Inducing Point Operator Transformer: A Flexible and Scalable Architecture for Solving PDEs

Seungjun Lee, Taeil Oh

Alsemy, South Korea
seungjun.lee@alsemy.com, taeil.oh@alsemy.com

Abstract

Solving partial differential equations (PDEs) by learning the solution operators has emerged as an attractive alternative to traditional numerical methods. However, implementing such architectures presents two main challenges: flexibility in handling irregular and arbitrary input and output formats and scalability to large discretizations. Most existing architectures are limited by their desired structure or infeasible to scale large inputs and outputs. To address these issues, we introduce an attention-based model called an inducing point operator transformer (IPOT). Inspired by inducing points methods, IPOT is designed to handle any input function and output query while capturing global interactions in a computationally efficient way. By detaching the inputs/outputs discretizations from the processor with a smaller latent bottleneck, IPOT offers flexibility in processing arbitrary discretizations and scales linearly with the size of inputs/outputs. Our experimental results demonstrate that IPOT achieves strong performances with manageable computational complexity on an extensive range of PDE benchmarks and real-world weather forecasting scenarios, compared to state-of-the-art methods. Our code is publicly available at https://github.com/7tl7qns7ch/IPOT.

Introduction

Partial differential equations (PDEs) are widely used for mathematically modeling physical phenomena by representing pairwise interactions between infinitesimal segments. Once formulated, PDEs allow us to analyze and predict the physical system, making them essential tools in various scientific fields (Strauss 2007). However, formulating accurate PDEs can be a daunting task without domain expertise where there remain numerous unknown processes for many complex systems. Moreover, traditional numerical methods for solving PDEs can require significant computational costs and may sometimes be intractable. In recent years, data-driven approaches have emerged as an alternative to the conventional procedures for solving PDEs, since they provide much faster predictions and only require observational data. In particular, operator learning, learning mapping between infinite-dimensional function spaces, generalizes well to unseen system inputs with their own flexibility in a discretization-invariant way (Lu et al. 2021, 2022; Kovachki et al. 2021; Li et al. 2020a,b, 2021b).

Observational data in solving PDEs often comes in much more diverse measurement formats, making it challenging to handle. These formats may include sparse and irregular measurements, different input and output domains, and complex geometries due to environmental conditions (Belbute-Peres, Economon, and Kolter 2020; Lienen and Günnemann 2022; Li et al. 2022; Tran et al. 2023). In addition, adaptive remeshing schemes are often deployed when different regions require different resolutions (Brenner, Scott, and Scott 2008; Huang and Russell 2010). However, most existing operator learners have their own restrictions. For instance, some require fixed discretization for the input function (Lu, Jin, and Karniadakis 2019; Lu et al. 2021), assume that input and output discretizations are the same (Lu et al. 2022; Li et al. 2020b, 2021b; Brandstetter, Worrall, and Welling 2022), assume local connectivity (Li et al. 2020a,b), or have uniform regular grids (Li et al. 2021b; Gupta, Xiao, and Bogdan 2021; Cao 2021). They can be limited when the observations have discrepancies between their own settings and given measurements. In order to be flexible to handle a variety of discretization schemes, the model should impose minimal assumptions on locality or data structure.

Our approach aims to address the challenges of handling arbitrary input and output formats by developing a mesh-agnostic architecture that treats observations as individual elements without problem-specific modifications. As flexible in processing data (Jaegle et al. 2021) and efficient in modeling long-range dependencies (Tsai et al. 2019), Transformer (Vaswani et al. 2017) can be an appropriate starting point for our approach. A core building block of the Transformers, the attention mechanism corresponds to and even outperforms the kernel integral operation of the existing operator learners due to its nature of capturing long-range interactions (Cao 2021; Liu, Xu, and Zhang 2022; Guo, Cao, and Chen 2023). However, their quadratic growth in computational complexity with input size can make it impractical for real-world applications, high-fidelity modeling, or long-term forecasting without imposing problem-specific locality.

To address the issue, we took inspiration from inducing-point methods which aim to reduce the effective number of input data points for computational efficiency (Quiñonero-Candela and Rasmussen 2005; Snelson and Ghahramani 2005; Titsias 2009; Li et al. 2020b), and from the extension of the methods to Transformer by cross-attention with a

small number of learnable queries (Lee et al. 2019; Tang and Ha 2021; Jaegle et al. 2021, 2022; Rastogi et al. 2023). In this paper, we introduce a fully attention-based model called an inducing-point operator transformer (IPOT) to capture long-range interactions and provide flexibility for arbitrary input and output formats with feasible computational complexity. IPOT consists of an encoder-processor-decoder (Sanchez-Gonzalez et al. 2020; Pfaff et al. 2021), where the encoder compresses the input function into a smaller fixed number of latent bottlenecks inspired by inducing-point methods, the processor operates on the latent features, and the decoder produces solutions at any output queries from the latent features. This architecture achieves scalability by separating input and output discretizations from the latent processor. It allows the architecture to avoid quadratic complexity and decouples the depth of the processor from the size of the inputs or outputs. This approach can be used for real-world applications with large, complex systems or long-term forecasting tasks, making it feasible and practical to use.

Finally, we conducted several experiments on the extensive PDE benchmarks and a real-world dataset. Compared to state-of-the-art methods, IPOT achieves competitive performances with feasible computational complexity. It can handle uniform regular grids, irregular grids, real-world weather forecasting, and their variants with arbitrary discretization formats.

Related Works

Operator Learning. Based on a pioneer work (Chen and Chen 1995), deep operator networks (DeepONets) were presented to extend the architectures for operator learning with modern deep networks (Lu, Jin, and Karniadakis 2019; Lu et al. 2021). DeepONets consists of a branch network and a trunk network and can be queried out at any coordinate from the trunk network. However, they require the fixed discretization of the input functions from the branch network (Lu et al. 2022). Another promising framework is neural operators, which consist of several integral operators with parameterized kernels to map between infinite-dimensional functions (Kovachki et al. 2021; Li et al. 2020a,b, 2021b). Message passings on graphs (Li et al. 2020a,b; Pfaff et al. 2021), convolutions in the Fourier domain (Li et al. 2021b), or wavelets domain (Gupta, Xiao, and Bogdan 2021; Tripura and Chakraborty 2023) are used to approximate the kernel integral operations. However, the implemented architectures typically require having the same sampling points for input and output (Lu et al. 2022; Kovachki et al. 2021; Li et al. 2020b, 2021b). In addition, graph-based methods do not converge when problems become complex, and convolutional in the spectral domains methods are limited to the uniform regular grids due to the usage of FFT (Li et al. 2021b). To handle irregular grids, (Li et al. 2022; Tran et al. 2023) apply adaptive coordinate maps before and after FNO to make it extend to irregular meshes. Recently, Transformers have been recognized as flexible and accurate operator learners (Cao 2021). However, their quadratic scaling poses a significant challenge to their practical use. (Cao 2021) removes the softmax normalization and introduces Galerkin-type attention to achieve linear scaling. Although efficient transformers (Liu, Xu, and Zhang 2022; Li, Meidani, and Farimani 2022; Hao et al. 2023) have been proposed with preserving permutation symmetries (Lee 2022), the demand for flexibility and scalability is still not met for practical use in the real world.

Inducing-Point Methods. Inducing-point methods have been commonly used to approximate the Gaussian process through m inducing points, where they involve replacing the entire dataset with smaller subsets that are representative of the data. These inducing points methods have been widely used in regression (Cao et al. 2013), kernel machines (Nguyen, Chen, and Lee 2021), and matrix factorizations (He and Xu 2019). A similar idea is also employed in the existing neural operator (Li et al. 2020b), which is based on graph networks and uses sub-sampled nodes from the original graph as inducing points to reduce the computational complexity of the previous graph neural operator (Li et al. 2020a). However, this approach did not converge for complex problems (Li et al. 2021b). Recently, the methods have been extended to Transformers by incorporating cross-attention with a reduced number of learnable query vectors (Lee et al. 2019; Tang and Ha 2021; Jaegle et al. 2021, 2022; Rastogi et al. 2023).

Preliminaries

Neural Operators

Let us consider a set of N input-output pairs, denoted by $(a_i, u_i)_{i=1}^N$, where $a_i = a_i|_X$ and $u_i = u_i|_Y$ are finite discretizations of the input function $a_i \in \mathcal{A}$ at the set of input points $X = \{x_1, \ldots, x_{n_x}\}$ and output function $u_i \in \mathcal{U}$ at the set of output points $Y = \{y_1, \ldots, y_{n_y}\}$ with the number of discretized points n_x and n_y, respectively. Here, $\mathcal{A}$ and $\mathcal{U}$ are input and output function spaces, respectively. The objective of operator learning is to minimize the empirical loss $E_{a\sim\mu}\left[\mathcal{L}\left(\mathcal{G}_\theta(a), u\right)\right] \approx \frac{1}{N}\sum_{i=1}^N \|u_i - \mathcal{G}_\theta(a_i)\|^2$ to learn a mapping $\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}$. The architectures of the neural operator (Kovachki et al. 2021; Li et al. 2021b), usually consist of three components, namely lifting, iterative updates, and projection, which correspond to the encoder-processor-decoder, respectively.

$$u = \mathcal{G}_\theta(a) = (\mathcal{Q} \circ \mathcal{G}_{L-1} \circ \cdots \circ \mathcal{G}_1 \circ \mathcal{P})(a) \qquad (1)$$

where the lifting (encoder) $\mathcal{P} = \mathcal{G}_{enc}$ and projection (decoder) $\mathcal{Q} = \mathcal{G}_{dec}$ are local transformations that map input features to target dimensional features, implemented by point-wise feed-forward neural networks. The iterative updates $\mathcal{G}_l : v_l \mapsto v_{l+1}, l \in [0, L-1]$ are global transformations that capture the interactions between the elements, implemented by a sequence of following transformations,

$$v_{l+1}(x) = \sigma\Big(\mathcal{W}_l v_l(x) + \left[\mathcal{K}_l(v_l)\right](x)\Big), \quad x \in \Omega, \qquad (2)$$

where σ are nonlinear functions, $\mathcal{W}_l$ are point-wise linear transformations, $\mathcal{K}_l$ are kernel integral operations on $v_l(x)$. The existing implementations of the neural operators typically use the same discretizations for the input and output (Lu et al. 2022; Li et al. 2020b, 2021b; Brandstetter, Worrall, and Welling 2022), which leads to the predicted outputs only being queried at the input meshes. Our core idea is to replace the encoder $\mathcal{G}_{enc}$ and decoder $\mathcal{G}_{dec}$ to accommodate arbitrary size and structure of the input functions and output queries.

Kernel Integral Operation & Attention Mechanism

The kernel integral operations are generally implemented by integration of input values weighted by kernel values κ representing the pairwise interactions between the elements on input domain $x \in \Omega_x$ and output domain $y \in \Omega_y$,

$$[\mathcal{K}(v)](y) = \int_{\Omega_x} \kappa(y,x)v(x)dx, \quad (x,y) \in \Omega_x \times \Omega_y, \tag{3}$$

where the kernel κ is defined on $\Omega_y \times \Omega_x$. The transform $\mathcal{K}$ can be interpreted as mapping a function $v(x)$ defined on a domain $x \in \Omega_x$ to the function $[\mathcal{K}(v)](y)$ defined on a domain $y \in \Omega_y$. Recently, it has been proved that the kernel integral operation can be successfully approximated by the attention mechanism of Transformers both theoretically and empirically (Kovachki et al. 2021; Cao 2021; Guibas et al. 2021; Pathak et al. 2022). Intuitively, let input vectors $X \in \mathbb{R}^{n_x \times d_x}$ and query vectors $Y \in \mathbb{R}^{n_y \times d_y}$, then the attention can be expressed as

$$\textit{Attn}(Y,X,X) = \sigma(QK^T)V \approx \int_{\Omega_x} (q(Y) \cdot k(x))v(x)dx, \tag{4}$$

where $Q = YW^q \in \mathbb{R}^{n_y \times d}$, $K = XW^k \in \mathbb{R}^{n_x \times d}$, $V = XW^v \in \mathbb{R}^{n_x \times d}$, and σ are the query, key, value matrices, and softmax function, respectively. W^q, W^k, and W^v are learnable weight matrices that operate in a pointwise way which makes the attention module not depend on the input and output discretizations. A brief derivation of Equation 4 can be found in the Appendix. The weighted sum of V with the attention matrix $\sigma(QK^T)$ can be interpreted as the kernel integral operation in which the parameterized kernel is approximated by the attention matrix (Tsai et al. 2019; Cao 2021; Xiong et al. 2021; Choromanski et al. 2021). This attention is also known as cross-attention, where the input vectors are projected to query embedding space by the attention, $\textit{Attn}(Y,X,X)$. However, the computational complexity of the mechanism is proportional to $O(n_x n_y d)$, and it can cause quadratic complexity $O(n^2 d)$ for large n_x and n_y ($n_x, n_y \approx n$).

Approach

The goal of this work is to develop a mesh-agnostic architecture that can handle any input and output discretizations with reduced computational complexity. We allow for different, irregular, and arbitrary input and output formats.

Handling Arbitrary Input Function and Output Queries. The output solution evaluated at output query y can be expressed as $u(y) = [\mathcal{G}_\theta(a)](y)$ which can be viewed as the operating $\mathcal{G} : \mathcal{A} \times Y \to \mathcal{U}$ an input function $a \in \mathcal{A}$ at the corresponding output query $y \in Y$. We treat the representations of discretized input $a = a|_X \in \mathbb{R}^{n_x \times d_a}$, output $u = u|_Y \in \mathbb{R}^{n_y \times d_u}$ and corresponding output queries $Y \in \mathbb{R}^{n_y \times d_y}$ as sets of individual elements. In practice, they are represented by flattened arrays, without using structured bias to avoid our model bias toward specific data structures and flexibly process any discretization formats. The significant modifications from the existing neural operators (Equation 1) are mostly in the encoder and decoder for detaching the dependence of input and output discretizations by

$$Z_0 = \mathcal{G}_{enc}(a), \quad Z_{l+1} = \mathcal{G}_l(Z_l), \quad \tilde{u} = \mathcal{G}_{dec}(Y, Z_L). \tag{5}$$

where the encoder projects the input function into latent space, the sequence of the processing is operated in latent space, and then the decoder predicts the output solutions from the latent representations at the corresponding output queries.

Positional Encoding. In order to compensate for the positional information at each function value, we follow a common way of existing neural operators (Kovachki et al. 2021; Li et al. 2021b), which involves concatenating the position coordinates with the corresponding function values to form the input representation, $a = \{(x_1, a(x_1)), ..., (x_{n_x}, a(x_{n_x}))\}$. Additionally, instead of using raw position coordinates for both input $X = \{x_1, ..., x_{n_x}\}$ and outputs $Y = \{y_1, ..., y_{n_y}\}$, we concatenate additional Fourier embeddings for the position coordinates. This technique, which is commonly used to enrich positional information in neural networks (Vaswani et al. 2017; Mildenhall et al. 2020; Tancik et al. 2020; Sitzmann et al. 2020) involves exploiting sine and cosine functions with frequencies spanning from minimum to maximum frequencies, thereby covering the sampling rates for the corresponding dimensions.

Inducing Point Operator Transformer (IPOT)

We build our model with an attention-based architecture consisting of an encoder-processor-decoder, called an inducing point operator transformer (IPOT) to reduce the computational complexity of attention mechanisms (Figure 1). The key feature of IPOT is the use of a reduced number of inducing points ($n_z \ll n_x, n_y$) (Lee et al. 2019; Jaegle et al. 2021; Rastogi et al. 2023). This is typically achieved by employing n_z learnable query vectors into the encoder, allowing most of the attention mechanisms to be computed in the smaller latent space instead of the larger observational space. This results in a significant reduction in computational complexity. The encoder encodes the input function a to the fixed smaller number of latent feature vectors (discretization number: $n_x \mapsto n_z$), the processor processes the pairwise interactions between elements of the latent features vectors (discretization number: $n_z \mapsto n_z$), and the decoder decodes the latent features to output solutions at a set of output queries Y (discretization number: $n_z \mapsto n_y$).

Attention Blocks. The exact form of the attention blocks $\textit{Attention}(Y,X,X)$ used in the following sections are described in Appendix. The nonlinearity σ, pointwise linear transformations $\mathcal{W}_l$, and kernel integral operations $\mathcal{K}_l$ in Equation 2 are approximated by feed-forward neural networks, residual connections, and the attention modules, which are common block forms of Transformer-like architectures (Vaswani et al. 2017). In addition, layer normalization (Ba, Kiros, and Hinton 2016) is used to normalize the query, key, and values inputs, and multi-headed extensions of the attention blocks are employed to improve the model's performance.

Encoder. We use a cross-attention block as the encoder to encode inputs a with the size n_x to a smaller fixed number n_z of learnable query vectors (inducing points) $Z_q \in \mathbb{R}^{n_z \times d_z}$ (typically $n_z \ll n_x$). Then the result of the block is

$$Z_0 = \mathcal{G}_{enc}(a) = \textit{Attention}(Z_q, a, a) \in \mathbb{R}^{n_z \times d_z}, \tag{6}$$

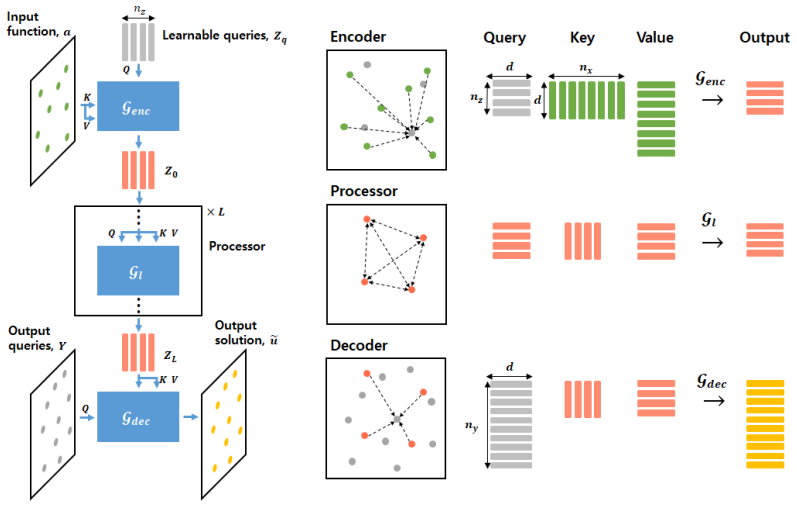

Figure 1: Inducing-Point Operator Transformer (IPOT) uses a smaller number of inducing points, enabling it to flexibly handle any discretization formats of input and output, and significantly reduce the computational costs. IPOT encodes input function discretized on $X = \{x_1, ..., x_{n_x}\}$ to a fixed smaller size n_z of learnable query vectors, and decoding them to output discretized on $Y = \{y_1, ..., y_{n_y}\}$. The number of size is varied as n_x (arbitrary) $\rightarrow$ n_z (fixed) $\rightarrow$ n_y (arbitrary).

where each learnable query vector of Z_q is randomly initialized by a normal distribution and learned along the architectures. The encoder maps the input domain to a latent domain consisting of n_z inducing points based on the correlations between input discretizations and inducing points. The computational complexity of the encoder is proportional to $O(n_x n_z d)$ which scales linearly with the input size n_x. Furthermore, the use of inducing points reduces the computational complexity of the subsequent attention blocks.

Processor. We use a series of self-attention blocks as the processor each of which takes $Z_l \in \mathbb{R}^{n_z \times d_z}$ as the input of the query, key, and value components. Then the output of each self-attention block for $l \in [0, L-1]$ is

$$Z_{l+1} = \mathcal{G}_l(Z_l) = \mathit{Attention}(Z_l, Z_l, Z_l) \in \mathbb{R}^{n_z \times d_z}, \quad (7)$$

which captures global interactions of inducing points in the latent space. Since the processor is decoupled from the input and output discretizations, IPOT is not only applicable to any input and output discretization formats but also significantly reduces computational costs. Processing in the latent space rather than in observational space reduces the costs in the processor to $O(Ln_z^2 d)$ from $O(Ln_x^2 d)$ for the original Transformers. This decouples the depth of the processor L from the size of input n_x or output n_y, allowing the construction of deep architectures or long-term forecasting models with large L.

Decoder. We use a cross-attention block as the decoder to decode the latent vectors from the processor $Z_L \in \mathbb{R}^{n_z \times d_z}$ at output queries $Y \in \mathbb{R}^{n_y \times d}$. Then, the output solutions predicted by IPOT at the corresponding output queries are

$$\tilde{u} = \mathcal{G}_{dec}(Y, Z_L) = \mathit{Attention}(Y, Z_L, Z_L) \in \mathbb{R}^{n_y \times d_u}. \quad (8)$$

The decoder maps the latent domain to the output domain based on the correlations between n_z inducing points and output queries Y. Since the result from the processor is independent of the discretization format of input function a, the entire model is also applicable to arbitrary input discretization and can be queried out at arbitrary output queries. In addition, the computational cost of the decoder is proportional to $O(n_z n_y d)$ which also scales linearly with the output size n_y.

Time-Stepping Through Latent Space

We model the time-dependent PDEs as an autoregressive process. The state of the system at time $t + dt$ is described as $u_{t+dt} = (\mathcal{G}_{dec} \circ \mathcal{G}_{dt} \circ \mathcal{G}_{enc})(u_t)$, where $\mathcal{G}_{enc}$ is an encoder that maps the current observational state to the latent state, $\mathcal{G}_{dec}$ is a decoder that maps the latent state back to the observational state, and $\mathcal{G}_{dt}$ is a processor that steps forward in time by dt implemented by a series of self-attention blocks on the latent states. We assume that the processor $\mathcal{G}_{dt}$ is independent of t (Sanchez-Gonzalez et al. 2020; Pfaff et al. 2021; Li et al. 2021a; Li, Meidani, and Farimani 2022). When setting the time step as $dt = 1$, the predicted trajectory of the system is obtained by the following recurrent relation at each time $t \in [0, T-1]$

$$Z_0 = \mathcal{G}_{enc}(u_0), \quad Z_{t+1} = \mathcal{G}_{dt}(Z_t), \quad \tilde{u}_{t+1} = \mathcal{G}_{dec}(Y, Z_{t+1}). \quad (9)$$

where u_0 is the initial state, Y are output queries, and identical processor $\mathcal{G}_{dt}$ is applied at each time step. Throughout the entire trajectory, by encoding the initial state into the latent space, we can significantly reduce the computational costs of subsequent processing compared to processing in the observational space.

Computational Complexity

The overall computational complexity of IPOT is proportional to $O(n_x n_z d + Ln_z^2 d + n_z n_y d)$ which scales linearly to n_x and n_y, and the depth of architecture L is decoupled from the input and output size. When $n_x, n_y \approx n$ and $L \gg 1$, the complexity becomes $O(Ln_z^2 d + 2nn_z d) \approx O(Ln_z^2 d)$,

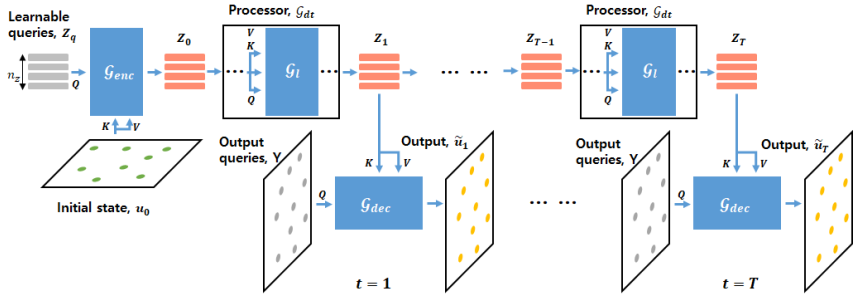

Figure 2: The state of the system at time T is denoted as $u_T = (\mathcal{G}_{dec} \circ [\mathcal{G}_{dt} \circ \cdots \circ \mathcal{G}_{dt}] \circ \mathcal{G}_{enc})(u_0)$, where $\mathcal{G}_{enc}$ is an encoder that maps the initial state u_0 to the latent state, $\mathcal{G}_{dec}$ is a decoder that maps the latent state back to the observational state, and $\mathcal{G}_{dt}$ is a processor that steps forward in time by dt implemented by a series of self-attention blocks on latent states.

which is significantly lower than that of standard Transformer $O(Ln^2d)$ (Vaswani et al. 2017), which has quadratic scaling and is coupled with L. Existing operator learners including Fourier neural operator $O(Ln \log n\, d)$ (Li et al. 2021b) or linear Transformers $O(Lnd^2)$ (Cao 2021; Li, Meidani, and Farimani 2022) have sub-quadratic scaling with the size n, but they are coupled with the depth L. The decoupling of the depth L from the size n makes it practical to construct deep architectures or apply them to long-term forecasting that requires a large L.

Experiments

We conduct experiments on several benchmark datasets, including PDEs on regular grids (Li et al. 2021b), irregular grids (Li et al. 2022; Tran et al. 2023; Yin et al. 2022), and real-world data from the ERA5 reanalysis dataset (Hersbach et al. 2020) to investigate the flexibility and scalability of our model through various downstream tasks. Details of the equations and the problems are described in the Appendix. Results for experiments already discussed on baselines were obtained from the related literature, and results for the extended tasks that have not been discussed before have been reproduced from their original codes. We provide some illustrations of the predictions of IPOT for the benchmarks on regular and irregular grids (Figure 3) and long-term dynamics (Figure 4). The implementation details and additional results can be found in the Appendix.

Baselines. We took several representative architectures as baselines including deep operator network (DeepONet) (Lu, Jin, and Karniadakis 2019), the mesh-based learner with graph neural networks (Meshgraphnet) (Pfaff et al. 2021), Fourier neural operator (FNO) (Li et al. 2021b), Factorized-Fourier neural operator (FFNO) (Tran et al. 2023), and operator Transformer (OFormer) (Li, Meidani, and Farimani 2022).

Evaluation Metric. We use relative L_2 error for the objective functions and evaluation metrics for test errors, where we follow the convention of related literature (Kovachki et al. 2021; Li et al. 2021b). When N is the dataset size of input-output pairs $(a_i, u_i)_{i=1}^N$, the relative L_2 error is defined as

$$E_{a\sim\mu}[\mathcal{L}(\mathcal{G}_\theta(a), u)] = \frac{1}{N}\sum_{i=1}^{N}\frac{\|u - \mathcal{G}_\theta(a)\|_2}{\|u_i\|_2}. \qquad (10)$$

Problems of PDEs Solved on Regular Grids

We conducted experiments on benchmark problems, where the PDEs for Darcy flow, and Navier-Stokes equation were solved on regular grids from (Li et al. 2021b). Shown in Table 1, IPOT consistently demonstrates competitive or strong performances with efficient computational resources, particularly in the case of Navier-Stokes benchmarks that involve higher complexity and time-dependence, resulting in larger n and requiring models with long sequences L of processors or iterative updates. These achievements are obtained without using any problem-specific layers such as pooling layers or convolutional filters, which are only compatible the regular grid structures. Consequently, IPOT offers greater flexibility in handling arbitrary inputs and output formats beyond the regular grids.

Problems of PDEs Solved on Irregular Grids

We conducted experiments on various problems, where the PDEs were solved on non-uniform and irregular grids. These problems include predicting flows around complex geometries (airfoil), estimating stresses on irregularly sampled point clouds (elasticity), and predicting displacements given initial boundary conditions for plastic forging problem (plasticity) as described in (Tran et al. 2023). The experimental results presented in Table 1 also demonstrate that our IPOT model demonstrates competitive or stronger performances with efficient computational resources, as illustrated in Figure 3.

Application to Real-World Data

Unlike the previous problems, we conducted experiments on a subset of the ERA5 reanalysis dataset from the European Centre for Medium-Range Weather Forecasts (Hersbach et al. 2020), where the governing PDEs are unknown. While the ERA5 database includes extensive hourly and monthly measurements for several parameters at various pressure levels, our focus is specifically on predicting the daily temperature at 2m from the surface T_{2m}. IPOT achieves superior accuracy compared to the baselines while maintaining comparable both time and memory costs. Using the decoupled n_z inducing points from the observational space, our approach mitigates the computational burden associated with large n and L, making it beneficial for complex and long-term forecasting tasks.

Model	Dataset	Params	Runtime	Memory	Error	Dataset	Params	Runtime	Memory	Error
DeepONet	Darcy	0.42M	2.73	2.40	4.61e-2	Airfoil	0.42M	3.13	2.75	3.85e–2
MeshGraphNet		0.21M	9.51	5.57	9.67e–2		0.22M	10.92	6.38	5.57e–2
FNO		1.19M	**1.88**	1.96	1.09e–2		1.19M	2.63	2.73	4.21e–2
FFNO		0.41M	3.36	1.99	**7.70e–3**		0.41M	4.71	2.78	**7.80e–3**
OFormer		1.28M	3.63	5.71	1.26e–2		1.28M	4.17	7.97	1.83e–2
IPOT (ours)		**0.15M**	2.70	**1.82**	1.73e–2		**0.12M**	**2.15**	**2.10**	8.79e–3
DeepONet	Navier	-	-	-	-	Elasticity	1.03M	3.72	1.18	9.65e–2
MeshGraphNet		0.29M	137.17	6.15	1.29e–1		0.46M	7.36	4.04	4.18e–2
FNO		0.93M	53.73	3.09	1.28e–2		0.57M	**1.04**	1.68	5.08e–2
FFNO		0.27M	53.82	3.40	1.32e–2		0.55M	2.42	2.08	2.63e–2
OFormer		1.85M	70.15	9.90	1.04e–2		2.56M	5.58	2.98	1.83e–2
IPOT (ours)		**0.12M**	**21.05**	**2.08**	**8.85e–3**		**0.12M**	1.99	**1.13**	**1.56e–2**
DeepONet	ERA5	-	-	-	-	Plasticity	-	-	-	-
MeshGraphNet		2.07M	51.75	18.45	7.16e–2		-	-	-	-
FNO		2.37M	**9.23**	13.04	1.21e–2		1.85M	10.40	16.81	5.08e–2
FFNO		1.12M	14.39	17.06	7.25e–3		0.57M	66.47	16.86	4.70e–3
OFormer		1.85M	71.18	10.90	1.15e–2		0.49M	28.43	14.11	1.83e–2
IPOT (ours)		**0.51M**	9.83	**10.58**	**6.64e–3**		**0.13M**	**10.14**	**5.35**	**3.25e–3**

Table 1: Performance and efficiency comparisons with baselines across various datasets. The efficiencies are compared in terms of the number of parameters, time spent per epoch (seconds), and CUDA memory consumption (GB). The missing entries occur when the methods are not able to handle the datasets or when encountering convergence issues.

Model	Different resolutions	Partial observed
	res= 4° / 1° / 0.25°	Masked land
FNO	1.30e–2 / 1.23e–2 / 1.24e–2	3.10e–2
OFormer	3.66e–2 / 1.65e–2 / 1.86e–2	4.37e–2
IPOT	**8.96e–3 / 7.78e–3 / 8.66e–3**	**2.83e–2**

Table 2: Results for comparing the generalization ability on discretizations. The models are evaluated on the task of different resolutions and masked inputs for the ERA5 dataset.

Model	n / n_z	Time	Error	Comp
OFormer	16.2K / 16.2K	71.18	1.15e–2	$\mathcal{O}(nd^2)$
IPOT w.o ip	16.2K / 16.2K	$\gg 100$	–	$\mathcal{O}(n^2d)$
IPOT (64)	16.2K / 64	7.44	1.45e–2	$\mathcal{O}(n_z^2d)$
IPOT (128)	16.2K / 128	7.61	1.30e–2	$\mathcal{O}(n_z^2d)$
IPOT (256)	16.2K / 256	7.91	6.87e–3	$\mathcal{O}(n_z^2d)$
IPOT (512)	16.2K / 512	9.83	6.44e–3	$\mathcal{O}(n_z^2d)$

Table 3: Performance of IPOT with varying the number of inducing points with respective of runtime (Time), error, and complexity (Comp). IPOT w.o ip denotes that IPOT without inducing points which emulates the standard Transformer.

Generalization Ability on Discretizations. Furthermore, we explore the model's generalization ability to different resolutions and to make predictions from partially observed inputs. The evaluation of different resolutions is motivated by the scenario where observations are collected at varying resolutions. The evaluation with masked inputs is motivated by situations when observations are only available for the sea surface while data from land areas are unavailable as shown in the bottom right of Figure 4. We employ bilinear interpolation methods to obtain interpolated input values corresponding to land coordinates and combine them with the masked inputs for some comparison models that necessitate a consistent grid structure for both input and output data. As shown in Table 2, IPOT consistently outperforms all the baselines in all scenarios, demonstrating stable and strong performance. These results highlight the exceptional flexibility and accuracy of IPOT, showcasing its remarkable generalization capability.

Ablation Study

Effect of the Number of Inducing Points. We conducted experiments on ERA5 data with varying numbers of latent query vectors, ranging from 64 to 512, to evaluate the effect of the number of inducing points, and to emulate the quadratic complexity in attention blocks of the standard Transformer (Vaswani et al. 2017), we also consider IPOT without inducing points. In this variant, we use self-attention where the observational input is injected into query, key, and value components instead of using cross-attention with learnable queries in the encoder. Table 3 demonstrates that increasing the number of latent query vectors improves the performance of IPOT. Notably, when the number is over 256, IPOT sufficiently outperforms other baselines. However, when the number of inducing points is too small, IPOT exhibits poor performance. Additionally, the quadratic complexity of attention in a standard Transformer hinders convergence and results in excessive computational time.

Computational Complexity on Different Resolutions. We compared the computational complexity of different models on the ERA5 data at different resolutions, as shown in Figure 5. The time complexities were assessed by measuring the inference time in seconds for processing observational data during testing, and the memory complexities were evaluated by recording CUDA memory allocation. It is observed that

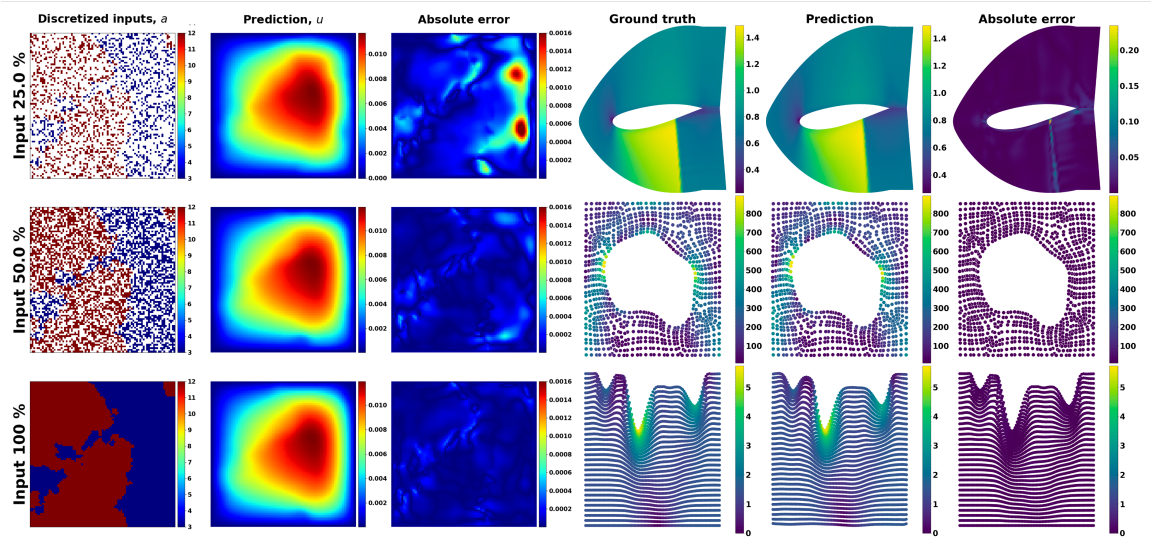

Figure 3: Predictions of IPOT on the problems of Darcy flow (left), airfoil (top right), elasticity (middle right), and plasticity (bottom right). In the case of Darcy flow, the partially observed inputs are randomly subsampled with the ratios of $\{100, 50, 25\}\%$.

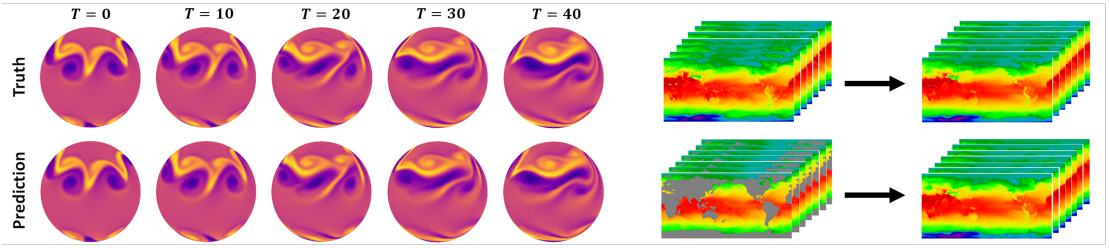

Figure 4: Long-term predictions of IPOT on spherical manifolds for the shallow-water equations (left), and on real-world weather forecasting when the inputs are spatially fully given (top right), and partially given (bottom right).

IPOT without latent inducing points, which serves as a surrogate for a standard Transformer with quadratic complexity in self-attention, does not scale well in terms of both time and memory costs. IPOT with 256 or 512 inducing points scale effectively up to $n = 10^6$, making them suitable for handling large-scale observational data due to their efficiency in both time and memory complexity.

Conclusions

In this work, we raise potential issues on existing operator learning models for solving PDEs in terms of flexibility in handling irregular and arbitrary input and output discretizations, as well as the computational complexity for complex and long-term forecasting real-world problems. To address these issues, we propose IPOT, an attention-based architecture capable of handling arbitrary input and output discretizations while maintaining feasible computational complexity. This is achieved by using a reduced number of inducing points in the latent space, effectively addressing the quadratic scaling issue in the attention mechanism. Our proposed model demonstrates superior performance on a wide range of benchmark problems, including PDEs solved on both regular and irregular grids. Furthermore, we show that IPOT outperforms other baseline models on real-world data with competitive efficiency. Moreover, IPOT exhibits strong generalization ability, providing consistent performance in challenging real-life scenarios, while other models often suffer from significant performance degradation or require additional pre-processing for compatibility with the data structure.

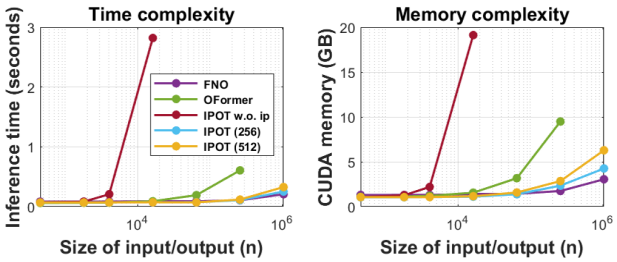

Figure 5: Complexity comparisons on different resolutions. We compare the different models in terms of inference time (left) and CUDA memory usage (right) with different sizes of input/output.

References

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

Belbute-Peres, F. D. A.; Economon, T.; and Kolter, Z. 2020. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, 2402–2411. PMLR.

Brandstetter, J.; Worrall, D. E.; and Welling, M. 2022. Message Passing Neural PDE Solvers. In *International Conference on Learning Representations*.

Brenner, S. C.; Scott, L. R.; and Scott, L. R. 2008. *The mathematical theory of finite element methods*, volume 3. Springer.

Cao, S. 2021. Choose a transformer: Fourier or galerkin. *Advances in Neural Information Processing Systems*, 34.

Cao, Y.; Brubaker, M. A.; Fleet, D. J.; and Hertzmann, A. 2013. Efficient optimization for sparse Gaussian process regression. *Advances in Neural Information Processing Systems*, 26.

Chen, T.; and Chen, H. 1995. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4): 911–917.

Choromanski, K. M.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J. Q.; Mohiuddin, A.; Kaiser, L.; Belanger, D. B.; Colwell, L. J.; and Weller, A. 2021. Rethinking Attention with Performers. In *International Conference on Learning Representations*.

Guibas, J.; Mardani, M.; Li, Z.; Tao, A.; Anandkumar, A.; and Catanzaro, B. 2021. Adaptive Fourier Neural Operators: Efficient Token Mixers for Transformers. *arXiv preprint arXiv:2111.13587*.

Guo, R.; Cao, S.; and Chen, L. 2023. Transformer Meets Boundary Value Inverse Problems. In *The Eleventh International Conference on Learning Representations*.

Gupta, G.; Xiao, X.; and Bogdan, P. 2021. Multiwavelet-based Operator Learning for Differential Equations. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Hao, Z.; Ying, C.; Wang, Z.; Su, H.; Dong, Y.; Liu, S.; Cheng, Z.; Zhu, J.; and Song, J. 2023. GNOT: A General Neural Operator Transformer for Operator Learning. *arXiv preprint arXiv:2302.14376*.

He, J.; and Xu, J. 2019. MgNet: A unified framework of multigrid and convolutional neural network. *Science china mathematics*, 62: 1331–1354.

Hersbach, H.; Bell, B.; Berrisford, P.; Hirahara, S.; Horányi, A.; Muñoz-Sabater, J.; Nicolas, J.; Peubey, C.; Radu, R.; Schepers, D.; et al. 2020. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730): 1999–2049.

Huang, W.; and Russell, R. D. 2010. *Adaptive moving mesh methods*, volume 174. Springer Science & Business Media.

Jaegle, A.; Borgeaud, S.; Alayrac, J.-B.; Doersch, C.; Ionescu, C.; Ding, D.; Koppula, S.; Zoran, D.; Brock, A.; Shelhamer, E.; Henaff, O. J.; Botvinick, M.; Zisserman, A.; Vinyals, O.; and Carreira, J. 2022. Perceiver IO: A General Architecture for Structured Inputs & Outputs. In *International Conference on Learning Representations*.

Jaegle, A.; Gimeno, F.; Brock, A.; Vinyals, O.; Zisserman, A.; and Carreira, J. 2021. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, 4651–4664. PMLR.

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

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, S. 2022. Mesh-Independent Operator Learning for Partial Differential Equations. In *ICML 2022 2nd AI for Science Workshop*.

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

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020a. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2021a. Markov neural operators for learning chaotic systems. *arXiv preprint arXiv:2106.06898*.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Stuart, A.; Bhattacharya, K.; and Anandkumar, A. 2020b. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33: 6755–6766.

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

Li, Z.; Meidani, K.; and Farimani, A. B. 2022. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*.

Lienen, M.; and Günnemann, S. 2022. Learning the Dynamics of Physical Systems from Sparse Observations with Finite Element Networks. In *International Conference on Learning Representations*.

Liu, X.; Xu, B.; and Zhang, L. 2022. HT-Net: Hierarchical Transformer based Operator Learning Model for Multiscale PDEs. *arXiv preprint arXiv:2210.10890*.

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

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.

Lu, L.; Meng, X.; Cai, S.; Mao, Z.; Goswami, S.; Zhang, Z.; and Karniadakis, G. E. 2022. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393: 114778.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 405–421. Springer.

Nguyen, T.; Chen, Z.; and Lee, J. 2021. Dataset Meta-Learning from Kernel Ridge-Regression. In *International Conference on Learning Representations*.

Pathak, J.; Subramanian, S.; Harrington, P.; Raja, S.; Chattopadhyay, A.; Mardani, M.; Kurth, T.; Hall, D.; Li, Z.; Azizzadenesheli, K.; et al. 2022. FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators. *arXiv preprint arXiv:2202.11214*.

Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; and Battaglia, P. 2021. Learning Mesh-Based Simulation with Graph Networks. In *International Conference on Learning Representations*.

Quiñonero-Candela, J.; and Rasmussen, C. E. 2005. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6(65): 1939–1959.

Rastogi, R.; Schiff, Y.; Hacohen, A.; Li, Z.; Lee, I.; Deng, Y.; Sabuncu, M. R.; and Kuleshov, V. 2023. Semi-Parametric Inducing Point Networks and Neural Processes. In *The Eleventh International Conference on Learning Representations*.

Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; and Battaglia, P. 2020. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, 8459–8468. PMLR.

Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; and Wetzstein, G. 2020. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33.

Snelson, E.; and Ghahramani, Z. 2005. Sparse Gaussian Processes using Pseudo-inputs. In Weiss, Y.; Schölkopf, B.; and Platt, J., eds., *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Strauss, W. A. 2007. *Partial differential equations: An introduction*. John Wiley & Sons.

Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; and Ng, R. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33.

Tang, Y.; and Ha, D. 2021. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. *Advances in Neural Information Processing Systems*, 34.

Titsias, M. 2009. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In van Dyk, D.; and Welling, M., eds., *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, 567–574. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR.

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

Tripura, T.; and Chakraborty, S. 2023. Wavelet Neural Operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404: 115783.

Tsai, Y.-H. H.; Bai, S.; Yamada, M.; Morency, L.-P.; and Salakhutdinov, R. 2019. Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel. *arXiv preprint arXiv:1908.11775*.

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

Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; and Singh, V. 2021. Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 35, 14138. NIH Public Access.

Yin, Y.; Kirchmeyer, M.; Franceschi, J.-Y.; Rakotomamonjy, A.; and Gallinari, P. 2022. Continuous PDE Dynamics Forecasting with Implicit Neural Representations. *arXiv preprint arXiv:2209.14855*.