Choose a Transformer: Fourier or Galerkin

Shuhao Cao
Department of Mathematics and Statistics
Washington University in St. Louis
s.cao@wustl.edu

Abstract

In this paper, we apply the self-attention from the state-of-the-art Transformer in *Attention Is All You Need* [87] for the first time to a data-driven operator learning problem related to partial differential equations. An effort is put together to explain the heuristics of, and to improve the efficacy of the attention mechanism. By employing the operator approximation theory in Hilbert spaces, it is demonstrated for the first time that the softmax normalization in the scaled dot-product attention is sufficient but not necessary. Without softmax, the approximation capacity of a linearized Transformer variant can be proved to be comparable to a Petrov-Galerkin projection layer-wise, and the estimate is independent with respect to the sequence length. A new layer normalization scheme mimicking the Petrov-Galerkin projection is proposed to allow a scaling to propagate through attention layers, which helps the model achieve remarkable accuracy in operator learning tasks with unnormalized data. Finally, we present three operator learning experiments, including the viscid Burgers' equation, an interface Darcy flow, and an inverse interface coefficient identification problem. The newly proposed simple attention-based operator learner, Galerkin Transformer, shows significant improvements in both training cost and evaluation accuracy over its softmax-normalized counterparts.

1 Introduction

Partial differential equations (PDEs) arise from almost every multiphysics and biological systems, from the interaction of atoms to the merge of galaxies, from the formation of cells to the change of climate. Scientists and engineers have been working on approximating the governing PDEs of these physical systems for centuries. The emergence of the computer-aided simulation facilitates a cost-friendly way to study these challenging problems. Traditional methods, such as finite element/difference [18, 20], spectral methods [11], etc., leverage a discrete structure to reduce an infinite dimensional operator map to a finite dimensional approximation problem. Meanwhile, in the field practice of many scientific disciplines, substantial data for PDE-governed phenomena available on discrete grids enable modern black-box models like Physics-Informed Neural Network (PINN) [70, 61, 47] to exploit measurements on collocation points to approximate PDE solutions.

Nonetheless, for traditional methods or data-driven function learners such as PINN, given a PDE, the focus is to approximate a single instance, for example, solving for an approximated solution for one coefficient with a fixed boundary condition. A slight change to this coefficient invokes a potentially expensive re-training of any data-driven function learners. In contrast, an operator learner aims to learn a map between infinite-dimensional function spaces, which is much more difficult yet rewarding. A well-trained operator learner can evaluate many instances without re-training or collocation points, thus saving valuable resources, and poses itself as a more efficient approach in the long run. Data-driven resolution-invariant operator learning is a booming new research direction [59, 5, 55, 63, 89, 56, 60, 90, 35, 73]. The pioneering model, DeepONet [59], attributes architecturally to a universal approximation theorem for operators [16]. Fourier Neural Operator (FNO) [56] notably

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

shows an awing state-of-the-art performance outclassing previous methods by orders of magnitudes in certain benchmarks.

Under a supervised setting, an operator learner is trained with the operator's input functions and their responses to the inputs as targets. Since both functions are sampled at discrete grid points, this is a special case of a `seq2seq` problem [80]. The current state-of-the-art `seq2seq` model is the Transformer first introduced in [87]. As the heart and soul of the Transformer, the scaled dot-product attention mechanism is capable of unearthing the hidden structure of an operator by capturing long-range interactions. Inspired by many insightful pioneering work in Transformers [48, 17, 74, 83, 95, 96, 94, 58, 75, 65], we have modified the attention mechanism minimally yet in a mathematically profound manner to better serve the purpose of operator learning.

Among our new Hilbert space-inspired changes of the scaled dot-product attention, the first and foremost change is: no softmax, or the approximation thereof. The softmax normalization succeeding the matrix multiplication convexifies the weights for combining different positions' latent representations, which is regarded as an indispensable in the positive kernel interpretation of the scaled dot-product attention [83]. However, softmax acts globally in the sequence length dimension for each row of the attention matrix, and further adds to the quadratic complexity of the attention in the classic Transformer. Theory-wise, instead of viewing each row as each word's latent representation in the Natural Language Processing (NLP) tradition, the columns of the query/keys/values are seen as functions in Hilbert spaces sampled on discretized grids. Thus, taking the softmax away allows us to verify a discrete Ladyzhenskaya–Babuška–Brezzi (LBB) condition, which amounts to the proof of the newly proposed Galerkin-type attention operator to have an approximation capacity independent of the sequence length (Theorem 4.3, see also Appendix D).

The removal of the softmax saves valuable computational resources. Numerically, the softmax-free models greatly outperform the ones with the softmax in terms of training FLOP and memory consumption (Section 5). Yet in an ablation study, without the presence of the softmax, the training becomes unstable (Table 9). To remedy this, a new Galerkin projection-type layer normalization scheme is proposed to act as a cheap alternative to the normalizations explicitly derived in the proof of the Petrov-Galerkin interpretation (equation (39)). Since a learnable scaling can now be propagated through the encoder layers, the attention-based operator learner with this new layer normalization scheme exhibits better comprehension of certain physical properties associated with the PDEs such as the energy decay. Combining with other mathematics-inspired tricks including a diagonally dominant rescaled initialization for the projection matrices and a layer-wise enrichment of the positional encodings, the evaluation accuracy is boosted by a significant amount.

Main contributions. The main contributions of this work are summarized as follows.

- **Attention without softmax.** We propose a new simple self-attention operator and its linear variant without the softmax normalization. Two new interpretations are offered, together with the approximation capacity of the linear variant proved comparable to a Petrov-Galerkin projection.
- **Operator learner for PDEs.** We combine the newly proposed attention operators with the current best state-of-the-art operator learner Fourier Neural Operator (FNO) [56] to significantly improve its evaluation accuracy in PDE solution operator learning benchmark problems. Moreover, the new model is capable of recovering coefficients based on noisy measurements that traditional methods or FNO cannot accomplish.
- **Experimental results.** We present three benchmark problems to show that operator learners using the newly proposed attentions are superior in computational/memory efficiency, as well as in accuracy versus those with the conventional softmax normalization. The PyTorch codes to reproduce our results are available as an open-source software. [1]

2 Related Works

Operator learners related to PDEs. In [4, 5], certain kernel forms of the solution operator of parametric PDEs are approximated using graph neural networks. The other concurrent notable approach is DeepONet [59, 60]. [55] further improves the kernel approach by exploiting the multilevel grid structure. [56] proposes a discretization-invariant operator learner to achieve a state-of-the-art

[1] https://github.com/scaomath/galerkin-transformer

performance in certain benchmark problems. [89, 90] proposed a DeepONet roughly equivalent to an additive attention, similar to the one in the Neural Turing Machine (NMT) in [6]. Model/dimension reduction combined with neural nets is another popular approach to learn the solution operator for parametric PDEs [9, 63, 54, 22]. Deep convolutional neural networks (DCNN) are widely applied to learn the solution maps with a fixed discretization size [1, 8, 38, 34, 33, 99, 85]. Recently, DCNN has been successfully applied in various inverse problems [33, 45] such as Electrical Impedance Tomography (EIT). To our best knowledge, there is no work on data-driven approaches to an inverse interface coefficient identification for a class of coefficients with random interface geometries.

Attention mechanism and variants. Aside from the ground-breaking scaled dot-product attention in [87], earlier [6] proposed an additive content-based attention, however, with a vanishing gradient problem due to multiple nonlinearity composition. [23] shows the first effort in removing the softmax normalization in [6] after the projection, however, it still uses a Sigmoid nonlinearity before the additive interpolation propagation stage, and performs worse than its softmax counterpart. The first kind of linearizing the attention leverages the assumption of the existence of a feature map to approximate the softmax kernel [48, 17, 69]. The second type of linearization exploits the low-rank nature of the attention using various methods such as sampling or projection [72, 10, 78, 91], or fast multipole decomposition [64]. The conjecture in [74] inspires us to remove the softmax overall. [75] first proposed the inverse sequence length scaling normalization for a linear complexity attention without the softmax, however, the scaling normalization has not been extensively studied in examples and performs worse.

Various studies on Transformers. The kernel interpretation in [83] inspires us to reformulate the attention using the Galerkin projection. [94, Theorem 2] gives a theoretical foundation of removing the softmax normalization to formulate the Fourier-type attention. The Nyström approximation [96] essentially acknowledges the similarity between the attention matrix and an integral kernel. The experiments of various initializations in [58] inspires us to propose the rescaled diagonal dominant initialization scheme used in this paper. [95, 65] inspires us to try different layer normalization schemes. The practices of enriching the positional encoding in multiple layers in our work trace back to [2, 24], and more recently, contribute to the success of AlphaFold 2 [46], as it is natural to exploit the universal approximation if the target is a function explicitly in the coordinate frame and/or transformation group (position). Other practices on improving the attention mechanisms to conserve important physical properties are in [81, 30, 42].

3 Operator learning related to PDEs

Closely following the setup in [55, 56], we consider a data-driven model to approximate a densely-defined operator $T : \mathcal{H}_1 \to \mathcal{H}_2$ between two Hilbert spaces with an underlying bounded spacial domain $\Omega \subset \mathbb{R}^m$. The operator T to be learned is usually related to certain physical problems, of which the formulation is to seek the solution to a PDE of the following two types.

Parametric PDE: given coefficient $a \in \mathcal{A}$, and $f \in \mathcal{Y}$, find $u \in \mathcal{X}$ such that $L_a(u) = f$.

(i) To approximate the nonlinear mapping from the varying parameter a to the solution with a fixed right-hand side, $T : \mathcal{A} \to \mathcal{X},\ a \mapsto u$.

(ii) The inverse coefficient identification problem to recover the coefficient from a noisy measurement $\tilde{u}$ of the steady-state solution u, in this case, $T : \mathcal{X} \to \mathcal{A},\ \tilde{u} \mapsto a$.

Nonlinear initial value problem: given $u_0 \in \mathcal{H}$, find $u \in C([0, T]; \mathcal{H})$ such that $\partial_t u + N(u) = 0$.

(iii) Direct inference from the initial condition to the solution. $T : \mathcal{H} \to \mathcal{H},\ u_0(\cdot) \mapsto u(t_1, \cdot)$ with $t_1 \gg \Delta t$ with t_1 much greater than the step-size in traditional explicit integrator schemes.

Using (i) as an example, based on the given N observations $\{a^{(j)}, u^{(j)}\}_{j=1}^N$ and their approximations $\{a_h^{(j)}, u_h^{(j)}\}$ defined at a discrete grid of size $h \ll 1$, the goal of our operator learning problem is to build an approximation T_θ to T, such that $T_\theta(a_h)$ is a good approximation to $u = L_a^{-1} f =: T(a) \approx u_h$ independent of the mesh size h, where a_h and u_h are in finite dimensional spaces $\mathbb{A}_h, \mathbb{X}_h$ on this grid. We further assume that $a^{(j)} \sim \nu$ for a measure ν compactly supported on $\mathcal{A}$, and the sampled data form a reasonably sized subset of $\mathcal{A}$ representative of field applications. The loss $\mathcal{J}(\theta)$ is

$$\mathcal{J}(\theta) := \mathbb{E}_{a\sim\nu}\left[\|\big(T_\theta(a) - u\|_{\mathcal{H}}^2 + \mathfrak{G}(a, u; \theta)\right] \tag{1}$$

and in practice is approximated using the sampled observations on a discrete grid

$$\mathcal{J}(\theta) \approx \frac{1}{N}\sum_{j=1}^{N}\Big\{\big\|\big(T_\theta(a_h^{(j)}) - u_h^{(j)}\big\|_{\mathcal{H}}^2 + \mathfrak{G}\big(a_h^{(j)}, u_h^{(j)}; \theta\big)\Big\}. \tag{2}$$

In example (i), $\|\cdot\|_{\mathcal{H}}$ is the standard L^2-norm, and $\mathfrak{G}(a, u; \theta)$ serves as a regularizer with strength γ and is problem-dependent. In Darcy flow where $L_a := -\nabla\cdot(a\nabla(\cdot))$, it is $\gamma\|a\nabla(T_\theta(a) - u)\|_{L^2(\Omega)}^2$, since $u \in H^{1+\alpha}(\Omega)$ ($\alpha > 0$ depends on the regularity of a) and $a\nabla u \in \boldsymbol{H}(\mathrm{div};\Omega)$ a priori. For the evaluation metric, we drop the $\mathfrak{G}(a, u; \theta)$ term, and monitor the minimization of (2) using $\|\cdot\|_{\mathcal{H}}$.

4 Attention-based operator learner

Feature extractor. We assume the functions in both inputs and targets are sampled on a uniform grid. In an operator learning problem on $\Omega \subset \mathbb{R}^1$, a simple feedforward neural network (FFN) is used as the feature extractor that is shared by every position (grid point).

Interpolation-based CNN. If $\Omega \subset \mathbb{R}^2$, inspired by the multilevel graph kernel network in [55], we use two 3-level interpolation-based CNNs (CiNN) as the feature extractor, but also as the downsampling and upsampling layer, respectively, in which we refer to restrictions/prolongations between the coarse/fine grids both as interpolations. For the full details of the network structure please refer to Appendix B.

Positional encoding. The Cartesian coordinates of the grid, on which the attention operator's input latent representation reside, are concatenated as additional feature dimension(s) to the input, as well as to the latent representations in every attention head.

Decoder. The decoder is a problem-dependent admissible network that maps the learned representations from the encoder back to the target dimension. For smooth and regular solutions in $H^{1+\alpha}(\Omega)$, we opt for a 2-layer spectral convolution that is the core component in [56]. A simple pointwise feedforward neural network (FFN) is used for nonsmooth targets in $L^\infty(\Omega)$.

4.1 Simple self-attention encoder

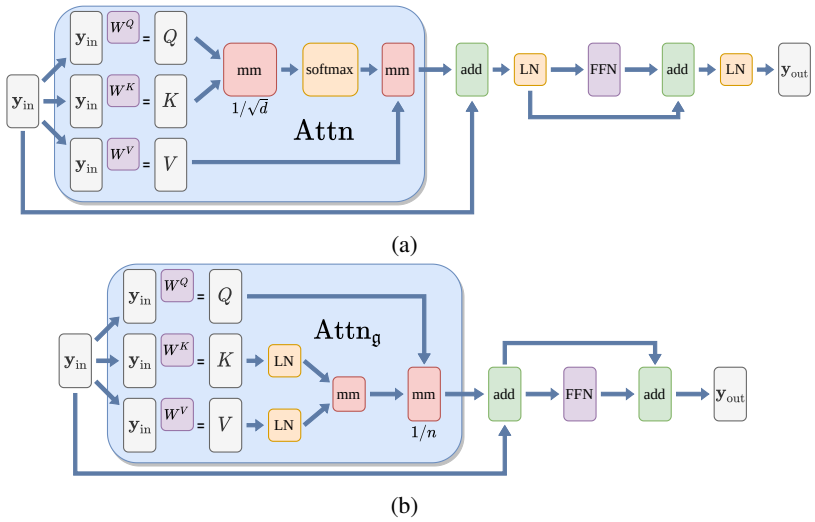

Figure 1: Comparison of the classic attention [87] with the Galerkin-type simple self-attention in a single head; (a) in the standard softmax attention, the softmax is applied row-wise after the matrix product `mm`; (b) a mesh-weighted normalization allows an integration-based interpretation.

The encoder contains a stack of identical simple attention-based encoder layers. For simplicity, we consider a single attention head that maps $\mathbf{y} \in \mathbb{R}^{n\times d}$ to another element in $\mathbb{R}^{n\times d}$, and define the

trainable projection matrices, and the latent representations $Q/K/V$ as follows.

$$W^Q, W^K, W^V \in \mathbb{R}^{d\times d}, \quad \text{and} \quad Q := \mathbf{y}W^Q, \quad K := \mathbf{y}W^K, \quad V := \mathbf{y}W^V. \tag{3}$$

We propose the following simple attention that (i) uses a mesh-weighted normalization without softmax, (ii) allows a scaling to propagate through the encoder layers.

$$\mathrm{Attn}_{\mathrm{sp}} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}, \quad \widetilde{\mathbf{y}} \leftarrow \mathbf{y} + \mathrm{Attn}_\dagger(\mathbf{y}), \quad \mathbf{y} \mapsto \widetilde{\mathbf{y}} + g(\widetilde{\mathbf{y}}), \tag{4}$$

where the head-wise normalizations are applied pre-dot-product: for $\dagger \in \{\mathfrak{f}, \mathfrak{g}\}$,

$$\text{(Fourier-type attention)} \qquad \mathbf{z} = \mathrm{Attn}_{\mathfrak{f}}(\mathbf{y}) := (\widetilde{Q}\widetilde{K}^\top)V/n, \tag{5}$$

$$\text{(Galerkin-type attention)} \qquad \mathbf{z} = \mathrm{Attn}_{\mathfrak{g}}(\mathbf{y}) := Q(\widetilde{K}^\top\widetilde{V})/n, \tag{6}$$

and $\widetilde{\diamond}$ denotes a trainable non-batch-based normalization. As in the classic Transformer [87], and inspired by the Galerkin projection interpretation, we choose $\widetilde{\diamond}$ as the layer normalization $\mathrm{Ln}(\diamond)$, and $g(\cdot)$ as the standard 2-layer FFN identically applied on every position (grid point). In simple attentions, the weight for each row of V, or column of Q in the linear variant, is not all positive anymore. This can be viewed as a cheap alternative to the cosine similarity-based attention.

Remark 4.1. *If we apply the regular layer normalization rule that eliminates any scaling:*

$$\mathbf{y} \mapsto \mathrm{Ln}\big(\mathbf{y} + \mathrm{Attn}_\dagger(\mathbf{y}) + g\big(\mathrm{Ln}(\mathbf{y} + \mathrm{Attn}_\dagger(\mathbf{y}))\big)\big), \quad \textit{where } \mathrm{Attn}_\dagger(\mathbf{y}) := Q(K^\top V)/n, \tag{7}$$

then this reduces to the efficient attention first proposed in [75].

4.1.1 Spacial structure-preserving feature map

Consider an operator learning problem with an underlying domain $\Omega \subset \mathbb{R}^1$. $\{x_l\}_{l=1}^n$ denotes the set of grid points in the discretized Ω such that the weight $1/n = h$ is the mesh size. Let $\zeta_q(\cdot), \phi_k(\cdot), \psi_v(\cdot) : \Omega \to \mathbb{R}^{1\times d}$ denote the feature maps of Q, K, V, i.e., the i-th row of Q, K, V written as $\boldsymbol{q}_i = \zeta_q(x_i)$, $\boldsymbol{k}_i = \phi_k(x_i)$, $\boldsymbol{v}_i = \psi_v(x_i)$. They are, in the NLP convention, viewed as the feature (embedding) vector at the i-th position, respectively. The inter-position topological structure such as continuity/differentiability in the same feature dimension is learned thus not explicit. The following ansatz for $Q/K/V$ in the same attention head is fundamental to our new interpretations.

Assumption 4.2. *The columns of $Q/K/V$, respectively, contain the vector representations of the learned basis functions spanning certain subspaces of the latent representation Hilbert spaces.*

Using $V \in \mathbb{R}^{n\times d}$ with a full column rank as an example, its columns contain potentially a set of bases $\{v_j(\cdot)\}_{j=1}^d$ evaluated at the grid points (degrees of freedom, or DoFs). Similarly, the learned bases whose DoFs form the columns of Q, K are denoted as $\{q_j(\cdot)\}_{j=1}^d$, $\{k_j(\cdot)\}_{j=1}^d$, as well as $\{z_j(\cdot)\}_{j=1}^d$ for the outputs in (5) and (6). To be specific, the j-th column of V, denoted by $\boldsymbol{v}^j$, then stands for a vector representation of the j-th basis function evaluated at each grid point, i.e., its l-th position stands for $(\boldsymbol{v}^j)_l = v_j(x_l)$. Consequently, the row $\boldsymbol{v}_i = (v_1(x_i), \ldots, v_d(x_i))$ can be alternatively viewed as the evaluation of a vector latent basis function at x_i.

4.1.2 Fourier-type attention of a quadratic complexity

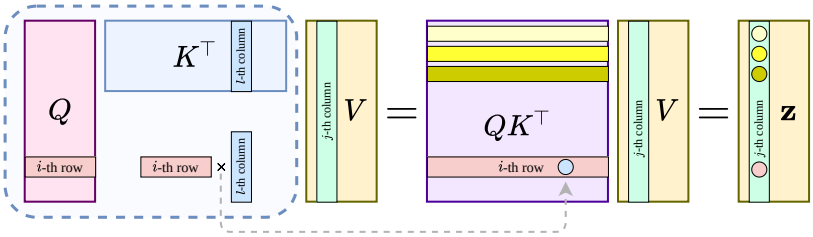

Figure 2: A dissection of Fourier-type attention's output.

In the Fourier-type attention (5), Q, K are assumed to be normalized for simplicity, the j-th column ($1 \le j \le d$) in the i-th row ($1 \le i \le n$) of $\mathbf{z}$ is computed by (see Figure 2):

$$\begin{aligned}(\boldsymbol{z}_i)_j &= h(QK^\top)_{i\bullet}\, \boldsymbol{v}^j = h\big(\boldsymbol{q}_i\cdot\boldsymbol{k}_1, \ldots, \boldsymbol{q}_i\cdot\boldsymbol{k}_l, \ldots, \boldsymbol{q}_i\cdot\boldsymbol{k}_n\big)^\top \cdot \boldsymbol{v}^j \\ &= h\sum_{l=1}^n (\boldsymbol{q}_i\cdot\boldsymbol{k}_l)(\boldsymbol{v}^j)_l \approx \int_\Omega \big(\zeta_q(x_i)\cdot\phi_k(\xi)\big) v_j(\xi)\,\mathrm{d}\xi,\end{aligned} \tag{8}$$

where the h-weight facilitates the numerical quadrature interpretation of the inner product. Concatenating columns $1 \leq j \leq d$ yields the i-row $\boldsymbol{z}_i$ of the output $\mathbf{z}$: $\boldsymbol{z}_i \approx \int_\Omega \big(\zeta_q(x_i) \cdot \phi_k(\xi)\big)\psi_v(\xi)\,\mathrm{d}\xi$. Therefore, without the softmax nonlinearity, the local dot-product attention output at i-th row computes approximately an integral transform with a non-symmetric learnable kernel function $\kappa(x, \xi) := \zeta_q(x)\phi_k(\xi)$ evaluated at x_i, whose approximation property has been studied in [94, Theorem 2], yet without the logits technicality due to the removal of the softmax normalization.

After the skip-connection, if we further exploit the learnable nature of the method and assume $W^V = \mathrm{diag}\{\delta_1, \cdots, \delta_d\}$ such that $\delta_j \neq 0$ for $1 \leq j \leq d$, under Assumption 4.2:

$$\delta_j^{-1} v_j(x) \approx z_j(x) - \int_\Omega \kappa(x, \xi) v_j(\xi)\,\mathrm{d}\xi, \quad \text{for } j = 1, \cdots, d, \quad \text{and } x \in \{x_i\}_{i=1}^n. \tag{9}$$

This is the forward propagation of the Fredholm equation of the second-kind for each $v_j(\cdot)$. When using an explicit orthogonal expansion such as Fourier to solve for $\{v_j(\cdot)\}_{j=1}^d$, or to seek for a better set of $\{v_j(\cdot)\}$ in our case, it is long known being equivalent to the Nyström's method with numerical integrations [7] (similar to the $h = 1/n$ weighted sum). Therefore, the successes of the random Fourier features in [17, 69] and the Nyströmformer's approximation [96] are not surprising.

Finally, we name this type of simple attention "Fourier" is due to the striking resemblance between the local scaled dot-product operation and a Fourier-type kernel [29] integral transform, since eventually the target resides in a Hilbert space with an underlying spacial domain Ω, while the latent representation space parallels a "frequency" domain on Ω^*. This also bridges the structural similarity of the scaled dot-product attention with the Fourier Neural Operator [56] where the Fast Fourier Transform (FFT) can be viewed as a non-learnable change of basis.

4.1.3 Galerkin-type attention of a linear complexity

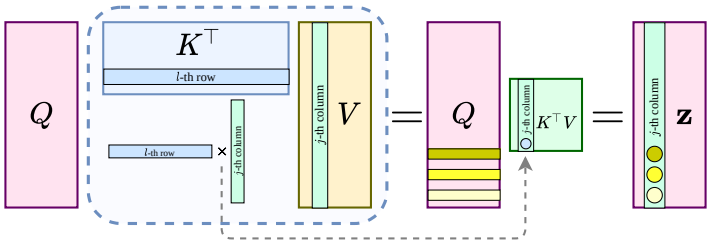

Figure 3: A dissection of Galerkin-type attention's output.

For the Galerkin-type simple attention in (6), K, V are assumed to be normalized for simplicity, we first consider the i-th entry in the j-th column $\boldsymbol{z}^j$ of $\mathbf{z}$ (see Figure 3):

$$(\boldsymbol{z}^j)_i = h\,\boldsymbol{q}_i^\top \cdot (K^\top V)_{\bullet j}, \tag{10}$$

which is the inner product of the i-th row of Q and the j-th column of $K^\top V$. Thus,

$$\boldsymbol{z}^j = h \begin{pmatrix} | & | & | & | \\ \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \\ | & | & | & | \end{pmatrix}^\top (K^\top V)_{\bullet j} = h \left((K^\top V)_{\bullet j}^\top \begin{pmatrix} — & \boldsymbol{q}^1 & — \\ — & \vdots & — \\ — & \boldsymbol{q}^d & — \end{pmatrix} \right)^\top \tag{11}$$

This reads as: $(K^\top V)_{\bullet j}$ contains the coefficients for the linear combination of the vector representations $\{\boldsymbol{q}^l\}_{l=1}^d$ of the bases stored in Q's column space to form the output $\mathbf{z}$. Meanwhile, the j-th column $(K^\top V)_{\bullet j}$ of $K^\top V$ consists the inner product of j-th column of V with every column of K.

$$\boldsymbol{z}^j = h \sum_{l=1}^d \boldsymbol{q}^l (K^\top V)_{lj}, \quad \text{where } (K^\top V)_{\bullet j} = \big(\boldsymbol{k}^1 \cdot \boldsymbol{v}^j, \boldsymbol{k}^2 \cdot \boldsymbol{v}^j, \cdots, \boldsymbol{k}^d \cdot \boldsymbol{v}^j\big)^\top. \tag{12}$$

As a result, using Assumption 4.2, and for simplicity the latent Hilbert spaces $\mathcal{Q}, \mathcal{K}, \mathcal{V}$ are assumed to be defined on the same spacial domain Ω, i.e., $k_l(\cdot)$, $v_j(\cdot)$ evaluated at every x_i are simply their vector representations $\boldsymbol{k}^l$ $(1 \leq l \leq d)$ and $\boldsymbol{v}^j$, we have the functions represented by the columns of the output $\mathbf{z}$ can be then compactly written as: rewriting $\langle v_j, k_l \rangle := (K^\top V)_{lj}$

$$z_j(x) := \sum_{l=1}^d \langle v_j, k_l \rangle\, q_l(x), \quad \text{for } j = 1, \cdots, d, \quad \text{and } x \in \{x_i\}_{i=1}^n, \tag{13}$$

where the bilinear form $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{K} \to \mathbb{R}$. (13) can be also written in a componentwise form:

$$z_j(x_i) := (\boldsymbol{z}^j)_i = h \sum_{l=1}^{d} (\boldsymbol{v}^j \cdot \boldsymbol{k}^l)(\boldsymbol{q}^l)_i \approx \sum_{l=1}^{d} \left(\int_\Omega v_j(\xi) k_l(\xi) \, \mathrm{d}\xi \right) q_l(x_i). \tag{14}$$

Therefore, when $\{\diamond_j(\cdot)\}_{j=1}^d, \diamond \in \{q, k, v\}$ consist approximations to three sets of bases for potentially different subspaces, and if we set the trial spaces as the column spaces of Q and the test space as that of K, respectively, the forward propagation of the Galerkin-type attention is a recast of a learnable Petrov–Galerkin-type projection (cf. Appendix D.1) for every basis represented by the columns of V.

Essentially, this linearized dot-product attention is a change of basis: the new set of basis is the column space of Q, and how to linearly combine the bases in Q is based on the inner product (response) of the corresponding feature dimension's basis in V against every basis in K. While the form of (14) suggests the orthonormality of the basis represented by Q, K, V, as well as being of full column ranks, the learnable nature of the method suggests otherwise (see Appendix D). At last, we have the following strikingly simple yet powerful approximation result.

Theorem 4.3 (Céa-type lemma, simplified version)**.** *Consider a Hilbert space $\mathcal{H}$ defined on a bounded domain $\Omega \subset \mathbb{R}^m$ discretized by n grid points, and $f \in \mathcal{H}$. $\mathbf{y} \in \mathbb{R}^{n \times d}$ is the current latent representation for $n > d > m$ and full column rank. $\mathbb{Q}_h \subset \mathcal{Q} \subset \mathcal{H}$ and $\mathbb{V}_h \subset \mathcal{V} \subset \mathcal{H}$ are the latent approximation subspaces spanned by basis functions with the columns of Q and V in (3) as degrees of freedom, respectively, and $0 < \dim \mathbb{Q}_h = r \leq \dim \mathbb{V}_h = d$. Let $\mathfrak{b}(\cdot, \cdot) : \mathcal{V} \times \mathcal{Q} \to \mathbb{R}$ be a continuous bilinear form; $g_\theta(\cdot)$ is a learnable map that is the composition of the Galerkin-type attention operator with a new set of attention matrices, and a pointwise universal approximator. If for any fixed $q \in \mathbb{Q}_h$ the functional norm of $\mathfrak{b}(\cdot, q)$ is bounded below by $c > 0$, then it holds for $f_h \in \mathbb{Q}_h$ being the best approximation of f in $\| \cdot \|_{\mathcal{H}}$:*

$$\min_\theta \|f - g_\theta(\mathbf{y})\|_{\mathcal{H}} \leq c^{-1} \min_{q \in \mathbb{Q}_h} \max_{v \in \mathbb{V}_h} \frac{|\mathfrak{b}(v, f_h - q)|}{\|v\|_{\mathcal{H}}} + \|f - f_h\|_{\mathcal{H}}. \tag{15}$$

Remarks on and interpretations of the best approximation result. Theorem 4.3 states that the Galerkin-type attention has capacity to give a quasi-optimal approximation in $\| \cdot \|_{\mathcal{H}}$ in the current subspace $\mathbb{Q}_h$. For the mathematically rigorous complete set of notations and the full details of the proof we refer the readers to Appendix D. Even though Theorem 4.3 is presented for a single instance of $f \in \mathcal{H}$ for simplicity, the proof in Appendix D.3 shows that the attention operator is fully capable of simultaneously approximating a collection of functions.

Estimate (15) comes with great scalability with respect to the sequence length in that it all boils down to whether c is independent of n in the lower bound of $\|\mathfrak{b}(\cdot, q)\|_{\mathbb{V}_h'}$. The existence of an n-independent lower bound is commonly known as the discrete version of the Ladyzhenskaya–Babuška–Brezzi (LBB) condition [19, Chapter 6.12], also referred as the Banach-Nečas-Babuška (BNB) condition in Galerkin methods on Banach spaces [27, Theorem 2.6].

As the cornerstone of the approximation to many PDEs, the LBB condition establishes the surjectivity of a map from $\mathbb{V}_h$ to $\mathbb{Q}_h$. In a simplified context (15) above of approximating functions using this linear attention variant (Q: values, query, V: keys), it roughly translates to: for an incoming "query" (function f in a Hilbert space), to deliver its best approximator in "value" (trial function space), the "key" (test function space) has to be sufficiently rich such that there exists a key to unlock every possible value. While from the perspective of a layer-wise dynamical basis update (14) (Q: values, K: keys, V: query), through testing against the "keys", a latent representation is sought such that "query" (input trial space) and "values" (output trial space) can achieve the minimum possible difference under a dual norm; for details please refer to Appendix D.4.5.

The role of feed-forward networks and positional encodings. Due to the presence of the concatenated coordinates $\mathbf{x} := \|_{i=1}^n x_i \in \mathbb{R}^{n \times m}$ to the latent representation $\mathbf{y}$, the pointwise subnetwork $g_s(\cdot) : \mathbb{R}^{n \times m} \to \mathbb{R}^{n \times d}$ of the nonlinear universal approximator (FFN) in each attention block is one among many magics of the attention mechanism. In every attention layer, the subspaces $\mathbb{Q}_h / \mathbb{K}_h / \mathbb{V}_h$ are being constantly enriched by $\mathrm{span}\{w_j \in \mathbb{X}_h : w_j(x_i) = (g_s(\mathbf{x}))_{ij}, 1 \leq j \leq d\} \subset \mathcal{H}$ to try to capture how an operator of interest responses to the subset of inputs. Despite the fact that the FFNs, when being viewed as a class of functions, bear no linear structure within, the basis functions produced this way act as a building block to characterize a linear space for a learnable projection. This heuristic shows to be effective when the target is a function of the (relative) positional encodings

(coordinates, transformation groups, etc.), in that this is incorporated in many other attention-based learners with applications in physical sciences [81, 30, 42, 46].

5 Experiments

In this section we perform a numerical study the proposed Fourier Transformer (**FT**) with the Fourier-type encoder, and the Galerkin Transformer (**GT**) with the Galerkin-type encoder, in various PDE-related operator learning tasks. The models we compare our newly proposed models with are the operator learners with the simple attention replaced by the standard softmax normalized scaled dot-product attention (**ST**) [87], and a linear variant (**LT**) [75] in which two independent softmax normalizations are applied on Q, K separately.[2] The data are obtained courtesy of the PDE benchmark under the MIT license.[3] For full details of the training/evaluation and model structures please refer to Appendix C.

Instead of the standard Xavier uniform initialization [32], inspired by the interpretations of Theorem 4.3 in Appendix D.3.4, for the projection matrices W^Q, W^K, W^V, we modify the initialization slightly as follows

$$W^{\diamond}_{\text{init}} \leftarrow \eta U + \delta I, \quad \text{for } \diamond \in \{Q, K, V\}, \tag{16}$$

where $U = (x_{ij})$ is a random matrix using the Xavier initialization with gain 1 such that $x_{ij} \sim \mathcal{U}([-\sqrt{3/d}, \sqrt{3/d}])$, and δ is a small positive number. In certain operator learning tasks, we found that this tiny modification boosts the evaluation performance of models by up to 50% (see Appendix C.2) and improves the training stability acting as a cheap remedy to the lack of a softmax normalization. We note that similar tricks have been discovered concurrently in [21].

Unsurprisingly, when compared the memory usage and the speed of the networks (see Table 1), the Fourier-type attention features a 40%–50% reduction in memory versus the attention with a softmax normalization. The Galerkin attention-based models have a similar memory profile with the standard linear attention, it offers up to a 120% speed boost over the linear attention in certain tests.

The baseline models for each example are the best operator learner to-date, the state-of-the-art Fourier Neural Operator (FNO) in [56] but without the original built-in batch normalization. All attention-based models match the parameter quota of the baseline, and are trained using the loss in (2) with the same `1cycle` scheduler [77] for 100 epochs. For fairness, we have also included the results for the standard softmax normalized models (ST and LT) using the new layer normalization scheme in (5) and (6). We have retrained the baseline with the same `1cycle` scheduler using the code provided in [56], and listed the original baseline results using a step scheduler of 500 epochs of training from [56] Example 5.1 and Example 5.2, respectively.

Table 1: The memory usage/FLOP/complexity comparison of the models. Batch size: 4; the CUDA mem (GB): the sum of the `self_cuda_memory_usage`; GFLOP: Giga FLOP for 1 backpropagation; both are from the PyTorch `autograd` profiler for 1 BP averaging from 1000 BPs; the mem (GB) is recorded from `nvidia-smi` of the memory allocated for the active Python process during profiling; the speed (iteration per second) is measured during training; the exponential operation has an explicit complexity of $e > 1$ [13].

	Example 1: $n = 8192$				Encoders only: $n = 8192, d = 128, l = 10$				Dot-product complexity per layer
	Mem	CUDA Mem	Speed	GFLOP	Mem	CUDA Mem	Speed	GFLOP	
ST	18.39	31.06	5.02	1393	18.53	31.34	4.12	1876	$O(n^2ed)$
FT	10.05	22.92	6.10	1138	10.80	22.32	5.46	1610	$O(n^2d)$
LT	2.55	2.31	12.70	606	2.73	2.66	10.98	773	$O(n(d^2 + ed))$
GT	**2.36**	**1.93**	**27.15**	**275**	**2.53**	**2.33**	**19.20**	**412**	$O(nd^2)$

5.1 Example 1: viscous Burgers' equation

In this example, we consider a benchmark problem of the viscous Burgers' equation with a periodic boundary condition on $\Omega := (0, 1)$ in [56]. The nonlinear operator to be learned is the discrete

[2] https://github.com/lucidrains/linear-attention-transformer

[3] https://github.com/zongyi-li/fourier_neural_operator

approximations to the solution operator $T : C_p^0(\Omega) \cap L^2(\Omega) \to C_p^0(\Omega) \cap H^1(\Omega)$, $u_0(\cdot) \mapsto u(\cdot, 1)$. The initial condition $u_0(\cdot)$'s are sampled following a Gaussian Random Field (GRF).

The result can be found in Table 2. All attention-based operator learners achieve a resolution-invariant performance similar with FNO1d in [56]. The new Galerkin projection-type layer normalization scheme significantly outperforms the regular layer normalization rule in this example, in which both inputs and targets are unnormalized. For full details please refer to Appendix C.2.

Table 2: Evaluation relative error ($\times 10^{-3}$) of Burgers' equation 5.1.

	$n = 512$ ($b = 8$)	$n = 2048$ ($b = 8$)	$n = 8192$ ($b = 4$)
FNO1d [56]	15.8	14.6	13.9
FNO1d `1cycle`	4.373	4.126	4.151
FT regular Ln	1.400	1.477	1.172
GT regular Ln	2.181	1.512	2.747
ST regular Ln	1.927	2.307	1.981
LT regular Ln	1.813	1.770	1.617
FT Ln on Q, K	**1.135**	**1.123**	**1.071**
GT Ln on K, V	**1.203**	**1.150**	**1.025**
ST Ln on Q, K	1.271	1.266	1.330
LT Ln on K, V	1.139	1.149	1.221

5.2 Example 2: Darcy flow

In this example, we consider another well-known benchmark $-\nabla \cdot (a \nabla u) = f$ for $u \in H_0^1(\Omega)$ from [9, 56, 55, 63], and the operator to be learned is the approximations to $T : L^\infty(\Omega) \to H_0^1(\Omega), a \mapsto u$, in which a is the coefficient with a random interface geometry, and u is the weak solution.

The result can be found in Table 3. As the input/output are normalized, in contrast to Example 5.1, the Galerkin projection-type layer normalization scheme does not significantly outperform the regular layer normalization rule in this example. The attention-based operator learners achieve on average 30% to 50% better evaluation results than the baseline FNO2d using the same trainer. For full details please refer to Appendix C.3.

Table 3: Evaluation relative error ($\times 10^{-2}$) of Darcy interface problem 5.2.

	$n_f = 141, n_c = 43$	$n_f = 211, n_c = 61$
FNO2d [56]	1.09	1.09
FNO2d `1cycle` (only n_f)	1.419	1.424
FT regular Ln	**0.838**	**0.847**
GT regular Ln	0.894	0.856
ST regular Ln	1.075	1.131
LT regular Ln	1.024	1.130
FT Ln on Q, K	0.873	0.921
GT Ln on K, V	**0.839**	**0.844**
ST Ln on Q, K	0.946	0.959
LT Ln on K, V	0.875	0.970

5.3 Example 3: inverse coefficient identification for Darcy flow

In this example, we consider an inverse coefficient identification problem based on the same data used in Example 5.2. The input (solution) and the target (coefficient) are reversed from Example 5.2, and the noises are added to the input. The inverse problems in practice are a class of important tasks in many scientific disciplines such as geological sciences and medical imaging but much more difficult due to poor stability [49]. We aim to learn an approximation to an ill-posed operator $T : H_0^1(\Omega) \to L^\infty(\Omega), u + \epsilon N_\nu(u) \mapsto a$, where $N_\nu(u)$ stands for noises related to the sampling distribution and the data. $\epsilon = 0.01$ means 1% of noise added in both training and evaluation, etc.

The result can be found in Table 4. It is not surprising that FNO2d, an excellent smoother which filters higher modes in the frequency domain, struggles in this example to recover targets consisting of

high-frequency traits (irregular interfaces) from low-frequency prevailing data (smooth solution due to ellipticity). We note that, the current state-of-the-art methods [14] for inverse interface coefficient identification need to carry numerous iterations to recover a single instance of a simple coefficient with a regular interface, provided that a satisfactory denoising has done beforehand. The attention-based operator learner has capacity to unearth structurally how this inverse operator's responses on a subset, with various benefits articulated in [55, 56, 5, 63, 9].

Table 4: Evaluation relative error ($\times 10^{-2}$) of the inverse problem 5.3.

	$n_f = 141, n_c = 36$			$n_f = 211, n_c = 71$		
	$\epsilon = 0$	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0$	$\epsilon = 0.01$	$\epsilon = 0.1$
FNO2d (only n_f)	13.71	13.78	15.12	13.93	13.96	15.04
FNO2d (only n_c)	14.17	14.31	17.30	13.60	13.69	16.04
FT regular Ln	**1.799**	**2.467**	6.814	1.563	2.704	8.110
GT regular Ln	2.026	2.536	**6.659**	1.732	2.775	8.024
ST regular Ln	2.434	3.106	7.431	2.069	3.365	8.918
LT regular Ln	2.254	3.194	9.056	2.063	3.544	9.874
FT Ln on Q, K	1.921	2.717	6.725	**1.523**	**2.691**	8.286
GT Ln on K, V	1.944	2.552	6.689	1.651	2.729	**7.903**
ST Ln on Q, K	2.160	2.807	6.995	1.889	3.123	8.788
LT Ln on K, V	2.360	3.196	8.656	2.136	3.539	9.622

6 Conclusion

We propose a general operator learner based on a simple attention mechanism. The network is versatile and is able to approximate both the PDE solution operator and the inverse coefficient identification operator. The evaluation accuracy on the benchmark problems surpasses the current best state-of-the-art operator learner Fourier Neural Operator (FNO) in [56]. However, we acknowledge the limitation of this work: (i) similar to other operator learners, the subspace, on which we aim to learn the operator's responses, may be infinite dimensional, but the operator must exhibit certain low-dimensional attributes (e.g., smoothing property of the higher frequencies in GRF); (ii) it is not efficient for the attention operator to be applied at the full resolution for a 2D problem, and this limits the approximation to a nonsmooth subset such as functions in L^∞; (iii) due to the order of the matrix product, the proposed linear variant of the scaled dot-product attention is non-causal thus can only apply to encoder-only applications.

7 Broader Impact

Our work introduces the state-of-the-art self-attention mechanism the first time to PDE-related operator learning problems. The new interpretations of attentions invite numerical analysts to work on a more complete and delicate approximation theory of the attention mechanism. We have proved the Galerkin-type attention's approximation capacity in an ideal Hilbertian setting. Numerically, the new attention-based operator learner has capacity to approximate the difficult inverse coefficient identification problem with an extremely noisy measurements, which was not attainable using traditional iterative methods for nonlinear mappings. Thus, our method may pose a huge positive impact in geoscience, medical imaging, etc. Moreover, traditionally the embeddings in Transformer-based NLP models map the words to a high dimensional space, but the topological structure in the same feature dimension between different positions are learned thereby not efficient. Our proof provides a theoretical guide for the search of feature maps that preserve, or even create, structures such as differentiability or physical invariance. Thus, it may contribute to the removal of the softmax nonlinearity to speed up significantly the arduous training or pre-training of larger encoder-only models such as BERT [25], etc. However, we do acknowledge that our research may negatively impact on the effort of building a cleaner future for our planet, as inverse problems are widely studied in reservoir detection, and we have demonstrated that the attention-based operator learner could potentially help to discover new fossil fuel reservoirs due to its capacity to infer the coefficients from noisy measurements.

Acknowledgments and Disclosure of Funding

The hardware to perform this work is kindly donated by Andromeda Saving Fund. The first author was supported in part by the National Science Foundation under grants DMS-1913080 and DMS-2136075. No additional revenues are related to this work. We would like to thank the anonymous reviewers and the area chair for the suggestions on improving this article. We would like to thank Dr. Long Chen (Univ of California Irvine) for the inspiration of and encouragement on the initial conceiving of this paper, as well as numerous constructive advices on revising this paper, not mentioning his persistent dedication of making publicly available tutorials [15] on writing beautiful vectorized code. [4] We would like to thank Dr. Ari Stern (Washington Univ in St. Louis) for the help on the relocation during the COVID-19 pandemic. We would like to thank Dr. Likai Chen (Washington Univ in St. Louis) for the invitation to the Stats and Data Sci seminar at WashU that resulted the reboot of this study. [5] We would like to thank Dr. Ruchi Guo (Univ of California Irvine) and Dr. Yuanzhe Xi (Emory Univ) for the invaluable feedbacks on the choice of the numerical experiments. We would like to thank the Kaggle community, including but not limited to Jean-François Puget (CPMP@Kaggle) for sharing a simple Graph Transformer in TensorFlow,[6] Murakami Akira (mrkmakr@Kaggle) for sharing a Graph Transformer with a CNN feature extractor in Tensorflow, [7] and Cher Keng Heng (hengck23@Kaggle) for sharing a Graph Transformer in PyTorch.[8] We would like to thank daslab@Stanford, OpenVaccine, and Eterna for hosting the COVID-19 mRNA Vaccine competition and Deng Lab (Univ of Georgia) for collaborating in this competition. We would like to thank CHAMPS (Chemistry and Mathematics in Phase Space) for hosting the J-coupling quantum chemistry competition and Corey Levinson (Eligo Energy, LLC) for collaborating in this competition. We would like to thank Zongyi Li (Caltech) for sharing some early dev code in the updated PyTorch `fft` interface and the comments on the viscosity of the Burgers' equation. We would like to thank Ziteng Pang (Univ of Michigan) and Tianyang Lin (Fudan Univ) to update us with various references on Transformers. We would like to thank Joel Schlosser (Facebook) to incorporate our change to the PyTorch `transformer` module to simplify our testing pipeline. We would be grateful to the PyTorch community for selflessly code sharing, including Phil Wang(lucidrains@github) and Harvard NLP group [50]. We would like to thank the `chebfun` [26] for integrating powerful tools into a simple interface to solve PDEs. We would like to thank Dr. Yannic Kilcher (ykilcher@twitter) and Dr. Hung-yi Lee (National Taiwan Univ) for frequently covering the newest research on Transformers in video formats. We would also like to thank the Python community [86, 67] for sharing and developing the tools that enabled this work, including PyTorch [68], NumPy [37], SciPy [88], Plotly [43] Seaborn [92], Matplotlib [41], and the Python team for Visual Studio Code. We would like to thank `draw.io` [44] for providing an easy and powerful interface for producing vector format diagrams. For details please refer to the documents of every function that is not built from the ground up in our open-source software library.[9]

[4] https://github.com/lyc102/ifem

[5] Transformer: A Dissection from an Amateur Applied Mathematician

[6] https://www.kaggle.com/cpmpml/graph-transfomer

[7] https://www.kaggle.com/mrkmakr/covid-ae-pretrain-gnn-attn-cnn

[8] https://www.kaggle.com/c/stanford-covid-vaccine/discussion/183518

[9] https://github.com/scaomath/galerkin-transformer

References

[1] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[2] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 3159–3166, Jul. 2019.

[3] Giovanni Alessandrini. An identification problem for an elliptic equation in two variables. *Annali di matematica pura ed applicata*, 145(1):265–295, 1986.

[4] Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. Graph element networks: adaptive, structured computation and memory. In *International Conference on Machine Learning*, pages 212–222. PMLR, 2019.

[5] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[6] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[7] Jean-Paul Berrut and Manfred R Trummer. Equivalence of Nyström's method and Fourier methods for the numerical solution of Fredholm integral equations. *Mathematics of computation*, 48(178):617–623, 1987.

[8] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

[9] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric PDEs. *arXiv preprint arXiv:2005.03180*, 2020.

[10] Guy Blanc and Steffen Rendle. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*, pages 590–599. PMLR, 2018.

[11] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[12] Susanne C Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer, 2008.

[13] Richard P Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In *Analytic computational complexity*, pages 151–176. Elsevier, 1976.

[14] Tony F Chan and Xue-Cheng Tai. Identification of discontinuous coefficients in elliptic problems using total variation regularization. *SIAM Journal on Scientific Computing*, 25(3):881–904, 2003.

[15] Long Chen. iFEM: an integrated finite element methods package in MATLAB. Technical report, 2008.

[16] Tianping Chen and Hong Chen. 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, 1995.

[17] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with Performers. In *International Conference on Learning Representations (ICLR)*, 2021.

[18] Philippe G Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[19] Philippe G Ciarlet. *Linear and nonlinear functional analysis with applications*, volume 130. SIAM, 2013.

[20] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[21] Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Punta Cana, Dominican Republic, November 2021.

[22] Niccolò Dal Santo, Simone Deparis, and Luca Pegolotti. Data driven approximation of parametrized pdes by reduced basis and neural networks. *Journal of Computational Physics*, 416:109550, 2020.

[23] Alexandre de Brébisson and Pascal Vincent. A cheap linear attention mechanism with fast lookups and fixed-size representations. *arXiv preprint arXiv:1609.05866*, 2016.

[24] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1*, pages 4171–4186. Association for Computational Linguistics, 2019.

[26] Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. Chebfun guide, 2014.

[27] Alexandre Ern and Jean-Luc Guermond. *Theory and Practice of Finite Elements*. Springer, 2004.

[28] Robert D Falgout. An introduction to algebraic multigrid. *IEEE Annals of the History of Computing*, 8 (06):24–33, 2006.

[29] Charles Fox. The G and H functions as symmetrical Fourier kernels. *Transactions of the American Mathematical Society*, 98(3):395–429, 1961.

[30] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 1970–1981, 2020.

[31] Vivette Girault and P-A Raviart. Finite element approximation of the Navier-Stokes equations. *Lecture Notes in Mathematics, Berlin Springer Verlag*, 749, 1979.

[32] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[33] Ruchi Guo and Jiahua Jiang. Construct deep neural networks based on direct sampling methods for solving electrical impedance tomography. *SIAM Journal on Scientific Computing*, 43(3):B678–B711, 2021.

[34] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.

[35] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *arXiv preprint arXiv:2109.13459*, 2021.

[36] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2013. ISBN 9783662024270.

[37] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825): 357–362, September 2020.

[38] Juncai He and Jinchao Xu. Mgnet: A unified framework of multigrid and convolutional neural network. *Science China Mathematics*, 62(7):1331–1354, 2019.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[40] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[41] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[42] Michael J Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. LieTransformer: equivariant self-attention for Lie groups. pages 4533–4543, 2021.

[43] Plotly Technologies Inc. plotly, 2015.

[44] JGraph. draw.io, 2021.

[45] Jiahua Jiang, Yi Li, and Ruchi Guo. Learn an index operator by cnn for solving diffusive optical tomography: a deep direct sampling method. *arXiv preprint arXiv:2104.07703*, 2021.

[46] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[47] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 2021.

[48] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[49] A. Kirsch. *An Introduction to the Mathematical Theory of Inverse Problems*. Applied Mathematical Sciences. Springer New York, 2011. ISBN 9781441984746.

[50] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proc. ACL*, 2017.

[51] Annika Lang and Jürgen Potthoff. Fast simulation of Gaussian random fields. *MonteCarlo Methods Appl.*, 17(3):195–214, 2011.

[52] Peter D Lax and Robert D Richtmyer. Survey of the stability of linear finite difference equations. *Communications on pure and applied mathematics*, 9(2):267–293, 1956.

[53] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. FNet: Mixing tokens with Fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021.

[54] Sijing Li, Zhiwen Zhang, and Hongkai Zhao. A data-driven approach for multiscale elliptic PDEs with random coefficients based on intrinsic dimension reduction. *Multiscale Modeling & Simulation*, 18(3): 1242–1271, 2020.

[55] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems*, volume 33, pages 6755–6766, 2020.

[56] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[57] Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *Advances in Neural Information Processing Systems (NIPS 2018)*, 31:6169–6178, 2018.

[58] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

[59] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[60] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[61] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[62] Peter Monk et al. *Finite element methods for Maxwell's equations*. Oxford University Press, 2003.

[63] Nicholas H Nelsen and Andrew M Stuart. The random feature model for input-output maps between banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021.

[64] Tan M. Nguyen, Vai Suliafu, Stanley J. Osher, Long Chen, and Bao Wang. FMMformer: Efficient and Flexible Transformer via Decomposed Near-field and Far-field Attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[65] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

[66] Akifumi Okuno, Tetsuya Hada, and Hidetoshi Shimodaira. A probabilistic framework for multi-view feature learning with many-to-many associations via neural networks. In *International Conference on Machine Learning*, pages 3888–3897. PMLR, 2018.

[67] Travis E. Oliphant. Python for scientific computing. *Computing in Science Engineering*, 9(3):10–20, 2007.

[68] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Workshop Autodiff*, 2017.

[69] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

[70] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[71] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[72] Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[73] Nicholas Roberts, Mikhail Khodak, Tri Dao, Liam Li, Christopher Ré, and Ameet Talwalkar. Rethinking neural operations for diverse tasks. *arXiv preprint arXiv:2103.15798*, 2021.

[74] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021.

[75] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021.

[76] Jonathan W Siegel and Jinchao Xu. Approximation rates for neural networks with general activation functions. *Neural Networks*, 128:313–321, 2020.

[77] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

[78] Kyungwoo Song, Yohan Jung, Dongjun Kim, and Il-Chul Moon. Implicit kernel attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):9713–9721, May 2021.

[79] Ari Stern. Banach space projections and Petrov–Galerkin estimates. *Numerische Mathematik*, 130(1): 125–133, 2015.

[80] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[81] Kai Sheng Tai, Peter Bailis, and Gregory Valiant. Equivariant transformer networks. In *International Conference on Machine Learning*, pages 6086–6095. PMLR, 2019.

[82] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. MLP-mixer: An all-MLP architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[83] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4344–4353, Hong Kong, China, November 2019.

[84] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[85] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020.

[86] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[87] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[88] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[89] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[90] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. *arXiv preprint arXiv:2103.10974*, 2021.

[91] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[92] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[93] H. Whitney. *Geometric Integration Theory*. Princeton Legacy Library. Princeton University Press, 2015. ISBN 9781400877577.

[94] Matthew A Wright and Joseph E Gonzalez. Transformers are deep infinite-dimensional non-Mercer binary kernel machines. *arXiv preprint arXiv:2106.01506*, 2021.

[95] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[96] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based algorithm for approximating self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14138–14148, May 2021.

[97] Jinchao Xu and Ludmil Zikatanov. Algebraic multigrid methods. *Acta Numerica*, 26:591–721, 2017.

[98] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020.

[99] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Section 6 and in each experiment's detailed description in Appendix C.3 and C.4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The key assumption of the new interpretations of the attention mechanism is in Assumption 4.2. Due to the page limit, the assumptions and settings are fully elaborated in a mathematical rigorous fashion in Appendix D for the proof of Theorem 4.3.
   (b) Did you include complete proofs of all theoretical results? [Yes] In Appendix D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The data are publicly available. For instruction to reproduce our result please refer to `README.md` in `https://github.com/scaomath/galerkin-transformer/tree/main/examples`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We include all details in the supplemental material Appendix B and C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We found that our method is numerically seed-invariant, and we have reported the error bands in Appendix C.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please refer to Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Multiple creators of or contributing work to the dataset used in our work are cited [56, 5, 55, 63]. The code base we have implemented our model upon is in the footnote of Section 5, and cited in the Acknowledgement [50]. In the publicly available code repository, in the document of every function, we have credited every author we can find for even a small code snippet, if that function is not built from ground up by us.
   (b) Did you mention the license of the assets? [Yes] In Section 5.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The full code base to replicate our results is publicly available as an open-source software.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The data is obtained from `https://github.com/zongyi-li/fourier_neural_operator` with the MIT license as well as the consent from the author per a personal communication.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]