Journal of Computational Physics 506 (2024) 112944

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Mitigating spectral bias for the multiscale operator learning

Xinliang Liu [a,b,1], Bo Xu [c,1], Shuhao Cao [d], Lei Zhang [e,*]

[a] School of Mathematical Sciences, Shenzhen University, Shenzhen 518060, China
[b] Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal, 23955, Saudi Arabia
[c] School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai, 200240, China
[d] School of Science and Engineering, University of Missouri-Kansas City, Kansas City, 64110, MO, United States
[e] School of Mathematical Sciences, Institute of Natural Sciences and MOE-LSC, Shanghai Jiao Tong University, Shanghai, 200240, China

ARTICLE INFO

Keywords:
Partial differential equations
Operator learning
Transformer
Multiscale PDE

ABSTRACT

Neural operators have emerged as a powerful tool for learning the mapping between infinite-dimensional parameter and solution spaces of partial differential equations (PDEs). In this work, we focus on multiscale PDEs that have important applications such as reservoir modeling and turbulence prediction. We demonstrate that for such PDEs, the spectral bias towards low-frequency components presents a significant challenge for existing neural operators. To address this challenge, we propose a hierarchical attention neural operator (HANO) inspired by the hierarchical matrix approach. HANO features a scale-adaptive interaction range and self-attentions over a hierarchy of levels, enabling nested feature computation with controllable linear cost and encoding/decoding of multiscale solution space. We also incorporate an empirical H^1 loss function to enhance the learning of high-frequency components. Our numerical experiments demonstrate that HANO outperforms state-of-the-art (SOTA) methods for representative multiscale problems.

1. Introduction

In recent years, operator learning methods have emerged as powerful tools for computing parameter-to-solution maps of partial differential equations (PDEs). In this paper, we focus on the operator learning for multiscale PDEs (MsPDEs) that encompass multiple temporal/spatial scales. MsPDE models arise in applications involving heterogeneous and random media, and are crucial for predicting complex phenomena such as reservoir modeling, atmospheric and ocean circulation, and high-frequency scattering. Important prototypical examples include multiscale elliptic partial differential equations, where the diffusion coefficients vary rapidly. The coefficient can be potentially rapidly oscillatory, have high contrast ratio, or even bear a continuum of non-separable scales.

MsPDEs, even with fixed parameters, present great challenges for classical numerical methods [1], as their computational cost typically scales inversely proportional to the finest scale ϵ of the problem. To overcome this issue, multiscale solvers have been developed by incorporating microscopic information to achieve computational cost independent of ϵ. One such technique is *numerical homogenization* [2–7], which identifies low-dimensional approximation spaces adapted to the corresponding multiscale operator. Similarly, fast solvers like *multilevel/multigrid methods* [8,9] and *wavelet-based multiresolution methods* [10,11] may face limitations

[1] The first two authors contributed equally.

* Corresponding author.
E-mail address: lzhang2012@sjtu.edu.cn (L. Zhang).

https://doi.org/10.1016/j.jcp.2024.112944
Received 22 December 2023; Received in revised form 12 March 2024; Accepted 13 March 2024

Available online 19 March 2024

when applied to multiscale PDEs [1], while multilevel methods based on numerical homogenization techniques, such as Gamblets [12], have emerged as a way to discover scalable multilevel algorithms and operator-adapted wavelets for multiscale PDEs. Low-rank decomposition-based methods are another popular approach to exploit the low-dimensional nature of MsPDEs. Notable example includes the fast multipole method [13], hierarchical matrices ($\mathscr{H}$ and $\mathscr{H}^2$ matrices) [14], and hierarchical interpolative factorization [15]. These methods can achieve (near-)linear scaling and high computational efficiency by exploiting the low-rank approximation of the (elliptic) Green's function [16].

Neural operators, unlike traditional solvers that operate with fixed parameters, are capable of handling a range of input parameters, making them promising for data-driven forward and inverse solving of PDE problems. Pioneering work in operator learning methods includes [17–20]. Nevertheless, they are limited to problems with fixed discretization sizes. Recently, infinite-dimensional operator learning has been studied, which learns the solution operator (mapping) between infinite-dimensional Banach spaces for PDEs. Most notably, the Deep Operator Network (DeepONet) [21] was proposed as a pioneering model to leverage deep neural networks' universal approximation for operators [22]. Taking advantage of the Fast Fourier Transform (FFT), Fourier Neural Operator (FNO) [23] constructs a learnable parametrized kernel in the frequency domain to render the convolutions in the solution operator more efficient. Other developments include the multiwavelet extension of FNO [24], Message-Passing Neural Operators [25], dimension reduction in the latent space [26], Gaussian Processes [27], Clifford algebra-inspired neural layers [28], and Dilated convolutional residual network [29].

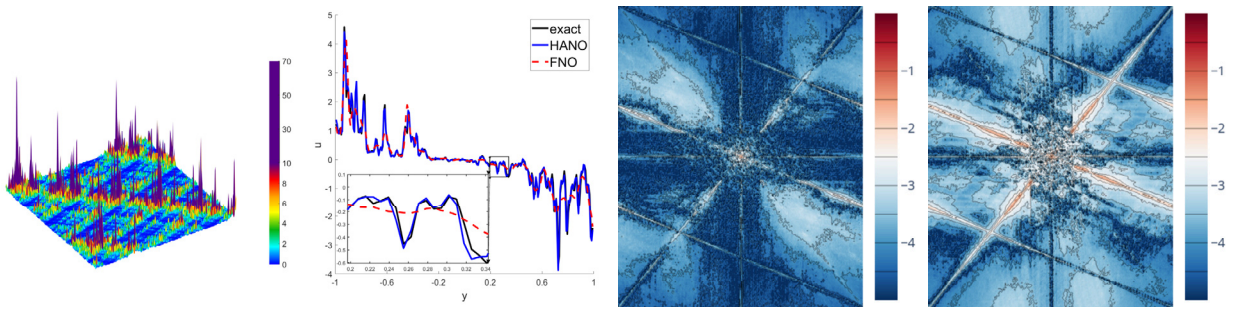
Attention neural architectures, popularized by the Transformer deep neural network [30], have emerged as universal backbones in the field of Deep Learning. These architectures serve as the foundation for numerous state-of-the-art models, including GPT [31], Vision Transformer (ViT) [32], and Diffusion models [33,34]. More recently, Transformers have been studied and become increasingly popular in PDE operator learning problems, e.g., in [35–41] and many others. There are several advantages in the attention architectures. Attention can be viewed as a parametrized instance-dependent kernel integral to learn the "basis" [35] similar to those in the numerical homogenization; see also the exposition featured in neural operators [42]. This layerwise latent updating resembles the learned "basis" in DeepONet [39], or frame [43]. It is flexible to encode the non-uniform geometries in the latent space [44]. In [45,46], advanced Transformer architectures (ViT) and Diffusion models are combined with the neural operator framework. In [47], Transformers are combined with reduced-order modeling to accelerate the fluid simulation for turbulent flows. In [48], tensor decomposition techniques are employed to enhance the efficiency of attention mechanisms in solving high-dimensional partial differential equation (PDE) problems.

Among these data-driven operator learning models, under certain circumstances, the numerical results could sometimes overtake classical numerical methods in terms of efficiency or even in accuracy. For instance, full wave inversion is considered in [49] with the fusion model of FNO and DeepOnet (Fourier-DeepONet); direct methods-inspired DNNs are applied to the boundary value Calderón problems achieve much more accurate reconstruction with the help of data [50–52]; in [53], the capacity of FNO to jump significantly large time steps for spatialtemporal PDEs is exploited to infer the wave packet scattering in quantum physics and achieves magnitudes more efficient result than traditional implicit Euler marching scheme. [54] exploits the capacity of graph neural networks to accelerate particle-based simulations. [55] investigates the integration of the neural operator DeepOnet with classical relaxation techniques, resulting in a hybrid iterative approach. Meanwhile, Wu et al. [56] introduce an asymptotic-preserving convolutional DeepOnet designed to capture the diffusive characteristics of multiscale linear transport equations.

For multiscale PDEs, operator learning methods can be viewed as an advancement beyond multiscale solvers such as numerical homogenization. Operator learning methods have two key advantages: (1) They can be applied to an ensemble of coefficients/parameters, rather than a single set of coefficients, which allows the methods to capture the stochastic behaviors of the coefficients; (2) The decoder in the operator learning framework can be interpreted as a data-driven basis reduction procedure from the latent space (high-dimensional) that approximates the solution data manifold (often lower-dimensional) of the underlying PDEs. This procedure offers automated data-adaptation to the coefficients, enabling accurate representations of the solutions' distributions. In contrast, numerical homogenization typically relies on a priori bases that are not adapted to the ensemble of coefficients. In this regard, the operator learning approach has the potential to yield more accurate reduced-order models for multiscale PDEs with parametric/random coefficients.

However, for multiscale problems, current operator learning methods have primarily focused on representing the smooth parts of the solution space. This results in the so-called "spectral bias", leaving the resolution of intrinsic multiscale features as a significant challenge. The spectral bias, also known as the frequency principle [57–59], states that deep neural networks (DNNs) often struggle to learn high-frequency components of functions that vary at multiple scales. In this regard, Fourier or wavelet-based methods are not always effective for MsPDEs, even for fixed parameters. Neural operators tend to fit low-frequency components faster than high-frequency ones, limiting their ability to accurately capture fine details. When the elliptic coefficients are smooth, the coefficient to solution map can be well resolved by the FNO parameterization [23]. Nevertheless, existing neural operators have difficulty learning high-frequency components of multiscale PDEs, as is shown in Fig. 1 and detailed in Section 3. While the universal approximation theorems can be proven for FNO type models (see e.g., [60]), achieving a meaningful decay rate requires "extra smoothness", which may be absent or lead to large constants for MsPDEs. For FNO, this issue was partially addressed in [61], yet the approach there needs an ad-hoc manual tweak on the weights for the modes chosen.

We note that for fixed parameter MsPDEs, In recent years, there has been increasing exploration of neural network methods for solving multiscale PDEs despite the spectral bias or frequency principle [57–59] indicating that deep neural networks (DNNs) often struggle to effectively capture high-frequency components of functions. Specifically designed neural solvers [62–64] have been developed to mitigate the spectral bias and accurately solve multiscale PDEs with fixed parameters.



(a) multiscale trigonometric coefficient, (b) slices of the derivatives $\frac{\partial u}{\partial y}$ at $x = 0$, (c) absolute error spectrum of HANO in $\log_{10}$ scale (d) absolute error spectrum of FNO in $\log_{10}$ scale.

Fig. 1. We illustrate the effectiveness of the HANO scheme on the challenging multiscale trigonometric benchmark, with the coefficients and corresponding solution derivative shown in (a) and (b), see Appendix 3.1.2 for problem description. We notice that HANO can capture the solution derivatives more accurately, whereas FNO only captures their averaged or homogenized behavior. In (c) and (d), we analyze the error by decomposing it into the frequency domain $[-256\pi, 256\pi]^2$ and plotting the absolute error spectrum. This shows the spectral bias in the existing state-of-the-art model, and also our method achieves superior performance in predicting fine-scale features, especially accurately capturing derivatives. We refer readers to Fig. 7 in Section 3.1 and Figs. 8, 9, 7.

Motivated by aforementioned challenges, we investigate the spectral bias present in existing neural operators. Inspired by conventional multilevel methods and numerical homogenization, we propose a new **H**ierarchical **A**ttention **N**eural **O**perator (HANO) architecture to mitigate it for multiscale operator learning. We also test our model on standard operator learning benchmark including the Navier-Stokes equation in the turbulent regime, and the Helmholtz equation in the high wave number regime. Our main contributions can be summarized as follows:

- We introduce HANO, that decomposes input-output mapping into hierarchical levels in an automated fashion, and enables nested feature updates through hierarchical local aggregation of self-attentions with a controllable linear computational cost.
- We use an empirical H^1 loss function to further reduce the spectral bias and improve the ability to capture the oscillatory features of the multiscale solution space;
- We investigate the spectral bias in the existing neural operators and empirically verify that HANO is able to mitigate the spectral bias. HANO substantially improves accuracy, particularly for approximating derivatives, and generalization for multiscale tasks, compared with state-of-the-art neural operators and efficient attention/transformers.

2. Methods

In this section, to address the spectral bias for multiscale operator learning, and motivated by the remarkable performance of attention-based models [30,65] in computer vision and natural language processing tasks, as well as the effectiveness of hierarchical matrix approach [14] for multiscale problems, we propose the Hierarchical Attention Neural Operator (HANO) model.

2.1. Operator learning problem

We follow the setup in [23,21] to approximate the operator $\mathcal{S} : a \mapsto u := \mathcal{S}(a)$, with the input/parameter $a \in \mathcal{A}$ drawn from a distribution μ and the corresponding output/solution $u \in \mathcal{U}$, where $\mathcal{A}$ and $\mathcal{U}$ are infinite-dimensional Banach spaces, respectively. Our aim is to learn the operator $\mathcal{S}$ from a collection of finitely observed input-output pairs through a parametric map $\mathcal{N} : \mathcal{A} \times \Theta \to \mathcal{U}$ and a loss functional $\mathcal{L} : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, such that the optimal parameter

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{a \sim \mu} \left[\mathcal{L} \left(\mathcal{N}(a, \theta), \mathcal{S}(a) \right) \right].$$

2.1.1. Hierarchical discretization

To develop a hierarchical attention, first we assume that there is a hierarchical discretization of the spatial domain D. For an input feature map that is defined on a partition of D, for example, of resolution 8×8 patches, we define $\mathcal{I}^{(3)} := \{i = (i_1, i_2, i_3) \,|\, i_1, i_2, i_3 \in \{0, 1, 2, 3\}\}$ as the finest level index set, in which each index i corresponds to a patch *token* characterized by a feature vector $\mathbf{f}_i^{(3)} \in \mathbb{R}^{C^{(3)}}$. For a token $i = (i_1, i_2, i_3)$, its parent token $j = (i_1, i_2)$ aggregates finer level tokens (e.g., $(1,1)$ is the parent of $(1,1,0), (1,1,1), (1,1,2), (1,1,3)$ in Fig. 2), characterized by a feature vector $\mathbf{f}_j^{(2)} \in \mathbb{R}^{C^{(2)}}$. We postpone describing the aggregation scheme in the following paragraph. In general, we write $\mathcal{I}^{(m)} := \{i = (i_1, i_2, ..., i_m) \,|\, i_\ell \in \{0, 1, 2, 3\} \text{ for } \ell = 1, ..., m\}$ as the index set of m-th level tokens and $\mathcal{I}^{(r)}$ denotes the index set of the finest level tokens. Note that the hierarchy is not restricted to quadtree setting.

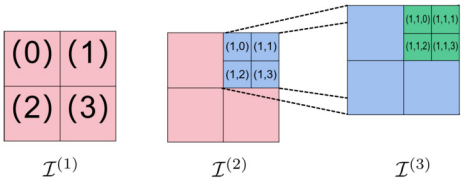


Fig. 2. Hierarchical discretization and index tree. The 2D unit square is discretized hierarchically into three levels with corresponding index sets $\mathcal{I}^{(1)}, \mathcal{I}^{(2)}$, and $\mathcal{I}^{(3)}$. To illustrate, $(1)^{(1,2)}$ represents the second level child nodes of node (1) and is defined as $(1)^{(1,2)} = \{(1,0),(1,1),(1,2),(1,3)\}$.

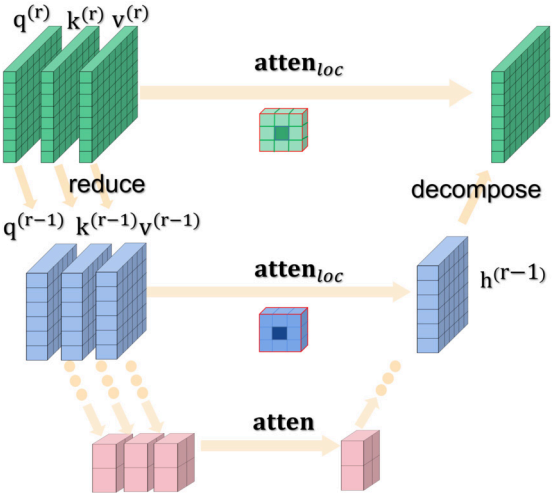


Fig. 3. Hierarchically nested attention.

2.2. Vanilla attention mechanism

In this section, we first revisit the vanilla scaled dot-product attention mechanism for a single-level discretization. For example, the finest level tokens, denoted as $\mathbf{f}_i^{(r)} \in \mathbb{R}^{C^{(r)}}$, are indexed by $i \in \mathcal{I}^{(r)}$. The token aggregation formula on this level can then be expressed as:

$$\textbf{atten}: \mathbf{h}_i^{(r)} = \sum_{j \in \mathcal{I}^{(r)}} \mathcal{G}(\mathbf{q}_i^{(r)}, \mathbf{k}_j^{(r)}) \mathbf{v}_j^{(r)}, \tag{1}$$

where $\mathbf{q}_i^{(r)} = \mathbf{W}^Q \mathbf{f}_i^{(r)}$, $\mathbf{k}_i^{(r)} = \mathbf{W}^K \mathbf{f}_i^{(r)}$, $\mathbf{v}_i^{(r)} = \mathbf{W}^V \mathbf{f}_i^{(r)}$, and $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V \in \mathbb{R}^{C^{(r)} \times C^{(r)}}$ are learnable matrices. Here, for simplicity, we use the function $\mathcal{G}$ represents a pairwise interaction between queries and keys in the self-attention mechanism. Note that in conventional self-attention mechanism, the pairwise interaction potential is defined by $\mathcal{G}(\mathbf{q}_i^{(r)}, \mathbf{k}_j^{(r)}) := \exp(\mathbf{q}_i^{(r)} \cdot \mathbf{k}_j^{(r)})/C^{(r)}$ and further normalized by a row scaling factor via softmax function. To be more specific, the vanilla self-attention is defined by

$$\textbf{vanilla atten}: \mathbf{h}_i^{(r)} = \sum_{j \in \mathcal{I}^{(r)}} \frac{\mathcal{G}(\mathbf{q}_i^{(r)}, \mathbf{k}_j^{(r)})}{\sum_{j \in \mathcal{I}^{(r)}} \mathcal{G}(\mathbf{q}_i^{(r)}, \mathbf{k}_j^{(r)})} \mathbf{v}_j^{(r)}. \tag{2}$$

2.3. Hierarchical attention

In this section, we present HANO in Algorithm 1, a hierarchically nested attention scheme with $\mathcal{O}(N)$ cost inspired by $\mathscr{H}^2$ matrices [66], which is much more efficient than the vanilla attention above that scales with $\mathcal{O}(N^2)$. The overall HANO scheme (e.g., for a three-level example see Fig. 3) resembles the V-cycle operations in multigrid methods, and it comprises four key operations: reduce, multilevel local attention and decompose&mix. In this procedure, instead of using global attention aggregation as in equation (1), we utilize a local aggregation formula inspired by the $\mathcal{H}$ matrix approximation in the step of multilevel local attention. This approximation decomposes global interactions into local interactions at different scales (levels of tokens, denoted by m of $\mathcal{I}^{(m)}$). Empirically in Section 3, this decomposition has a very minimal loss of expressivity.

2.3.1. *Reduce operation* using the quadtree hierarchy

The *reduce* operation aggregates finer-level tokens into coarser-level tokens in the hierarchy. We denote $i^{(m,m+1)}$ as the set of indices of the $(m+1)$-th level child tokens of the m-th level token i, where $i \in \mathcal{I}^{(m)}$. In the quadtree case, $i^{(m,m+1)} = \{(i,0),(i,1),(i,2),(i,3)\}$, where (i,j) is the concatenation of i and $0 \le j \le 3$. The reduce map can be defined as

$$\mathbf{q}_i^{(m)} = \mathcal{R}^{(m)}(\{\mathbf{q}_j^{(m+1)}\}_{j\in i^{(m,m+1)}}),$$

which maps the $(m+1)$-th level tokens with indices in $i^{(m,m+1)}$ to the m-th level token i. We implement $\mathcal{R}^{(m)}$ as a linear layer, namely,

$$\mathbf{q}_i^{(m)} = \mathbf{R}_0^{(m)}\mathbf{q}_{(i,0)}^{(m+1)} + \mathbf{R}_1^{(m)}\mathbf{q}_{(i,1)}^{(m+1)} + \mathbf{R}_2^{(m)}\mathbf{q}_{(i,2)}^{(m+1)} + \mathbf{R}_3^{(m)}\mathbf{q}_{(i,3)}^{(m+1)},$$

where $\mathbf{R}_0^{(m)}, \mathbf{R}_1^{(m)}, \mathbf{R}_2^{(m)}, \mathbf{R}_3^{(m)} \in \mathbb{R}^{C^{(m-1)}\times C^{(m)}}$ are matrices. The reduce operation is applied to $\mathbf{q}_i^{(m)}$, $\mathbf{k}_i^{(m)}$, and $\mathbf{v}_i^{(m)}$ for any $i \in \mathcal{I}^{(m)}$, and $m = r-1, \cdots, 1$. This step corresponds to the downwards arrow in Fig. 3.

2.3.2. Multilevel local attention

Instead of using global attention aggregation as in equation (1), we utilize a local aggregation formula in each single level. The local aggregation at the m-th level $\mathbf{atten}_{\text{loc}}^{(m)}$ is written using the nested $\mathbf{q}_i^{(m)}, \mathbf{k}_j^{(m)}, \mathbf{v}_j^{(m)}$ for $m = r, \ldots, 1$ as follows: for $i \in \mathcal{I}^{(m)}$,

$$\mathbf{atten}_{\text{loc}}^{(m)} : \mathbf{h}_i^{(m)} = \sum_{j\in\mathcal{N}^{(m)}(i)\cup i} \mathcal{G}(\mathbf{q}_i^{(m)} \cdot \mathbf{k}_j^{(m)})\mathbf{v}_j^{(m)}, \tag{3}$$

where $\mathcal{N}^{(m)}(i)$ denotes the set of m-th level neighbors of $i \in \mathcal{I}^{(m)}$. We define $\mathcal{N}^{(m)}(i)$ as the set of tokens within a specific window centered on the i-th token with a fixed window size for each level. This configuration ensures that attention aggregation mirrors the localized scope characteristic of convolution operations.

2.3.3. Decompose&mix operation using the quadtree hierarchy

The *decompose* operation reverses the reduce operation from level 1 to level $r-1$. The decompose operator $\mathcal{D}^{(m)} : \mathbf{h}_i^{(m)} \mapsto \{\tilde{\mathbf{h}}_j^{(m+1)}\}_{j\in i^{(m,m+1)}}$, maps the m-th level feature $\mathbf{h}_i^{(m)}$ with index i and $1 \le m \le r-1$ to $(m+1)$-th level tokens associated to its child set $i^{(m,m+1)}$. The presentation above provides an equivalent matrix form of $\mathcal{R}^{(m)}$ and $\mathcal{D}^{(m)}$ from fine to coarse levels. $\tilde{\mathbf{h}}_i^{(m+1)}$ is further aggregated to $\mathbf{h}_i^{(m+1)}$ in the *mix* operation such that $\mathbf{h}_i^{(m+1)} += \tilde{\mathbf{h}}_i^{(m+1)}$ for $i \in \mathcal{I}^{(m+1)}$. In the current implementation, we use a simple linear layer such that $\tilde{\mathbf{h}}_{(i,s)}^{(m+1)} = \mathbf{D}_s^{(m),T}\mathbf{h}_i^{(m)}$, for $s = 0, 1, 2, 3$, with parameter matrices $\mathbf{D}_s^{(m)} \in \mathbb{R}^{C^{(m)}\times C^{(m+1)}}$.

At this point, we can summarize the hierarchically nested attention algorithm as follows.

Algorithm 1 Hierarchically Nested Attention.

Input: $\mathcal{I}^{(r)}$, $\mathbf{f}_i^{(r)}$ for $i \in \mathcal{I}^{(r)}$.
STEP 0: Compute $\mathbf{q}_i^{(r)}$, $\mathbf{k}_i^{(r)}$, $\mathbf{v}_i^{(r)}$ for $i \in \mathcal{I}^{(r)}$.
STEP 1: For $m = r-1, \cdots, 1$, **Do** the reduce operations $\mathbf{q}_i^{(m)} = \mathcal{R}^{(m)}(\{\mathbf{q}_j^{(m+1)}\}_{j\in i^{(m,m+1)}})$ and also for $\mathbf{k}_i^{(m)}$ and $\mathbf{v}_i^{(m)}$, for any $i \in \mathcal{I}^{(m)}$.
STEP 2: For $m = r, \cdots, 1$, **Do** the local aggregation by equation (3) to compute $\mathbf{h}_i^{(m)}, m = 1, ..., r$, for any $i \in \mathcal{I}^{(m)}$.
STEP 3: For $m = 1, \cdots, r-1$, **Do** the decompose operations $\{\tilde{\mathbf{h}}_j^{(m+1)}\}_{j\in i^{(m,m+1)}} = \mathcal{D}^{(m)}(\mathbf{h}_i^{(m)})$, for any $i \in \mathcal{I}^{(m)}$; then $\mathbf{h}_i^{(m+1)} += \tilde{\mathbf{h}}_i^{(m+1)}$, for any $i \in \mathcal{I}^{(m+1)}$.
Output: $\mathbf{h}_i^{(r)}$ for any $i \in \mathcal{I}^{(r)}$.

2.3.4. Hierarchical matrix perspective

The hierarchically nested attention in Algorithm 1 resembles the celebrated hierarchical matrix method [66], in particular, the $\mathscr{H}^2$ matrix from the perspective of matrix operations. In the following, we take the one-dimensional binary tree-like hierarchical discretization shown in Fig. 4 as an example to illustrate the reduce operation, decompose operation, and multilevel token aggregation in **STEP 0-4** of Algorithm 1 using matrix representations.

STEP 0 Given the input features $\mathbf{f}^{(r)}$, compute the queries $\mathbf{q}_i^{(r)}$, keys $\mathbf{k}_j^{(r)}$, and values $\mathbf{v}_i^{(r)}$ for $j \in \mathcal{I}^{(r)}$.

Starting from the finest level features $\mathbf{f}_i^{(r)}, i \in \mathcal{I}^{(r)}$, the queries $\mathbf{q}^{(r)}$ can be obtained by

$$\left[\begin{array}{c}\vdots\\ \mathbf{q}_i^{(r)}\\ \vdots\end{array}\right] = \underbrace{\left[\begin{array}{ccc}\mathbf{W}^{Q,(r)} & & \\ & \mathbf{W}^{Q,(r)} & \\ & & \ddots \\ & & \mathbf{W}^{Q,(r)}\end{array}\right]}_{|\mathcal{I}^{(r)}|} \left[\begin{array}{c}\vdots\\ \mathbf{f}_i^{(r)}\\ \vdots\end{array}\right] \Bigg\} |\mathcal{I}^{(r)}|,$$

and for the keys $\mathbf{k}^{(r)}$ and values $\mathbf{v}^{(r)}$, similar procedures follow.

STEP 1 **For** $m = r-1:1$, **Do** the reduce operations $\mathbf{q}_i^{(m)} = \mathcal{R}^{(m)}(\{\mathbf{q}_j^{(m+1)}\}_{j\in i^{(m,m+1)}})$ and also for $\mathbf{k}_i^{(m)}$ and $\mathbf{v}_i^{(m)}$, for any $i \in \mathcal{I}^{(m)}$.

If $\mathcal{R}^{(m)}$ is linear, the reduce operations correspond to $\left[\begin{array}{c}\vdots\\ \mathbf{q}_i^{(m)}\\ \vdots\end{array}\right] = R^{(m)}\left[\begin{array}{c}\vdots\\ \mathbf{q}_i^{(m+1)}\\ \vdots\end{array}\right]$. In matrix form, the reduce operation is given by multiplying with

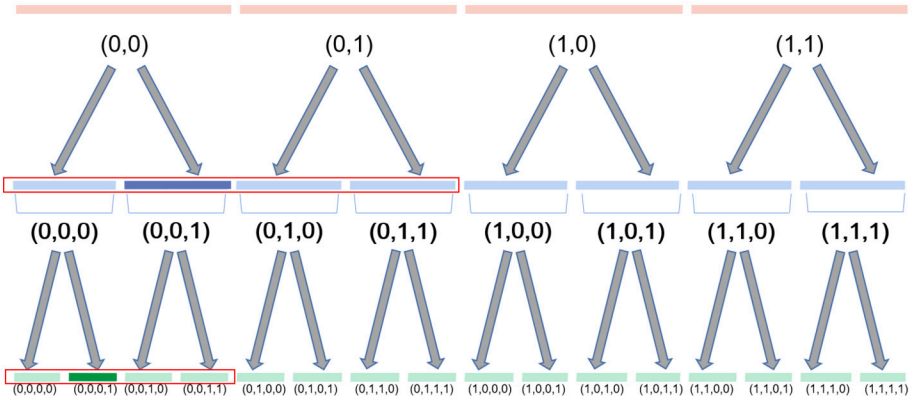


Fig. 4. Hierarchical discretization of 1D domain. The coarsest level partition is plotted as the top four segments in pink. The segment $(0,0)$ is further partitioned into two child segments $(0,0,0)$ and $(0,0,1)$. During the reducing process, the computation proceeds from bottom to top to obtain coarser level representations. For example, the $(0,0)$ representations are obtained by applying learnable reduce operations $\mathcal{R}^{(2)}$ and $\mathcal{R}^{(1)}$ on $(0,0,0)$ and $(0,0,1)$ respectively. When generating the high-resolution representations, the computation proceeds from top to bottom by applying learnable decomposition operations $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(2)}$. The red frames show examples of attention windows at each level.

$$R^{(m)} := \underbrace{\begin{bmatrix} \mathbf{R}_0^{(m)} & \mathbf{R}_1^{(m)} & & & & & \\ & & \mathbf{R}_0^{(m)} & \mathbf{R}_1^{(m)} & & & \\ & & & & \ddots & \ddots & \\ & & & & & \mathbf{R}_0^{(m)} & \mathbf{R}_1^{(m)} \end{bmatrix}}_{|\mathcal{I}^{(m+1)}|} \Big\} |\mathcal{I}^{(m)}|,$$

and $\mathbf{R}_0^{(m)}, \mathbf{R}_1^{(m)} \in \mathbb{R}^{C(m-1)\times C(m)}$ are matrices parametrized by linear layers. In practice, queries, keys, and values use different $\mathbf{R}_0^{(m)}, \mathbf{R}_1^{(m)}$ to enhance the expressivity. In general, these operators $\mathcal{R}^{(m)}$ are not limited to linear operators. The composition of nonlinear activation functions would help increase the expressivity. The nested learnable operators $\mathcal{R}^{(m)}$ also induce the channel mixing and are equivalent to a structured parameterization of $\mathbf{W}^Q, \mathbf{W}^V, \mathbf{W}^K$ matrices for the coarse level tokens, in the sense that, inductively,

$$\begin{aligned} \begin{bmatrix} \vdots \\ \mathbf{q}_i^{(m)} \\ \vdots \end{bmatrix} &= R^{(m)} \cdots R^{(r-1)} \begin{bmatrix} \vdots \\ \mathbf{q}_i^{(r)} \\ \vdots \end{bmatrix} \\ &= R^{(m)} \cdots R^{(r-1)} \begin{bmatrix} \mathbf{W}^{Q,(r)} & & & \\ & \mathbf{W}^{Q,(r)} & & \\ & & \ddots & \\ & & & \mathbf{W}^{Q,(r)} \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{f}_i^{(r)} \\ \vdots \end{bmatrix}. \end{aligned} \tag{4}$$

STEP 2 With the m-th level queries and keys, we can calculate the local attention matrix $\mathbf{G}_{\text{loc}}^{(m)}$ at the m-th level with $(\mathbf{G}_{\text{loc}}^{(m)})_{i,j} := \exp(\mathbf{q}_i^{(m)} \cdot \mathbf{k}_j^{(m)})$ for $i \in \mathcal{N}^{(m)}(j)$, or $i \sim j$ using the following local aggregation at the m-th level $\mathbf{atten}_{\text{loc}}^{(m)}$. For the nested $\mathbf{q}_i^{(m)}, \mathbf{k}_j^{(m)}, \mathbf{v}_j^{(m)}$ for $m = r, \ldots, 1$, the local aggregation is for $i \in \mathcal{I}^{(m)}$,

$$\mathbf{atten}_{\text{loc}}^{(m)} : \mathbf{h}_i^{(m)} = \sum_{j \in \mathcal{N}^{(m)}(i) \cup i} \mathcal{G}(\mathbf{q}_i^{(m)}, \mathbf{k}_j^{(m)}) \mathbf{v}_j^{(m)}, \tag{5}$$

where $\mathcal{N}^{(m)}(i)$ is the set of m-th level neighbors of $i \in \mathcal{I}^{(m)}$.

STEP 3 The decompose operations, opposite to the reduce operations, correspond to the transpose of the following matrix in the linear case,

$$D^{(m)} := \underbrace{\begin{bmatrix} \mathbf{D}_0^{(m)} & \mathbf{D}_1^{(m)} & & & & & \\ & & \mathbf{D}_0^{(m)} & \mathbf{D}_1^{(m)} & & & \\ & & & & \ddots & \ddots & \\ & & & & & \mathbf{D}_0^{(m)} & \mathbf{D}_1^{(m)} \end{bmatrix}}_{|\mathcal{I}^{(m+1)}|} \Big\} |\mathcal{I}^{(m)}|.$$

The m-th level aggregation in Fig. 3 contributes to the final output $\mathbf{f}^{(r)}$ in the form

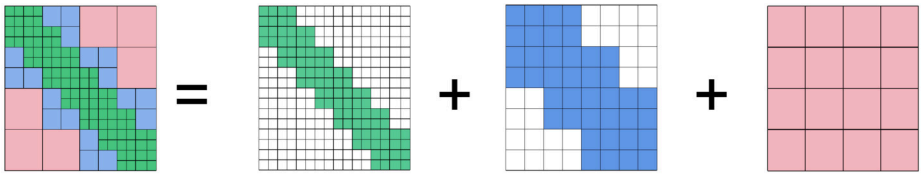


Fig. 5. A demonstration of the decomposition of attention matrix into three levels of local attention matrix.

$$D^{(r-1),T}\cdots D^{(m),T}\mathbf{G}_{\text{loc}}^{(m)}R^{(m)}\cdots R^{(r-1)}\begin{bmatrix}\vdots\\ \mathbf{v}_i^{(r)}\\ \vdots\end{bmatrix}.$$

Eventually, aggregations at all r levels in one V-cycle can be summed up as

$$\begin{bmatrix}\vdots\\ \mathbf{h}_i^{(r)}\\ \vdots\end{bmatrix}=\left(\sum_{m=1}^{r-1}(D^{(r-1),T}\cdots D^{(m),T}\mathbf{G}_{loc}^{(m)}R^{(m)}\cdots R^{(r-1)})+\mathbf{G}_{\text{loc}}^{(r)}\right)\begin{bmatrix}\vdots\\ \mathbf{v}_i^{(r)}\\ \vdots\end{bmatrix}. \tag{6}$$

The hierarchical attention matrix

$$\mathbf{G}_h:=\sum_{m=1}^{r-1}(D^{(r-1),T}\cdots D^{(m),T}\mathbf{G}_{loc}^{(m)}R^{(m)}\cdots R^{(r-1)})+\mathbf{G}_{\text{loc}}^{(r)},$$

in equation (6) resembles the three-level $\mathscr{H}^2$ matrix decomposition illustrated in Fig. 5 (see also [66] for a detailed description). The sparsity of $\mathbf{G}_h$ lies in the fact that the attention matrix is only computed for pairs of tokens within the neighbor set. The $\mathscr{H}^2$ matrix-vector multiplication in equation (6) implies the $\mathcal{O}(N)$ complexity of Algorithm 1.

Note that, the local attention matrix at level $\mathcal{I}^{(1)}$ (pink), level $\mathcal{I}^{(2)}$ (blue) and level $\mathcal{I}^{(3)}$ (green) are $\mathbf{G}_{loc}^{(1)}$, $\mathbf{G}_{loc}^{(2)}$ and $\mathbf{G}_{loc}^{(3)}$, respectively. However, when considering their contributions to the finest level, they are equivalent to the attention matrix $D^{(2),T}D^{(1),T}\mathbf{G}_{loc}^{(1)}R^{(1)}R^{(2)}\in\mathbb{R}^{\mathcal{I}^{(3)}}\times\mathbb{R}^{\mathcal{I}^{(3)}}$ (pink), $D^{(2),T}\mathbf{G}_{loc}^{(2)}R^{(2)}\in\mathbb{R}^{\mathcal{I}^{(3)}}\times\mathbb{R}^{\mathcal{I}^{(3)}}$ (blue) and $\mathbf{G}_{loc}^{(3)}$ (green), as demonstrated in Fig. 5. Each pink block and blue block are actually low-rank sub-matrices with rank $C^{(1)}$ and rank $C^{(2)}$, respectively, by definition.

2.3.5. Complexity

We conclude the section by estimating the complexity of Algorithm 1.

Proposition 2.1. *The reduce operation, multilevel local attention, and decomposition/mix operations together form a V-cycle for updating tokens, as illustrated in Fig. 3. The cost of one V-cycle is $O(N)$ if $\mathcal{I}$ is a quadtree, as implemented in the paper.*

Proof. For each level m, the cost to compute equation (3) is $c(|\mathcal{I}^{(m)}|C^{(m)})$ since for each $i\in\mathcal{I}^{(m)}$ the cardinality of the neighbor set $\mathcal{N}^{(m)}(i)$ is bounded by a constant c. The reduce operation $\mathbf{f}_i^{(k-1)}=\mathcal{R}^{(k-1)}(\{\mathbf{f}_j^{(m)}\}_{j\in i^{(k-1,k)}})$ costs at most $|\mathcal{I}^{(m)}|C^{(m)}C^{(k-1)}$ flops and so does the decompose operation at the same level. Therefore, for each level, the operation cost is $c(|\mathcal{I}^{(m)}|C^{(m)})+2|\mathcal{I}^{(m)}|C^{(m)}C^{(m-1)}$. When $\mathcal{I}$ is a quadtree, $\mathcal{I}^{(r)}=N$, $\mathcal{I}^{(r-1)}=N/4,\cdots,\mathcal{I}^{(1)}=4$, therefore the total computational cost $\sim\mathcal{O}(N)$. □

2.4. Overall architecture

The overall neural network architecture uses the standard Transformer [32] architecture for computer vision tasks, and the HANO attention is a drop-in replacement of the attention mechanism therein. The input $\mathbf{a}$ is first embedded into $n\times n$ tokens represented as a tensor of size $n\times n\times C^{(r)}$ using patch embedding, for a dataset with resolution $n_f\times n_f$, such as in the multiscale elliptic equation benchmark. These tokens are then processed by a multi-level hierarchically nested attention, as described in Section 2, resulting in hidden features $\mathbf{h}_i^{(r)}, i\in\mathcal{I}^{(r)}$. Finally, a decoder maps the hidden features to the solution $\mathbf{u}$. Different decoders can be employed depending on prior knowledge of the PDE model. For example, [21] uses a simple feedforward neural network (FFN) to learn a "basis" set, [67] employs a data-driven SVD-based decoder, and in our work, the compose and mix operations function as the decoder.

In this paper, we choose $r=5$ as the depth of the HANO, window size of 3×3 for the definition of the neighborhood $\mathcal{N}^{(\cdot)}(\cdot)$ in equation (3), GELU as the activation function, and a CNN-based patch embedding module to transfer the input data into features/tokens.

For a dataset with resolution $n_f\times n_f$, such as in the multiscale elliptic equation benchmark 3.1, the input feature $\mathbf{f}^{(5)}$ is represented as a tensor of size $n\times n\times C$ via patch embedding. The self-attention is first computed within a local window on level 5. Then the reduce layer concatenates the features of each group of 2×2 neighboring tokens and applies a linear transformation on the $4C$-dimensional concatenated features on $\frac{n}{2}\times\frac{n}{2}$ level 2 tokens, to obtain level 2 features $\mathbf{f}^{(2)}$ as a tensor of the size $\frac{n}{2}\times\frac{n}{2}\times 2C$. The procedure is repeated from level 2 to level 1 with $\mathbf{f}^{(1)}$ of size $\frac{n}{4}\times\frac{n}{4}\times 4C$.

Table 1
Hyperparameters configurations.

Module	Hyperparameters
Patch embedding	patch size: 4, padding: 0
Hierarchical Attention	number of levels: 5 down sampling ratio $\lvert\mathcal{I}^{m+1}\rvert/\lvert\mathcal{I}^{m}\rvert$: 4 feature dimension at each level: $\{32, 32, 32, 32, 32\}$ window size at each level: $\{3, 3, 3, 3, 3\}$ LayerNorm position: after attention number of cycles: 2

For the decompose operations, starting at level 1, a linear layer is applied to transform the $4C$-dimensional features $\mathbf{f}^{(1)}$ into $8C$-dimensional features. Each level 1 token with $8C$-dimensional features is decomposed into four level 2 tokens with $2C$-dimensional features. These four level 2 tokens are added to the existing level 2 feature $\mathbf{f}^{(2)}$ with output size of $\frac{n}{2} \times \frac{n}{2} \times 2C$. The decomposition procedure is repeated from level 2 to level 3. The output of level 3 is $\mathbf{f}^{(3)}$, which has a size of $n \times n \times C$. We call the above procedures a cycle and we repeat k cycles by the same set up with layer normalizations between cycles.

The detailed configuration for HANO, which may consist of different levels and feature dimensions for each task, is presented in Table 1.

2.5. Comparison with existing multilevel transformers

In vision transformers like [65,68] with a multilevel architecture, attentions are performed at each level separately, resulting in no multilevel attention-based aggregation. This may lead to the loss of fine-scale information in the coarsening process, which is not ideal for learning multiscale operators where fine-scale features are crucial. The following components in the HANO approach we proposed could potentially address this issue:

(1) Attention-based local aggregations at each level, followed by summation of features from all levels to form the updated fine-scale features;
(2) The reduce/local aggregation/decompose/mix operations, inspired by the $\mathcal{H}^2$ hierarchical matrix method, enable the recovery of fine details with a linear cost;
(3) Nested computation of features at all levels, with simultaneous parameterization of the learnable matrices $\mathbf{W}^Q, \mathbf{W}^V, \mathbf{W}^K$ in a nested manner. Those components highlight the novelty of our method.

Meanwhile, the nested token calculation approach also differs from existing multilevel vision transformers [65,68], as we perform the reduce operation before attention aggregation, resulting in nested $\mathbf{q}, \mathbf{k}, \mathbf{v}$ tokens. Additionally, our approach differs from UNet [69], which utilizes a maxpooling for the reduce operation. For a numerical ablation study in which UNet and SWIN have the same general architecture, but different ways to aggregate features in each level, please refer to Table 2.

Our attention matrix has a global interaction range but features low-rank off-diagonal blocks at each level, as shown in Section 2.3. Note that the overall attention matrix itself is not necessarily low-rank, distinguishing it from efficient attention models using kernel tricks or low-rank projections [70–74].

3. Experiments

In this section, we tested HANO's evaluation accuracy and efficiency compared to other state-of-the-art neural operators and other Transformers in several standard operator learning benchmarks. In Section 3.1, a new operator learning benchmark is created for solving multiscale elliptic PDEs, and common neural operators are tested. HANO demonstrates higher accuracy and robustness for coefficients with different degrees of roughness/multiscale features. In Section 3.4, HANO is tested in the Navier-Stokes equation benchmark problem with a high Reynolds number. In Section 3.5, HANO is tested in a benchmark for the Helmholtz equation.

3.1. Data generation for porous media benchmark

We apply the HANO model to learn the mapping from coefficient functions to solution operators for multiscale elliptic equations. We use the porous media benchmark with a two-phase coefficient produced from a log-normal random field following e.g. [75,76], and popularized by [77,23] as a standard task in operator learning. The model equation in divergence form writes

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } D \\ u = 0 & \text{on } \partial D \end{cases} \tag{7}$$

where the coefficient $0 < a_{\min} \le a := a(x) \le a_{\max}, \forall x \in D$, and the forcing term $f \in L^2(D; \mathbb{R})$. By the Lax-Milgram lemma, the coefficient to solution map $\mathcal{S} : L^\infty(D; \mathbb{R}_+) \to H_0^1(D; \mathbb{R})$, $u \mapsto \mathcal{S}(a)$ is well-defined.

Table 2
The baseline methods are implemented with their official implementation if publicly available. Performance is measured with relative L^2 errors ($\times 10^{-2}$) and relative H^1 errors ($\times 10^{-2}$). For the Darcy rough case, we run each experiment 3 times to calculate the mean and the standard deviation (after $\pm$) of relative L^2 and relative H^1 errors. All experiments use a fixed train-val-test split setup, see Section 3.2 for details.

Model	Runtime (s)	Darcy smooth		Darcy rough		Multiscale	
		L^2	H^1	L^2	H^1	L^2	H^1
FNO2D	**7.278**	0.706	3.131	1.782 ±0.021	9.318 ±0.088	1.949	14.535
FNO2D H^1	7.391	0.684	2.583	1.613 ±0.010	7.516 ±0.049	1.800	9.619
UNet	9.127	2.169	4.885	3.591 ±0.127	6.479 ±0.311	1.425	5.012
U-NO	11.259	0.678	2.580	1.185 ±0.005	5.695 ±0.005	1.350	8.577
U-NO H^1	11.428	0.492	1.276	1.023 ±0.013	3.784 ±0.016	1.187	5.380
MWT	19.715	—	—	1.138 ±0.010	4.107 ±0.008	1.021	7.245
GT	38.219	0.945	3.365	1.790 ±0.012	6.269 ±0.418	1.052	8.207
SWIN	41.417	—	—	1.622 ±0.047	6.796 ±0.359	1.489	13.385
HANO L^2	9.620	0.490	1.311	0.931 ±0.021	2.612 ±0.059	0.842	4.842
HANO H^1	9.620	0.218	0.763	0.343±0.006	1.846±0.023	0.580	1.749

— MWT [24] only supports resolution with powers of two.

We also include experiments for multiscale trigonometric coefficients with higher contrast. The newly generated benchmark using rough or multiscale coefficient $a(x)$ test the capacity of neural operators to capture fast oscillation (e.g. the diffusion coefficient becomes $a_\epsilon(x) = a(x/\epsilon)$ with $\epsilon \ll 1$), higher contrast ratio $a_{\max}/a_{\min}$, and even a continuum of non-separable scales. See Section 3.1.2 for details. Results for three benchmarks are summarized in Table 2.

3.1.1. Two-phase coefficient

The two-phase coefficients $\{a\}$ and approximations to solutions $\{u\}$ in Section 3.1 are generated according to https://github.com/zongyi-li/fourier_neural_operator/tree/master/data_generation as a standard benchmark. The forcing term is fixed as $f(x) \equiv 1$ in D. The coefficients $a(x)$ are generated according to $a \sim \mu := \psi_{\#}\mathcal{N}\left(0, (-\Delta + cI)^{-2}\right)$, where the covariance is inverting this elliptic operator with zero Neumann boundary conditions. The mapping $\psi : \mathbb{R} \to \mathbb{R}$ takes the value $a_{\max}$ on the positive part of the real line and $a_{\min}$ on the negative part. The push-forward is defined in a pointwise manner, thus a takes the two values inside D and jumps from one value to the other randomly with likelihood characterized by the covariance. Consequently, $a_{\max}$ and $a_{\min}$ can control the contrast of the coefficient. The parameter c controls the "roughness" of the coefficient; a larger c results in a coefficient with rougher two-phase interfaces, as shown in Fig. 6. Solutions u are obtained by using a second-order finite difference scheme on a staggered grid with respect to a.

In [23] and all subsequent work benchmarking this problem for operator learning, the coefficient is determined using $a_{\max} = 12$, $a_{\min} = 3$, and $c = 9$, which results in a relative simply topology of the interface. In this case, the solutions are also relatively smooth (which is referred to as "*Darcy smooth*"). To show the architectural advantage of HANO, we adjust the parameters to increase the likelihood of the random jumps of the coefficients, which results in much more complicated topology of the interfaces, and solutions generated show more "roughness" (which is referred to as "*Darcy rough*"). See Fig. 6 for an example.

3.1.2. Multiscale trigonometric coefficient

We also consider equation (7) with multiscale trigonometric coefficient adapted from [12], as one of the multiscale elliptic equation benchmarks. The domain D is $(-1, 1)^2$, and the coefficient $a(x)$ is defined as

$$a(x) = \prod_{k=1}^{6}(1 + \frac{1}{2}\cos(a_k \pi(x_1 + x_2)))(1 + \frac{1}{2}\sin(a_k \pi(x_2 - 3x_1))),$$

where a_k is uniformly distributed between 2^{k-1} and $1.5 \times 2^{k-1}$ for each k, and the forcing term is fixed as $f(x) \equiv 1$. The reference solutions are obtained using the linear Lagrange finite element methods on uniform triangulation cut from a 1023×1023 Cartesian grid. Datasets of lower resolution are created by downsampling the higher resolution dataset using bilinear interpolation. The experiment results for the multiscale trigonometric case with different resolutions are shown in Table 2. HANO obtains the best relative L^2 error compared to other neural operators. See Figs. 1 and 8 for illustrations of the coefficient and comparison of the solutions/derivatives at the slice $x = 0$.

3.2. Training setup

We consider pairs of functions $\{(a_j, u_j)\}_{j=1}^N$, where a_j is drawn from a probability measure specified in Sections 3.1.1 and 3.1.2, and $u_j = S(a_j)$. During training and evaluation, a_j and u_j are evaluated pointwisely on a uniform 2D grid $\mathsf{G}^2 := \{(x_1, x_2) = (ih, jh) \mid i, j = 0, \dots, n-1\}$ as matrices $\mathbf{a}_j$ and $\mathbf{u}_j$. We generate the hierarchical index tree $\mathcal{I}$ using a quadtree representation of nodes with depth r, where the finest level objects are pixels or patches aggregated by pixels.

We apply the ADAM optimizer with a maximum learning rate 10^{-3}, weight decay 10^{-4}, and a 1-cycle scheduler from [78]. We choose batch size 8 for experiments in Sections 3.1 and 3.4, and batch size 4 for experiments in Sections 3.1.2 and 3.5.

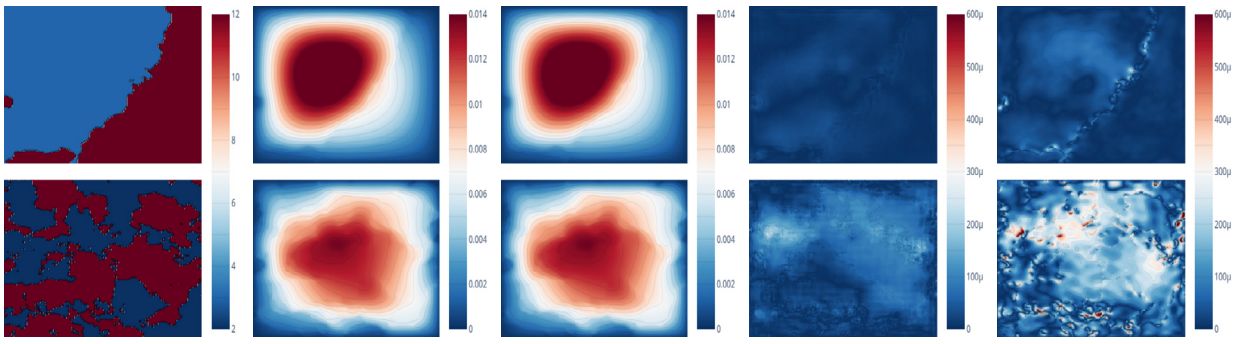


Fig. 6. Top: (a) smooth coefficient in [23], with $a_{\max} = 12$, $a_{\min} = 3$, $c = 9$, (b), reference solution, (c) HANO solution, (d) HANO, absolute (abs.) error, (e) FNO2D, abs. error; **Bottom:** (a) rough coefficients with $a_{\max} = 12$, $a_{\min} = 2$, $c = 20$, (b) reference solution, (c) HANO solution, (d) HANO, abs. error, (e) FNO2D, abs. error, the maximal error of FNO2D is around $900\mu = 9\text{e}{-4}$.

For the Darcy rough case, we use a train-validation-test split of 1280, 112, and 112, respectively, with a max of 500 epochs. For Darcy smooth and multiscale trigonometric cases, we use a split of 1000, 100, and 100, respectively, with a max of 500 epochs for the Darcy smooth case and 300 for the multiscale trigonometric case.

Baseline models are taken from the publicly available official implementations, and changes are detailed in each subsection if there is any. All experiments are run on a NVIDIA A100 GPU.

Empirical H^1 loss function For multiscale problems, we adopt an H^1 loss function instead of the conventional L^2 loss, which places greater emphasis on high-frequency components. Empirically, we observe that the model's training is more efficient and the generalization is more robust than those without. First, the empirical L^2 loss function is defined as

$$\mathcal{L}^L(\{(\mathbf{a}_j, \mathbf{u}_j)\}_{j=1}^N; \theta) := \frac{1}{N} \sum_{i=1}^N \|\mathbf{u}_j - \mathcal{N}(\mathbf{a}_j; \theta)\|_{l^2} / \|\mathbf{u}_j\|_{l^2},$$

where $\|\cdot\|_{l^2}$ is the canonical l^2 vector norm. The normalized discrete Fourier transform (DFT) coefficients of f are given by

$$\mathcal{F}(f)(\xi) := \frac{1}{\sqrt{n}} \sum_{x \in \mathsf{G}^2} f(x) e^{-2i\pi x \cdot \xi}, \quad \xi \in \mathbb{Z}_n^2 := \left\{\xi = (\xi_1, \xi_2) \in \mathbb{Z}^2 \mid -n/2 + 1 \leqslant \xi_j \leqslant n/2, j = 1, 2\right\} \tag{8}$$

The empirical H^1 loss function is thus given by,

$$\mathcal{L}^H(\{(\mathbf{a}_j, \mathbf{u}_j)\}_{j=1}^N; \theta) := \frac{1}{N} \sum_i \|\mathbf{u}_j - \mathcal{N}(\mathbf{a}_j; \theta)\|_h / \|\mathbf{u}_j\|_h, \tag{9}$$

where $\|\mathbf{u}\|_h := \sqrt{\sum_{\xi \in \mathbb{Z}_n^2} |\xi|^2 (\mathcal{F}(\mathbf{u})(\xi))^2}$. $\mathcal{L}^H$ can be viewed as a weighted $\mathcal{L}^L$ loss using $|\xi|^2$ weights to balance the error in low- and high-frequency components. Note that the frequency domain representation of the discrete H^1 norm is used following the practices in e.g. [79,80]. Here the discrete H^1 norm approximated using difference quotient in the physical space can also be employed, however, from the numerical quadrature point of view, by Parseval identity equation (9) is exact with no quadrature error.

3.3. Empirical study on the spectral bias in operator learning

We compare HANO and FNO in terms of prediction error dynamics across frequencies from epoch 0 to epoch 100 (end) in Fig. 7 for a comprehensive comparison. The subfigures (c,d) in Fig. 7 suggest that existing methods can learn low frequencies quickly but struggle with higher frequencies. At the end of the training, plenty of high-frequency components are still not well resolved as shown in Fig. 1(b,c,d) and Fig. 9. This phenomenon is often referred to as the *spectral bias*, well-documented for training neural networks for conventional classification tasks, and here we observe it in operator learning tasks. On the contrary, HANO's error decays faster for higher frequencies and more uniformly overall. It also achieves lower testing errors. Experimentally, the ablation suggests that the hierarchical nested attention allows the model to capture finer-scale variations better, which also helps HANO outperform existing methods as is shown in Fig. 1. We also observe that, with the H^1 loss function, spectral bias is further mitigated, which applies to FNO-based variants as well.

To better illustrate the spectral bias of multiscale operator learning, we record the training dynamics in the frequency domain. Recall that, the spatial domain $D = [0, 1]^2$ is discretized uniformly with $h = 1/n$ to yield a Cartesian grid G^2 in our experiments. For any $\xi \in \mathbb{Z}_n^2$, consider the normalized discrete Fourier transform (DFT) coefficients $\mathcal{F}(f)(\cdot)$ of f in equation (8), the mean absolute prediction error for a given frequency $\xi \in \mathbb{Z}_n^2$ is measured by

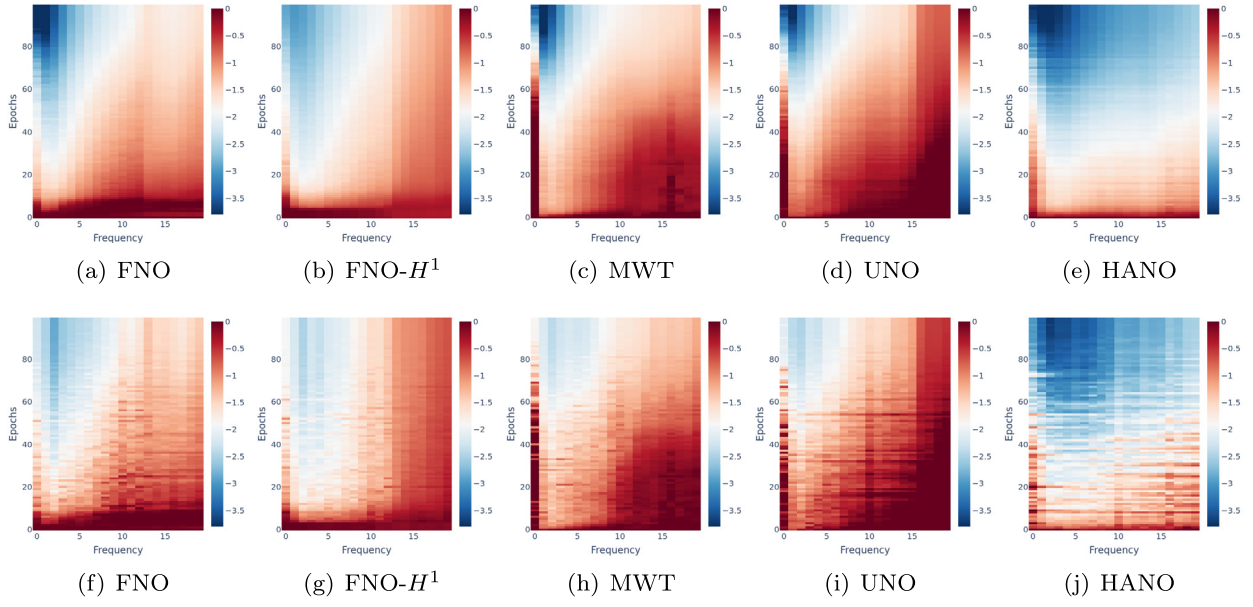


Fig. 7. Top: (a)-(e) show the training error dynamics in the frequency domain. The x-axis shows the first 20 dominating frequencies, from low frequency (left) to high frequency (right). The y-axis shows the number of training epochs. The colorbar shows the normalized L^2 error (with respect to the error at epoch 0) in $\log_{10}$ scale. We compare five different methods; **Bottom:** (f)-(j) Corresponding testing error dynamics in the frequency domain for different methods.

$$\mathcal{E}^{\text{train}}(\mathcal{N};\xi) := \frac{1}{N}\sum_{i=1}^{N_{train}} |\mathcal{F}(\mathbf{u}_i^{\text{train}} - \mathcal{N}(\mathbf{a}_i^{\text{train}}))(\xi)|,$$

$$\mathcal{E}^{\text{test}}(\mathcal{N};\xi) := \frac{1}{N}\sum_{i=1}^{N_{\text{test}}} |\mathcal{F}(\mathbf{u}_i^{\text{test}} - \mathcal{N}(\mathbf{a}_i^{test}))(\xi)|,$$

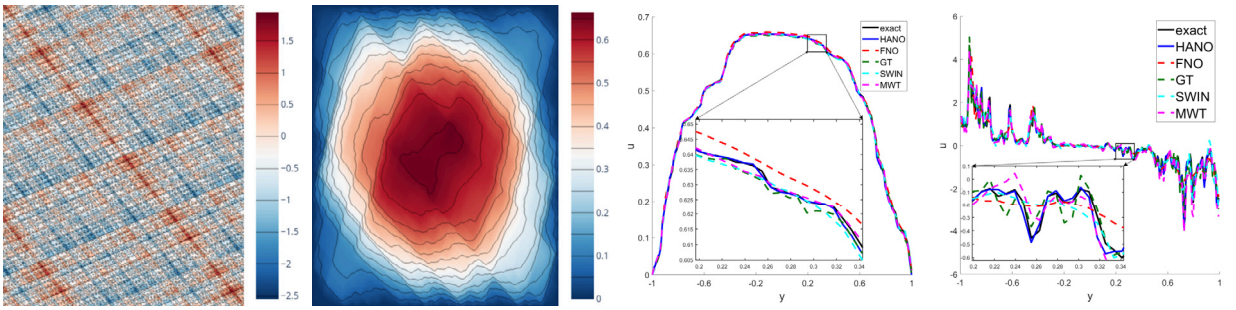
where $\{\mathbf{a}_i^{\text{train}}, \mathbf{u}_i^{\text{train}}\}_{i=1}^{N_{\text{train}}}$ and $\{\mathbf{a}_i^{\text{test}}, \mathbf{u}_i^{\text{test}}\}_{i=1}^{N_{\text{test}}}$ are the training and testing datasets of Darcy rough task. Heuristically, for low frequencies ξ, $\mathcal{E}(\mathcal{N};\xi)$ represents the capability of the neural network for predicting the "global trend". Conversely, for high frequencies ξ, $\mathcal{E}(\mathcal{N};\xi)$ represents the capability for predicting variations on smaller scales. During training, we record $\mathcal{E}^{\text{train}}(\mathcal{N};\xi)$ and $\mathcal{E}^{\text{test}}(\mathcal{N};\xi)$ for each $\xi \in \mathbb{Z}_n^2$ at each epoch.

From Fig. 7, we conclude that existing methods struggle with learning higher frequencies. UNet and UNO mitigate this to some extent, likely due to their UNet-like multi-level structure. HANO's error pattern shows faster decay than others for higher frequencies and is more uniform overall. It also achieves lower testing errors. The mathematical heuristics of MWT, UNet, and UNO can be attributed to multigrid methods [9] and wavelet-based multiresolution methods [10,11]. However, these methods may have limitations for multiscale PDEs [1] because they apply instance-independent kernel integration regardless of the input data (or latent representations). In contrast, attention-based operations in the HANO architecture, which becomes more efficient enabled by the data-driven reduce/decompose operations, have the potential to address this limitation by adapting the kernel to a specific input instance.

Comparison with existing methods Our comprehensive evaluation incorporates several contemporary methods:

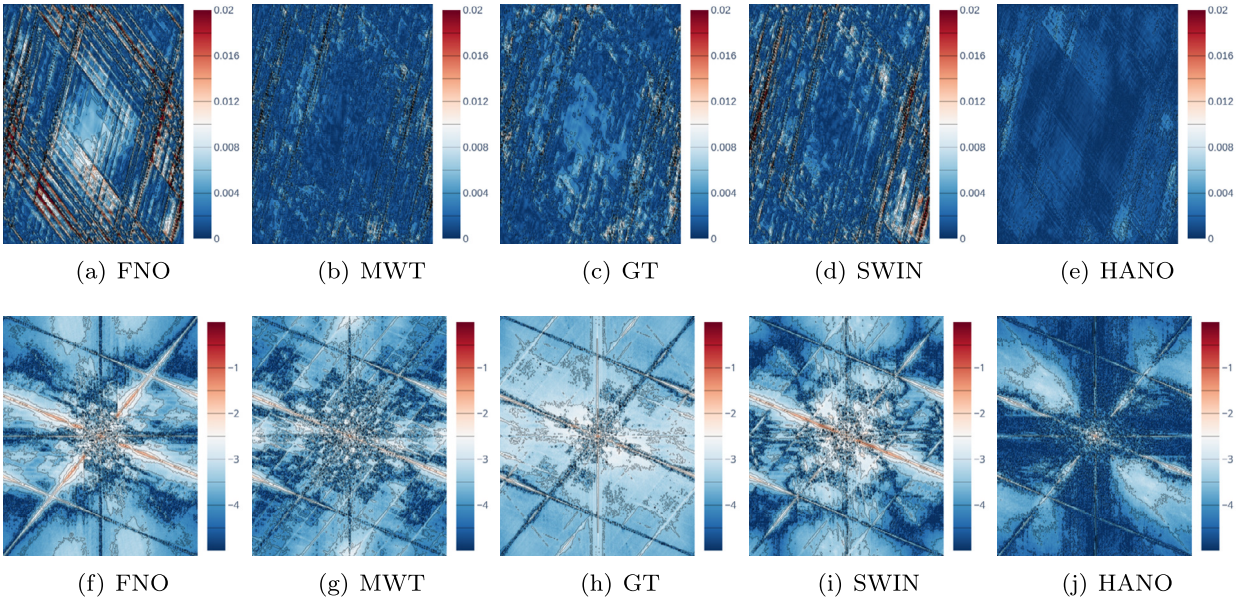
- FNO Variants: The multiwavelet neural operator (MWT) [24], which implements wavelet convolutions on top of FNO's FFT architecture, is also included in this study.
- UNet-based Models: We include the original UNet [69] and U-NO [81], a U-shaped neural operator.
- Transformer-based Neural Operators: We also tested Galerkin Transformer (GT) [35], and SWIN Transformer [65], a general-architectured multiscale vision transformer.
- H^1-loss evaluation ablation study: we train specific models (FNO2D, U-NO) using the H^1 loss to understand its effect, leading to the variants FNO2D H^1 and U-NO H^1. We have also trained a variant of HANO using the L^2 loss function, which we refer to as HANO L^2. This variant, along with the other variants we have discussed, is included in Table 2.

In experiments, we found that HANO outperforms other neural operators in all tasks. The efficacy of the H^1 loss is noticeable across architectures; for instance, observe the performance difference between FNO2D, U-NO and their respective H^1-loss-trained variants. Additionally, the modifications in FNO-CNN notably elevate its performance with only a modest increase in runtime. As depicted in Fig. 6, transitioning from Darcy smooth to Darcy rough and then to multiscale trigonometric problems, the enhancement in high-frequency components highlights HANO's increasing advantage over other methods.



(a) coefficient in $\log_{10}$ scale (b) reference solution in 2D (c) 1D slices of the predicted solutions (d) 1D slices of the predicted solution derivatives

Fig. 8. (a) multiscale trigonometric coefficient, (b) reference solution, (c) comparison of predicted solutions on the slice $x = 0$, (d) comparison of predicted derivative $\frac{\partial u}{\partial y}$ on the slice $x = 0$.



(a) FNO (b) MWT (c) GT (d) SWIN (e) HANO

(f) FNO (g) MWT (h) GT (i) SWIN (j) HANO

Fig. 9. Top: (a)-(e) absolute error of different operator learning methods; **Bottom:** (f)-(j) absolute error spectrum in $\log_{10}$ scale of different operator learning methods.

Comparison of solutions/derivatives for more neural operators We show the coefficient, reference solution from Multiscale trigonometric dataset, and the comparison with other operator learning models such as GT, SWIN, and MWT in Fig. 8. HANO resolves the finer scale oscillations more accurately, as reflected by the predicted derivatives in (d) of Fig. 8.

Comparison of error spectrum for more neural operators In Figs. 1 (c) and (d), we decompose the error into the frequency domain $[-256\pi, 256\pi]^2$ and plot the absolute error spectrum for HANO and FNO. Here, in Fig. 9, we also include the absolute error and absolute error spectrum for other baseline models, such as MWT, GT, and SWIN. The comprehensive comparison also demonstrates that existing methods exhibit the phenomena of spectral bias to some degree. This empirical evidence also demonstrates the reason that HANO has the best accuracy in evaluation.

3.4. Navier-Stokes equation

In this section, we consider the 2D Navier-Stokes equation (NSE) dataset benchmarked in [23]. This dataset contains data generated for NSE in the vorticity-streamfunction formulation approximated by a pseudo-spectral solver with a Crank-Nicholson time stepping on the unit torus T^2. For $x \in \mathsf{T}^2$ and $t \in [0, T]$, $\mathbf{u}(x,t) = \nabla^{\perp}\psi(x,t)$ is the velocity, and $\omega(x,t)$ denotes the vorticity. This formulation then writes

Table 3
Benchmark for the Navier-Stokes equation. 64×64 resolution is used for both training and testing. N is the training sample size, either $N = 1000$ or $N = 10000$. The number of testing samples is 100 or 1000, respectively. We use the L^2 loss function, the Adam optimizer, and the OneCycleLR scheduler with a cosine annealing strategy. The learning rate starts with 1×10^{-3} and decays to 1×10^{-5}. All models were trained for 500 epochs and use L^2 loss function.

Model	#Parameters	$T = 50$ $\nu = 1e-3$ $N = 1000$	$T = 30$ $\nu = 1e-4$ $N = 1000$	$T = 30$ $\nu = 1e-4$ $N = 10000$	$T = 20$ $\nu = 1e-5$ $N = 1000$	$T = 20$ (new) $\nu = 1e-5$ $N = 1000$
FNO-3D	6,558,537	0.0086	0.1918	0.0820	0.1893	—
FNO-2D	2,368,001	0.0128	0.1559	0.0834	0.1556	0.0624
U-Net	24,950,491	0.0245	0.2051	0.1190	0.1982	0.1058
TF-Net	7,451,724	0.0225	0.2253	0.1168	0.2268	0.1241
ResNet	266,641	0.0701	0.2871	0.2311	0.2753	0.1518
DilResNet	586,753	0.0315	0.2561	0.2081	0.2315	0.1641
HANO	7,629,350	**0.0074**	**0.0557**	**0.0179**	**0.0482**	**0.0265**

$$\partial_t \omega + \mathbf{u} \cdot \nabla \omega = \nu \Delta \omega + f,$$
$$\Delta \psi + \omega = 0,$$
$$\omega(x, 0) = \omega_0(x),$$

where ω_0 is the initial vorticity field, ν is the viscosity, f is the rotation of a vector forcing term, and Re is the Reynolds number, defined as $\mathrm{Re} := \frac{\rho u L}{\nu}$ with the density ρ (= 1 here). The length scale of the fluid L is set to 1 here. The Reynolds number is a dimensionless parameter and is inversely proportional to the viscosity ν. The operator to be learned is the approximation to

$$S : \omega(\cdot, 0 \le t \le 9) \to \omega(\cdot, 10 \le t \le T),$$

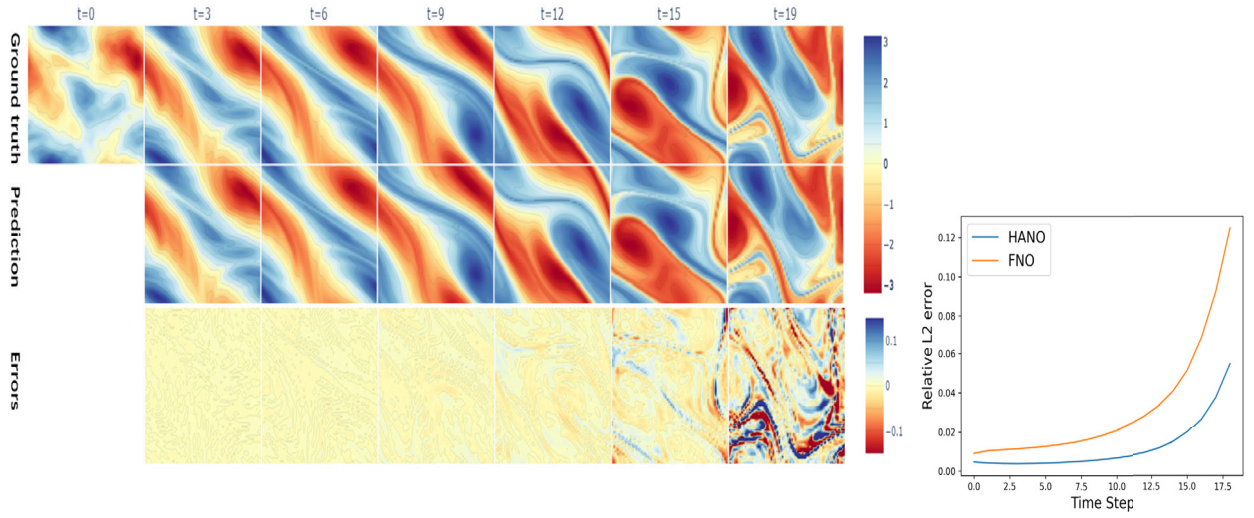
mapping the vorticity up to time 9 to the vorticity up to some later time T. We experiment with viscosities $\nu = 10^{-3}, 10^{-4}, 10^{-5}$, and decrease the final time T accordingly as the dynamics becomes more turbulent with increasing Reynolds number.

Time dependent neural operator Following the standard setup in [23], we fix the resolution as 64×64 for both training and testing. Ten time-slices of solutions $\omega(\cdot, t)$ at $t = 0, ..., 9$ are taken as the input data to the neural operator $\mathcal{N}$ which maps the solutions at 10 given timesteps to their subsequent time step. This procedure, often referred to as the rolled-out prediction, can be repeated recurrently until the final time T. For example, the k-th rollout is to obtain $\{\omega(\cdot, t_i)\}_{i=k-9}^{k} \mapsto \omega(\cdot, t_{k+1})$. In Table 3, the results are listed for HANO, FNO-3D (FFT in space-time), FNO-2D (FFT in space, and time rollouts), U-Net [69], TF-Net [82], ResNet [83] and DilResNet [29], and HANO achieves the best performance.

Furthermore, we also test models on the same Navier-Stokes task with $\nu = 10^{-5}$ but introduce an alternative training configuration labeled as $T = 20$ (new). Note that all models incorporating these specialized training techniques show consistently performance enhancement when compared to the original setup. The inclusion of these training tricks contributes to more stable generalization errors across various models, justifying their use in performance comparisons. Hence, both evaluation methods offer an equitable basis for contrasting the efficacy of our approach with existing baselines. We present the comprehensive overview of the training configurations here (Fig. 10).

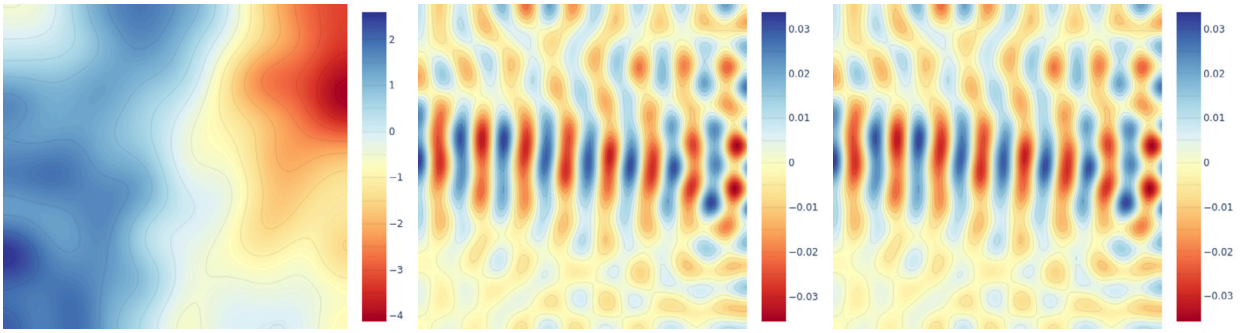
Two training setups

- **Original training setup:** Samples consist of 20 sequential time steps, with the goal of predicting the subsequent 10 time steps from the preceding 10. Using the roll-out prediction approach as described by [23], the neural operator uses the initial 10-time steps to forecast the immediate next time step. This predicted time step is then merged with the prior 9 time steps to predict the ensuing time step. This iterative process continues to forecast the remaining 10 time steps.
- **New training setup:** Our findings indicate that an amalgamation of deep learning strategies is pivotal for the optimal performance of time-dependent tasks. While [23] employed the previous 10-time steps as inputs for the neural operator, our approach simplifies this. We find that leveraging just the current step's data, similar to traditional numerical solvers, is sufficient. During training, we avoid model unrolling. Originally, we had 1000 samples, each consisting of 20 sequential time steps. We have transformed these into 19,000 samples, where each sample now comprises a pair of sequential time steps. These are then shuffled and used to train the neural operator to predict the subsequent time step based on the current one. For testing, we revert to the roll-out prediction method as only the initial time step's ground truth is available. These methods align with some techniques presented in [25]. As shown in Table 3, the second approach provides better performance. Also note that FNO-3D performs FFT in both space and time, while other models are designed with more conventional marching-in-time schemes, such that they are applied in an autoregressive fashion. Therefore, the new alternative training configuration is not applicable to FNO-3D.



(a) Example rollout trajectories of the HANO model (b) Error vs. Time

Fig. 10. Benchmark for the Navier-Stokes equation with $\nu = 1e-5$.



(a) wavespeed field $\boldsymbol{c}$ (b) groud truth $\boldsymbol{u}$ (c) HANO prediction $\hat{\boldsymbol{u}}$

Fig. 11. The mapping $\mathbf{c} \mapsto \mathbf{u}$.

3.5. Helmholtz equation

We test the performance of HANO for the acoustic Helmholtz equation in highly heterogeneous media as an example of multiscale wave phenomena, whose solution is considerably expensive for complicated and large geological models. This example and training data are taken from [84], for the Helmholtz equation on the domain $\Omega := (0,1)^2$. Given frequency $\omega = 10^3$ and wavespeed field $c : \Omega \to \mathbb{R}$, the excitation field $u : \Omega \to \mathbb{R}$ solves the equation

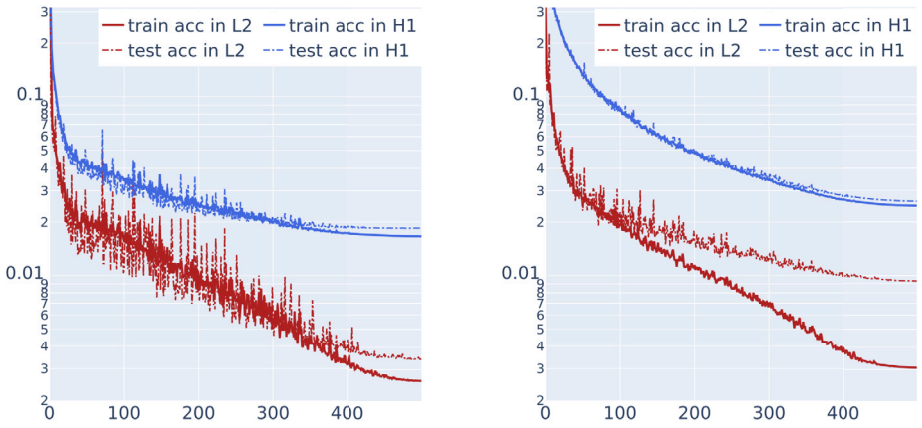
$$\begin{cases} \left(-\Delta - \dfrac{\omega^2}{c^2(x)}\right) u = 0 & \text{in } \Omega, \\ \dfrac{\partial u}{\partial n} = 0 & \text{on } \partial\Omega_1, \partial\Omega_2, \partial\Omega_4, \\ \dfrac{\partial u}{\partial n} = 1 & \text{on } \partial\Omega_3, \end{cases}$$

where $\partial\Omega_3$ is the top side of the boundary, and $\partial\Omega_{1,2,4}$ are other sides. The wave speed field is $c(x) = 20 + \tanh(\tilde{c}(x))$, where $\tilde{c}$ is sampled from the Gaussian random field $\tilde{c} \sim \mathcal{N}(0, \left(-\Delta + \tau^2\right)^{-d})$, where $\tau = 3$ and $d = 2$ are chosen to control the roughness. The Helmholtz equation is solved on a 100×100 grid by finite element methods. We aim to learn the mapping from $\mathbf{c} \in \mathbb{R}^{100\times100}$ to $\mathbf{u} \in \mathbb{R}^{100\times100}$ as shown in Fig. 11. In this example, following the practice in [84], a training dataset of size 4000 examples is adopted, while the test dataset contained 800 examples. All models were trained for 100 epochs.

For issues like high-frequency problems, especially Helmholtz equations with a large wavenumber, HANO might exhibit limitations due to the inherent challenge in approximating high-frequency components locally. Such scenarios highlight the difficulty of capturing the propagation of high-frequency solutions through local attention mechanisms alone. This challenge is analogous to the

Table 4
Performance on the Helmholtz benchmark.

Model	time	params(m)	$L^2(\times 10^{-2})$	$H^1(\times 10^{-2})$
FNO2D	6.2	16.93	1.25	7.66
UNet	7.1	17.26	3.81	23.31
DilResNet	10.8	1.03	4.34	34.21
U-NO	21.5	16.39	1.26	8.03
LSM	28.2	4.81	2.55	10.61
HANO	13.1	11.35	**0.95**	**6.10**



(a) Training curve by using H^1 loss (b) Training curve by using L^2 loss

Fig. 12. Comparison of training dynamics between HANO trained with H^1 loss and L^2 loss.

limitations faced by classical numerical methods in devising straightforward hierarchical matrix formulations for the Green's function of Helmholtz equations with large wavenumbers. It is noteworthy that the wavenumber is modest in this benchmark, and HANO surpasses other baseline methods, by a relatively modest margin as shown in Table 4. The error of our rerun FNO is comparable with the reported results in [84]. We note that the four models benchmarked for the Helmholtz equation in [84], including FNO and DeepONet, failed to reach a relative error less than 1×10^{-2}. We also compare the evaluation time of the trained models in Table 4. Compared to HANO, FNO has both a larger error and takes longer to evaluate. Moreover, FNO is known to be prone to overfitting the data when increasing the stacking of spectral convolution layers deeper and deeper [85]. UNet, as a CNN-based method, can evaluate much faster (30 times faster than HANO) but has the worst error (60 times higher than HANO).

3.6. Training dynamics

We present the dynamics of training and testing error over 100 epochs of training in Fig. 12. We compare HANO trained with H^1 and L^2 loss functions, and show the evolution of errors as well as the loss curves during the training process. The comparison shows that HANO with H^1 loss achieves lower training and testing errors. It also suggests that the H^1 loss function reduces the generalization gap (measured by the difference between training error and testing error), while L^2 loss function fails to do so.

3.7. Memory usage

We report the memory usage of different models for the Darcy smooth (with resolution 211×211) and Darcy rough (with resolution 256×256) benchmarks in Table 5. The table shows that the memory usage of HANO remains stable across resolutions. For the higher resolution of 256×256, both MWT and GT consume more CUDA memory than HANO, even though HANO achieves much higher accuracy.

3.8. Discretization invariance

FNO achieves discretization invariance through Fourier interpolation, enabling models trained on low-resolution data to handle high-resolution input. By incorporating suitable interpolation operators, HANO can achieve a comparable capability. Specifically, it can be trained on lower resolution data but evaluated at higher resolution, without requiring any higher resolution data during training (achieving zero-shot super-resolution).

Table 5
The memory usage (GB) of different models by using torchinfo.

Model	Darcy smooth	Darcy rough
FNO	0.72	1.04
GT	1.40	4.53
MWT	—	1.27
HANO	0.89	1.21

Table 6
Comparison of discretization invariance property for HANO and FNO for the multiscale trigonometric coefficient benchmark. The relative L^2 error ($\times 10^{-2}$) with respect to the reference solution on the testing resolution is measured.

	FNO			HANO		
Train \ Test	128	256	512	128	256	512
64	5.2808	7.9260	9.1054	1.3457	1.3557	1.3624
128		3.9753	6.0156		0.6715	0.6835
256			3.1871			0.5941

We conducted the experiments following the same setup as in [23] for the multiscale trigonometric coefficient benchmark. The models were trained on 64×64, 128×128, and 256×256 resolutions, and tested on 128×128, 256×256, and 512×512 resolutions, respectively. Results in Table 6 show that HANO incorporating linear interpolation is more stable than FNO.

3.9. Datasets and code

The code and datasets can be accessed at the following location: https://github.com/xlliu2017/HANO.

4. Conclusion

In this work, we investigated the "spectral bias" phenomenon commonly observed in multiscale operator learning. To our best knowledge, we conducted the first in-depth numerical study of this issue. We proposed HANO, a hierarchical attention-based model to mitigate the spectral bias. HANO employs a fine-coarse-fine V-cycle update and an empirical H^1 loss to recover fine-scale features in the multiscale solutions. Our experiments show that HANO outperforms existing neural operators on multiscale benchmarks in terms of accuracy and robustness.

Limitation and outlook: (1) HANO's current implementation requires a regular grid, and extending it to data clouds and graph neural networks could offer new opportunities to exploit its hierarchical representation. (2) The current attention-based operator in HANO can achieve discretization invariance using simple interpolation [42] (e.g. see Section 3.8). However, either simple interpolation or Fourier interpolation (used by FNO) may suffer from aliasing errors in the frequency domain, as indicated by our experiments and recent analysis [60,86]. Better balance between discretization invariance and model accuracy may be achieved with proper operator-adaptive sampling and interpolation techniques.

CRediT authorship contribution statement

Xinliang Liu: Writing – original draft, Methodology, Formal analysis, Conceptualization. **Bo Xu:** Visualization, Validation, Software, Methodology, Investigation, Data curation. **Shuhao Cao:** Writing – review & editing, Methodology, Formal analysis. **Lei Zhang:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

BX and LZ are partially supported by the Shanghai Municipal Science and Technology Project 22JC1401600, National Key Research and Development Program of China 2023YFF0805200, NSFC grant 12271360, and the Fundamental Research Funds for the Central Universities. SC is partially supported by the National Science Foundation award DMS-2309778.

References

[1] L.V. Branets, S.S. Ghai, X.-H. Wu, Challenges and technologies in reservoir modeling, Commun. Comput. Phys. 6 (1) (2009) 1–23.
[2] B. Engquist, P.E. Souganidis, Asymptotic and numerical homogenization, Acta Numer. 17 (2008) 147–190.
[3] T.Y. Hou, X.-H. Wu, Z. Cai, Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients, Math. Comput. 68 (227) (1999) 913–943.
[4] Y. Efendiev, T. Hou, Multiscale Finite Element Methods: Theory and Applications, vol. 4, Springer Science & Business Media, 2009.
[5] Y. Efendiev, J. Galvis, T.Y. Hou, Generalized multiscale finite element methods (gmsfem), J. Comput. Phys. 251 (2013) 116–135, https://doi.org/10.1016/j.jcp.2013.04.045, http://www.sciencedirect.com/science/article/pii/S0021999113003392.
[6] E. Chung, Y. Efendiev, T.Y. Hou, Adaptive multiscale model reduction with generalized multiscale finite element methods, J. Comput. Phys. 320 (2016) 69–95.
[7] E. Chung, Y. Efendiev, T.Y. Hou, Multiscale Model Reduction: Multiscale Finite Element Methods and Their Generalizations, vol. 212, Springer Nature, 2023.
[8] W. Hackbusch, Multigrid Methods and Applications, Springer Series in Computational Mathematics, vol. 4, Springer-Verlag, Berlin, 1985.
[9] J. Xu, L. Zikatanov, Algebraic multigrid methods, Acta Numer. 26 (2017) 591–721.
[10] M.E. Brewster, G. Beylkin, A multiresolution strategy for numerical homogenization, Appl. Comput. Harmon. Anal. 2 (4) (1995) 327–349.
[11] G. Beylkin, N. Coult, A multiresolution strategy for reduction of elliptic PDEs and eigenvalue problems, Appl. Comput. Harmon. Anal. 5 (2) (1998) 129–155.
[12] H. Owhadi, Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games, SIAM Rev. 59 (1) (2017) 99–149.
[13] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (2) (1987) 325–348.
[14] W. Hackbusch, L. Grasedyck, S. Börm, An introduction to hierarchical matrices, in: Proceedings of Equadiff 10, Masaryk University, 2002, pp. 101–111.
[15] K. Ho, L. Ying, Hierarchical interpolative factorization for elliptic operators: differential equations, Commun. Pure Appl. Math. 69 (8) (2016) 1415–1451.
[16] M. Bebendorf, Efficient inversion of the galerkin matrix of general second-order elliptic operators with nonsmooth coefficients, Math. Comput. 74 (251) (2005) 1179–1199.
[17] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.
[18] Y. Fan, J. Feliu-Fabá, L. Lin, L. Ying, L. Zepeda-Nunez, A multiscale neural network based on hierarchical nested bases, Res. Math. Sci. 6 (21) (2019).
[19] Y. Fan, L. Lin, L. Ying, L. Zepeda-Núnez, A multiscale neural network based on hierarchical matrices, Multiscale Model. Simul. 17 (4) (2019) 1189–1213.
[20] Y. Khoo, J. Lu, L. Ying, Solving parametric pde problems with artificial neural networks, Eur. J. Appl. Math. 32 (3) (2020) 421–435.
[21] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[22] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (4) (1995) 911–917.
[23] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: The International Conference on Learning Representations, 2021.
[24] G. Gupta, X. Xiao, P. Bogdan, Multiwavelet-based operator learning for differential equations, Adv. Neural Inf. Process. Syst. 34 (2021) 24048–24062.
[25] J. Brandstetter, D.E. Worrall, M. Welling, Message passing neural PDE solvers, in: International Conference on Learning Representations, 2022, https://openreview.net/forum?id=vSix3HPYKSU.
[26] J. Seidman, G. Kissas, P. Perdikaris, G.J. Pappas, Nomad: nonlinear manifold decoders for operator learning, Adv. Neural Inf. Process. Syst. 35 (2022) 5601–5613.
[27] Y. Chen, B. Hosseini, H. Owhadi, A.M. Stuart, Solving and learning nonlinear pdes with gaussian processes, J. Comput. Phys. 447 (2021) 110668.
[28] J. Brandstetter, R. van den Berg, M. Welling, J.K. Gupta, Clifford neural layers for PDE modeling, in: The Eleventh International Conference on Learning Representations, 2023, https://openreview.net/forum?id=okwxL_c4x84.
[29] K. Stachenfeld, D.B. Fielding, D. Kochkov, M. Cranmer, T. Pfaff, J. Godwin, C. Cui, S. Ho, P. Battaglia, A. Sanchez-Gonzalez, Learned simulators for turbulence, in: International Conference on Learning Representations, 2022.
[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[31] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.
[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: transformers for image recognition at scale, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=YicbFdNTTy.
[33] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.
[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10684–10695.
[35] S. Cao, Choose a transformer: Fourier or galerkin, Adv. Neural Inf. Process. Syst. 34 (2021).
[36] N. Geneva, N. Zabaras, Transformers for modeling physical systems, Neural Netw. 146 (2022) 272–289.
[37] G. Kissas, J.H. Seidman, L.F. Guilhoto, V.M. Preciado, G.J. Pappas, P. Perdikaris, Learning operators with coupled attention, J. Mach. Learn. Res. 23 (1) (2022) 9636–9698.
[38] Z. Li, K. Meidani, A.B. Farimani, Transformer for partial differential equations' operator learning, Trans. Mach. Learn. Res. (2023), https://openreview.net/forum?id=EPPqt3uERT.
[39] Z. Hao, Z. Wang, H. Su, C. Ying, Y. Dong, S. Liu, Z. Cheng, J. Song, J. Zhu, GNOT: a general neural operator transformer for operator learning, in: International Conference on Machine Learning, PMLR, 2023, pp. 12556–12569.
[40] A.H. De Oliveira Fonseca, E. Zappala, J. Ortega Caro, D.V. Dijk, Continuous spatiotemporal transformer, in: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (Eds.), Proceedings of the 40th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 202, PMLR, 2023, pp. 7343–7365.
[41] Z. Xiao, Z. Hao, B. Lin, Z. Deng, H. Su, Improved operator learning by orthogonal attention, arXiv:2310.12487, 2023.
[42] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: learning maps between function spaces with applications to pdes, J. Mach. Learn. Res. 24 (89) (2023) 1–97, URL http://jmlr.org/papers/v24/21-1524.html.
[43] F. Bartolucci, E. de Bézenac, B. Raonić, R. Molinaro, S. Mishra, R. Alaifari, Are neural operators really neural operators? Frame theory meets operator learning, arXiv preprint arXiv:2305.19913, 2023.

[44] S. Liu, Z. Hao, C. Ying, H. Su, Z. Cheng, J. Zhu, Nuno: a general framework for learning parametric pdes with non-uniform data, arXiv preprint arXiv:2305.18694, 2023.
[45] O. Ovadia, E. Turkel, A. Kahana, G.E. Karniadakis Ditto, Diffusion-inspired temporal transformer operator, arXiv preprint arXiv:2307.09072, 2023.
[46] O. Ovadia, A. Kahana, P. Stinis, E. Turkel, G.E. Karniadakis, Vito: vision transformer-operator, arXiv preprint arXiv:2303.08891, 2023.
[47] A. Hemmasian, A. Barati Farimani, Reduced-order modeling of fluid flows with transformers, Phys. Fluids 35 (5) (2023).
[48] Z. Li, D. Shu, A.B. Farimani, Scalable transformer for pde surrogate modeling, Adv. Neural Inf. Process. Syst. 36 (2023).
[49] M. Zhu, S. Feng, Y. Lin, L. Lu, Fourier-deeponet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness, arXiv preprint arXiv:2305.17289, 2023.
[50] R. Guo, J. Jiang, Construct deep neural networks based on direct sampling methods for solving electrical impedance tomography, SIAM J. Sci. Comput. 43 (3) (2021) B678–B711.
[51] R. Guo, S. Cao, L. Chen, Transformer meets boundary value inverse problems, in: The Eleventh International Conference on Learning Representations, 2023, https://openreview.net/forum?id=HnlCZATopvr.
[52] R. Guo, J. Jiang, Y. Li, Learn an index operator by cnn for solving diffusive optical tomography: a deep direct sampling method, J. Sci. Comput. 95 (1) (2023) 31.
[53] S. Mizera, Scattering with neural operators, arXiv preprint arXiv:2308.14789, 2023.
[54] Z. Li, A. Barati Farimani, Graph neural network-accelerated lagrangian fluid simulation, Comput. Graph. 103 (2022) 201–211.
[55] E. Zhang, A. Kahana, E. Turkel, R. Ranade, J. Pathak, G.E. Karniadakis, A hybrid iterative numerical transferable solver (hints) for pdes based on deep operator network and relaxation methods, arXiv preprint, arXiv:2208.13273, 2022.
[56] K. Wu, X.-B. Yan, S. Jin, Z. Ma, Capturing the diffusive behavior of the multiscale linear transport equations by asymptotic-preserving convolutional deeponets, Comput. Methods Appl. Mech. Eng. 418 (2024) 116531.
[57] N. Rahaman, D. Arpit, A. Baratin, F. Draxler, M. Lin, F.A. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of deep neural networks, in: International Conference on Machine Learning, 2019.
[58] B. Ronen, D. Jacobs, Y. Kasten, S. Kritchman, The convergence rate of neural networks for learned functions of different frequencies, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 4761–4771.
[59] Z.-Q.J. Xu, Y. Zhang, T. Luo, Y. Xiao, Z. Ma, Frequency principle: Fourier analysis sheds light on deep neural networks, Commun. Comput. Phys. 28 (5) (2020) 1746–1767.
[60] N. Kovachki, S. Lanthaler, S. Mishra, On universal approximation and error bounds for Fourier neural operators, J. Mach. Learn. Res. 22 (2021).
[61] J. Zhao, R.J. George, Y. Zhang, Z. Li, A. Anandkumar, Incremental Fourier neural operator, arXiv preprint, arXiv:2211.15188, 2022.
[62] X.-A. Li, Z.-Q.J. Xu, L. Zhang, A multi-scale dnn algorithm for nonlinear elliptic equations with multiple scales, arXiv preprint, arXiv:2009.14597, 2020.
[63] S. Wang, H. Wang, P. Perdikaris, On the eigenvector bias of fourier feature networks: from regression to solving multi-scale pdes with physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 384 (2021) 113938.
[64] X.-A. Li, Z.-Q.J. Xu, L. Zhang, Subspace decomposition based dnn algorithm for elliptic type multi-scale pdes, J. Comput. Phys. 488 (2023) 112242.
[65] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10012–10022.
[66] W. Hackbusch, Hierarchical Matrices: Algorithms and Analysis, Springer, Berlin, 2015.
[67] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric pdes, SMAI J. Comput. Math. 7 (2021) 121–157.
[68] Z. Zhang, H. Zhang, L. Zhao, T. Chen, S.Ö. Arık, T. Pfister, Nested hierarchical transformer: towards accurate, data-efficient and interpretable visual understanding, in: AAAI Conference on Artificial Intelligence (AAAI), 2022.
[69] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.
[70] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al., Rethinking attention with performers, arXiv preprint arXiv:2009.14794, 2020.
[71] S. Wang, B. Li, M. Khabsa, H. Fang, H. Ma Linformer, Self-attention with linear complexity, arXiv:2006.04768, 2020.
[72] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N.A. Smith, L. Kong, Random feature attention, arXiv preprint arXiv:2103.02143, 2021.
[73] T. Nguyen, V. Suliafu, S. Osher, L. Chen, B. Wang, Fmmformer: efficient and flexible transformer via decomposed near-field and far-field attention, Adv. Neural Inf. Process. Syst. 34 (2021) 29449–29463.
[74] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, V. Singh, Nyströmformer: a Nyström-based algorithm for approximating self-attention, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 14138–14148.
[75] A. Guadagnini, S.P. Neuman, Nonlocal and localized analyses of conditional mean steady state flow in bounded, randomly nonuniform domains: 1. Theory and computational approach, Water Resour. Res. 35 (10) (1999) 2999–3018.
[76] C.J. Gittelson, Stochastic galerkin discretization of the log-normal isotropic diffusion problem, Math. Models Methods Appl. Sci. 20 (02) (2010) 237–263.
[77] N.H. Nelsen, A.M. Stuart, The random feature model for input-output maps between banach spaces, SIAM J. Sci. Comput. 43 (5) (2021) A3212–A3243.
[78] L.N. Smith, N. Topin, Super-convergence: very fast training of neural networks using large learning rates, in: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11006, SPIE, 2019, pp. 369–386.
[79] B. Ronen, D. Jacobs, Y. Kasten, S. Kritchman, The convergence rate of neural networks for learned functions of different frequencies, Adv. Neural Inf. Process. Syst. 32 (2019).
[80] M. Tancik, P.P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J.T. Barron, R. Ng, Fourier features let networks learn high frequency functions in low dimensional domains, arXiv preprint arXiv:2006.10739, 2020.
[81] M.A. Rahman, Z.E. Ross, K. Azizzadenesheli, U-no: U-shaped neural operators, arXiv e-prints arXiv:2204.11127, 2022.
[82] R. Wang, K. Kashinath, M. Mustafa, A. Albert, R. Yu, Towards physics-informed deep learning for turbulent flow prediction, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &, Data Mining, 2020, pp. 1457–1466.
[83] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[84] M.V.d. Hoop, D.Z. Huang, E.Q. Null, A.M. Stuart, The cost-accuracy trade-off in operator learning with neural networks, J. Mach. Learn. 1 (3) (2022) 299–341, https://doi.org/10.4208/jml.220509.
[85] A. Tran, A. Mathews, L. Xie, C.S. Ong, Factorized Fourier neural operators, in: The Eleventh International Conference on Learning Representations, 2023, https://openreview.net/forum?id=tmIiMPl4IPa.
[86] S. Lanthaler, S. Mishra, G.E. Karniadakis, Error estimates for deeponets: a deep learning framework in infinite dimensions, Trans. Math. Appl. 6 (1) (2022) tnac001.