Transolver Is a Linear Transformer: Revisiting Physics-Attention Through the Lens of Linear Attention

Wenjie Hu[1,2*], **Sidun Liu**[1,2*], **Peng Qiao**[1,2†], **Zhenglun Sun**[1,2], **Yong Dou**[1,2†]

[1]College of Computer Science and Technology, National University of Defense Technology, Changsha, China
[2]National Key Laboratory of Parallel and Distributed Computing, National University of Defense Technology, Changsha, China
{hwjb127,liusidun,pengqiao,zhenglun_sun,yongdou}@nudt.edu.cn

Abstract

Recent advances in Transformer-based Neural Operators have enabled significant progress in data-driven solvers for Partial Differential Equations (PDEs). Most current research has focused on reducing the quadratic complexity of attention to address the resulting low training and inference efficiency. Among these works, Transolver stands out as a representative method that introduces Physics-Attention to reduce computational costs. Physics-Attention projects grid points into slices for slice attention, then maps them back through deslicing. However, we observe that Physics-Attention can be reformulated as a special case of linear attention, and that the slice attention may even hurt the model performance. Based on these observations, we argue that its effectiveness primarily arises from the slice and deslice operations rather than interactions between slices. Building on this insight, we propose a two-step transformation to redesign Physics-Attention into a canonical linear attention, which we call *Linear Attention Neural Operator* (LinearNO). Our method achieves state-of-the-art performance on six standard PDE benchmarks, while reducing the number of parameters by an average of 40.0% and computational cost by 36.2%. Additionally, it delivers superior performance on two challenging, industrial-level datasets: AirfRANS and Shape-Net Car.

Code — https://github.com/HiPRL/LinearNO

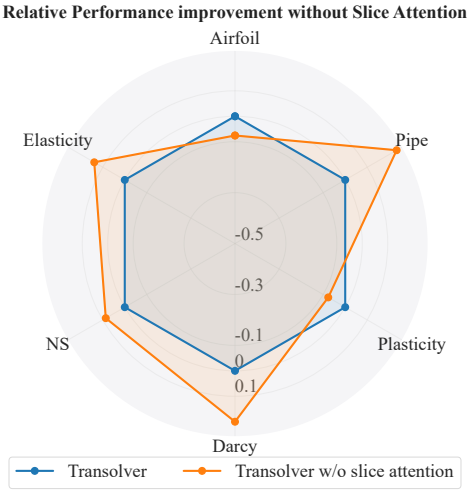


Figure 1: We observe that in most scenarios, removing the slice attention in Physics-Attention leads to performance improvement. This suggests that its effectiveness mainly stems from the slice and deslice operations, rather than the interactions between slices.

Introduction

Solving Partial Differential Equations is a fundamental task in many fields of science and engineering. Due to their complexity, these equations often require discretization into meshes and solving with numerical methods (Piomelli 1999; Alfonsi 2009; Lee and Moser 2015), which are computationally expensive and time-consuming. Recent advances in deep learning provide new approaches for PDE solving (Chen et al. 2022; Liu et al. 2024; Huang et al. 2025).

The essence of PDE solving is to find the solution function corresponding to the PDE and its boundary or initial conditions. This can be formulated as a supervised learning problem, where neural networks learn the mappings from boundary or initial functions to solution functions. Neural Operator (Lu et al. 2021; Kovachki et al. 2023) was proposed to learn the mappings between those two function spaces while guaranteeing both discretization-invariance and universal approximation.

The integral kernel operator of Neural Operator (Kovachki et al. 2023) is a non-local operation that passes messages between discrete grid points in a discretization-invariant manner, which is analogous to the transformer's self-attention (Vaswani et al. 2017). Therefore, a series of works have introduced transformers into PDE solving and Neural Operator construction. However, when considering discrete grid points as tokens, the quadratic complexity of self-attention seriously limits the scale of PDE problems. Several strategies are employed to optimize the complexity of self-attention. Some methods (Li, Meidani, and Farimani 2023; Li, Shu, and Farimani 2023; Hao et al. 2023) introduce linear attention (Katharopoulos et al. 2020) to decrease the quadratic complexity to linear. Other methods (Wang and

*These authors contributed equally.
†Co-corresponding author.

Wang 2024; Serrano et al. 2024; Alkin et al. 2024) follow the Set Transformer (Lee et al. 2019), projecting grid points into a latent space to reduce their length and performing attention in that space for PDE solving. All of the above works aim to construct low-rank patterns to achieve complexity reduction. Transolver (Wu et al. 2024), as a representative model, proposes a Physics-Attention module which contains a slicing operation to project the grid points into several slices and perform slice attention, and a deslicing operation to project the slices back to grid points. However, the design of this module is still intuitive, and lacks in-depth analysis.

In this paper, we reveal that the Transolver's Physics-Attention is actually a special case of linear attention. We further conduct preliminary experiments and observe that in most scenarios, removing the slice attention in Physics-Attention leads to performance improvement, as shown in Figure 1. This suggests that the effectiveness of Physics-Attention mainly comes from the slice and deslice operations rather than interactions between slices. Based on this insight, we conduct a two-step transformation to make Physics-Attention a canonical linear attention. Specifically, in the generalization step, we relax certain constraints of the Physics-Attention to align it with linear attention. In the simplification step, we demonstrate the non-essential nature of slice attention and consequently remove it. The derived *Linear Attention Neural Operator*, or LinearNO, retains a canonical structure and achieves higher PDE solving accuracy than Physics-Attention with fewer parameters and computational cost. We further prove that LinearNO is a Monte Carlo approximation of the integral kernel operator.

Our main contributions can be summarized as follows:

- We reveal that the Physics-Attention proposed by Transolver is essentially a special case of linear attention.
- We propose a two-step transformation to make Physics-Attention a canonical linear attention, and illustrate the rationale behind it.
- The derived linear attention model outperforms Transolver with fewer parameters and computational cost, achieving state-of-the-art on multiple PDE solving tasks.

Related Work

Neural Operators

Neural Operators are a class of models that aim to solve PDE by learning a mapping from boundary or initial functions to solution functions. They achieve mappings between function spaces by ensuring universal approximation and discretization-invariance (Kovachki et al. 2023). DeepONet (Lu et al. 2021) uses a two-branch architecture to learn the mapping between functions. Graph Neural Operator (GNO) (Li et al. 2020) implements the integral kernel operator using message passing in graph neural networks. Fourier Neural Operator (FNO) (Li et al. 2021) performs the integral kernel operator by mapping input functions to the frequency domain via Fourier transforms. GEO-FNO (Li et al. 2023a) extends FNO to unstructured meshes by mapping irregular domains onto structured grids. U-FNO (Wen et al. 2022) enhances the multi-scale feature extraction capability of FNO by incorporating a U-Net architecture.

Transformers for PDE Solving

Transformer (Vaswani et al. 2017) has been shown to be a special case of integral kernel operators (Kovachki et al. 2023). However, its quadratic computational complexity makes it difficult to apply to large-scale PDE problems efficiently. Some methods, such as OFormer (Li, Meidani, and Farimani 2023), FactFormer (Li, Shu, and Farimani 2023), and GNOT (Hao et al. 2023), adopt linear attention to reduce the computational cost while maintaining the modeling capacity. Another line of methods aims to reduce complexity by compressing the length of the token sequence. Methods such as LNO (Wang and Wang 2024), AROMAP (Serrano et al. 2024), UPT (Alkin et al. 2024), and AeroGTO (Liu et al. 2025) reduce computational complexity by introducing a learnable set of latent tokens to project discrete grid points into a latent space using cross-attention, followed by self-attention within the latent space. LSM (Wu et al. 2023) also employs the same compression strategy to reduce sequence length, and replaces self-attention with spectral transformation for compressed tokens. Transolver (Wu et al. 2024) introduces Physics-Attention, which uses a learnable slice-weight matrix to assign discrete grid points to physical slices, grouping similar grid points together. Self-attention is then applied across slices to exchange information and uncover deeper physical relationships within the complex geometry. The slice-weight matrix is used to decode the slices back to grid points. Transolver++ (Luo et al. 2025) proposes local adaptation for better slice distinction.

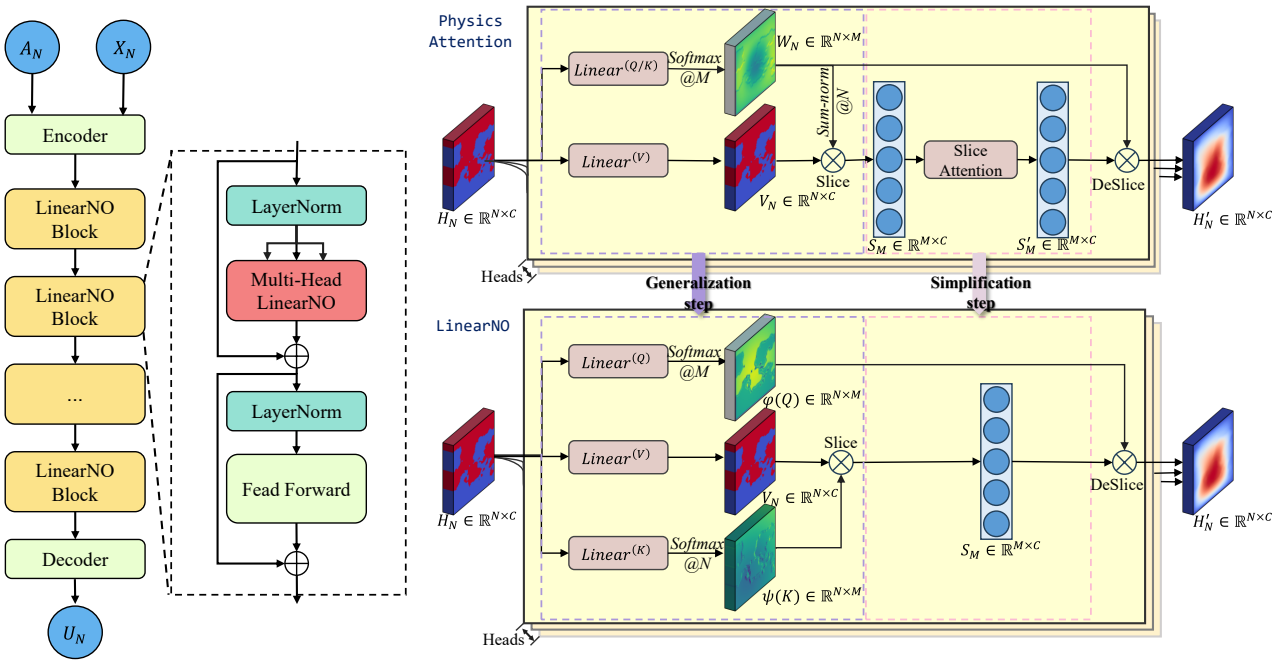
Linear Attention

The Softmax operation on the attention matrix is the key factor contributing to the quadratic complexity of attention. To address this issue, linear attention uses kernel functions to approximate the Softmax function, allowing it to change the computation order of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ in full attention, thereby reducing the computational complexity from quadratic to linear. The Linear Transformer (Katharopoulos et al. 2020) replaces the softmax operation with a kernel function, achieving performance comparable to standard attention. Both Performer (Choromanski et al. 2020) and RFA (Peng et al. 2021) employ random feature methods to approximate the softmax function, making the approximation mathematically equivalent to full attention. TSSA (Wu et al. 2025) derives a new linear attention by optimizing a variational form of the Maximum Coding Rate Reduction objective. RALA (Fan, Huang, and He 2025) introduces a rank-augmented method to enhance the performance.

Methodology

Preliminaries

Problem Setting. For a PDE problem defined on the spatial domain $\Omega \subseteq \mathbb{R}^d$, the domain Ω is discretized into N grid points, denoted as $\{\boldsymbol{x}_i\}_{i=1}^N = \boldsymbol{X}_N \in \mathbb{R}^{N\times d}$, where $\boldsymbol{X}_N$ contains the coordinates of all grid points. Correspondingly, we obtain the discretized form of the initial/boundary condition function, $\{a(\boldsymbol{x}_i)\}_{i=1}^N = \boldsymbol{A}_N \in \mathbb{R}^{N\times d_a}$, and the discretized solution, $\{u(\boldsymbol{x}_i)\}_{i=1}^N = \boldsymbol{U}_N \in \mathbb{R}^{N\times d_u}$. Our objective is to learn the mapping from the coordinates and



(a) The overall design

(b) Physics-Attention vs. LinearNO

Figure 2: (a) The overall design of our network. The encoder and decoder modules follow the same architectural design as those in Transolver. (b) Comparison between Physics-Attention and LinearNO. The top shows the Physics-Attention in Transolver, while the bottom depicts our LinearNO. *Softmax@M* and *Softmax@N* indicate softmax operations along dimensions M and N, respectively. *Sum-norm@N* refers to the standard normalization $x'_i = \frac{x_i}{\sum_{j=1}^{N} x_j}$ for each row.

initial/boundary condition function $(\boldsymbol{X}_N, \boldsymbol{A}_N)$ to the corresponding solution function $\boldsymbol{U}_N$, as governed by the operator L_a. The inclusion of $\boldsymbol{A}_N$ is problem-dependent.

Transolver Revisiting. Transolver (Wu et al. 2024) proposes a Transformer-based Neural Operator to capture the physical properties of PDE solutions through the Physics-Attention. In this section, we briefly revisit its core components. The input to the Physics-Attention is a feature matrix $\boldsymbol{H}_N = \{\boldsymbol{h}_i\}_{i=1}^N \in \mathbb{R}^{N\times d_h}$, obtained from $\boldsymbol{A}_N$ and $\boldsymbol{X}_N$, where N is the number of spatial grid points and d_h is the feature dimension. First, the input feature matrix $\boldsymbol{H}_N$ is passed through a linear layer followed by a softmax function to produce a slice-weight matrix $\boldsymbol{W}_N = \{\boldsymbol{w}_i\}_{i=1}^N \in \mathbb{R}^{N\times M}$, where M is a hyperparameter denoting the number of slices. A channel-wise linear layer is subsequently applied to $\boldsymbol{H}_N$ to obtain $\boldsymbol{V}_N = \{\boldsymbol{v}_i\}_{i=1}^N \in \mathbb{R}^{N\times d_h}$. Using this slice-weight matrix, Physics-Attention computes a weighted average of $\boldsymbol{V}_N$ to obtain a slice feature matrix $\boldsymbol{S}_M = \{\boldsymbol{s}_j\}_{j=1}^M \in \mathbb{R}^{M\times d_h}$. This operation can be interpreted as projecting the original N grid points onto M physical slices, effectively compressing the sequence length from N to M. The computation is formally given by:

$$\boldsymbol{w}_i = \text{Softmax}(\text{Linear}^{(Q/K)}(\boldsymbol{h}_i)) \quad i = 1, \ldots, N \tag{1}$$

$$\boldsymbol{v}_i = \text{Linear}^{(V)}(\boldsymbol{h}_i) \quad i = 1, \ldots, N \tag{2}$$

$$\boldsymbol{s}_j = \frac{\sum_{i=1}^N \boldsymbol{w}_{ij}\boldsymbol{v}_i}{\sum_{i=1}^N \boldsymbol{w}_{ij}} \quad j = 1, \ldots, M \tag{3}$$

The $\text{Linear}^{(Q/K)}(\cdot)$ and $\text{Linear}^{(V)}(\cdot)$ are single-layer MLPs with output dimensions of M and d_h, respectively. A self-attention on the slices $\boldsymbol{S}_M$ is then performed to produce an updated slices $\boldsymbol{S}'_M = \{\boldsymbol{s}'_j\}_{j=1}^M \in \mathbb{R}^{M\times d_h}$. Finally, using the slice-weight matrix $\boldsymbol{W}_N$, the slices $\boldsymbol{S}'_M$ are decoded back to the original sequence length, yielding the output $\boldsymbol{H}'_N = \{\boldsymbol{h}'_i\}_{i=1}^N \in \mathbb{R}^{N\times d_h}$. The process is summarized as follows:

$$\boldsymbol{S}'_M = \text{Self-Attention}(\boldsymbol{S}_M) \tag{4}$$

$$\boldsymbol{h}'_i = \sum_{j=1}^M \boldsymbol{w}_{ij}\boldsymbol{s}'_j \quad i = 1, \ldots, N \tag{5}$$

Linear Attention. As shown in Eq. (6), linear attention aims to approximate the softmax function by designing different kernel functions $\varphi(\cdot)$ and $\psi(\cdot)$, allowing it to change the computation order of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$. In OFormer (2023) and FactFormer (2023), the kernel functions correspond to vector normalization and the identity function, respectively. The choice of kernel functions directly affects the performance of linear attention.

$$\begin{aligned}\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})\boldsymbol{V} \\ &\approx \varphi(\boldsymbol{Q})(\psi^\top(\boldsymbol{K})\boldsymbol{V})\end{aligned} \tag{6}$$

Some improvements of linear attention insert additional intermediate operations between $\psi^\top(\boldsymbol{K})\boldsymbol{V}$ and $\varphi(\boldsymbol{Q})$, i.e.,

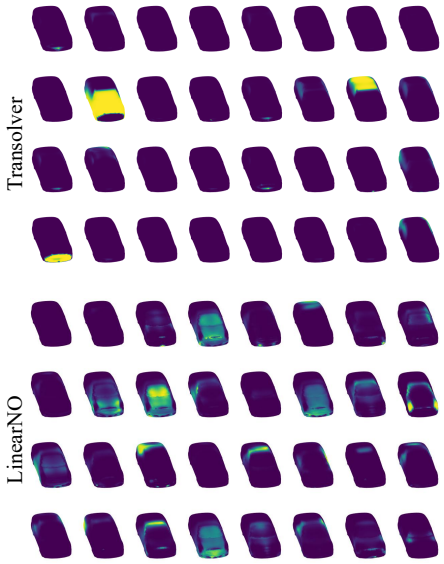


Figure 3: Visualization of Slice-Weight Matrix W_N

$\varphi(\boldsymbol{Q}) \circ \mathcal{G} \circ \psi^\top(\boldsymbol{K})\boldsymbol{V}$. The operation $\mathcal{G}$ can be rank augmentation (Fan, Huang, and He 2025), forget gate (Yang et al. 2023), etc.

Equivalence of Transolver to Linear Attention

By comparing the two definitions, we observe that Physics-Attention is essentially a special case of linear attention. Specifically, we can construct equivalent feature mappings $\varphi(\boldsymbol{Q})$, $\psi(\boldsymbol{K})$, and $\boldsymbol{V}$ as in Eq.(7),

$$\begin{aligned} \varphi(\boldsymbol{Q}) &= \left\{ \text{Softmax}(\text{Linear}^{(Q/K)}(\boldsymbol{h}_i)) \right\}_{i=1}^{N} = \{\boldsymbol{w}_i\}_{i=1}^{N} \\ \psi(\boldsymbol{K}) &= \left\{ \frac{\boldsymbol{w}_i}{\sum_{j=1}^{N} \boldsymbol{w}_j} \right\}_{i=1}^{N} \\ \boldsymbol{V} &= \left\{ \text{Linear}^{(V)}(\boldsymbol{h}_i) \right\}_{i=1}^{N} \end{aligned} \tag{7}$$

and cast the intermediate operation $\mathcal{G}$ as self-attention to reformulate Physics-Attention into a linear attention form. The linear attention term $\varphi(\boldsymbol{Q})$ is equivalent to the slice-weight matrix $\boldsymbol{W}_N$ used in Physics-Attention. In the following sections, we will use these two notations interchangeably depending on the context.

As a result, Physics-Attention is a special case of linear attention where (1) $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ are from the same learnable layer $\text{Linear}^{(Q/K)}(\cdot)$ and differ only in their respective normalization schemes, and (2) self-attention is applied as the intermediate operation $\mathcal{G}$. However, the above two characteristics do not help the model performance, which we will discuss further later.

LinearNO: Linear Attention Neural Operator

In this section, we present a more generalized analysis of the Physics-Attention. Based on this, we redesign it through two key steps—generalization and simplification—to derive a more flexible architecture called the *Linear Attention Neural Operator*, or LinearNO. Figure 2b shows a comparison between the proposed LinearNO and the Physics-Attention.

Generalization step. Luo et al. (2025) observed that Physics-Attention tends to produce uniform slice weights, thus generating less distinguishable slices in some cases, which damages the model's performance, as shown in Figure 3. We hypothesize that this may be attributed to the intended asymmetry between $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ in linear attention; enforcing the shared learnable layer between $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ could blur their roles and result in overly averaged representations. Based on this hypothesis, we relaxed the weight-sharing constraint in the generalization step, resulting in asymmetric $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$. Specifically, we allow $\psi(\boldsymbol{K})$ to be learned independently of $\varphi(\boldsymbol{Q})$. Following Physics-Attention, $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ are normalized along the M and N dimensions, respectively. This ensures that each row of attention matrix $\varphi(\boldsymbol{Q})\psi^\top(\boldsymbol{K})$ is normalized, which is consistent with full attention (Shen et al. 2021). Formally, we define $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ as follows:

$$\begin{aligned} \varphi(\boldsymbol{Q}) &= \text{Softmax}(\text{Linear}^{(Q)}(\boldsymbol{H}_N)) \\ \psi^\top(\boldsymbol{K}) &= \text{Softmax}(\text{Linear}^{(K)}(\boldsymbol{H}_N)^\top) \end{aligned} \tag{8}$$

where $\boldsymbol{H}_N \in \mathbb{R}^{N\times d}$ is the input feature, $\text{Linear}^{(Q)}(\cdot)$ and $\text{Linear}^{(K)}(\cdot)$ both contain learnable parameters of shape $\mathbb{R}^{d_h \times M}$, and Softmax is computed row by row. After this step, we observed more diverse attention patterns and a more saturated rank in the attention matrix.

Simplification step. In Physics-Attention, the slice ($\psi(\boldsymbol{K})$) and deslice ($\varphi(\boldsymbol{Q})$) operations are symmetric, which limits the scope of feature interaction to a token itself and those similar to it. As a result, cross-slice feature interaction is not possible, making slice attention necessary. However, after the generalization step lifts the symmetry constraint between $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$, each token can interact with all others during the slice and deslice processes. This makes the slice attention unnecessary. Besides, the experiment in Figure 1 also indicates that even under symmetric constraints, slice attention still fails to deliver consistent performance gains. Therefore, in the simplification step, we remove this attention entirely and set the intermediate operation $\mathcal{G}$ to the identity.

LinearNO. Through the two steps above, we derive a canonical form of linear attention, which we refer to as the *Linear Attention Neural Operator* (LinearNO). It is formalized as follows:

$$\text{LinearNO}(\boldsymbol{H}_N) = \varphi(\boldsymbol{Q}) \left(\psi^\top(\boldsymbol{K}) \cdot \text{Linear}^{(V)}(\boldsymbol{H}_N) \right) \tag{9}$$

Here we provide a theorem to illustrate that LinearNO is a standard Neural Operator.

Theorem 1. *Let $\{\boldsymbol{x}_i\}_{i=1}^{+\infty}$ be a sequence of refined meshes on Ω with $\boldsymbol{x}_i \sim \mu_\Omega$, and assume that the function $v(\boldsymbol{x})$ is bounded on Ω. As $n \to +\infty$, for any $\epsilon > 0$, the proposed LinearNO converges in probability to a continuous integral kernel operator:*

Model	Airfoil	Pipe	Plasticity	NS	Darcy	Elasticity
FNO (2021)	/	/	/	0.1556	0.0108	/
U-FNO (2022)	0.0269	0.0056	0.0039	0.2231	0.0183	0.0239
GEO-FNO (2023a)	0.0138	0.0067	0.0074	0.1556	0.0108	0.0229
F-FNO (2023)	0.0078	0.0070	0.0047	0.2322	0.0077	0.0263
Galerkin (2021)	0.0118	0.0098	0.0120	0.1401	0.0084	0.0240
OFormer (2023)	0.0183	0.0168	0.0017	0.1705	0.0124	0.0183
GNOT (2023)	0.0076	0.0047	0.0336	0.1380	0.0105	0.0086
FactFormer (2023)	0.0071	0.0060	0.0312	0.1214	0.0109	/
ONO (2024)	0.0061	0.0052	0.0048	0.1195	0.0076	0.0118
LSM (2023)	0.0059	0.0050	0.0025	0.1535	0.0065	0.0218
LNO* (2024)	0.0053	0.0031	0.0028	<u>0.0830</u>	0.0063	0.0066
Transolver* (2024)	0.0053	0.0030	<u>0.0013</u>	0.0882	<u>0.0055</u>	0.0065
Transolver++† (2025)	<u>0.0051</u>	<u>0.0027</u>	0.0014	0.1010	0.0056	<u>0.0064</u>
LinearNO (ours)	**0.0049**	**0.0024**	**0.0011**	**0.0699**	**0.0050**	**0.0050**

Table 1: Comparison of relative L2 errors on six standard PDE benchmark tasks. / denotes the method is not applicable to the task. * denotes results reproduced by us. † denotes results reproduced based on descriptions in the paper due to unavailable source code. An underscore indicates the second-best result, and bold indicates the best result.

$$\lim_{n\to+\infty} \Pr\left\{\frac{1}{N}\|\textit{LinearNO}(\boldsymbol{x}) - \mathcal{F}(\boldsymbol{x})\| \le \epsilon\right\} = 1$$

$$\mathcal{F}(\boldsymbol{x}) := \int_{\Omega} \kappa(v(\boldsymbol{x}), v(\boldsymbol{y}))v(\boldsymbol{y})\boldsymbol{R}\, d\mu_{\Omega}(\boldsymbol{y})$$

Here, $\boldsymbol{R}$ are learnable parameters, $\mathcal{F}(\boldsymbol{x})$ is the continuous integral kernel operator, and the kernel function κ of LinearNO is defined as:

$$\kappa(v(\boldsymbol{x}), v(\boldsymbol{y})) = \frac{\varphi(v(\boldsymbol{x}))\exp(\boldsymbol{B}^{\top}v(\boldsymbol{y})^{\top})}{\int_{\Omega}\exp\left(\boldsymbol{B}^{\top}v(\boldsymbol{y})^{\top}\right)d\mu_{\Omega}(\boldsymbol{y})}$$

where $\boldsymbol{B}$ are learnable parameters, and $\varphi(\cdot)$ denotes the point-wise normalization function as defined in Eq. (8)*.*

This theorem shows that LinearNO satisfies the discretization-invariance of Neural Operators and serves as a Monte Carlo approximation in the form of the continuous integral kernel operator, enabling a mapping from function spaces to function spaces. We provide a detailed proof of Theorem 1 in Appendix A.

Experiments

Experimental Setup

Datasets We adopt six benchmark datasets, including Airfoil, Pipe, Plasticity, NS, Darcy and Elasticity, which are classical problems in fluid mechanics and solid mechanics. These datasets were introduced by FNO (Li et al. 2021) and Geo-FNO (Li et al. 2023a), and are now widely used as standard benchmarks for evaluating Neural Operators. In addition, we use two industrial-level datasets: AirfRANS (Bonnet et al. 2022) and ShapeNet-Car (Umetani and Bickel 2018) to assess the model's performance in aerodynamic shape design. More detailed information about all datasets can be found in Appendix B.

Baselines We compare our model against several representative baselines, including classical Neural Operator models: FNO (2021), Geo-FNO (2023a), F-FNO (2023) and U-FNO (2022), Transformer-based models: Galerkin (2021), OFormer (2023), GNOT (2023), FactFormer (2023), ONO (2024), LSM (2023), LNO (2024), Transolver (2024) and Transolver++ (2025), classical geometric deep models: PointNet (2017), GraphSage (2017) and GraphUNet (2019), MeshGraphNet (2020).

Setup To ensure fair comparisons, we follow the experimental settings of Transolver (Wu et al. 2024). We report the relative L2 error and Spearman correlation coefficient ρ as the evaluation metric. Detailed experimental settings and hyperparameter configurations are provided in Appendix B.

Main Results

Accuracy Comparison. Table 1 presents the relative L2 errors of LinearNO and SOTA models across six PDE benchmarks. LinearNO achieves the best performance on all tasks. Specifically, LinearNO achieves over a 10% relative improvement on the Pipe, Plasticity, NS and Elasticity task.

We further evaluate our method on two complex, industrial-level high-fidelity datasets: AirfRANS and ShapeNet Car, with results summarized in Table 2. Our model achieves the best results on both tasks, especially on the AirfRANS dataset, where LinearNO outperforms Transolver by more than 60% in predicting the lift coefficient C_L, achieving a Spearman's correlation coefficient of 0.9992. This demonstrates that our method can more accurately predict the aerodynamic properties of different geometries, thereby assisting engineers in the iterative design of shapes with superior aerodynamic characteristics. We also conducted generalization experiments on the AirfRANS dataset. Specifically, the training and testing sets have different ranges of Reynolds numbers and angles of attack. For more details, please refer to Appendix D.

Model	AirfRANS				Shape-Net Car			
	Volume ↓	Surface ↓	C_L ↓	ρ_L ↑	Volume ↓	Surface ↓	C_D ↓	ρ_D ↑
MLP	0.0081	0.0200	0.2108	0.9932	0.0512	0.1304	0.0307	0.9496
GraphSage (2017)	0.0087	0.0184	0.1476	0.9964	0.0461	0.1050	0.0270	0.9695
PointNet (2017)	0.0253	0.0996	0.1973	0.9919	0.0494	0.1104	0.0298	0.9583
GraphUNet (2019)	0.0076	0.0144	0.1677	0.9949	0.0471	0.1102	0.0226	0.9725
MeshGraphNet (2020)	0.0214	0.0387	0.2252	0.9945	0.0354	0.0781	0.0168	0.9840
GNO (2020)	0.0269	0.0405	0.2016	0.9938	0.0383	0.0815	0.0172	0.9834
GEO-FNO (2023a)	0.0361	0.0301	0.6161	0.9257	0.1670	0.2378	0.0664	0.8280
GALERKIN (2021)	0.0074	0.0159	0.2336	0.9951	0.0339	0.0878	0.0179	0.9764
GNOT (2023)	0.0049	0.0152	0.1992	0.9942	0.0329	0.0798	0.0178	0.9833
GINO (2023b)	0.0297	0.0482	0.1821	0.9958	0.0386	0.0810	0.0184	0.9826
LNO* (2024)	0.0214	0.0268	0.1480	0.9744	0.0269	0.0870	0.0174	0.9781
Transolver* (2024)	<u>0.0023</u>	<u>0.0085</u>	<u>0.1230</u>	<u>0.9978</u>	<u>0.0221</u>	<u>0.0785</u>	<u>0.0117</u>	<u>0.9914</u>
Transolver++† (2025)	0.0068	0.0159	0.1880	0.9910	0.0226	0.0800	0.0132	<u>0.9914</u>
LinearNO (ours)	**0.0011**	**0.0077**	**0.0491**	**0.9992**	**0.0194**	**0.0754**	**0.0106**	**0.9925**

Table 2: Performance on AirfRANS and Shape-Net Car. Volume and Surface represent the physical fields in the surrounding flow region and on the object surface, respectively. C_L and C_D denote the lift coefficient and drag coefficient, respectively. The table reports their relative L2 errors. Spearman's correlation coefficient ρ is closer to 1 indicating better performance. * indicates results reproduced by us. † denotes results reproduced based on descriptions in the paper due to unavailable source code. An underscore indicates the second-best result, and bold indicates the best result.

Metric	Model	Airfoil	Darcy	Elas*
Parameter (GB)	Transolver	2.81	2.83	0.71
	LinearNO	**1.77**	**1.77**	**0.59**
Computation (GFLOPs)	Transolver	32.38	20.87	0.76
	LinearNO	**21.34**	**13.68**	**0.69**

Table 3: Comparison on parameter count and computational cost between LinearNO and Transolver. Elas* is used as an abbreviation for Elasticity to fit the table layout. For all benchmark datasets, please refer to Appendix C.

No.	*Gen.*	*Sim.*	Airfoil	Darcy	Elasticity
1	×	×	0.0067	0.0064	0.0069
2	✓	×	0.0054	0.0061	0.0062
3	×	✓	0.0071	0.0052	0.0064
4	✓	✓	**0.0049**	**0.0050**	**0.0050**

Table 4: Ablation study of generalization and simplification. *Gen.* denotes the generalization step, and *Sim.* denotes the simplification step. For all benchmark datasets, please refer to Appendix C.

Overall, these results demonstrate the superior performance of LinearNO and provide strong empirical support for our theoretical generalization and simplification of the Physics-Attention.

Efficiency Analysis. We compare the number of parameters and computational cost between LinearNO and Transolver on six standard PDE tasks, as shown in Table 3. It can be observed that LinearNO has significantly fewer parameters and lower computational cost than Transolver. On average, our method reduces the number of parameters by 40.0% and the FLOPs by 36.2%. This demonstrates that LinearNO is more computationally efficient and lightweight, making it more suitable for resource-constrained environments.

Slice Analysis. The rank of attention matrix $\varphi(\boldsymbol{Q})\psi^\top(\boldsymbol{K})$ corresponds to the number of physical states effectively captured by the model, which influences its performance (Luo et al. 2025). This effective rank is bounded by the number of slices M. We hypothesize that the symmetric design of $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ in Transolver limits the effective utilization of slices. In this section, we present a quantitative analysis to support this claim.

We estimate the layer-wise rank of the attention matrix using Singular Value Decomposition (Golub and Kahan 1965), and the results are shown in Figure 4. The findings indicate that the asymmetric design of $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ facilitates more efficient utilization of slices. Furthermore, we visualize the distribution of the final-layer $\varphi(\boldsymbol{Q})$ matrix on the Airfoil dataset. Compared to Transolver, LinearNO exhibits more diverse physical slices. In addition, LinearNO consistently outperforms Transolver across different numbers of slices on the Airfoil dataset.

Ablation Study

Effect of generalization and simplification steps. We design ablation experiments to evaluate the impact of the generalization and simplification steps on model performance. Specifically, we set $\psi^\top(\boldsymbol{K}) = \text{Softmax}(\varphi^\top(\boldsymbol{Q}))$ to dis-

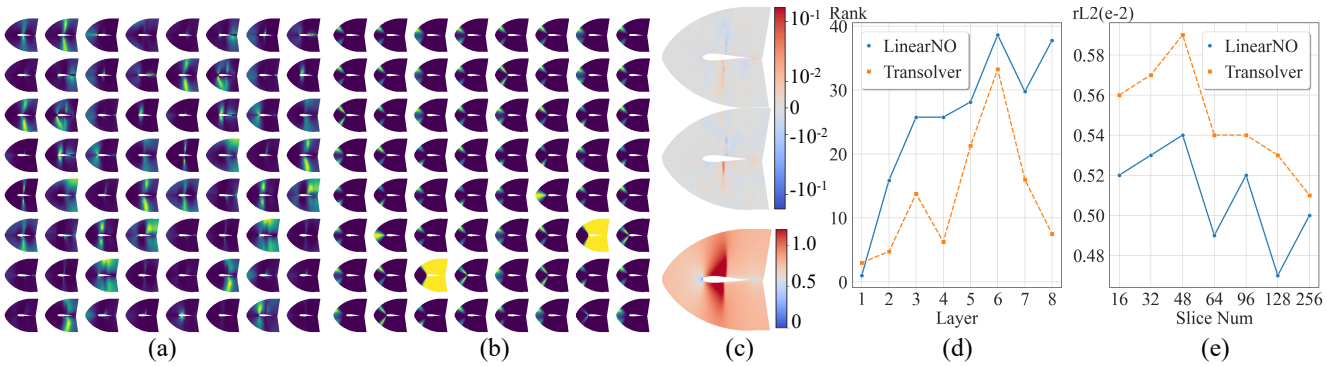


Figure 4: All experiments are conducted on the Airfoil dataset. (a) and (b) show the final-layer slice-Weight matrix W_N visualizations of LinearNO and Transolver, respectively. (c) shows the error distributions of LinearNO (top) and Transolver (middle), with the ground truth shown at the bottom. (d) shows the average rank of the attention matrix $\varphi(\boldsymbol{Q})\psi^\top(\boldsymbol{K})$ per head at each layer when slice numbers $M = 64$. (e) presents the prediction errors of both models under different slice numbers.

Num of Slice M	Airfoil	Darcy	Elasticity
16	0.0052	0.0056	0.0062
32	0.0053	0.0053	0.0056
48	0.0054	0.0051	0.0052
64	0.0049	0.0050	0.0050
96	0.0052	**0.0048**	0.0056
128	**0.0047**	0.0050	0.0051
256	0.0050	0.0049	**0.0046**

Table 5: Relative L2 errors of LinearNO with different numbers of slices M. For all benchmark datasets, please refer to Appendix C.

$\varphi(Q)$	$\psi(K)$	Airfoil	Darcy	Elasticity
N	M	0.0059	0.0055	0.0112
M	M	0.0061	0.0060	0.0081
N	N	0.0068	0.0056	0.0095
M	N	**0.0049**	**0.0050**	**0.0050**

Table 6: Comparison of relative L2 errors under different normalization dimensions on the Airfoil, Darcy, and Elasticity datasets.

able the generalization step, and we add a slice attention to disable the simplification step. We compare these variants with LinearNO, as shown in Table 4. By comparing No.1 vs. No.2 and No.3 vs. No.4, we observe that the generalization step consistently improves model performance in most cases, confirming the benefit of using asymmetric projections. Furthermore, the comparison between No.1 and No.3 shows that, under symmetric settings, slice attention does not consistently improve performance, which is also observed in Transolver. In contrast, comparing No.2 and No.4 reveals that applying slice attention under asymmetric projections consistently hurts the performance. Therefore, the simplification step is also justified.

Slice Number. The results in Table 5 demonstrate that increasing the number of slices M generally improves the prediction accuracy across different PDE tasks, with the most significant gains observed in tasks like NS and Elasticity. For most cases, the relative L2 error decreases as M increases, with $M = 256$ often yielding the best performance. However, the improvement exhibits diminishing returns beyond a certain point, and for some tasks such as Darcy and Airfoil, using an excessive number of slices might not lead to further improvement or may even degrade performance slightly. Therefore, selecting an optimal number of slices requires balancing accuracy gains with computational costs and task-specific characteristics.

Softmax Normalization. We also conduct experiments to investigate the impact of normalization dimension on model performance. Previous works present multiple combinations of normalization dimensions for $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ (Shen et al. 2021; Hao et al. 2023). In this section, compare different combinations of Softmax normalization. Here, we assume $\varphi(\boldsymbol{Q}), \psi(\boldsymbol{K}) \in \mathbb{R}^{N\times M}$. As shown in Table 6, applying Softmax to $\varphi(\boldsymbol{Q})$ and $\psi(\boldsymbol{K})$ on dimensions M and N performs best, because it maintains the row normalization property of the attention matrix $\varphi(\boldsymbol{Q})\psi^\top(\boldsymbol{K})$.

Conclusion

In this work, we first review the Physics-Attention and reveal its essence as a form of linear attention. Based on this insight, we propose a two-step transformation: a generalization step and a simplification step, which transform the original Physics-Attention into a more efficient model called LinearNO. LinearNO has fewer parameters and lower computational cost. It achieves state-of-the-art results on all six standard PDE benchmark tasks. In addition, LinearNO demonstrates superior performance on two industrial-level datasets compared to other models. We further show through experiments that our model achieves a higher rank utilization than Transolver. Finally, we conduct a series of ablation studies to validate the effectiveness.

Acknowledgments

This work was supported by National Key R&D Program 2025YFB3003600 and the Open Fund of National Key Laboratory of Parallel and Distributed Computing (PDL)NO.WDZC20255290101.

References

Alfonsi, G. 2009. Reynolds-Averaged Navier–Stokes Equations for Turbulence Modeling. *Applied Mechanics Reviews*, 62(4): 20.

Alkin, B.; Fürst, A.; Schmid, S.; Gruber, L.; Holzleitner, M.; and Brandstetter, J. 2024. Universal physics transformers: a framework for efficiently scaling neural operators. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 25152–25194.

Bonnet, F.; Mazari, J. A.; Cinnella, P.; and Gallinari, P. 2022. AIRFRANS: high fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier-stokes solutions. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 23463–23478.

Cao, S. 2021. Choose a transformer: fourier or galerkin. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 24924–24940.

Chen, D.; Gao, X.; Xu, C.; Wang, S.; Chen, S.; Fang, J.; and Wang, Z. 2022. FlowDNN: a physics-informed deep neural network for fast and accurate flow prediction. *Frontiers of Information Technology & Electronic Engineering*, 23(2): 207–219.

Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Belanger, D.; Colwell, L.; et al. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*.

Fan, Q.; Huang, H.; and He, R. 2025. Breaking the low-rank dilemma of linear attention. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 25271–25280.

Gao, H.; and Ji, S. 2019. Graph u-nets. In *international conference on machine learning*, 2083–2092. PMLR.

Golub, G.; and Kahan, W. 1965. Calculating the Singular Values and Pseudo-Inverse of a Matrix. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, 2(2): 205–224.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

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

Huang, S.; Feng, W.; Tang, C.; He, Z.; Yu, C.; and Lv, J. 2025. Partial differential equations meet deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

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

Kovachki, N.; Li, Z.; Liu, B.; Azizzadenesheli, K.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2023. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89): 1–97.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, 3744–3753. PMLR.

Lee, M.; and Moser, R. D. 2015. Direct numerical simulation of turbulent channel flow up to. *Journal of fluid mechanics*, 774: 395–415.

Li, Z.; Huang, D. Z.; Liu, B.; and Anandkumar, A. 2023a. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388): 1–26.

Li, Z.; Kovachki, N. B.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A. M.; and Anandkumar, A. 2020. Neural Operator: Graph Kernel Network for Partial Differential Equations. *CoRR*, abs/2003.03485.

Li, Z.; Kovachki, N. B.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A. M.; and Anandkumar, A. 2021. Fourier Neural Operator for Parametric Partial Differential Equations. In *9th International Conference on Learning Representations, ICLR 2021*.

Li, Z.; Kovachki, N. B.; Choy, C.; Li, B.; Kossaifi, J.; Otta, S. P.; Nabian, M. A.; Stadler, M.; Hundt, C.; Azizzadenesheli, K.; et al. 2023b. Geometry-informed neural operator for large-scale 3D PDEs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 35836–35854.

Li, Z.; Meidani, K.; and Farimani, A. B. 2023. Transformer for Partial Differential Equations' Operator Learning. *Trans. Mach. Learn. Res.*, 2023.

Li, Z.; Shu, D.; and Farimani, A. B. 2023. Scalable transformer for PDE surrogate modeling. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 28010–28039.

Liu, P.; Wang, P.; Ren, X.; Yuan, H.; Hao, Z.; Xu, C.; Cai, S.; and Ni, D. 2025. AeroGTO: An Efficient Graph-Transformer Operator for Learning Large-Scale Aerodynamics of 3D Vehicle Geometries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 18924–18932.

Liu, Y.; Zhang, Q.; Chen, X.; Xu, C.; Wang, Q.; and Liu, J. 2024. LKFlowNet: A deep neural network based on large kernel convolution for fast and accurate nonlinear fluid-changing prediction. *Physics of Fluids*, 36(9).

Lu, L.; Jin, P.; Pang, G.; Zhang, Z.; and Karniadakis, G. E. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3): 218–229.

Luo, H.; Wu, H.; Zhou, H.; Xing, L.; Di, Y.; Wang, J.; and Long, M. 2025. Transolver++: An Accurate Neural Solver for PDEs on Million-Scale Geometries. *arXiv preprint arXiv:2502.02414*.

Peng, H.; Pappas, N.; Yogatama, D.; Schwartz, R.; Smith, N. A.; and Kong, L. 2021. Random Feature Attention. In *9th International Conference on Learning Representations, ICLR 2021*.

Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; and Battaglia, P. 2020. Learning mesh-based simulation with graph networks. In *8th International Conference on Learning Representations, ICLR 2020*.

Piomelli, U. 1999. Large-eddy simulation: achievements and challenges. *Progress in aerospace sciences*, 35(4): 335–362.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Serrano, L.; Wang, T. X.; Le Naour, E.; Vittaut, J.-N.; and Gallinari, P. 2024. AROMA: preserving spatial structure for latent PDE modeling with local neural fields. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 13489–13521.

Shen, Z.; Zhang, M.; Zhao, H.; Yi, S.; and Li, H. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 3531–3539.

Tran, A.; Mathews, A. P.; Xie, L.; and Ong, C. S. 2023. Factorized Fourier Neural Operators. In *11th International Conference on Learning Representations, ICLR 2023*.

Umetani, N.; and Bickel, B. 2018. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (TOG)*, 37(4): 1–10.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of the 31th International Conference on Neural Information Processing Systems*, 5998–6008.

Wang, T.; and Wang, C. 2024. Latent neural operator for solving forward and inverse PDE problems. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 33085–33107.

Wen, G.; Li, Z.; Azizzadenesheli, K.; Anandkumar, A.; and Benson, S. M. 2022. U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163: 104180.

Wu, H.; Hu, T.; Luo, H.; Wang, J.; and Long, M. 2023. Solving High-Dimensional PDEs with Latent Spectral Models. In *International Conference on Machine Learning*, 37417–37438. PMLR.

Wu, H.; Luo, H.; Wang, H.; Wang, J.; and Long, M. 2024. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*.

Wu, Z.; Ding, T.; Lu, Y.; Pai, D.; Zhang, J.; Wang, W.; Yu, Y.; Ma, Y.; and Haeffele, B. D. 2025. Token Statistics Transformer: Linear-Time Attention via Variational Rate Reduction. In *13th International Conference on Learning Representations, ICLR 2025*.

Xiao, Z.; Hao, Z.; Lin, B.; Deng, Z.; and Su, H. 2024. Improved operator learning by orthogonal attention. In *Proceedings of the 41st International Conference on Machine Learning*, 54288–54299.

Yang, S.; Wang, B.; Shen, Y.; Panda, R.; and Kim, Y. 2023. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*.