol{f}_i$.

Given these, the orthogonal attention update the hidden states $\boldsymbol{h}_i^{(l)}$ of PDE solutions via:

$$\tilde{\boldsymbol{h}}_i^{(l+1)} = \hat{\boldsymbol{\psi}}_i^{(l)}\mathrm{diag}(\hat{\boldsymbol{\mu}}^{(l)})[\hat{\boldsymbol{\psi}}_i^{(l)\top}(\boldsymbol{h}_i^{(l)}\boldsymbol{w}_V^{(l)})], \tag{5}$$

where $\boldsymbol{w}_V^{(l)} \in \mathbb{R}^{d\times d}$ is a trainable linear weight to refine the hidden states, and $\hat{\boldsymbol{\mu}}^{(l)} \in \mathbb{R}_+^k$ denote positive eigenvalues associated with the induced kernel and are trainable in practice. This update rule is closely related to Mercer's theorem, as will be detailed in Section 3.3.

The non-linear transformation σ is implemented following the structure of the traditional attention mechanism, which involves residual connections (He et al., 2016) and FFN:

$$\boldsymbol{h}_i^{(l+1)} = \mathrm{FFN}(\mathrm{LN}(\tilde{\boldsymbol{h}}_i^{(l+1)} + \boldsymbol{h}_i^{(l)})). \tag{6}$$

The FFN in the final orthogonal attention serves as $\mathcal{P}$ to map hidden states to PDE solutions.

Implementation of $\mathrm{ort}(\cdot)$. As mentioned, we leverage $\mathrm{ort}(\cdot)$ to make $\hat{\boldsymbol{g}}_i^{(l)}$ follow the structure of the outputs of eigenfunctions. We highlight that the orthonormalization lies in the function space, i.e., among the output dimensions of the function $\hat{g}^{(l)} : \boldsymbol{f}_{i,j} \mapsto \hat{\boldsymbol{g}}_{i,j}^{(l)} \in \mathbb{R}^k$ instead of the column vectors. Thereby, we should not orthonormalize matrix $\hat{\boldsymbol{g}}_i^{(l)}$ over its columns but manipulate $\hat{g}^{(l)}$.

To achieve this, we first estimate the covariance over the output dimensions of $\hat{g}^{(l)}$, which can be approximated by Monte Carlo (MC) estimation:

$$\begin{aligned}\mathbf{C}^{(l)} &\approx \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}[\hat{g}^{(l)}(\boldsymbol{f}_{i,j})^\top \hat{g}^{(l)}(\boldsymbol{f}_{i,j})] \\ &= \frac{1}{NM}\sum_{i=1}^{N}[\hat{\boldsymbol{g}}_i^{(l)\top}\hat{\boldsymbol{g}}_i^{(l)}].\end{aligned} \tag{7}$$

Then, we orthonormalize $\hat{g}^{(l)}$ by right multiplying the matrix $\mathbf{L}^{(l)-\top}$, where $\mathbf{L}^{(l)}$ is the lower-triangular matrix arising from the Cholesky decomposition of $\mathbf{C}^{(l)}$, i.e., $\mathbf{C}^{(l)} = \mathbf{L}^{(l)}\mathbf{L}^{(l)\top}$. In the vector formula, there is

$$\hat{\boldsymbol{\psi}}_i^{(l)} := \hat{\boldsymbol{g}}_i^{(l)}\mathbf{L}^{(l)-\top}. \tag{8}$$

The covariance of the functions producing $\hat{\boldsymbol{\psi}}_i^{(l)}$ can be approximately estimated:

$$\begin{aligned}&\frac{1}{NM}\sum_{i=1}^{N}\left[\left(\hat{\boldsymbol{g}}_i^{(l)}\mathbf{L}^{(l)-\top}\right)^\top \hat{\boldsymbol{g}}_i^{(l)}\mathbf{L}^{(l)-\top}\right] \\ &\qquad = \mathbf{L}^{(l)-1}\mathbf{C}^{(l)}\mathbf{L}^{(l)-\top} = \mathbf{I},\end{aligned} \tag{9}$$

which conforms that these functions can be regarded as orthonormal eigenfunctions that implicitly define a kernel.

However, in practice, the model parameters evolve repeatedly, we cannot trivially estimate $\mathbf{C}^{(l)}$, which involves the whole training set, at a low cost per training iteration. Instead, we propose to approximately estimate $\mathbf{C}^{(l)}$ via the exponential moving average trick—similar to the update rule in batch normalization (Ioffe & Szegedy, 2015), we maintain a buffer tensor $\mathbf{C}^{(l)}$ and update it with training mini-batches. We reuse the recorded training statistics to ensure the stability of inference.

The aforementioned process involves a cubic complexity with respect to k due to the Cholesky decomposition. However, it is worth noting that empirically, k is significantly smaller than the number of measurement points M. Consequently, the overall complexity of the proposed orthogonal attention mechanism remains moderate (see the empirical results in Table 2).

3.3. Theoretical Insights

This section provides the theoretical insights behind orthogonal attention. We abuse notations when there is no mis-

leading. Consider a kernel integral operator $\mathcal{K}$ as follow:

$$(\mathcal{K}h)(\boldsymbol{x}) := \int_D \kappa(\boldsymbol{x}, \boldsymbol{x}')h(\boldsymbol{x}')\, d\boldsymbol{x}', \quad \forall \boldsymbol{x} \in D, \tag{10}$$

where κ is a positive semi-definite kernel and h is the input function. Given ψ_i as the eigenfunction of $\mathcal{K}$ corresponding to the i-th largest eigenvalue μ_i, we have:

$$\begin{aligned} \int_D \kappa(\boldsymbol{x}, \boldsymbol{x}')\psi_i(\boldsymbol{x}')\, d\boldsymbol{x}' = \mu_i\psi_i(\boldsymbol{x}), \quad \forall i \geq 1, \forall \boldsymbol{x} \in D \\ \langle \psi_i, \psi_j \rangle = \mathbb{1}[i = j], \quad \forall i, j \geq 1, \end{aligned} \tag{11}$$

where $\langle a, b \rangle := \int a(\boldsymbol{x})b(\boldsymbol{x})\, d\boldsymbol{x}$ denotes the inner product in D. By Mercer's theorem, there is:

$$\begin{aligned} (\mathcal{K}h)(\boldsymbol{x}) &= \int_D \kappa(\boldsymbol{x}, \boldsymbol{x}')h(\boldsymbol{x}')\, d\boldsymbol{x}' \\ &= \int \sum_{i\geq 1} \mu_i\psi_i(\boldsymbol{x})\psi_i(\boldsymbol{x}')h(\boldsymbol{x}')\, d\boldsymbol{x}' \\ &= \sum_{i\geq 1} \mu_i \langle \psi_i, h \rangle \psi_i(\boldsymbol{x}). \end{aligned} \tag{12}$$

Although we cannot trivially estimate the eigenfunctions ψ_i in the absence of κ's expression, Equation 12 offers us new insights on how to parameterize a kernel integral operator. In particular, we can truncate the summation in Equation 12 and introduce a parametric model $\hat{\psi}(\cdot) : D \to \mathbb{R}^k$ with orthogonal outputs and build a neural operator $\hat{\mathcal{K}}$ with the following definition:

$$(\hat{\mathcal{K}}h)(\boldsymbol{x}) := \sum_{i=1}^{k} \langle \hat{\psi}_i, h \rangle \hat{\psi}_i(\boldsymbol{x}). \tag{13}$$

We demonstrate the convergence of $\hat{\mathcal{K}}$ towards the ground truth $\mathcal{K}$ under MSE loss in the Appendix A. In practice, we first consider $\mathbf{X} := \{\boldsymbol{x}_j\}_{1\leq j\leq M}$ and $\mathbf{Y} := \{\boldsymbol{x}_j\}_{1\leq j\leq M'}$ as two sets of measurement points to discretize the input and output functions. We denote $\hat{\boldsymbol{\psi}} \in \mathbb{R}^{M\times k}$ and $\hat{\boldsymbol{\psi}}' \in \mathbb{R}^{M'\times k}$ as the evaluation of the model $\hat{\psi}$ on $\mathbf{X}$ and $\mathbf{Y}$ respectively. Let $\boldsymbol{h} \in \mathbb{R}^M$ represent the evaluation of h on $\mathbf{X}$. There is:

$$(\hat{\mathcal{K}}h)(\mathbf{Y}) \approx \sum_{i=1}^{k} [\hat{\psi}_i(\mathbf{X})^\top h(\mathbf{X})]\hat{\psi}_i(\mathbf{Y}) = \hat{\boldsymbol{\psi}}'\hat{\boldsymbol{\psi}}^\top \boldsymbol{h}. \tag{14}$$

Comparing Equation 12 and Equation 13, we can see that the scaling factors μ_i are omitted, which may undermine the model flexibility in practice. To address this, we introduce a learnable vector $\hat{\boldsymbol{\mu}} \in \mathbb{R}_+^k$ to Equation 14, resulting in:

$$(\hat{\mathcal{K}}h)(\mathbf{Y}) \approx \hat{\boldsymbol{\psi}}'\mathrm{diag}(\hat{\boldsymbol{\mu}})\hat{\boldsymbol{\psi}}^\top \boldsymbol{h}. \tag{15}$$

As shown, there is an attention structure—$\hat{\boldsymbol{\psi}}'\mathrm{diag}(\hat{\boldsymbol{\mu}})\hat{\boldsymbol{\psi}}^\top$ corresponds to the attention matrix that defines how the output function evaluations attend to the input. Besides, the orthonormalization regularization arises from the nature of eigenfunctions, benefitting to alleviate overfitting and boost generalization. When $\mathbf{X} = \mathbf{Y}$, the above attention structure is similar to regular self-attention mechamism with a symmetric attention matrix. Otherwise, it boils down to a cross-attention, which enables our approach to query output functions at arbitrary locations independent of the inputs. Find more details regarding this in Appendix A.

4. Experiments

We conduct extensive experiments on diverse and challenging benchmarks across various domains to showcase the effectiveness of our method.

Benchmarks. We first evaluate our model's performance on Darcy and NS2d (Li et al., 2020) benchmarks to evaluate its capability on regular grids. Subsequently, we extend our experiments to benchmarks with irregular geometries, including Airfoil, Plasticity, and Pipe, which are represented in structured meshes, as well as Elasticity, presented in point clouds (Li et al., 2022).

Baselines. We compare our model with several baseline models, including the well-recognized FNO (Li et al., 2020) and its variants Geo-FNO (Li et al., 2022), F-FNO (Tran et al., 2023), and U-FNO (Wen et al., 2022). Furthermore, we consider other models such as Galerkin Transformer (Cao, 2021), LSM (Wu et al., 2023), and GNOT (Hao et al., 2023). It's worth noting that LSM and GNOT are the latest state-of-the-art (SOTA) neural operators.

Implementation details. We use the l_2 relative error in Equation 1 as the training loss and evaluation metric. We train all models for 500 epochs. Our training process employs the AdamW optimizer (Loshchilov & Hutter, 2018) and the OneCycleLr scheduler (Smith & Topin, 2019). We initialize the learning rate at 10^{-3} and explore batch sizes within the range of $\{2, 4, 8, 16\}$. The model's width is set to 128, while the orthogonalization process employs dimension d as either 8 or 16. Unless specified otherwise, we choose either the Linear Transformer block from (Katharopoulos et al., 2020) or the Nyström Transformer block from (Xiong et al., 2021) as the NN block in our model. Further implementation details of the baselines are provided in Appendix B. We also provide a run-time comparison of different neural operators in Table 2. Our experiments are conducted on a single NVIDIA RTX 3090 GPU.

4.1. Main Results

Table 1 reports the results. Remarkably, our model achieves SOTA performance on three of these benchmarks, reducing the average prediction error by 13%. Specifically, it reduces the error by 31% and 10% on Pipe and Airfoil, respectively.

Table 1. The main results on six benchmarks compared with seven baselines. Lower scores indicate superior performance, and the best results are highlighted in bold. "*" means that the results of the method are reproduced by ourselves. "-" means that the baseline cannot handle this benchmark.

MODEL	NS2d	Airfoil	Pipe	Darcy	Elasticity	Plasticity
FNO	0.1556	-	-	0.0108	-	-
Galerkin	0.1401	-	-	0.0084	-	-
Geo-FNO	0.1556	0.0138	0.0067	0.0108	0.0229	0.0074
U-FNO*	0.2182	0.0137	0.0050	0.0266	0.0226	**0.0028**
F-FNO*	0.1213	0.0079	0.0063	0.0318	0.0316	0.0048
LSM*	0.1693	0.0062	0.0049	**0.0069**	0.0225	0.0035
GNOT	0.1380	0.0076	-	0.0105	**0.0086**	-
ONO	**0.1195**	**0.0056**	**0.0034**	0.0072	0.0118	0.0048

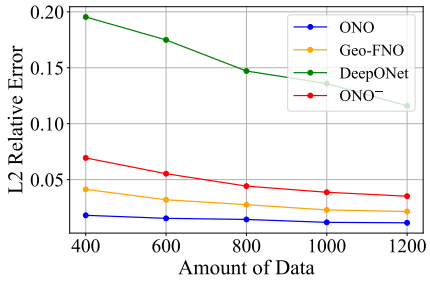

Figure 3. Comparison on the l_2 relative error for different training data amounts on Elasticity.

In the case of NS2d, which involves temporal predictions, our model surpasses all baselines. We attribute it to the temporal generalization enabled by our orthogonal attention. We conjecture that the efficacy of orthogonal regularization contributes to our model's excellent performance in these three benchmarks by mitigating overfitting the limited training data. These three benchmarks encompass both regular and irregular geometries, demonstrating the versatility of our model across various geometric settings.

Our model achieves the second-lowest prediction error on Darcy and Elasticity benchmarks, albeit with a slight margin compared to the SOTA baselines. We notice that our model and the other attention operator (GNOT) demonstrate a significant reduction in error when compared to other operators that utilize a learnable mapping to convert the irregular geometries into or back from uniform meshes. This mapping process can potentially introduce errors. However, attention operators naturally handle irregular meshes for sequence input without requiring mapping, leading to superior performance. Our model also exhibits competitive performance on plasticity, involving the mapping of a shape vector to the complex mesh grid with a dimension of deformation. These results highlight the versatility and effectiveness of our model as a framework for operator learning.

Table 2. Runtime comparison. "LT" refers to using the Linear Transformer block for specifying the NN block. All models use a batch size of 8. FNO, LSM, and ONO are fixed as 4 layers. The width of ONO d is set to 128, and the number of eigenfunctions k is 16.

MODEL	FNO	Galerkin	LSM	ONO (LT)
Runtime (s)	3.81	9.88	9.05	7.87

Training on limited data. We investigate the influence of limited training data using the Elasticity dataset and make comparisons with FNO and DeepONet, two widely recognized neural operators. To demonstrate the effectiveness of the orthogonalization process, we additionally utilize ONO without the orthogonalization, referred to as ONO$^-$.

The results are shown in Figure 3. ONO outperforms the baselines with different training data amounts, followed by Geo-FNO, ONO$^-$, and DeepONet. Each of the neural operators demonstrates a degradation in performance as the training data amount decreases. The reduction in training data from 1200 to 400 results in significant increases in prediction error for ONO$^-$ (97.1%, $0.0352 \rightarrow 0.0694$) and Geo-FNO ($92\%$, $0.0215 \rightarrow 0.0413$). DeepONet demonstrates a 68% increase ($0.0215 \rightarrow 0.0413$), while ONO demonstrates the lowest increase of 58% ($0.0114 \rightarrow 0.0181$). Notably, ONO$^-$ exhibits considerable performance deterioration when trained on reduced training data compared to ONO. The superior generalization ability of ONO, relative to the baselines, highlights the effectiveness of the orthogonalization operation for deep learning-based operator learning.

Runtime comparison. Table 2 provides a comparison on the runtime of different neural operators, revealing that ONO with a Linear Transformer block has a comparable computational cost to the linear Galerkin Transformer.

Table 3. Comparison on the l_2 relative error for Zero-shot super-resolution on darcy benchmark. s denotes the resolution of the evaluation data. The models are trained on data of 43×43 resolution (s=43).

MODEL	$s = 61$	$s = 85$	$s = 141$	$s = 211$	$s = 421$
FNO	0.1164	0.1797	0.2679	0.3160	0.3631
Ours	**0.0204**	**0.0259**	**0.0315**	**0.0349**	**0.0386**

Table 4. Comparison on the l_2 relative error for seen and unseen timesteps on NS2d.

MODEL	Seen	Unseen
FNO	0.0982	0.2446
Ours	**0.0889**	**0.2143**

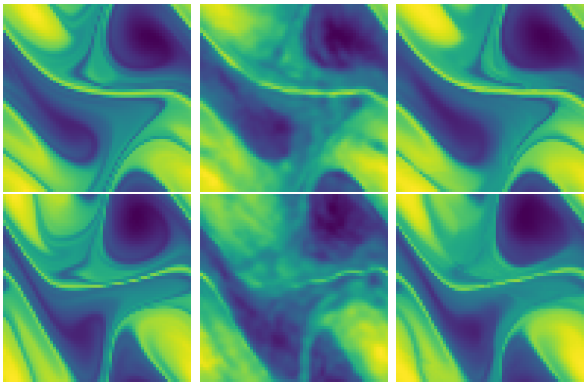

Figure 4. The two rows refer to the results of models, trained to predict timesteps 11-18, for timesteps 19 and 20 on NS2d. From left to right: ground truth, prediction of FNO, and that of ONO.

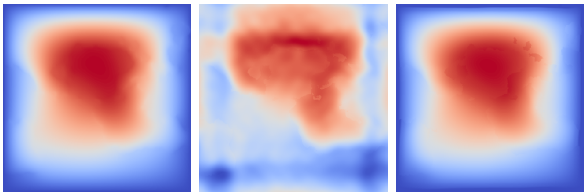

Figure 5. Zero-shot super-resolution results on Darcy. Models are trained on 43×43 data and evaluated on 421×421. From left to right: ground truth, prediction of FNO, and that of ONO.

4.2. Generalization Experiments

We conduct experiments on the generalization performance in both the spatial and temporal axes. First, a zero-shot super-resolution experiment is conducted on Darcy. The model is trained on 43×43 resolution data and evaluated on resolutions up to nearly ten times that size (421×421). Subsequently, we train the model to predict timesteps 11-18 and evaluate it on two subsequent intervals: timesteps 11-18, denoted as "Seen", and timesteps 19-20, denoted as "Unseen". We choose the FNO (Li et al., 2020) as the baseline due to its well-acknowledged mesh-invariant property and is use of the orthogonal Fourier basis functions, which may potentially offer regularization benefits.

The results are shown in Table 3 and Table 4. On Darcy, the prediction error of FNO increases dramatically as the evaluation resolution grows. In contrast, our model exhibits a much slower increase in error and maintains a low prediction error even with excessively enlarged resolution, notably reducing the prediction error by 89% compared to FNO on the 421×421 resolution. On NS2d, Our model outperforms in both time intervals, reducing the prediction error by 9% and 12%. We further visualize some generalization results in these two scenarios in Figure 5 and Figure 4. The results are consistent with the reported values. These results demonstrate that our model exhibits remarkable generalization capabilities in both temporal and spatial domains. Acquiring high-resolution training data can be computationally expensive. Our model's mesh-invariant property enables effective high-resolution performance after being trained on low-resolution data, potentially resulting in significant computational cost savings.

Table 5. Comparison on the l_2 relative error for different NN blocks on Airfoil, Elasticity, and Pipe benchmarks.

DESIGNS	Airfoil	Elasticity	Pipe
Linear	0.0079	0.0137	0.0060
Nystrom	**0.0056**	**0.0118**	**0.0034**
Galerkin	0.0122	0.0133	0.0075

4.3. Ablation experiments

To assess the effectiveness of various components in ONO, we conduct a comprehensive ablation study on three benchmarks: Airfoil, Elasticity, and Pipe.

Influence of NN Block. To show the compatibility of our model, we conduct experiments with different NN blocks. We choose the Galerkin Transformer block in operator learning (Cao, 2021) and two linear transformer blocks in other domains, including the Linear Transformer block in (Xiong et al., 2021) and the Nyström Transformer block in (Katharopoulos et al., 2020).

Table 5 showcases the results. The Nyström Transformer block performs better on all three benchmarks and reduces the error up to 43% on Pipe. Linear Transformer block exhibits superior performance on Airfoil and Pipe compared to the Galerkin Transformer block while demonstrating similar performance on Elasticity. We notice that the Linear Transformer block and Galerkin Transformer block are both

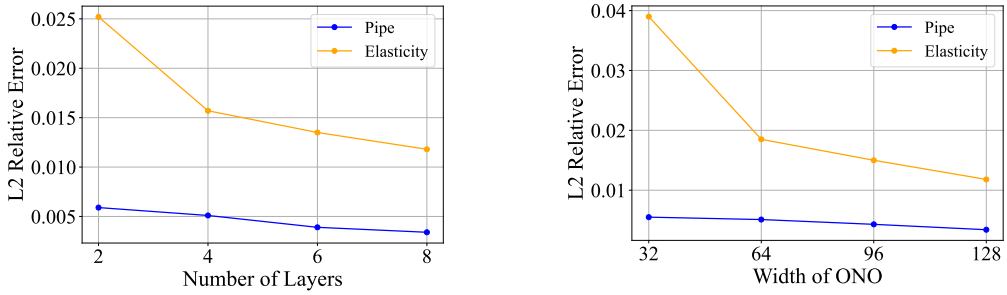

Figure 6. l_2 relative error varies w.r.t. the number of layers (Left) and width (right) of ONO on Pipe and Elasticity benchmarks.

Table 6. Comparison on the l_2 relative error for orthogonalization and normalization techniques on Airfoil, Elasticity, and Pipe.

DESIGNS	Airfoil	Elasticity	Pipe
BN	0.0808	0.0149	0.2151
LN	0.0288	0.0387	0.0056
Ortho	**0.0056**	**0.0118**	**0.0034**

kernel-based methods transformer methods. The Nyström attention uses a downsampling approach to approximate the softmax attention, which aids in capturing positional relationships and contributes to the feature extraction. However, all the variants consistently exhibit competitive performance, showcasing the flexibility and robustness of our model.

Influence of Orthogonalization. To further investigate the impact of the orthogonalization process, we carry out a series of experiments on three benchmarks. "BN" and "LN" denote the batch normalization (Ioffe & Szegedy, 2015) and the layer normalization (Ba et al., 2016), while "Ortho" signifies the orthogonalization process in the attention module. It's worth noting that the attention mechanism coupled with layer normalization assumes a structure resembling Fourier-type attention (Cao, 2021).

As shown in Table 6, our orthogonal attention consistently outperforms other attention mechanisms across all benchmarks, resulting in a remarkable reduction of prediction error, up to 81% on Airfoil and 39% on Pipe. We conjecture that the orthogonalization may benefit model training through feature scaling. Additionally, the inherent linear independence among orthogonal eigenfunctions aids the model in effectively distinguishing between various features, contributing to its superior performance compared to conventional normalizations.

4.4. Scaling Experiments

Our model's architecture offers scalability, allowing adjustments to both its depth and width for enhanced performance or reduced computational costs. We conduct scaling experiments to examine how the prediction error changes with the number of layers and the width.

Table 7. Comparison on the l_2 relative error for ONO with different depths on Elasticity and Plasticity benchmarks.

Model	Elasticity	Plasticity
ONO-8	0.0118	0.0048
ONO-30	**0.0047**	**0.0013**

Figure 6 shows the results. The left one depicts the change in prediction error with an increasing number of layers, while the right one shows how the error responds to a growth in the width of ONO. It is evident that error reduction correlates positively with both the number of layers and width. Nevertheless, diminishing returns become apparent when exceeding four layers or a width of 64 on Elasticity. We recommend employing a model configuration consisting of four layers and a width of 64 due to its favorable balance between performance and computational cost.

To further assess the scalability of our model, we increase the number of layers to 30 and the learnable parameters to 10 million while keeping the width at 128. We compare it to the 8-layer model, which has approximately 1 million parameters. The results are in Table 7. We denote the models as "ONO-30" and "ONO-8" respectively. The prediction of ONO-30 exhibits a remarkable decrease in both benchmarks, achieving reductions of 37% and 76%. The results demonstrate the potential of ONO as a large pre-trained neural operator.

5. Conclusion

This paper aims to address the performance decline stemming from the limited training data from classical PDE solvers and the complexity of deep models. Our main contribution is the introduction of regularization mechanisms for neural operators, which effectively enhance generalization performance with reduced training data. We propose an attention mechanism with orthogonalization regularization based on the kernel integral rewritten by orthonormal eigenfunctions. We further present a neural operator called ONO,

built upon this attention mechanism. Extensive experiments demonstrate the superiority of our approach compared to baselines. The study aims to mitigate the challenges associated with the small data regime and enhance the robustness of large PDE-solving models.

Acknowledgements

This work was supported by NSF of China (No. 62306176), Natural Science Foundation of Shanghai (No. 23ZR1428700), Key R&D Program of Shandong Province, China (2023CXGC010112), and CCF-Baichuan-Ebtech Foundation Model Fund.

Impact Statement

This work introduces a neural operator designed to effectively solve PDEs, which is of significance in scientific and engineering domains. As a foundational machine learning research, the immediate negative consequences are not evident, and the risk of misuse is currently low.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Cao, S. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, pp. 24924–24940, 2021.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., et al. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, 2018.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Ciarlet, P. G. *The finite element method for elliptic problems*. SIAM, 2002.

Courant, R., Friedrichs, K., and Lewy, H. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

Deng, Z., Shi, J., Zhang, H., Cui, P., Lu, C., and Zhu, J. Neural eigenfunctions are structured representation learners. *arXiv preprint arXiv:2210.12637*, 2022a.

Deng, Z., Shi, J., and Zhu, J. Neuralef: Deconstructing kernels by deep neural networks. In *International Conference on Machine Learning*, pp. 4976–4992. PMLR, 2022b.

Fonseca, A. H. d. O., Zappala, E., Caro, J. O., and van Dijk, D. Continuous spatiotemporal transformers. *arXiv preprint arXiv:2301.13338*, 2023.

Grady, T. J., Khan, R., Louboutin, M., Yin, Z., Witte, P. A., Chandra, R., Hewett, R. J., and Herrmann, F. J. Towards large-scale learned solvers for parametric pdes with model-parallel fourier neural operators. *arXiv e-prints*, pp. arXiv–2204, 2022.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, pp. 24048–24062, 2021.

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., Anandkumar, A., et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023a.

Li, Z., Shu, D., and Farimani, A. B. Scalable transformer for pde surrogate modeling. *arXiv preprint arXiv:2305.17560*, 2023b.

Liu, X., Xu, B., and Zhang, L. Ht-net: Hierarchical transformer based operator learning model for multiscale pdes. *arXiv preprint arXiv:2210.10890*, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Ovadia, O., Kahana, A., Stinis, P., Turkel, E., and Karniadakis, G. E. Vito: Vision transformer-operator. *arXiv preprint arXiv:2303.08891*, 2023.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. Image transformer. In *International conference on machine learning*, pp. 4055–4064, 2018.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, pp. e2016239118, 2021.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, pp. 369–386. SPIE, 2019.

Thomas, J. W. *Numerical partial differential equations: finite difference methods*. Springer Science & Business Media, 2013.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, pp. 104180, 2022.

Wu, H., Hu, T., Luo, H., Wang, J., and Long, M. Solving high-dimensional pdes with latent spectral models. In *International Conference on Machine Learning*, 2023.

Xiong, W., Huang, X., Zhang, Z., Deng, R., Sun, P., and Tian, Y. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *arXiv preprint arXiv:2301.10022*, 2023.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 14138–14148, 2021.

Zachmanoglou, E. C. and Thoe, D. W. *Introduction to partial differential equations with applications*. Courier Corporation, 1986.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, pp. 17283–17297, 2020.

Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. *The finite element method: its basis and fundamentals*. Elsevier, 2005.

A. Theoretical supplement

Proof of the convergence of $\hat{\mathcal{K}}$. To push $\hat{\mathcal{K}}$ in Equation 13 towards to unknown ground truth $\mathcal{K}$, we solve the following minimization problem:

$$\begin{aligned} &\min_{\hat{\psi}} \ell,\ l := \mathbb{E}_{h\sim p(h)} \left(\int \Big[\sum_{i=1}^{k} \langle \hat{\psi}_i, h\rangle \hat{\psi}_i(\boldsymbol{x}) - (\mathcal{K}h)(\boldsymbol{x}) \Big]^2 d\boldsymbol{x} \right) \\ &\quad s.t. : \langle \hat{\psi}_i, \hat{\psi}_j \rangle = \mathbb{1}[i=j], \quad \forall i,j \in [1,k], \end{aligned} \tag{16}$$

We next prove that the above learning objective closely connects to eigenfunction recovery. To show that, we first reformulate the above loss:

$$\begin{aligned} \ell &= \mathbb{E}_{h\sim p(h)} \left(\sum_{i=1}^{k}\sum_{i'=1}^{k} \langle \hat{\psi}_i, h\rangle \langle \hat{\psi}_{i'}, h\rangle \langle \hat{\psi}_i, \hat{\psi}_{i'}\rangle - 2\sum_{i=1}^{k} \langle \hat{\psi}_i, h\rangle \langle \hat{\psi}_i, \mathcal{K}h\rangle + C \right) \\ &= \mathbb{E}_{h\sim p(h)} \left(\sum_{i=1}^{k}\sum_{i'=1}^{k} \langle \hat{\psi}_i, h\rangle \langle \hat{\psi}_{i'}, h\rangle \mathbb{1}[i=i'] - 2\sum_{i=1}^{k} \langle \hat{\psi}_i, h\rangle \langle \hat{\psi}_i, \mathcal{K}h\rangle + C \right) \\ &= \mathbb{E}_{h\sim p(h)} \left(\sum_{i=1}^{k} \langle \hat{\psi}_i, h\rangle^2 - 2\sum_{i=1}^{k} \langle \hat{\psi}_i, h\rangle \langle \hat{\psi}_i, \mathcal{K}h\rangle + C \right) \\ &= \mathbb{E}_{h\sim p(h)} \left(\sum_{i=1}^{k} \langle \hat{\psi}_i, h\rangle^2 - 2\sum_{i=1}^{k} \langle \hat{\psi}_i, h\rangle \left[\sum_{j\geq 1} \mu_j \langle \psi_j, h\rangle \langle \hat{\psi}_i, \psi_j\rangle \right] + C \right) \end{aligned} \tag{17}$$

where C denotes a constant agnostic to $\hat{\psi}$.

Represent $\hat{\psi}_i$ with its coordinates $\boldsymbol{a}_i := [\boldsymbol{a}_{i,1}, \ldots, \boldsymbol{a}_{i,j}, \ldots]$ in the space spanned by $\{\psi_j\}_{j\geq 1}$, i.e., $\hat{\psi}_i = \sum_{j\geq 1} \boldsymbol{a}_{i,j}\psi_j$. Thereby, $\langle \hat{\psi}_i, \hat{\psi}_{i'}\rangle = \boldsymbol{a}_i^\top \boldsymbol{a}_{i'} := \sum_{j\geq 1} \boldsymbol{a}_{i,j}\boldsymbol{a}_{i',j}$ and $\boldsymbol{a}_i^\top \boldsymbol{a}_{i'} = \mathbb{1}[i=i']$. Likewise, we represent h with coordinates $\boldsymbol{a}_h := [\boldsymbol{a}_{h,1}, \ldots, \boldsymbol{a}_{h,j}, \ldots]$. Let $\boldsymbol{\mu} := [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots]$ and $\boldsymbol{a}_h := \mathbb{E}_{h\sim p(h)} \boldsymbol{a}_h \boldsymbol{a}_h^\top$. There is (we omit the above constant)

$$\begin{aligned} \ell &= \mathbb{E}_{h\sim p(h)} \left(\sum_{i=1}^{k} (\boldsymbol{a}_i^\top \boldsymbol{a}_h)^2 - 2\sum_{i=1}^{k} (\boldsymbol{a}_i^\top \mathrm{a}_h) \left[\sum_{h\geq 1} \mu_j \boldsymbol{a}_{h,j}\boldsymbol{a}_{i,h} \right] \right) \\ &= \mathbb{E}_{h\sim p(h)} \left(\sum_{i=1}^{k} (\boldsymbol{a}_i^\top \boldsymbol{a}_h)^2 - 2\sum_{i=1}^{k} (\boldsymbol{a}_i^\top \boldsymbol{a}_h)(\boldsymbol{a}_i^\top \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_h) \right) \\ &= \mathbb{E}_{h\sim p(h)} \left(\sum_{i=1}^{k} \boldsymbol{a}_i^\top (\boldsymbol{a}_h \boldsymbol{a}_h^\top)\boldsymbol{a}_i - 2\sum_{i=1}^{k} \boldsymbol{a}_i^\top (\boldsymbol{a}_h \boldsymbol{a}_h^\top)\mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_i \right) \\ &= \sum_{i=1}^{k} \boldsymbol{a}_i^\top \left[\mathbb{E}_{h\sim p(h)} \boldsymbol{a}_h \boldsymbol{a}_h^\top \right] \boldsymbol{a}_i - 2\sum_{i=1}^{k} \boldsymbol{a}_i^\top \left[\mathbb{E}_{h\sim p(h)} \boldsymbol{a}_h \boldsymbol{a}_h^\top \right] \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_i \\ &= \sum_{i=1}^{k} \left[\boldsymbol{a}_i^\top \boldsymbol{a}_h \boldsymbol{a}_i - 2\boldsymbol{a}_i^\top \boldsymbol{a}_h \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_i \right] \\ &= \sum_{i=1}^{k} \left[\boldsymbol{a}_i^\top \boldsymbol{a}_h \boldsymbol{a}_i - \boldsymbol{a}_i^\top \boldsymbol{a}_h \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_i - \boldsymbol{a}_i^\top \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_h \boldsymbol{a}_i \right] \\ &= \sum_{i=1}^{k} \boldsymbol{a}_i^\top \left[\boldsymbol{a}_h - \boldsymbol{a}_h \mathrm{diag}(\boldsymbol{\mu}) - \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_h \right] \boldsymbol{a}_i. \end{aligned} \tag{18}$$

$\boldsymbol{a}_h$ and $\boldsymbol{a}_h - \boldsymbol{a}_h \mathrm{diag}(\boldsymbol{\mu}) - \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_h$ are both symmetric positive semidefinite matrices with infinity rows and columns. Considering the orthonoramlity constraint on $\{\boldsymbol{a}_i\}_{i=1}^k$, minimizing ℓ will push $\{\boldsymbol{a}_i\}_{i=1}^k$ towards the k eigenvectors with smallest eigenvalues of $\boldsymbol{a}_h - \boldsymbol{a}_h \mathrm{diag}(\boldsymbol{\mu}) - \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_h$. In the case that $\boldsymbol{a}_h$ equals to the identity matrix, i.e.,

$\mathbb{E}_{h\sim p(h)}\langle h,\psi_i\rangle\langle h,\psi_j\rangle = \mathbb{1}[i=j]$, there is :

$$\ell = \sum_{i=1}^{k} \boldsymbol{a}_i^\top \left[\boldsymbol{a}_h - \boldsymbol{a}_h \mathrm{diag}(\boldsymbol{\mu}) - \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_h\right] \boldsymbol{a}_i = k - 2\sum_{i=1}^{k} \boldsymbol{a}_i^\top \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{a}_i. \tag{19}$$

Then $\{\boldsymbol{a}_i\}_{i=1}^k$ will converge to the k principal eigenvectors of $\mathrm{diag}(\boldsymbol{\mu})$, i.e., the one-hot vectors with i-th element equaling 1. Given that $\hat{\psi}_i = \sum_{j\geq 1} \boldsymbol{a}_{i,j}\psi_j$, the deployed parametric model $\hat{\psi}$ will converge to the k principal eigenfunctions of the unknown ground-truth kernel integral operator.

Cross-attention Variant. For a pair of functions $(\boldsymbol{f}_n, \boldsymbol{u}_n)$, the data points used to discretize them are different, denoted as $\mathbf{X}$ and $\mathbf{Y}$. Let $\mathcal{H}^{(l)}, l \in [1, L]$ denote the specified operators at various modeling stages. We define the propagation rule as

$$\begin{aligned}
(\mathcal{H}^{(1)}\boldsymbol{h}^{(1)})(\mathbf{Y}) &\approx \mathrm{FFN}(\mathrm{LN}(\left(\hat{\boldsymbol{\psi}}^{(1)}(\mathbf{Y})\hat{\boldsymbol{\psi}}^{(1)}(\mathbf{X})^\top \left[\boldsymbol{h}^{(1)}(\mathbf{X})\right]\right))) \\
(\mathcal{H}^{(l)}\boldsymbol{h}^{(l)})(\mathbf{Y}) &\approx \mathrm{FFN}(\mathrm{LN}(\left(\hat{\boldsymbol{\psi}}^{(l)}(\mathbf{Y})\hat{\boldsymbol{\psi}}^{(l)}(\mathbf{Y})^\top \left[\boldsymbol{h}^{(l)}(\mathbf{Y})\right] + \boldsymbol{h}^{(l)}(\mathbf{Y})\right))),\ l \in [2, L]
\end{aligned} \tag{20}$$

where $\mathrm{FFN}(\cdot)$ denotes a two-layer FFN and $\mathrm{LN}(\cdot)$ denotes the layer normalization.

B. Hyperparameters and Details for Models

FNO and its Variants. For FNO and its variants (Geo-FNO, F-FNO, U-FNO), we employ 4 layers with modes of 12 and widths from $\{20, 32\}$. Notably, Geo-FNO reverts to the vanilla FNO when applied to benchmarks with regular grids, resulting in equivalent performance for Darcy and NS2d benchmarks. For U-FNO, the U-Net path is appended in the last layer. FNO-2D is implemented in generalization experiments. The batch size is selected from $\{10, 20\}$.

LSM. We configure the model with 8 basis operators and 4 latent tokens. The width of the first scale is set to 32, and the downsampling ratio is 0.5. The batch size is selected from $\{10, 20\}$.

Table 8. Comparison of parameter count and memory requirements between ONO and baseline models.

MODEL	FNO	U-FNO	LSM	Galerkin	ONO (Linear)	ONO (Nyström)
Params (M)	0.9-18.9	1.0-19.4	4.8-13.9	2.2-2.5	0.8-2.0	0.8-2.0
Memory (G)	1-3	1-4	1-6	2-9	2-14	8-16

C. Limitation.

One limitation of this study is the mamory requirement as shown in Table 8. As shown, despite with comparable or fewer parameters, ONO exhibits higher memory requirements than the baselines, which is attributed to its dual-pathway architecture. To mitigate the memory overhead, we can properly lighten the lower pathway of ONO. We can also include sub- and up-sampling mechanisms in the front and end of ONO to shorten the sequence length for memory reduction.