# HT-NET: HIERARCHICAL TRANSFORMER BASED OPERATOR LEARNING MODEL FOR MULTISCALE PDES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Complex nonlinear interplays of multiple scales give rise to many interesting physical phenomena and pose major difficulties for the computer simulation of multiscale PDE models in areas such as reservoir simulation, high frequency scattering and turbulence modeling. In this paper, we introduce a hierarchical transformer (HT-Net) scheme to efficiently learn the solution operator for multiscale PDEs. We construct a hierarchical architecture with scale adaptive interaction range, such that the features can be computed in a nested manner and with a controllable linear cost. Self-attentions over a hierarchy of levels can be used to encode and decode the multiscale solution space over all scale ranges. In addition, we adopt an empirical $H^1$ loss function to counteract the spectral bias of the neural network approximation for multiscale functions. In the numerical experiments, we demonstrate the superior performance of the HT-Net scheme compared with state-of-the-art (SOTA) methods for representative multiscale problems.

## 1 INTRODUCTION

Partial differential equation (PDE) models with multiple temporal/spatial scales are ubiquitous in physics, engineering, and other disciplines. They are of tremendous importance in making predictions for challenging practical problems such as reservoir modeling, fracture propagation, high frequency scattering, atmosphere and ocean circulation, to name a few. The complex non-linear interplays of characteristic scales cause major difficulties for the computer simulation of multiscale PDEs. While the resolution of all characteristic scales is prohibitively expensive, sophisticated multiscale methods have been developed to efficiently and accurately solve the multiscale PDEs by incorporating microscopic information, though most of them are designed for problems with fixed input parameters.

Recently, several novel methods such as Fourier neural operator (FNO) Li et al. (2021), Galerkin transformer (GT) Cao (2021) and deep operator network (DeepONet) Lu et al. (2021) are developed to directly learn the operator (mapping) between infinite dimensional spaces for PDE problems, by taking advantages of the enhanced expressibility of deep neural networks and advanced architectures such as feature embedding, channel mixing and self-attentions. Such methods can deal with an ensemble of input parameters, and have great potential for the fast forward and inverse solves of PDE problems. However, for multiscale problems, most existing operator learning schemes essentially capture the smooth part of the solution space, and how to resolve the intrinsic multiscale features remains to be a major challenge.

In this paper, we design a hierarchical transformer based operator learning method, so that the accurate, efficient and robust computer simulation of multiscale PDE problems with an ensemble of input parameters becomes feasible. Our main contribution can be summarized in the following,

- we develop a novel transformer architecture which allows the decomposition of the input-output mapping to a hierarchy of levels, the features can be updated in a nested manner based on the hierarchical local aggregation of self-attentions with linear computational cost;
- we adopt the empirical $H^1$ loss which avoid the spectral bias and enhance the ability to capture the oscillatory features of the multiscale solution space;
- the resulting scheme has significantly better accuracy as well as better generalization properties for multiscale input parameters, compared with state-of-the-art (SOTA) models.

## 2 Background and Related Work

We briefly introduce multiscale PDEs in Section § 2.1, then summarize relevant multiscale numerical methods in § 2.2, and neural solvers, in particular neural operators in § 2.3.

### 2.1 Multiscale PDEs

Multiscale PDEs, in a narrower sense, refer to PDEs with rapidly varying coefficients, which may arise from a wide range of applications in heterogenous and random media. In a broader sense, they may form a hierarchy of models at different scales by a systematic derivation, starting from fundamental laws of physics such as quantum mechanics E & Engquist (2003).

Multiscale elliptic equations is an important class of prototypical examples, such as the following second order elliptic equation of divergence form,

$$\begin{aligned} -\nabla \cdot (a(x)\nabla u(x)) &= f(x) \quad & x \in D \\ u(x) &= 0 \quad & x \in \partial D \end{aligned} \tag{2.1}$$

where $0 < a_{\min} \le a(x) \le a_{\max}, \forall x \in D$, and the forcing term $f \in H^{-1}(D;\mathbb{R})$. The coefficient to solution map is $\mathcal{S} : L^\infty(D;\mathbb{R}_+) \to H_0^1(D;\mathbb{R})$, such that $u = \mathcal{S}(a)$. In Li et al. (2021), smooth coefficient $a(x)$ is considered and $\mathcal{S}$ can be well resolved by the FNO parameterization. The setup $a(x) \in L^\infty$ allows rough coefficients with fast oscillation (e.g. $a(x) = a(x/\varepsilon)$ with $\varepsilon \ll 1$), high contrast ratio with $a_{\max}/a_{\min} \gg 1$, and even a continuum of non-separable scales. Rough coefficient case is much harder from both scientific computing Branets et al. (2009) and operator learning perspectives.

Other outstanding multiscale PDE models may include:

- Incompressible Navier-Stokes equation, which models the fluid dynamics, when the Reynolds number is large, the flow becomes turbulent due to the simultaneous interaction of a wide range of scales of motion, both in time and in space.
- Helmholtz equation, which models the time-harmonic acoustic wave-propagation, its numerical solution exhibits severe difficulties in the high wave number regime due to the interaction of high frequency waves and numerical mesh;

### 2.2 Multiscale Solvers for Multiscale PDEs

For multiscale PDEs, computational cost of classical numerical methods such as finite element methods, finite difference methods etc. usually scales proportional to $1/\varepsilon \gg 1$. Multiscale solvers have been developed such that their computational costs are independent of $\varepsilon$ by incorporating microscopic information.

**Asymptotic and numerical homogenization** Asymptotic homogenization Bensoussan et al. (1978) is an elegant analytical approach for multiscale PDEs with scale separation, e.g., $a(x) = a(x/\varepsilon)$ with $\varepsilon \ll 1$

in equation 2.1. For more general multiscale PDEs with possibly a continuum of scales, numerical homogenization Engquist & Souganidis (2008) offers an effective numerical approach which aims to identify low dimensional approximation spaces such that the approximation bases are adapted to the corresponding multiscale operator, and can be efficiently constructed (e.g. localized). See Appendix A for details.

**Multilevel and multiresolution method** Multilevel/multigrid methods Hackbusch (1985); Xu & Zikatanov (2017) have been successfully applied to PDEs, and classical wavelet based methods Brewster & Beylkin (1995); Beylkin & Coult (1998) enable a multi-resolution decomposition. However, the convergence of those methods for multiscale problems can be severely affected by the regularity of coefficients Branets et al. (2009) because they are not operator adapted. Recently, the introduction of gamblets Owhadi (2017) opens an avenue to automatically discover scalable multilevel algorithms and operator adapted wavelets for linear PDEs with rough coefficients. Gamblets can be seen as the multilevel extension of numerical homogenization, see also Appendix A.2.

**Low Rank Decomposition based Methods** It is well-known that the elliptic Green's function has low rank approximation Bebendorf (2005), which lays the theoretical foundation of the low rank decomposition based numerical methods such as fast multipole method Greengard & Rokhlin (1987); Ying et al. (2004), hierarchical matrices ($\mathscr{H}$ and $\mathscr{H}^2$ matrices) Hackbusch et al. (2002); Bebendorf (2008) and hierarchical interpolative factorization Ho & Ying (2016). Numerical evidence suggests the robustness and effectiveness of low rank matrix decomposition based methods. See Appendix B for the connection with our method presented in this paper.

**Tensor Numerical Method** Analytical approaches such as periodic unfolding Cioranescu et al. (2008) suggest that low dimensional multiscale operator can be transformed to a high dimensional PDE, and tensor numerical methods, e.g, the sparse tensor finite element method Harbrecht & Schwab (2011) and the quantic tensor train (QTT) method Kazeev et al. (2022), provide efficient numerical procedures to find the low rank tensor representation of the multiscale solution.

### 2.3 Neural Operator for PDEs

**Neural PDE solvers and Spectral Bais** Various neural solvers have been proposed in E et al. (2017); Sirignano & Spiliopoulos (2018); Raissi et al. (2019) to solve PDEs with fixed parameters. The difficulty to solve multiscale PDEs has already been exhibited by the so-called spectral bias or frequency principle Rahaman et al. (2019); Ronen et al. (2019); Xu et al. (2019), which shows that DNN-based algorithms often suffer from the "curse of high-frequency" as they are inefficient to learn high-frequency components of multi-scale functions. A series of algorithms have been developed to solve multi-scale PDEs and to overcome the high-frequency curse of general DNNs (Cai et al., 2020; Li et al., 2020; Wang et al., 2021).

**Operator learning for PDEs** Instead of solving PDEs with fixed parameters, DNN algorithms demonstrate more potential to learn the input-output mapping of parametric PDEs. Finite dimensional operator learning methods such as Zhu & Zabaras (2018); Fan et al. (2019a;b); Khoo et al. (2020) can be applied to problems with fixed discretization. In particular, Fan et al. (2019a;b) combines $\mathscr{H}$ or $\mathscr{H}^2$ matrices linear operations with nonlinear activation functions, while the complex nonlinear geometrical interaction/aggregation is absent, which limits the expressivity of the resulting neural operator. The infinite dimensional operator learning methods Li et al. (2021); Gupta et al. (2021) aim to learn the mapping between infinite dimensional Banach spaces, and the convolutions in the construction of neural operators are parametrized by, e.g., Fourier or wavelet transform, which is efficient to implement. Nevertheless, even in the linear case, those methods do not always work if multiscale features are present. On the other hand, universal approximation can be rigorously proved for FNO rigorously for FNO operators Kovachki et al. (2021), while "extra smoothness" of input parameters is required to achieve meaningful decay rate, which either is absent or gives rise to large constants for multiscale PDEs. This motivates us to construct a new architecture for multiscale operator learning.

**Efficient Attention** The vanilla multi-head self-attention in Vaswani et al. (2017) scales quadratically with the number of tokens, which is prohibitive for high-resolution problems. Many efficient attention methods are proposed using the kernel trick (low rank projection) Choromanski et al. (2020); Wang et al. (2020); Peng et al. (2021); Nguyen et al. (2021) to reduce the computation cost arisen in the dense matrix operation. In the context of operator learning, Galerkin transformer Cao (2021) proposed to remove the softmax normalization in self-attention and introduced a linearized self-attention variant with Petrov-Galerkin projection normalization. Furthermore, hierarchical attention using local window aggregation is proposed in Liu et al. (2021); Zhang et al. (2022) for NLP and vision applications.

## 3 METHODS

In this section, we will introduce our hierarchical transformer model, in particular, the computing process over hierarchical levels of complexity, as shown in Figure 3.1. We follow the setup for operator learning in the pioneering work Li et al. (2021); Lu et al. (2021) to approximate the operator $\mathcal{S}: \boldsymbol{f} \mapsto \boldsymbol{u} := \mathcal{S}(\boldsymbol{f})$, with the input $\boldsymbol{f} \in \mathcal{A}$ drawn from a distribution $\mu$ and the output $\boldsymbol{u} \in \mathcal{U}$, where $\mathcal{A}$ and $\mathcal{U}$ are infinite dimensional Banach spaces respectively. We aim to learn the operator $\mathcal{S}$ from a finite collection of finitely observed input-output pairs by constructing a parametric map $\mathcal{N}: \mathcal{A} \times \Theta \to \mathcal{U}$ and using a loss functional $\mathcal{L}: \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, such that the optimal parameter $\theta^* = \arg\min_{\theta\in\Theta} \mathbb{E}_{\boldsymbol{f}\sim\mu}\left[\mathcal{L}\left(\mathcal{N}(\boldsymbol{f},\theta), S(\boldsymbol{f})\right)\right]$.

Our method is motivated by the celebrated hierarchical matrix method, which is an efficient numerical method for the solution of integral and differential equations. Unlike the kernel trick used in efficient attentions, the $\mathscr{H}$ and $\mathscr{H}^2$ matrices use the divide-and-conqure strategy and treat the interactions in space separately by geometrical scales and locality. The idea is reflected in Liu et al. (2021); Zhang et al. (2022) to a certain degree. In the following, we are going to give a hierarchical nested feature update scheme and rethink the transformer from the hierarchical matrix perspective.

**Hierarchical Discretization** We first introduce the hierarchical discretization of the spatial domain $D$, which can be used directly for time independent PDEs such as equation 2.1. Time dependent PDEs can be treated by taking time sliced data as feature channels. Let $\mathcal{I}^{(r)}$ be the finest level index set, such that each index $i = (i_1, \ldots, i_r) \in \mathcal{I}^{(r)}$ denotes the finest level spatial objects such as image pixels, finite difference points, etc. For any $i = (i_1, \ldots, i_r) \in \mathcal{I}^{(r)}$, and $1 \le m \le r$, we denote $i$'s $m$-th level parent node as $i^{(m)} = (i_1, \ldots, i_m)$ which represents $m$-th level objects such as superpixels/aggregates/patches of finer level objects. The $m$-th level index set is $\mathcal{I}^{(m)} := \{i^{(m)} : i \in \mathcal{I}^{(r)}\}$. For $i^{(m)} \in \mathcal{I}^{(m)}$, we denote $i^{(m,m')}$ as the $m'$-th level parent node of $i^{(m)}$ if $m > m' \ge 1$, and $i^{(m,m')} := \{j \in \mathcal{I}^{(m')} | j^{(m,m')} = i^{(m)}\}$ as the set of the $m'$-th level child nodes of $i^{(m)}$ if $m \le m' \le r$. The index (cluster) tree $\mathcal{I}$ is induced by the index sets $\mathcal{I}^{(m)}$, $1 \le m \le r$, and the parent/child relation. Each index $i \in \mathcal{I}^{(m)}$ corresponds to a spatial object (sometimes denoted as token) which can be characterized e.g. by its position $\boldsymbol{x}_i^{(m)}$ and a feature vector $\boldsymbol{f}_i^{(m)}$ with number of channels $\mathcal{C}^{(m)}$.

**Reduce Operation** The reduce operation defines the map from finer level features to coarser level features. Given $i^{(m)} \in \mathcal{I}^{(m)}$ and its child nodes $i^{(m,m+1)} \subset \mathcal{I}^{(m+1)}$, the operator $\mathcal{R}^{(m)}$ maps the $m+1$-th level features with indices in $i^{(m,m+1)}$ to the $m$-th level feature with index $i^{(m)}$, namely, $\boldsymbol{f}_i^{(m)} = \mathcal{R}^{(m)}(\{\boldsymbol{f}_j^{(m+1)}\}_{j\in i^{(m,m+1)}})$. The reduce operations can be done recursively from the finest level to the coarsest level, which leads to a nested representation. In the following, we present a self-attention based local aggregation scheme and a token mixing scheme across multiple levels to go from the coarsest level back to the finest level, which completes a V-cycle of feature update.

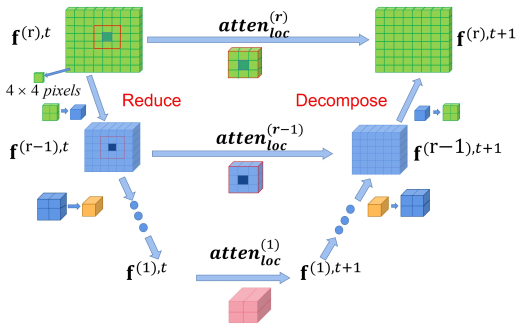


Figure 3.1: Architecture for one V-cycle of feature update.

**Multilevel Local Aggregation** We introduce the self-attention based aggregation. The update of the feature $\boldsymbol{f}_i^{(r),t} \in \mathbb{R}^{\mathcal{C}^{(r)}}$ with index $i \in \mathcal{I}^{(r)}$ can be given by the following $r$-th level token aggregation formula,

$$\text{atten} : \boldsymbol{f}_i^{(r),t+1} = \sum_{j=1}^{N^{(r)}} \mathcal{G}(\boldsymbol{f}_i^{(r),t}, \boldsymbol{f}_j^{(r),t}) v(\boldsymbol{f}_j^{(r),t}), \text{ for } i \in \mathcal{I}^{(r)}, \tag{3.1}$$

where the index $t$ denotes the evolution of features, and $N^{(r)} := |\mathcal{I}^{(r)}|$ is the number of features at level $r$. In the context of vanilla attention, we ignore the softmax normalizing factor and only consider 1 head attention for simplicity of notations, the interaction kernel $\mathcal{G}$ can be simplified as $\mathcal{G}(\boldsymbol{f}_i, \boldsymbol{f}_j) = \exp(\boldsymbol{W}^Q \boldsymbol{f}_i \cdot \boldsymbol{W}^K \boldsymbol{f}_j) = \exp(\boldsymbol{q}_i \cdot \boldsymbol{k}_j)$, where $\boldsymbol{q}_i := \boldsymbol{W}^Q \boldsymbol{f}_i$, $\boldsymbol{k}_i := \boldsymbol{W}^K \boldsymbol{f}_i$, $\boldsymbol{v}_i := v(\boldsymbol{f}_i) = \boldsymbol{W}^V \boldsymbol{f}_i$, and $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, $\boldsymbol{W}^V \in \mathbb{R}^{\mathcal{C}^{(r)} \times \mathcal{C}^{(r)}}$ are learnable matrices. Instead of calculating equation 3.1 explicitly with $\mathcal{O}(N^2)$ cost, we propose a self-attention based local aggregation scheme, inspired by the $\mathscr{H}^2$ matrices Hackbusch (2015), see also Appendix B. The <u>local aggregation</u> at the $r$-th level writes,

$$\text{atten}_{\text{loc}}^{(r)} : \boldsymbol{f}_i^{(r),t+1} = \sum_{j \in \mathcal{N}^{(r)}(i)} \exp(\boldsymbol{q}_j^{(r),t} \cdot \boldsymbol{k}_j^{(r),t}) \boldsymbol{v}_j^{(r),t}, \text{ for } i \in \mathcal{I}^{(r)}, \tag{3.2}$$

where $\mathcal{N}^{(r)}(i)$ is the set of the $r$-th level neighbor tokens of $i \in \mathcal{I}^{(r)}$, $\boldsymbol{q}_i^{(r),t} := \hat{W}^Q \boldsymbol{f}_i^{(r),t}$, $\boldsymbol{k}_i^{(r),t} := \hat{W}^K \boldsymbol{f}_i^{(r),t}$, $\boldsymbol{v}_i^{(r),t} := \hat{W}^V \boldsymbol{f}_i^{(r),t}$ with learnable matrices $\hat{W}^Q$, $\hat{W}^K$, $\hat{W}^V \in \mathbb{R}^{\mathcal{C}^{(r)} \times \mathcal{C}^{(r)}}$. Then for $m = r-1, ..., 1$, we calculate the aggregation in each level with nested $\boldsymbol{q}_i^{(m),t}$, $\boldsymbol{k}_j^{(m),t}$, $\boldsymbol{v}_j^{(m),t}$,

$$\text{atten}_{\text{loc}}^{(m)} : \boldsymbol{f}_i^{(m),t+1} = \sum_{j \in \mathcal{N}^{(m)}(i)} \exp(\boldsymbol{q}_i^{(m),t} \cdot \boldsymbol{k}_j^{(m),t}) \boldsymbol{v}_j^{(m),t}, \text{ for } i \in \mathcal{I}^{(m)} \tag{3.3}$$

where $\boldsymbol{q}_i^{(m),t} = \mathcal{R}^{(m)}(\{\boldsymbol{q}_j^{(m+1),t}\}_{j \in i^{(m,m+1)}})$, $\boldsymbol{k}_i^{(m),t} = \mathcal{R}^{(m)}(\{\boldsymbol{k}_j^{(m+1),t}\}_{j \in i^{(m,m+1)}})$, $\boldsymbol{v}_i^{(m),t} = \mathcal{R}^{(m)}(\{\boldsymbol{v}_j^{(m+1),t}\}_{j \in i^{(m,m+1)}})$, $\mathcal{N}^{(m)}(i)$ is the set of the $m$-th level neighbor tokens of $i \in \mathcal{I}^{(m)}$.

**Remark 1** *In our paper, for simplicity, the learnable operators $\mathcal{R}^{(m)}$ and $\mathcal{D}^{(m)}$ only consist of fully connected linear layers or convolution layers without nonlinear activation function. In general, these operators are not limited to linear operators. The nested learnable operators $\mathcal{R}^{(m)}$ also induce the channel mixing and is equivalent to a structured parameterization of $\boldsymbol{W}^Q, \boldsymbol{W}^V, \boldsymbol{W}^K$ matrix for the coarse level tokens. See Appendix B for the discussion.*

**Decompose Operation** To mix the multilevel features $\boldsymbol{f}_i^{(m)}, m = 1, ..., r$, we propose a decompose operation which reverses the reduce operation. The decompose operator $\mathcal{D}^{(m)} : \boldsymbol{f}_i^{(m),t} \mapsto \{\boldsymbol{f}_j^{(m+1),t+\frac{1}{2}}\}_{j\in i^{(m,m+1)}}$, maps the $m$-th level feature with index $i$ to $m+1$-th level features associated to its child set $i^{(m,m+1)}$. $\boldsymbol{f}_j^{(m+1),t+\frac{1}{2}}$ is further aggregated to $\boldsymbol{f}_j^{(m+1),t+1} += \boldsymbol{f}_j^{(m+1),t+\frac{1}{2}}$. In summary, we have the following algorithm for hierarchical attention, which can drop-in substitute the canonical attention.

**Algorithm 1** One V-cycle of Hierarchical Attention

```
Input: $\mathcal{I}^{(r)}$, $\boldsymbol{f}_i^{(r),t}$ for $i \in \mathcal{I}^{(r)}$.
STEP 0: Get the $\boldsymbol{q}_i^{(r),t}$, $\boldsymbol{k}_i^{(r),t}$, $\boldsymbol{v}_i^{(r),t}$ for $i \in \mathcal{I}^{(r)}$.
STEP 1: For $m = r-1 : -1 : 1$, Do the reduce operations $\boldsymbol{q}_i^{(m),t} = \mathcal{R}^{(m)}(\{\boldsymbol{q}_j^{(m+1),t}\}_{j\in i^{(m,m+1)}})$ and also for
    $\boldsymbol{k}_i^{(m),t}$ and $\boldsymbol{v}_i^{(m),t}$, for any $i \in \mathcal{I}^{(m)}$.
STEP 2: For $m = r : -1 : 1$, Do the local aggregation by equation 3.2 and equation 3.3 to get $\boldsymbol{f}_i^{(m),t+1}$ for any
    $i \in \mathcal{I}^{(m)}$.
STEP 3: For $m = 1 : r-1$, Do the decompose operations $\{\boldsymbol{f}_j^{(m+1),t+\frac{1}{2}}\}_{j\in i^{(m,m+1)}} = \mathcal{D}^{(m)}(\boldsymbol{f}_i^{(m),t})$, for any
    $i \in \mathcal{I}^{(m)}$; then $\boldsymbol{f}_i^{(m+1),t+1} += \boldsymbol{f}_i^{(m+1),t+\frac{1}{2}}$, for any $i \in \mathcal{I}^{(m+1)}$
Output: $\boldsymbol{f}_i^{(r),t+1}$ for any $i \in \mathcal{I}^{(r)}$.
```

**Proposition 3.1 (Complexity of Algorithm 1)** *The reduce operation, multilevel aggregation, and decomposition operation together form a V-cycle for the update of features, as illustrated in Figure 3.1. The cost of one V-cycle is $O(N)$ if $\mathcal{I}$ is a quadtree as implemented in the paper. See Appendix C for the proof.*

**Decoder** The decoder maps the features $\boldsymbol{f}$ (at the last update step) to the solution $\boldsymbol{u}$. The decoder is often chosen with prior knowledge of the PDE. A simple feedforward neural network (FFN) is used in Lu et al. (2021) to learn a basis set as the decoder. A good decoder can also be learned from the data using SVD Bhattacharya et al. (2021). In this paper, we use the spectral convolution layers in Li et al. (2021).

**Loss functions** The choice of loss functions is crucial for the efficient training process and the generalization. Due to the multiscale nature of the problem considered in this paper, we prefer to choose $H^1$ loss instead of the usual $L^2$ loss function, as it puts more ”weights” on the high frequency components. We refer the readers to Adams & Fournier (2003) for detailed definition of Sobolev space and $H^1$ norm, and to the next section for the implementation.

## 4 Experiments

We demonstrate the effectiveness of the hierarchical transformer model (HT-Net) on operator learning tasks for multiscale PDEs to showcase its qualities. All our examples are in 2D. We first study the ability of the model on the multiscale elliptic equation with two phase coefficients which is a widely used benchmark problem for operator learning Li et al. (2021), and the “multiscaleness” of the coefficients such as the oscillation, contrast and roughness of the two phase interface can be tuned in the model. We demonstrate the accuracy and robustness of the HT-NET for both smooth and rough coefficients, and also show that HT-NET is able to provide better generalization error for out-of-distribution input parameters. We also test the Navier-Stokes equation with large Reynolds number, as an example with non-linearity and time dependence.

### 4.1 Setup

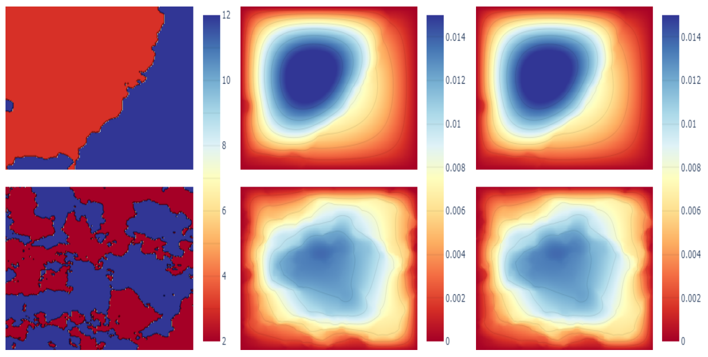
In our experiments, the spatial domain is $D := [0,1]^2$ and is discretized uniformly with $h = 1/n$. Let the uniform grid be $\mathsf{G}^2 := \{(x_i, x_j) = (ih, jh) \mid i, j = 0, ..., n-1\}$. $\{(a_j, u_j)\}_{j=1}^N$ are functions pairs such that $u_j = \mathcal{S}(a_j)$, and $a_j$ is drown from some probability measure $\mu$. The actual data pairs for training and testing are pointwise evaluations of $a_j$ and $u_j$ on the grid $\mathsf{G}^2$, denoted by $\boldsymbol{a}_j$ and $\boldsymbol{u}_j$ respectively. The comparison with all the baselines are consistent with (in most time better than) references. The hierarchical index tree $\mathcal{I}$ can be generated by the corresponding quadtree representation of the nodes with depth $r$, such that the finest level objects are pixels or patches aggregated by pixels. See Appendix D for more detail.

**Empirical $H^1$ loss function** Empirical $L^2$ loss function is defined as $\mathcal{L}^L(\{(\boldsymbol{a}_j, \boldsymbol{u}_j)\}_{j=1}^N; \theta) := \frac{1}{N}\sum_{i=1}^N \|\boldsymbol{u}_j - \mathcal{N}(\boldsymbol{a}_j;\theta)\|_{l^2}/\|\boldsymbol{u}_j\|_{l^2}$, where $\|\cdot\|_{l^2}$ is the canonical $l^2$ vector norm. For any $\xi \in \mathbb{Z}_n^2 := \left\{\xi \in \mathbb{Z}^2 \mid -n/2+1 \leqslant \xi_j \leqslant n/2, j = 1,2\right\}$, the normalized discrete Fourier transform (DFT) coefficients of $f$ writes $\mathcal{F}(f)(\xi) := \frac{1}{\sqrt{n}}\sum_{x\in\mathsf{G}^2} f(x)e^{-2i\pi x\cdot\xi}$. Then we define $\|u\|_h := \sqrt{\sum_{\xi\in\mathbb{Z}_n^2}|\xi|^2(\mathcal{F}(u)(\xi))^2}$. The empirical $H^1$ loss function is given by,

$$\mathcal{L}^H(\{(\boldsymbol{a}_j, \boldsymbol{u}_j)\}_{j=1}^N; \theta) := \frac{1}{N}\sum_{i=1}^N \|\boldsymbol{u}_j - \mathcal{N}(\boldsymbol{a}_j;\theta)\|_h/\|\boldsymbol{u}_j\|_h, \tag{4.1}$$

$\mathcal{L}^H$ can be viewed as a weighted $\mathcal{L}^L$ loss with weights $|\xi|^2$, which force the operator to capture the high frequency components in the solution. In this work, we use the equivalent frequency space representation of discrete $H^1$ norm, and in general, discrete $H^1$ norm in real space can also be adapted.

### 4.2 Multiscale Elliptic Equation



(a) coefficients (b) reference solutions (c) HT-Net solutions

Figure 4.1: Top: (a) smooth coefficient in Li et al. (2021), with $a_{\max} = 12$, $a_{\min} = 3$ and $c = 9$, (b), reference solution, (c) HT-Net solution; bottom: rough coefficients with $a_{\max} = 12$, $a_{\min} = 2$ and $c = 20$, (b), reference solution, (c) HT-Net solution. See Appendix E for details.

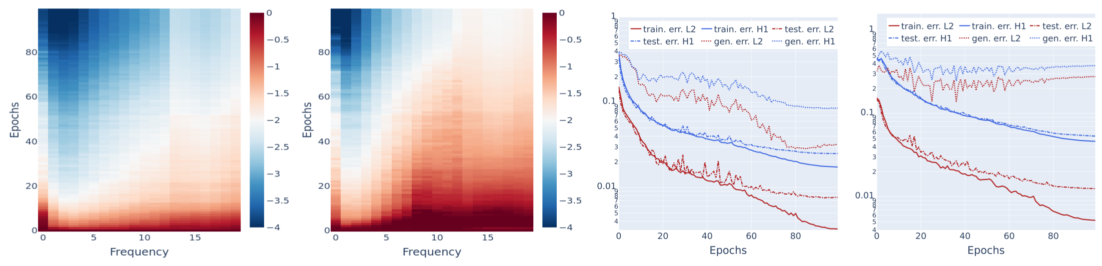
We apply the HT-Net to learn the coefficient to solution operator for equation 2.1. We use the two-phase coefficient model in the FNO paper Li et al. (2021), which is also benchmarked in Gupta et al. (2021); Cao (2021). In previous works, the coefficients do not oscillate much and the solutions look smooth. We change the parameters which control the smoothness and contrast of the coefficients, such that the solutions contain more roughness, see Figure 4.1. We refer the readers to Appendix E for data generation details. We also include experiments for multiscale trigonometric coefficients with higher contrast in Appendix F.1.

We list the relative $H^1$ and $L^2$ testing errors for the cases with smooth and rough two-phase coefficients in Table 1, which demonstrate that HT-Net outperforms other methods by a huge margin, for both smooth and rough cases. In particular, the comparison between HT-Net and its two level version indicates that increasing the number of levels can increase accuracy.

| | smooth | | | rough | | |
|---|---|---|---|---|---|---|
| | epochs=100 | epochs=500 | | epochs=100 | | epochs=500 |
| | $n = 141$ | $n = 141$ | $n = 211$ | $n = 256$ | $n = 512$ | $n = 256$ |
| FNO2d | 0.811/4.893 | 0.687/4.547 | 0.692/4.547 | 1.872/12.698 | 1.794/12.443 | 1.716/12.421 |
| FNO2d-$H^1$ | 0.761/3.155 | 0.675/3.931 | 0.664/4.040 | 1.126/4.232 | 1.236/5.821 | 0.974/3.851 |
| MWT | — | — | — | 1.324/4.687 | 1.291/4.290 | 1.279/4.446 |
| GT Ln on $K, V$ | 1.870/5.710 | 1.026/3.899 | 1.093/3.450 | 2.092/6.802 | 2.256/7.685 | 2.025/6.168 |
| HT-Net | $\mathbf{0.445/1.190}$ | $\mathbf{0.313/0.838}$ | $\mathbf{0.291/0.815}$ | $\mathbf{0.759/2.566}$ | $\mathbf{0.778/2.582}$ | $\mathbf{0.528/1.892}$ |
| HT-Net 2-level | — | — | — | $0.840/2.600$ | $0.907/3.213$ | $0.604/2.013$ |

Table 1: Relative $L^2$ and $H^1$ testing error ($\times 10^{-2}$ ) of the smooth and rough cases. FNO2d-$H^1$ is FNO2d with $H^1$ loss; MWT Gupta et al. (2021) only supports resolution with powers of two; HT-Net has 3 levels; HT-Net 2-level is HT-Net with 2 levels. $n$ is the one dimensional resolution for both training and testing.

**Spectral Bias in Operator Learning** We have observed the spectral bias phenomena in the training of neural operators. In Figure 4.2, we compare HT-Net trained with $H^1$ and $L^2$ losses, and show the evolutions of errors in the frequency domain as well as the loss curves. Comparison shows that HT-Net with $H^1$ loss has faster decay for high frequencies, and the errors over all the frequencies decay more uniformly. Also, the HT-Net with $H^1$ loss has better testing and generalization errors, although the training errors are comparable.



(a) HT-Net with $H^1$ loss (b) HT-Net with $L^2$ loss (c) HT-Net with $H^1$ loss (d) HT-Net with $L^2$ loss

Figure 4.2: In (a) HT-Net trained with $H^1$ loss, and (b) HT-Net trained with $L^2$ loss, we show the evolution of the errors with x-axis for frequency, y-axis for training epochs, and colorbar for the magnitude of $L^2$ error on each frequency in $\log_{10}$ scale, the error for each frequency is normalized frequency-wise by the error at epoch 0. The loss curves with training, testing, and generalization errors are shown in (c) for HT-Net trained with $H^1$ loss, and in (d) for HT-Net trained with $L^2$ loss.

**Generalization Errors** HT-Net generalizes better on test data from samples out of distribution. In addition to what we have showed in (c,d) of Figure 4.2, we compare HT-Net with other models in Table 2. The models are trained with the rough two-phase dataset in Table 4.1, and tested on out-of-distribution data. HT-Net outperforms other models by an order of magnitude.

| | FNO2d | FNO2d $H^1$ | MWT | HT-NET |
|---|---|---|---|---|
| $n = 256$ | 20.27 | 11.49 | 21.901 | **3.182** |

Table 2: Relative $L^2$ error ($\times 10^{-2}$) for out of distribution data, with $a_{\max} = 12$, $a_{\min} = 3$ and $c = 18$.

### 4.3 Navier Stokes

We consider the 2D Navier-Stokes equation in vorticity form on the unit torus, which is also benchmarked in Li et al. (2021)

$$\begin{aligned} \partial_t w(x,t) + u(x,t)\cdot\nabla w(x,t) &= \nu\Delta w(x,t) + f(x), && x \in (0,1)^2, t \in (0,T] \\ \nabla\cdot u(x,t) &= 0, && x \in (0,1)^2, t \in [0,T] \\ w(x,0) &= w_0(x), && x \in (0,1)^2 \end{aligned}$$

with velocity $u$, vorticity $w = \nabla\times u$, initial vorticity $w_0$, viscosity $\nu > 0$ ($\sim \mathrm{Re}^{-1}$ with Re being the Reynolds number), and forcing term $f$. We learn the operator $\mathcal{S} : w(\cdot, 0 \le t \le 10) \to w(\cdot, 10 \le t \le T)$, mapping the vorticity up to time 10 to the vorticity up to some later time $T > 10$. We experiment with viscosities $\nu = 1\mathrm{e}-3, 1\mathrm{e}-4, 1\mathrm{e}-5$, and decrease the final time $T$ accordingly as the Reynolds numbers increase and the dynamics become chaotic.

**Time dependent neural operator** Following the set up in Li et al. (2021), we fix the resolution to be $64 \times 64$ for both training and testing. The ten time slices of solution $w(\cdot, t)$ at $t = 0, ..., 9$ are taken as the input data to the neural operator $\mathcal{N}$ which maps the solution at the previous 10 time steps to the next time step (2D functions to 2D functions). This procedure can be repeated recurrently until the final time $T$, which is often called the rolled-out prediction. We list the results of HT-Net together with FNO-3D (convolution in space-time), FNO-2D, U-Net in Ronneberger et al. (2015) in Table 3, and we observe that HT-Net is significantly better than other methods.

| | #Parameters | $T = 50$, $\nu = 1\mathrm{e}-3$, $N = 1000$ | $T = 30$, $\nu = 1\mathrm{e}-4$, $N = 1000$ | $T = 30$, $\nu = 1\mathrm{e}-4$, $N = 10000$ | $T = 20$, $\nu = 1\mathrm{e}-4$, $N = 10000$ | $T = 20$, $\nu = 1\mathrm{e}-5$, $N = 1000$ |
|---|---|---|---|---|---|---|
| FNO-3D | $6,558,537$ | 0.0086 | 0.1918 | 0.0820 | — | 0.1893 |
| FNO-2D | 414,517 | 0.0128 | 0.1559 | 0.0834 | 0.0189 | 0.1556 |
| U-Net | $24,950,491$ | 0.0245 | 0.2051 | 0.1190 | 0.0229 | 0.1982 |
| HT-Net | $10,707,204$ | **0.0050** | **0.0517** | **0.0194** | **0.0053** | **0.0690** |

Table 3: Benchmarks for the Navier Stokes equation. The resolution is $64 \times 64$ for both training and testing, and all models are trained for 500 epochs. $N$ is the size of training samples.

## 5 Conclusion

We have built HT-Net, a hierarchical transformer based operator learning model for multiscale PDEs, which allows nested computation of features and self-attentions, and in turn provide a representation for the multiscale solution space. The reduce operations, multilevel local aggregations, and decompose operations form a fine-coarse-fine V-cycle for the feature update. We also introduced the empirical $H^1$ loss to reduce the spectral bias for the approximation of multiscale functions. HT-Net can provide much better accuracy and robustness compared with state-of-the-art (SOTA) neural operators, which is demonstrated by the multiscale elliptic and Naiver-Stokes benchmarks.

In this paper, we use regular grid for the discretization of PDE, HT-Net can be extended to more flexible setups such as scattered points and graph neural networks, which offers more opportunities to take advantage of the hierarchical representation.

## Ethics Statement

This work proposes a hierarchical transformer operator learning model for multiscale PDEs. As stated in the introduction, solving multiscale PDEs is associated with some of the most challenging practical problems such as reservoir modeling, fracture and fatigue prediction, high frequency scattering, weather forecasting, etc. Potentially, HT-Net solvers could help to reduce the prohibitively expensive computational cost of those simulations. The negative consequences are not obvious. Though in theory any technique can be misused, it is not likely to happen at the current stage.

## Reproducibility Statement

The datasets for Section 4 are either downloaded (for smooth two-phase coefficients and Navier-Stokes) from https://github.com/zongyi-li/fourier_neural_operator, or generated (for rough Darcy coefficients) using the code from the same website. We implemented $\mathcal{P}_1$ finite element method in MATLAB to solve equation 2.1 with multiscale trigonometric coefficients in Appendix F.1, and in FreeFEM to solve the Helmholtz equation in Appendix F.2. We have included introductions of the relevant mathematical and data generation concepts in the appendix.

We put the code for and also a link for datasets at the anonymous Github page https://github.com/Shengren-Kato/VFMM-ICLR2023.git. Supplementary descriptions of the code are also provided in the page.

## References

R. A. Adams and J. J. Fournier. Sobolev spaces, volume 140. Elsevier, 2003.

M. Bebendorf. Efficient inversion of the galerkin matrix of general second-order elliptic operators with nonsmooth coefficients. Math. Comp., 74(251):1179–1199, 2005.

M. Bebendorf. Hierarchical matrices, volume 63 of Lecture Notes in Computational Science and Engineering. Springer-Verlag, Berlin, 2008. A means to efficiently solve elliptic boundary value problems.

A. Bensoussan, J. L. Lions, and G. Papanicolaou. Asymptotic analysis for periodic structure. North Holland, Amsterdam, 1978.

L. Berlyand and H. Owhadi. Flux norm approach to finite dimensional homogenization approximations with non-separated scales and high contrast. Arch. Ration. Mech. Anal., 198(2):677–721, 2010.

G. Beylkin and N. Coult. A multiresolution strategy for reduction of elliptic PDEs and eigenvalue problems. Appl. Comput. Harmon. Anal., 5(2):129–155, 1998. ISSN 1063-5203.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. The SMAI journal of computational mathematics, 7:121–157, 2021.

L. V. Branets, S. S. Ghai, L. L., and X.-H. Wu. Challenges and technologies in reservoir modeling. Commun. Comput. Phys., 6(1):1–23, 2009.

M. E. Brewster and G. Beylkin. A multiresolution strategy for numerical homogenization. Appl. Comput. Harmon. Anal., 2(4):327–349, 1995. ISSN 1063-5203.

Max Budninskiy, Houman Owhadi, and Mathieu Desbrun. Operator-adapted wavelets for finite-element differential forms. Journal of Computational Physics, 388:144–177, 2019.

Wei Cai, Xiaoguang Li, and Lizuo Liu. A phase shift deep neural network for high frequency approximation and wave problems. SIAM Journal on Scientific Computing, 42(5):A3285–A3312, 2020.

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. arXiv preprint arXiv:2009.14794, 2020.

Doina Cioranescu, Alain Damlamian, and Georges Griso. The periodic unfolding method in homogenization. SIAM Journal on Mathematical Analysis, 40(4):1585–1620, 2008.

Weinan E and B. Engquist. Multiscale modeling and computation. Notices Amer. Math. Soc., 50(9):1062–1070, 2003.

Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Communications in Mathematics and Statistics, 5(4):349–380, 2017.

B. Engquist and P. E. Souganidis. Asymptotic and numerical homogenization. Acta Numerica, 17:147–190, 2008.

Yuwei Fan, Jordi Feliu-Fabá, Lin Lin, Lexing Ying, and Leonardo Zepeda-Nunez. A multiscale neural network based on hierarchical nested bases. Res. Math. Sci., 6(21), 2019a.

Yuwei Fan, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núnez. A multiscale neural network based on hierarchical matrices. Multiscale Modeling & Simulation, 17(4):1189–1213, 2019b.

L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. J. Comput. Phys., 73(2):325–348, 1987.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. Advances in Neural Information Processing Systems, 34:24048–24062, 2021.

W. Hackbusch. Multigrid Methods and Applications, volume 4 of Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 1985.

Wolfgang Hackbusch. Hierarchical matrices: algorithms and analysis. Springer, Berlin, 2015.

Wolfgang Hackbusch, Lars Grasedyck, and Steffen Börm. An introduction to hierarchical matrices. In Proceedings of Equadiff 10, pp. 101–111. Masaryk University, 2002.

Helmut Harbrecht and Christoph Schwab. Sparse tensor finite elements for elliptic multiple scale problems. Computer Methods in Applied Mechanics and Engineering, 200(45):3100–3110, 2011.

Moritz Hauck and Daniel Peterseim. Multi-resolution localized orthogonal decomposition for helmholtz problems. Multiscale Model. Simul., 20(2):657–684, 2022.

K. Ho and L. Ying. Hierarchical interpolative factorization for elliptic operators: Differential equations. Communiations on Pure and Applied Mathematics, 69(8):1415–1451, 2016.

T. Y. Hou, X.-H. Wu, and Z. Cai. Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients. Math. Comp., 68(227):913–943, 1999.

Vladimir Kazeev, Ivan Oseledets, Maxim V. Rakhuba, and Christoph Schwab. Quantized tensor fem for multiscale problems: Diffusion problems in two and three dimensions. Multiscale Modeling & Simulation, 20(3):893–935, 2022.

Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. European Journal of Applied Mathematics, 32(3):421–435, 2020.

Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. Journal of Machine Learning Research, 22:Art–No, 2021.

Xi-An Li, Zhi-Qin John Xu, and Lei Zhang. A multi-scale dnn algorithm for nonlinear elliptic equations with multiple scales. Communications in Computational Physics, 28(5):1886–1906, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. The International Conference on Learning Representations, 2021.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022, 2021.

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.

A. Målqvist and D. Peterseim. Localization of elliptic multiscale problems. Math. Comp., 83(290):2583–2603, 2014.

Tan Nguyen, Vai Suliafu, Stanley Osher, Long Chen, and Bao Wang. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. Advances in neural information processing systems, 34:29449–29463, 2021.

H. Owhadi. Multigrid with Rough Coefficients and Multiresolution Operator Decomposition from Hierarchical Information Games. SIAM Rev., 59(1):99–149, 2017.

H. Owhadi and L. Zhang. Metric-based upscaling. Comm. Pure Appl. Math., 60(5):675–723, 2007.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. arXiv preprint arXiv:2103.02143, 2021.

Nasim Rahaman, Devansh Arpit, Aristide Baratin, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of deep neural networks. International Conference on Machine Learning, 2019.

Maziar Raissi, Paris Perdikaris, and George 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.

Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In Advances in Neural Information Processing Systems, volume 32, pp. 4761–4771, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pp. 234–241. Springer, 2015.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. Journal of Computational Physics, 375:1339–1364, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

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

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 384:113938, 2021.

H. Xie, L. Zhang, and H. Owhadi. Fast eigenpairs computation with operator adapted wavelets and hierarchical subspace corrections. SIAM Journal on Numerical Analysis, 57(6):2519–2550, 2019. preprint.

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

Zhi-Qin J Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. International Conference on Neural Information Processing, pp. 264–274, 2019.

L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole method in two and three dimensions. Journal of Computational Physics, 196(2):591–626, 2004.

Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, , Sercan Ö. Arık, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In AAAI Conference on Artificial Intelligence (AAAI), 2022.

Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. J. Comput. Phys., 366:415–447, 2018.

## A Multiscale PDE

In this section, we introduce some mathematical and numerical concepts related to multiscale PDEs.

### A.1 Asymptotic Homogenization

Here we introduce some basic formulation of asymptotic homogenization. We suppose that $a(x)$ in equation 2.1 takes a special form $a(\frac{x}{\varepsilon})$, namely,

$$\begin{cases} -\text{div}\left(a\left(\frac{x}{\varepsilon}\right)\nabla u^{\varepsilon}\right) = f, \quad x \in D, \subset \mathbb{R}^d, a(\cdot) \text{ periodic in } y = x/\varepsilon, \\ u^{\varepsilon} = 0, \qquad\qquad\qquad\qquad\qquad\qquad\qquad x \in \partial D. \end{cases}$$

It can be derived from asymptotic expansion of the two-scale function $u^{\varepsilon} = u(x, x/\varepsilon)$ in $\varepsilon$, and justified rigorously that $u^{\varepsilon} \rightharpoonup u_0$ in $H^1$, with $u_0(x)$ the solution of the homogenized problem

$$\begin{cases} -\text{div}_x(a^0\nabla_x u_0) = f, \quad x \in D, \\ u_0 = 0, \qquad\qquad\qquad x \in \partial D. \end{cases}$$

which contains only coarse scale information. The homogenized coefficient $a^0$ can be computed by the formula $a^0_{ij} = \int_Y (e_i + \nabla\chi_i)^T a(y)(e_j + \nabla\chi_j)dy$, where $\chi_i$ solves the cell problems given by

$$\begin{cases} -\mathrm{div}_y\left(a\left(\nabla_y \chi_i + e_i\right)\right) = 0, & x \in Y, 1 \le i \le d, \\ \chi_i \in H^1_{\text{per}}(Y). \end{cases}$$

with $Y$ being the $d$ dimensional torus. Asymptotic homogenization provides an approximate solution $\hat{u}^\varepsilon = u_0 - \varepsilon \sum \chi_i\left(\frac{x}{\varepsilon}\right)\frac{\partial u_0}{\partial x_i}$ such that $\|u^\varepsilon - \hat{u}^\varepsilon\|_{H^1} \le c\varepsilon^{\frac{1}{2}}$. We note that to find the homogenization approximation $\hat{u}^\varepsilon$ only requires the coarse scale solution $u_0$ and precomputation of $d$ cell problems for $\chi_i$ which do not depend on $f$ and $D$.

### A.2 Numerical Homogenization and Multilevel/Multigrid Methods

Given the smallest scale of the multiscale problem $\varepsilon$ and a coarse computational scale determined by the available computational power and the desired precision, with $\varepsilon \ll H \ll 1$. The goal of numerical homogenization is to construct a finite dimensional approximation space $V_H$ and to seek an approximate solution $u_H \in V_H$, such that, the accuracy estimate $\|u - u_H\| \le CH^\alpha$ holds for optimal choices of the norm $\|\cdot\|$ and the exponent $\alpha$, and optimal computational cost holds with $V_H$ constructed via precomputed subproblems which are localized, independent and do not depend on the RHS and boundary condition of the problem.

In recent two decades, great progress have been made in this area Hou et al. (1999); Målqvist & Peterseim (2014); Owhadi & Zhang (2007), approximation spaces with optimal accuracy (in the sense of Kolmogorov $N$-width Berlyand & Owhadi (2010)) and cost can be constructed for elliptic equation with fixed rough coefficients. For multiscale PDEs, operator learning methods can be seen as a step forward from numerical homogenization, since they can be applied to an ensemble of coefficients, and the decoder can be interpreted as the basis of the underlying problem as well.

For multiscale PDEs, multilevel/multigrid methods can be seen as the multilevel generalization of the numerical homogenization methods. Numerical homogenization can provide coarse spaces with optimal approximation and localization properties. Recently, operator-adapted wavelets (gamblets) Owhadi (2017); Xie et al. (2019) have been developed, and enjoy three properties that are ideal for the construction of efficient direct methods: scale orthogonality, well-conditioned multi-resolution decomposition, and localization. Gamblets has been generalized to solve Navier-Stokes equation Budninskiy et al. (2019) and Helmholtz equation Hauck & Peterseim (2022) efficiently.

## B Hierarchical Matrix Perspective

The hierarchical nested attention Algorithm 1 resembles the celebrated hierarchical matrix method Hackbusch (2015), in particular, the $\mathscr{H}^2$ matricies from the perspective of matrix operations. In the following, we demonstrate Algorithm 1 in matrix formulas.

**STEP 0:** Get the $\boldsymbol{q}_i^{(r),t}$, $\boldsymbol{k}_j^{(r),t}$, $\boldsymbol{v}_i^{(r),t}$ for $j \in \mathcal{I}^{(r)}$.

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

$$\begin{bmatrix} \vdots \\ \boldsymbol{q}_i^{(r)} \\ \vdots \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{W}^Q & & & \\ & \boldsymbol{W}^Q & & \\ & & \ddots & \\ & & & \boldsymbol{W}^Q \end{bmatrix}}_{|\mathcal{I}^{(r)}|} \left.\begin{bmatrix} \begin{bmatrix} \vdots \\ \boldsymbol{f}_i^{(r)} \\ \vdots \end{bmatrix} \end{bmatrix}\right\} |\mathcal{I}^{(r)}|,$$

the keys $\boldsymbol{k}^{(r)}$ and values $\boldsymbol{v}^{(r)}$ can follow the similar procedure.

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

The reduce operations corresponds to $\begin{bmatrix} \vdots \\ \boldsymbol{q}_i^{(m)} \\ \vdots \end{bmatrix} = \mathbf{R}^{(m)} \begin{bmatrix} \vdots \\ \boldsymbol{q}_i^{(m+1)} \\ \vdots \end{bmatrix}$, where the reduce matrix is given by

$$\mathbf{R}^{(m)} := \underbrace{\begin{bmatrix} \boldsymbol{R}_0^{(m)} & \boldsymbol{R}_1^{(m)} & \boldsymbol{R}_2^{(m)} & \boldsymbol{R}_3^{(m)} & & & & & & & & & & \\ & & & & \boldsymbol{R}_0^{(m)} & \boldsymbol{R}_1^{(m)} & \boldsymbol{R}_2^{(m)} & \boldsymbol{R}_3^{(m)} & & & & & & \\ & & & & & & & & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & & & & & & \boldsymbol{R}_0^{(m)} & \boldsymbol{R}_1^{(m)} & \boldsymbol{R}_2^{(m)} & \boldsymbol{R}_3^{(m)} \end{bmatrix}}_{|\mathcal{I}^{(m+1)}|} \Bigg\} |\mathcal{I}^{(m)}|,$$

and $\boldsymbol{R}_0^{(m)}, \boldsymbol{R}_1^{(m)}, \boldsymbol{R}_2^{(m)}, \boldsymbol{R}_3^{(m)} \in \mathbb{R}^{C^{(m-1)}\times C^{(m)}}$ are learnable parameter matrix (in practice, queries, keys, and values use different $\boldsymbol{R}_0^{(m)}, \boldsymbol{R}_1^{(m)}, \boldsymbol{R}_2^{(m)}, \boldsymbol{R}_3^{(m)}$ to enhance the expressivity).

and inductively, $\begin{bmatrix} \vdots \\ \boldsymbol{q}_i^{(m)} \\ \vdots \end{bmatrix} = \mathbf{R}^{(m)} \cdots \mathbf{R}^{(r)} \begin{bmatrix} \vdots \\ \boldsymbol{q}_i^{(r)} \\ \vdots \end{bmatrix}$, and also $\begin{bmatrix} \vdots \\ \boldsymbol{v}_i^{(m)} \\ \vdots \end{bmatrix} = \mathbf{R}^{(m)} \cdots \mathbf{R}^{(r)} \begin{bmatrix} \vdots \\ \boldsymbol{v}_i^{(r)} \\ \vdots \end{bmatrix}$.

**STEP 2:** With the $m$-th level queries and keys, we can calculate the attention matrix $\boldsymbol{G}_{\text{loc}}^{(m)}$ at $m$-th level with $G_{\text{loc,i,j}}^{(m)} := \exp(\boldsymbol{q}_i^{(m),t} \cdot \boldsymbol{k}_j^{(m),t})$ for $i \in \mathcal{N}^{(m)}(j)$, or $i \sim j$.

**STEP 3:** The decompose operations, opposite to the reduce operations, corresponds to the transpose of the following matrix

$$\mathbf{D}^{(m)} := \underbrace{\begin{bmatrix} \boldsymbol{D}_0^{(m)} & \boldsymbol{D}_1^{(m)} & \boldsymbol{D}_2^{(m)} & \boldsymbol{D}_3^{(m)} & & & & & & & & & & \\ & & & & \boldsymbol{D}_0^{(m)} & \boldsymbol{D}_1^{(m)} & \boldsymbol{D}_2^{(m)} & \boldsymbol{D}_3^{(m)} & & & & & & \\ & & & & & & & & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & & & & & & \boldsymbol{D}_0^{(m)} & \boldsymbol{D}_1^{(m)} & \boldsymbol{D}_2^{(m)} & \boldsymbol{R}_3^{(m)} \end{bmatrix}}_{|\mathcal{I}^{(m+1)}|} \Bigg\} |\mathcal{I}^{(m)}|,$$

Eventually, the $m$-th level aggregation in Figure 3.1 contributes to the final output $\boldsymbol{f}^{(r,t+1)}$ in the form $\mathbf{D}^{(r),\mathbf{T}} \cdots \mathbf{D}^{(m),\mathbf{T}} \boldsymbol{G}_{\text{loc}}^{(m)} \mathbf{R}^{(m)} \cdots \mathbf{R}^{(r)} \begin{bmatrix} \vdots \\ \boldsymbol{v}_i^{(r),t} \\ \vdots \end{bmatrix}$ and

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

The matrix in equation B.1 resembles the three level $\mathscr{H}^2$ matrix decomposition illustrated in the following Figure B.1 for a three-level decomposition, we also refer to Hackbusch (2015) for detailed description. The sparsity of the matrix 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 B.1 also implies $\mathcal{O}(N)$ complexity of the Algorithm 1.

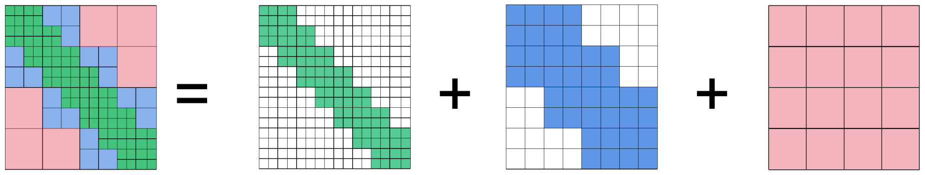


Figure B.1: The hierarchical admissible partition.

To this end, we connect our hierarchical attention with $\mathscr{H}^2$ matrix. From the perspective of $\mathscr{H}^2$ matrix, we view equation B.1 as an approximation (in the matrix form) of the finest level global attention equation 3.1. But one should note that, we take $\boldsymbol{W}^Q \boldsymbol{f}_i^{(m)}$ and $\boldsymbol{R}_0^{(m)} \boldsymbol{q}_i^{(m)}$ as the matrix vector multiplications with $\boldsymbol{q}_i^{(m)} \in \mathbb{R}^{C^{(m)}}$ and $\boldsymbol{R}_0^{(m)} \in \mathbb{R}^{C^{(m-1)} \times C^{(m)}}$, while in the $\mathscr{H}^2$ matrix setting, $\boldsymbol{W}^Q$, $\boldsymbol{R}_0^{(m)}$, $\boldsymbol{f}_i^{(m)}$ are all scalars.

## C PROOF OF PROPOSITION 3.1

*Proof* For each level $m$, the cost to compute equation 3.3 is $c(|\mathcal{I}^{(m)}|\mathcal{C}^{(m)})$ since for each $i \in \mathcal{I}^{(m)}$ the cardinality of the neighbour set $\mathcal{N}^{(m)}(i)$ is bounded by a constant $c$. The reduce operation $\boldsymbol{f}_i^{(k-1)} = \mathcal{R}^{(k-1)}(\{\boldsymbol{f}_j^{(m)}\}_{j \in i^{(k-1,k)}})$ costs at most $|\mathcal{I}^{(m)}|\mathcal{C}^{(m)}\mathcal{C}^{(k-1)}$ flops and so does the decompose operation at the same level. Therefore, for each level, the operation cost is $c(|\mathcal{I}^{(m)}|\mathcal{C}^{(m)}) + 2|\mathcal{I}^{(m)}|\mathcal{C}^{(m)}\mathcal{C}^{(m-1)}$. 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 $\sim \mathcal{O}(N)$. □

## D IMPLEMENTATION DETAILS

In the HT-Net implementation, we follow the window attention scheme in Liu et al. (2021) for the definition of the neighborhood $\mathcal{N}^{(\cdot)}(\cdot)$ in equation 3.2 and equation 3.3. In this paper, we choose $r = 3$ for the depth of the HT-Net.

For dataset with resolution $n_f \times n_f$, such as in example 4.2, the input feature $\boldsymbol{f}^{(3)}$ is represented as a tensor of size $n \times n \times C$. The self-attention is first computed within a local window on level 3. Then the reduce layer concatenates the features of each group of $2 \times 2$ 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 $\boldsymbol{f}^{(2)}$ 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 $\boldsymbol{f}^{(1)}$ of size $\frac{n}{4} \times \frac{n}{4} \times 4C$.

For the decompose process, starting at level 1, a linear layer is applied to transform the $4C$-dimensional features $\boldsymbol{f}^{(1)}$ into $8C$-dimensional features. Each level 1 token with $8C$-dimensional features is decomposed into four level 2 tokens with $2C$-dimensional features and added to the level 2 feature $\boldsymbol{f}^{(2)}$ with output size of $\frac{n}{2} \times \frac{n}{2} \times 2C$. The procedure is repeated from level 2 level to level 3 with the output $\boldsymbol{f}(3)$ of size $n \times n \times C$.

We optimize models using the Adam optimizer with learning rate 1e$-$3 and the 1-cycle schedule as in Cao (2021). We use batch size 8 for experiments in Sections 4.2 and 4.3, and batch size 4 for experiments in Appendices F.1 and F.2.

For the rough coefficient experiment in Section 4.2, there are 1280 samples in the training set and 112 samples in the testing set. For the Navier-Stokes experiment in Section 4.3, we have tried $N = 1000$ and $N = 10000$ training samples as shown in Table 3, and the number of testing samples is 100 and 1000 respectively. For other experiments, the number of training samples is 1000, and the number of testing samples is 100.

All experiments are run on one NVIDIA A100 GPU.

## E Data Generation for the Two-Phase Coefficient in Section 4.2

The two-phase coefficients and solutions are generated according to `https://github.com/zongyi-li/fourier_neural_operator/tree/master/data_generation`, and used as operator learning benchmarks in Li et al. (2021); Gupta et al. (2021); Cao (2021). The coefficients $a(x)$ are generated according to $a \sim \mu := \psi_{\#}\mathcal{N}\left(0, (-\Delta + cI)^{-2}\right)$ with zero Neumann boundary conditions on the Laplacian. 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. The forcing term is fixed as $f(x) \equiv 1$. Solutions $u$ are obtained by using a second-order finite difference scheme on a suitable grid. The parameters $a_{\max}$ and $a_{\min}$ can control the contrast of the coefficient. The parameter $c$ can control the roughness (oscillation) of the coefficient, and the coefficient with a larger $c$ has rougher two-phase interfaces, as shown in Figure 4.1.

## F Additional Experiments

### F.1 Multiscale trigonometric coefficient

In this experiment, we consider equation 2.1 with multiscale trigonometric coefficient adapted from Owhadi (2017), such that $D = [-1, 1]^2$, $a(x) = \prod_{k=1}^{6} (1 + \frac{1}{2}\cos(a_k\pi(x_1 + x_2)))(1 + \frac{1}{2}\sin(a_k\pi(x_2 - 3x_1)))$, with $a_k = \text{rand}(2^{k-1}, 1.5 \times 2^{k-1})$, and fixed $f(x) \equiv 1$. The reference solutions are obtained using $\mathcal{P}_1$ FEM on a $1023 \times 1023$ grid. Datasets of lower resolution are sampled from the higher resolution dataset by linear interpolation. The experiment results for the multiscale trigonometric case for different resolutions are shown in Table 4. HT-Net obtains the best relative $L^2$ error compared to other neural operators at various resolutions at 600 epochs. Multiwavelet neural operator MWT has the second best performance as it also possess a multiresolution structure, but it does not adapt to the PDE. It is not surprising that FNO, an excellent smoother which filters higher frequency modes, fails to capture the high frequency oscillations of the solution. In contrast, our method has a better performance in this respect. See Figure F.1 for illustrations of the coefficient and comparison of the solutions at the slice $x = 0$.

| | Time per epoch | epochs=300 | | | epochs=600 | | |
|---|---|---|---|---|---|---|---|
| | for n=512 | n=128 | n=256 | n=512 | n=128 | n=256 | n=512 |
| FNO | 25.75s | 1.996 | 1.842 | 1.817 | 2.017 | 1.820 | 1.806 |
| GT | 106.54s | 1.524 | 1.070 | 1.093 | 1.448 | 0.938 | 0.970 |
| MWT | 40.57s | **1.115** | 1.007 | 1.006 | 1.112 | 0.985 | 0.977 |
| SWIN | 42.62s | 1.768 | 2.378 | 4.513 | 1.579 | 2.216 | 4.365 |
| HT-net | 52.93s | 1.204 | **0.923** | **0.902** | **1.030** | **0.766** | **0.745** |

Table 4: Relative error ($\times 10^{-2}$ ) of the multiscale trigonometric example.

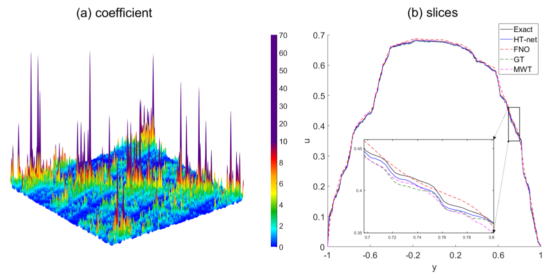


Figure F.1: (a) multiscale trigonometric coefficient, (b) slice of the solutions at $x = 0$.

## F.2 HELMHOLTZ EQUATION

We test the performance of HT-Net for the acoustic Helmholtz equation in highly heterogeneous media as an example for multiscale wave phenomena, whose solution is considerably expensive for complicated and large geological models. We adapt the setup from Hauck & Peterseim (2022),

$$\begin{cases} -\mathrm{div}(a(x)\nabla u(x)) - \kappa^2 u = f(x), & x \in D, \\ u(x) = 0, & x \in \partial D. \end{cases}$$

where the coefficient $a(x)$ takes the value 1 or $\varepsilon$ as shown in Figure F.2 with $\varepsilon^{-1} \in \mathrm{rand}(128, 256)$ , $\kappa = 9$, and

$$f(x_1, x_2) = \begin{cases} 10^4 \exp\left(\dfrac{-1}{1 - \frac{(x_1-0.125)^2+(x_2-0.5)^2}{0.05^2}}\right), & (x_1 - 0.125)^2 + (x_2 - 0.5)^2 < 0.05^2, \\ 0, & \text{else.} \end{cases}$$

In this example, MWT outperforms HT-net at the intermediate training stages and at the highest resolution ($s = 512$), it may attribute to the wave propagation nature of the problem or the regular arrangement of the coefficients $a(x)$. We will investigate this example further in our future studies.

| | epochs=300 | | | epochs=600 | | |
|---|---|---|---|---|---|---|
| | n=128 | n=256 | n=512 | n=128 | n=256 | n=512 |
| FNO | 7.508 | 8.125 | 8.408 | 6.844 | 7.267 | 7.896 |
| GT | 16.177 | 14.551 | 12.626 | 11.265 | 11.492 | 11.797 |
| MWT | **5.102** | **4.678** | **4.341** | 2.034 | 3.327 | **1.818** |
| SWIN | 25.640 | 30.263 | 31.015 | 20.175 | 26.024 | 28.474 |
| HT-net | 10.913 | 10.097 | 9.662 | **1.773** | **2.567** | 2.878 |

Table 5: Relative error ($\times 10^{-2}$ ) of the Helmholtz equation.

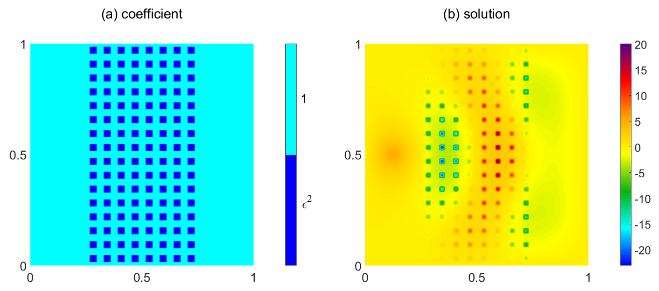


Figure F.2: (a) Heterogeneous coefficient $a(x)$, (b) the corresponding solution for $\varepsilon^{-1} = 225.6$, which is solved by $\mathcal{P}_1$ FEM implemented in FreeFEM.