

# On understanding and overcoming spectral biases of deep neural network learning methods for solving PDEs

Zhi-Qin John Xu<sup>a,1</sup>, Lulu Zhang<sup>a,1</sup>, Wei Cai<sup>b,\*</sup>

<sup>a</sup>*Institute of Natural Sciences and School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai, 200240, China*

<sup>b</sup>*Dept of Mathematics, Southern Methodist University, Dallas, 75252, Texas, USA*

---

## Abstract

In this review, we survey the latest approaches and techniques developed to overcome the spectral bias towards low frequency of deep neural network learning methods in learning multiple-frequency solutions of partial differential equations. Open problems and future research directions are also discussed.

*Keywords:* neural networks, spectral bias, deep learning, PDEs.

---

## 1. Introduction

With its remarkable successes in image classification [1], speech recognition [2], and natural language processing [3, 4, 5], the deep neural network (DNN) has recently also found many applications in scientific computing. These include modeling and predicting physical phenomena with various applications in material sciences, drug design, etc. [6, 7, 8]. Many of these applications involve solving partial differential equations (PDEs) such as the Schrodinger equation for quantum systems [9, 10], Maxwell's equations for electromagnetic phenomena [11], the Poisson-Boltzmann equation for molecular solvation [12, 13], the Navier-Stokes equations in fluid dynamics [14, 15], the elasticity dynamic equations in civil engineering and geophysics [16], the Hamilton-Jacobi-Bellman equations from optimal controls [17], to list just a

---

\*Corresponding author.

*Email addresses:* xuzhiqin@sjtu.edu.cn (Zhi-Qin John Xu ), zhangl9661@sjtu.edu.cn (Lulu Zhang ), cai@smu.edu (Wei Cai)

<sup>1</sup>Co-first author.

few. Traditionally, solutions to those equations in low dimensions are computed by finite element [18, 19, 20, 21], finite difference [22], spectral [23, 24] or integral equation methods [25] with much success and maturity. However, these methods face challenges when handling high-dimensional problems arising from treating stochastic effects, and material parameterization, or problems with complex geometries. Due to the compositional structure of the DNNs and the meshless random sampling during their learning, algorithms based on DNNs provide the potential of tackling high-dimensional problems without the curse of dimensionality [7, 8] where the traditional methods will not be able to handle at all, or of not suffering the costs and difficulties of meshing complex geometries.

Many physical problems in scientific computing engender solutions with a wide range of frequencies. Examples include turbulent flows [26, 27] where energy interactions among different scales in the form of vorticities and boundary layer effects; high-frequency waves in light-matter interactions from laser physics; and lithography and optical proximity correction in microchip designs at nanoscales and modern telecommunications [28, 29, 13]; quantum wave functions in many body electron systems [30]; Wigner quantum transport in nano-electronics [31, 13]; combustion dynamics [32], among many others. These wide-range frequency phenomena are described by various PDEs mentioned above. The multigrid (MG) methods [33, 34] developed to address the challenges of multiple-frequency issues in solving PDEs, though mainly in the low dimensional  $R^3$  space, took advantage of the fast high-frequency smoothing property of classical iterative methods (e.g., Jacobi and Gauss-Seidel) based on the splitting of the coefficient matrices from the discretization of the PDEs. The matrices from finite element and finite difference methods are sparse, thus making the iterative solvers [35] such as conjugate gradient and GMRES attractive, however, ill-conditioned with conditional numbers at the order of  $\frac{1}{h^2}$  ( and even worse for spectral methods [23]) for elliptic problems where the mesh size  $h$  is required to be fine enough to resolve the highest frequency component of the solution based on Shannon sampling theory [36]. Consequently, solving these linear systems necessitates the use of preconditioners [37, 38] with these iterative solvers. The main reason is that the large disparity in the spectra of the coefficient matrices creates difficulties in removing errors across all frequencies using traditional iterative solvers. However, the multigrid method with V- and W-cycles of smoothing and corrections between different grids can produce an efficient solution in achieving uniform reduction of errors over all possible frequen-

cies, and after many decades of research, the MG methods now provides a frequency uniform convergence for solving elliptic type of PDEs.

On the other hand, DNN as a new computational model for solving PDEs is still in its early stage of development. It lacks the sophistication, efficiency, reliability, and robustness of traditional numerical methods, even with recent intensive research in this area. The research community faces many challenges, one of them, the focus of this review, is the well-known spectral bias (or frequency principle) nature of the DNNs [39, 40, 41, 42]. This bias is evident in the approximation of functions or PDE solutions with a wide range of frequency content. Typically, the training of the DNNs demonstrates a preference towards the low frequency before the content of higher frequencies can be learned at latter stage of the training. This behavior contrasts with that of the classic iterative solvers. DNN-based computational methods developed for PDEs, particularly for high-dimensional problems common in many-body physics or problems with stochasticity in most engineering applications, or in complex geometry problems, face unique challenges due to the spectral bias. Thus, in order to make DNN-based algorithms to reach the same robustness and practicality as the classic MG method even for high dimensional problems, it is imperative to overcome the shortcoming of the spectral bias in learning wide-range frequency solutions. Intensive researches have been carried out by the computational community to improve the performance of DNNs. This review represents a first effort to summarize the state-of-the-art research directed towards this goal and aims to inspire further research to resolve the issue of spectral bias of DNN learning methods for PDEs more effectively.

The following lists background materials and surveyed methods, developed to overcome spectral biases of DNN based learning methods for PDEs:

- Preliminaries on DNN learning of PDE solutions
- Spectral bias of deep neural networks
- Frequency shifting approach
- Frequency scaling approach
- Hybrid approach
- Multi-scale neural operator and diffusion models.

## 2. Preliminaries on DNN learning of PDE solutions

### 2.1. Deep neural network (DNN)

We introduce some conventional notations for DNNs. A  $L$ -layer neural network is defined recursively as,

$$\begin{aligned} \mathbf{f}_{\boldsymbol{\theta}}^{[0]}(\mathbf{x}) &= \mathbf{x}, \\ \mathbf{f}_{\boldsymbol{\theta}}^{[l]}(\mathbf{x}) &= \sigma \circ (\mathbf{W}^{[l-1]} \mathbf{f}_{\boldsymbol{\theta}}^{[l-1]}(\mathbf{x}) + \mathbf{b}^{[l-1]}), \quad 1 \leq l \leq L-1, \\ \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) &\equiv \mathbf{f}_{\boldsymbol{\theta}}^{[L]}(\mathbf{x}) = \mathbf{W}^{[L-1]} \mathbf{f}_{\boldsymbol{\theta}}^{[L-1]}(\mathbf{x}) + \mathbf{b}^{[L-1]}, \end{aligned} \tag{1}$$

where  $\mathbf{x} \in \mathbb{R}^d$ ,  $\mathbf{W}^{[l]} \in \mathbb{R}^{m_{l+1} \times m_l}$ ,  $\mathbf{b}^{[l]} \in \mathbb{R}^{m_{l+1}}$ ,  $m_0 = d_{\text{in}} = d$  is the input dimension,  $m_L = d_o$  is the output dimension,  $\sigma$  is a scalar nonlinear function, and “ $\circ$ ” means entry-wise operation. We denote the set of all parameters in the network by  $\boldsymbol{\theta}$ .

The approximation capabilities of DNNs have been investigated extensively. Cybenko [43] demonstrated a universal approximation theorem of neural networks that any continuous function can be arbitrarily well approximated by sufficiently wide DNNs with one hidden layer with a continuous sigmoidal nonlinearity activation function. The sigmoidal function needs to saturate as the input variable goes to infinity. Almost simultaneously, Hornik et al. [44] proved an universal approximation theorem for squashing activation functions, which needs to be non-decreasing and also saturate as the input variable goes to infinity. Furthermore, Hornik [45] extended the activation functions to those arbitrarily bounded and non-constant ones. More generally, Leshno et al. [46] illustrated that DNNs with a locally bounded piecewise continuous activation function could approximate any continuous function to any degree of accuracy if and only if the network’s activation function is nonlinear and not a polynomial. E et al. [47] showed that functions from a Barron space can be approximated by two-layer neural networks without the curse of dimensionality. Lu et al. [48] established a quantitative approximation error characterization of deep ReLU networks for smooth functions in terms of arbitrary width and depth simultaneously. These collective works underscore the powerful approximation ability of DNNs. However, in practical applications, the use of appropriate optimization methods is also crucial for finding satisfactory solutions.

## 2.2. DNN based methods for solving PDEs

In solving PDEs, the empirical loss can be defined without labels as an unsupervised learning. In such cases, all possible data can be sampled with a non-zero probability, different from classical supervised learning problems. Let us consider the following equation,

$$\begin{aligned}\mathcal{L}[u](\mathbf{x}) &= f(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \Gamma[u](\mathbf{x}) &= g(\mathbf{x}), & \mathbf{x} \in \partial\Omega,\end{aligned}\tag{2}$$

where  $\Omega \subset \mathbb{R}^d$  is a domain and  $\partial\Omega$  is its boundary. Note that time-dependent problems can also be formulated as equations of this form with the time variable included as an additional dimension (though, only initial condition(s) at  $t = 0$  will be needed as an initial boundary value problem). Also, when dealing with time-dependent problems, it is crucial to consider the causality of the system, which refers to the causal connections between consecutive physical events. In this context, one event, the cause, must precede another event, the effect, in time. Taking causality into account when solving time-dependent problems is important, as it can significantly improve the performance and accuracy of the solutions [49].

### 2.2.1. PDE residual based learning – physics-informed neural network (PINN)

A common way to build a loss function borrows the idea of minimization of residuals from spectral methods [23] and the least square finite element (LSFE) method [21, 20]. Along this line, the earlier work of neural network methods [50, 51, 52, 53, 54] and later the physics-informed neural network (PINN) [14, 55] and deep least square methods using first order system formulations of PDEs [56, 57] attempt to find the solution of differential equations (DEs) by minimizing the least squares of the PDE’s residuals. And, the empirical risk with a least square loss for the PDE (2) can be defined by

$$\tilde{L}_{pde}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \|\mathcal{L}[u_{\boldsymbol{\theta}}](\mathbf{x}_i) - f(\mathbf{x}_i)\|_2^2,\tag{3}$$

where the dataset  $\{\mathbf{x}_i\}_{i=1}^n$  is randomly sampled from  $\Omega$  at each iteration step and, in addition, a boundary loss will be used to enforce the boundary condition, i.e., for the Dirichlet BC,

$$L_{bdry}(\boldsymbol{\theta}) = \frac{1}{N_{bdry}} \sum_{k=1}^{N_{bdry}} |u_{\boldsymbol{\theta}}(\mathbf{x}_k) - g(\mathbf{x}_k)|^2, \quad \mathbf{x}_k \in \partial\Omega.\tag{4}$$

where the dataset  $\{\mathbf{x}_k\}_{k=1}^{N_{bdry}}$  is also randomly sampled from  $\partial\Omega$  at each iteration step. Note that at each iteration, adaptive sampling based on  $\tilde{L}_{pde}(\boldsymbol{\theta})$  and  $L_{bdry}(\boldsymbol{\theta})$  might lead to more accurate results [58, 59], however, in this paper, for simplicity of presentation, a uniform distribution is chosen for the sampling.

### 2.2.2. Variational energy based Deep-Ritz method

Another very natural candidate for the loss function  $L(\boldsymbol{\theta})$  for the learning optimization procedure is the Ritz energy from the variational principle of elliptic operators [60].

The Deep-Ritz method [60] seeks a variational solution  $u(\mathbf{x})$  through the following minimization problem,

$$u = \arg \min_{v \in H_0^1(\Omega)} J(v), \quad (5)$$

where  $H_0^1$  is the Sobolev space for the set of admissible functions (also called trial functions, here represented by  $v$ ).

A typical variational problem is based on the following functional

$$J(v) = \int_{\Omega} \left( \frac{1}{2} |\nabla v(x)|^2 - f(x)v(x) \right) dx, \quad (6)$$

where  $f$  is a given function, representing external forcing to the system under consideration.

The PINN requires high-order derivatives in the definition of the loss function, which could potentially accelerate the convergence of high-frequency components during training as differentiation in the frequency domain is equivalent to a multiplication with the wave number  $k$  in the frequency domain, therefore increasing the weight of high-frequency components in the loss function. Lu et al. [61] and E et al. [62] have provided examples showing that differentiation accelerates the convergence of high frequencies. However, higher-order differentiation could also lead to training instability. In contrast, the DeepRitz, which reduces the differentiation by one order for second-order elliptic PDEs, offers a more stable training, albeit at the cost of slower convergence for high-frequency components. Generally, these methods still exhibit the characteristics of a preferential convergence in low-frequency components, and these formulations have been used for multi-frequency problems. The current literature lacks a rigorous and systemic comparison of the pros and cons of various formulations for different types of problems.

### 3. Spectral bias of deep neural networks

In the learning process of DNNs, there exists a spectral bias (studied in [63, 39] as a frequency principle) during training, supported by much empirical evidence [63, 39, 40, 42] and many theoretical works [39, 64, 65, 66, 67, 68, 41]. The spectral bias refers to the fact that DNNs tend to fit target functions from low to high frequencies, contrary to the behavior of conventional iterative schemes of numerical linear algebra [69]. As a result of the spectral bias, DNNs have been shown to face significant challenges in learning solutions with a wide range of frequencies.

#### 3.1. Empirical evidence of spectral bias

To gain insight into the training process, we visualize the training process in the frequency domain with a one-dimensional data with some given frequencies. The training samples are extracted from a one-dimensional target function with three frequencies  $f^*(x) = \sin(x) + \sin(3x) + \sin(5x)$  over the interval  $[-3.14, 3.14]$  in an evenly spaced range. The samples are represented as  $\{x_i, f(x_i)\}_{i=0}^{n-1}$ . We compute the discrete Fourier transform (DFT) of  $f^*(x)$  and the DNN output  $f_{\theta}(x)$ , and denote them as  $\hat{f}^*(k)$  and  $\hat{f}_{\theta}(k)$ , respectively, where  $k$  denotes the frequency. Fig. 1(a) illustrates the target function with its three distinct frequencies (indicated by red dots in Fig. 1(b)). The DNN is trained by using a mean squared error (MSE) loss and we consider the relative error of each frequency,  $\Delta_F(k) = \|\hat{h}_k - \hat{f}_k\|/\|\hat{f}_k\|$ . Fig. 1(c) shows the relative errors for the process, indicating that the DNN learns in the order of low to high frequencies.

The spectral bias of DNNs has been widely observed in high-dimensional problems and diverse settings in a series of experiments conducted in [39]. These experiments used common optimization methods, different network structures, various loss functions, and more. In addition, the spectral bias has also been found in the training of non-gradient based algorithms [70], such L-BFGS and quasi-Newton’s methods, etc.

#### 3.2. Qualitative theory of spectral bias

To gain an intuitive understanding of the underlying mechanism of the spectral bias, we review a simple analysis [71, 39, 64]. For this case, we

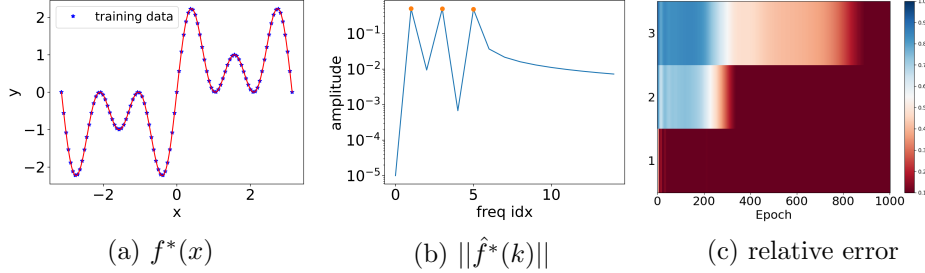


Figure 1: 1-d frequency principle. (a)  $f^*(x)$ . (b)  $\|\hat{f}^*(k)\|$ . (c)  $\Delta_F(k)$  of three important frequencies (indicated by red dots in the inset of (b)) against different training epochs (horizontal axis). Here the y-axis represents the 3 frequencies, and the colorbar indicates the loss values.

consider the activation function

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad x \in \mathbb{R},$$

and a DNN with one hidden layer, having  $m$  neurons, a 1-dimensional input  $x$ , and a 1-dimensional output, i.e.,

$$h(x) = \sum_{j=1}^m a_j \sigma(w_j x + b_j), \quad a_j, w_j, b_j \in \mathbb{R}, \quad (7)$$

where  $w_j$ ,  $a_j$ , and  $b_j$  are training parameters. We also denote  $\boldsymbol{\theta} = \{\theta_{ij}\}$  with  $\theta_{1j} = a_j$ ,  $\theta_{2j} = w_j$ , and  $\theta_{3j} = b_j$ ,  $j = 1, \dots, m$ .

Then, the Fourier transform of  $h(x)$  can be computed as follows

$$\hat{h}(k) = \sum_{j=1}^m \frac{2\pi a_j i}{|w_j|} \exp\left(\frac{i b_j k}{w_j}\right) \frac{1}{\exp(-\frac{\pi k}{2w_j}) - \exp(\frac{\pi k}{2w_j})}, \quad (8)$$

where  $k$  denotes the frequency. The loss at a frequency  $k$  is  $L(k) = \frac{1}{2} \|\hat{h}(k) - \hat{f}^*(k)\|^2$ , where  $f^*$  is the target function and the total loss function is defined as:  $L = \int_{-\infty}^{+\infty} L(k) dk$ . According to Parseval's theorem, this loss function in the Fourier domain is exactly the common mean squared error loss, that is,  $L = \int_{-\infty}^{+\infty} \frac{1}{2} (h(x) - f(x))^2 dx$ . Therefore, to analyse the error decay from a gradient descent training, we can use the loss in the frequency domain to see the error dependence on frequency. In practice, the weights are usually much smaller than 1, then, the last term in Eq. (8) is approximately



$\exp(-|\pi k/2w_j|)$ . Therefore, the absolute contribution from frequency  $k$  to the gradient for  $\theta_{lj}$  is

$$\left\| \frac{\partial L(k)}{\partial \theta_{lj}} \right\| \approx \|\hat{h}(k) - \hat{f}^*(k)\| \exp(-|\pi k/2w_j|) F_{lj}, \quad (9)$$

where  $\boldsymbol{\theta}_j \triangleq \{w_j, b_j, a_j\}$ ,  $\theta_{lj} \in \boldsymbol{\theta}_j$ ,  $F_{lj}$  is an  $O(1)$  function depending on  $\boldsymbol{\theta}_j$  and  $k$ . The term  $\exp(-|\pi k/2w_j|)$  shows that low frequency plays a dominant role for small training weights  $w_j$ , which leads to faster convergence of low-frequency components as manifested by the spectral bias of the DNNs.

The preceding analysis of the spectral bias relies on the form of the activation function, and most activation functions, such as tanh and ReLU, exhibit a similar decaying behavior in the frequency domain and, consequently, the spectral bias will also be readily observable. Also from this analysis, modifying the loss function to impose greater weight on high-frequency components can impact frequency convergence. One simple approach to mitigate spectral biases is incorporating derivatives of the DNN output with respect to the input in the loss function as the Fourier transform of  $\nabla_{\mathbf{x}} f_{\boldsymbol{\theta}}(x_i)$  equals the product of the transform of  $f_{\boldsymbol{\theta}}(\mathbf{x}_i)$  and frequency  $k$ , effectively prioritizing higher frequencies in the loss function, or contributing a factor  $k$  in Eq. (9). Approaches along this line can be found in [42, 61]. And, to accelerate the learning of high-frequency in fluid simulation, the loss of gradient information was also used in [72].

### 3.3. Linear quantitative theory of spectral bias

A series of works analyzed the spectral bias for two-layer wide DNNs with the Neural Tangent Kernel (NTK) approach [73, 74] with specific sample distributions [67, 75, 68] or any finite samples [76, 66, 65]. Additionally, in [77] the spectral bias was studied from the perspective of integral equation.

In this subsection, we review analysis work of the frequency dependence of the training error using the NTK approach together with linearization of network, i.e., a linear frequency principle (LFP) analysis for the spectral bias [65]. And, we consider the following gradient-descent flow dynamics of the empirical risk  $L_S$  of a network function  $f_{\boldsymbol{\theta}}(\cdot) = f(\cdot, \boldsymbol{\theta})$  parameterized by  $\boldsymbol{\theta}$  on a set of training data  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$

$$\begin{cases} \dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} L_S(\boldsymbol{\theta}), \\ \boldsymbol{\theta}(0) = \boldsymbol{\theta}_0, \end{cases} \quad (10)$$

where

$$L_S(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^n (f(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)^2. \quad (11)$$

Then, the training dynamics of the output function  $f(\cdot, \boldsymbol{\theta})$  are

$$\begin{aligned} \frac{d}{dt} f(\mathbf{x}, \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} f(\mathbf{x}, \boldsymbol{\theta}) \cdot \dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} f(\mathbf{x}, \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} L_S(\boldsymbol{\theta}) \\ &= -\nabla_{\boldsymbol{\theta}} f(\mathbf{x}, \boldsymbol{\theta}) \cdot \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta}) (f(\mathbf{x}_i, \boldsymbol{\theta}) - y_i) \\ &= -\sum_{i=1}^n K_m(\mathbf{x}, \mathbf{x}_i) (f(\mathbf{x}_i, \boldsymbol{\theta}) - y_i), \end{aligned} \quad (12)$$

where the NTK  $K_m(\mathbf{x}, \mathbf{x}')(t)$  [73] at time  $t$  and evaluated at  $(\mathbf{x}, \mathbf{x}') \in \Omega \times \Omega$  is defined by

$$K_m(\mathbf{x}, \mathbf{x}')(t) = \nabla_{\boldsymbol{\theta}} f(\mathbf{x}, \boldsymbol{\theta}(t)) \cdot \nabla_{\boldsymbol{\theta}} f(\mathbf{x}', \boldsymbol{\theta}(t)). \quad (13)$$

Now, consider a two-layer DNN

$$f(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{m}} \sum_{j=1}^m a_j \sigma(\mathbf{w}_j^{\top} \mathbf{x} + b_j), \quad (14)$$

where the vector of all parameters  $\boldsymbol{\theta}$  denotes all network parameters, initialized by a standard Gaussian distribution. As  $m \rightarrow \infty$ , the following linearization around initialization

$$f^{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}(t)) = f(\mathbf{x}; \boldsymbol{\theta}(0)) + \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}(0)) (\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)) \quad (15)$$

is an effective approximation of  $f(\mathbf{x}; \boldsymbol{\theta}(t))$ , i.e.,  $f^{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}(t)) \approx f(\mathbf{x}; \boldsymbol{\theta}(t))$  for any  $t$  [73, 78], which defines the linear regime. Note that,  $f^{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}(t))$ , linear in  $\boldsymbol{\theta}$  and nonlinear in  $\mathbf{x}$ , reserves the universal approximation power of  $f(\mathbf{x}; \boldsymbol{\theta}(t))$  as  $m \rightarrow \infty$ . In the rest of this sub-section, we do not distinguish  $f(\mathbf{x}; \boldsymbol{\theta}(t))$  from  $f^{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}(t))$ .

In the linear regime, one can prove [73]

$$K^*(\mathbf{x}, \mathbf{x}') := \lim_{m \rightarrow \infty} K_m(\mathbf{x}, \mathbf{x}')(t) = K(\mathbf{x}, \mathbf{x}')(0) \quad (16)$$

for any  $t$ . The gradient descent of the model thus becomes

$$\frac{d}{dt} \mathbf{u}(\mathbf{x}, t) = - \sum_{i=1}^n K^*(\mathbf{x}, \mathbf{x}_i) \mathbf{u}(\mathbf{x}_i, t), \quad (17)$$

where the residual  $\mathbf{u}(\mathbf{x}, t) = f(\mathbf{x}, \boldsymbol{\theta}(t)) - f^*(\mathbf{x})$ ,  $f^*(\mathbf{x})$  is the target function and  $y_i = f^*(\mathbf{x}_i)$ . Denote  $\mathbf{X} \in \mathbb{R}^{n \times d}$  and  $\mathbf{Y} \in \mathbb{R}^n$  as the training data,  $u(\mathbf{X}) := u(\mathbf{X}, \boldsymbol{\theta}(t)) \in \mathbb{R}^n$ ,  $\nabla_{\boldsymbol{\theta}} u(\mathbf{X}, \boldsymbol{\theta}(t)) \in \mathbb{R}^{n \times M}$  ( $M$  is the number of parameters), let  $K^* = [K^*(\mathbf{x}_i, \mathbf{x}_j)]_{ij} \in \mathbb{R}^{M \times M}$  with a slight abuse of the notation  $K^*$ , then,

$$\frac{du(\mathbf{X})}{dt} = -K^* u(\mathbf{X}). \quad (18)$$

In a continuous form, one can define the empirical density  $\rho(\mathbf{x}) = \sum_{i=1}^n \delta(\mathbf{x} - \mathbf{x}_i)/n$  and further denote  $u_\rho(\mathbf{x}) = u(\mathbf{x})\rho(\mathbf{x})$ . Therefore, the dynamics for  $u$  becomes

$$\frac{d}{dt} u(\mathbf{x}, t) = - \int_{\mathbb{R}^d} K^*(\mathbf{x}, \mathbf{x}') u_\rho(\mathbf{x}', t) d\mathbf{x}'. \quad (19)$$

This continuous form gives the integral equation analyzed in [77]. The convergence analysis of the dynamics in Eq. (18) can be done by performing eigen decomposition of  $K^*$ .

For the two-layer neural network (14), we assume that  $b \sim \mathcal{N}(0, \sigma_b^2)$  with  $\sigma_b \gg 1$ . As a special case, we have the exact LFP dynamics for the cases where the activation function is ReLU.

**Corollary 1 (Luo et al., (2022) [65]: LFP operator for ReLU).** *Under mild assumptions and if  $\sigma_b \gg 1$  and  $\sigma = \text{ReLU}$ , then the dynamics (19) satisfies the following expression,*

$$\langle \partial_t \mathcal{F}[u], \phi \rangle = - \langle \mathcal{L}[\mathcal{F}[u_\rho]], \phi \rangle + O(\sigma_b^{-3}), \quad (20)$$

where  $\phi \in \mathcal{S}(\mathbb{R}^d)$  is a test function and the LFP operator reads as

$$\begin{aligned} \mathcal{L}[\mathcal{F}[u_\rho]] &= \frac{\Gamma(d/2)}{2\sqrt{2}\pi^{(d+1)/2}\sigma_b} \mathbb{E}_{a,r} \left[ \frac{r^3}{16\pi^4 \|\boldsymbol{\xi}\|^{d+3}} + \frac{a^2 r}{4\pi^2 \|\boldsymbol{\xi}\|^{d+1}} \right] \mathcal{F}[u_\rho](\boldsymbol{\xi}) \\ &\quad - \frac{\Gamma(d/2)}{2\sqrt{2}\pi^{(d+1)/2}\sigma_b} \nabla \cdot \left( \mathbb{E}_{a,r} \left[ \frac{a^2 r}{4\pi^2 \|\boldsymbol{\xi}\|^{d+1}} \right] \nabla \mathcal{F}[u_\rho](\boldsymbol{\xi}) \right), \end{aligned} \quad (21)$$

where the expectations are taken w.r.t. initial parameter distribution,  $r = \|\mathbf{w}\|$ ,  $\mathcal{F}[\cdot]$  indicates Fourier transform.

These results explicitly estimate the convergence of each frequency, which confirms the observed spectral bias.

#### 4. Frequency shifting approach

To address the spectral bias of DNN, one approach is to use frequency domain manipulation. This strategy involves converting the higher frequency components of the data to lower frequencies prior to training, speeding up the learning process, and subsequently transforming the learned representations back to their original high-frequency range.

An example of this approach is the phase shift DNN (PhaseDNN) [79]. The PhaseDNN was developed to tackle the challenges of high-frequency learning for regressions and solving PDEs such as Helmholtz equations. It transforms high-frequency data components into a lower-frequency spectrum by using phase shifts in the frequency domain. This enables the network to learn from these modified data quickly and, upon completion, map them back to the original high frequencies by reverse phase shifts. This method has been shown to be effective in achieving an uniform convergence across a broad frequency spectrum for wave propagation problems. A typical PhaseDNN architecture is shown in Fig. 2, which comprises a series of smaller sub-DNNs, each tasked with approximating a specific range of high-frequency content. With the help of phase shifts in the frequency domain, these sub-DNNs are capable of learning high-frequency information at a convergence rate typical of low-frequency problems, thus improving the overall performance and accuracy of the network in handling wideband high-frequency data.

To illustrate how the PhaseDNN arises, let us consider an one-dimensional problem for a band-limited function  $f(x)$  within a frequency range  $[-K_0, K_0]$ , i.e.

$$\text{Supp}\hat{f}(k) \subset [-K_0, K_0] = [-M\Delta k, M\Delta k], \Delta k = K_0/M.$$

We first partition  $[-K_0, K_0]$  with uniform subintervals,

$$[-K_0, K_0] \subset \bigcup_{j=-M}^M A_j, A_j = [\omega_j - \frac{\Delta k}{2}, \omega_j + \frac{\Delta k}{2}], \omega_j = j\Delta k, j = -M, \dots, M. \quad (22)$$

We can decompose the target function  $f(x)$  in the Fourier space as follows,

$$\hat{f}(k) = \sum_{j=-M}^M \chi_{A_j}(k) \hat{f}(k) \triangleq \sum_{j=-M}^M \hat{f}_j(k), \quad (23)$$

where  $\chi_{A_j}$  is the indicator function for the interval  $A_j$ , which will give a

corresponding decomposition in  $x$ -space as

$$f(x) = \sum_{j=-M}^M f_j(x), \quad (24)$$

where

$$f_j(x) = \mathcal{F}^{-1}[\hat{f}_j](x). \quad (25)$$

The decomposition (24) involves  $2M+1$  functions  $f_j(x)$ , whose frequency spectrum is limited to  $[\omega_j - \frac{\Delta k}{2}, \omega_j + \frac{\Delta k}{2}]$ . Therefore, a simple phase shift by  $\omega_j$  could translate its spectrum to  $[-\Delta k/2, \Delta k/2]$ , allowing it to be learned quickly by a relatively small DNN  $T_j(x)$  with a few training epoches. Namely,  $\hat{f}_j(k + \omega_j)$  is supported in  $[-\Delta k/2, \Delta k/2]$ , then its inverse Fourier transform  $\mathcal{F}^{-1}[\hat{f}_j(k + \omega_j)]$ , denoted as

$$f_j^{\text{shift}}(x) = \mathcal{F}^{-1}[\hat{f}_j(k + \omega_j)](x) \quad (26)$$

can be learned quickly by a small DNN  $T_j(x, \boldsymbol{\theta}_j)$  by minimizing a loss function

$$L_j(\boldsymbol{\theta}_j) = \int_{-\infty}^{\infty} |f_j^{\text{shift}}(x) - T_j(x, \boldsymbol{\theta}_j)|^2 dx \quad (27)$$

in some  $n_0$ -epoches of training.

Meanwhile, in the physical domain, we have

$$f_j^{\text{shift}}(x) = e^{-i\omega_j x} f_j(x). \quad (28)$$

Equation (28) shows that once  $f_j^{\text{shift}}(x)$  is learned,  $f_j(x)$  is also learned by simply removing the phase factor, i.e.,

$$f_j(x) \approx e^{i\omega_j x} T_j(x, \boldsymbol{\theta}^{(n_0)}). \quad (29)$$

Now with all  $f_j(x)$  for  $-M \leq j \leq M$  learned after some  $n_0$  steps of training each, we have an approximation to  $f(x)$  over the whole frequency range  $[-M\Delta k, M\Delta k] = [-K_0, K_0]$  as follows

$$f(x) \approx T_{\boldsymbol{\theta}}(x) = \sum_{j=-M}^M e^{i\omega_j x} T_j(x, \boldsymbol{\theta}_j), \quad (30)$$

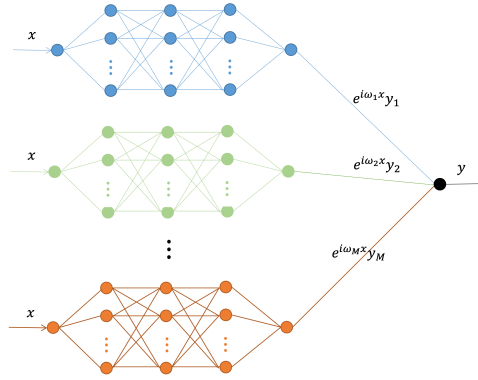


Figure 2: Illustration of a PhaseDNN structure.

where  $\theta = \{\theta_j\}_{j=-M}^M$  is the parameters of the combined DNN, as depicted in Fig. 2.

The discussion above suggests that a PhaseDNN can be sought in the following form (after converting to real functions),

$$T_{\theta}(x) = \sum_{m=0}^M A_m(x) \cos(\omega_m x) + B_m(x) \sin(\omega_m x), \quad (31)$$

where  $A_m, B_m$  are DNNs with their individual network parameters and the shift frequencies  $\omega_m$  in fact can be made as trainable parameters to best fit for the target functions.

As an illustration, we use the PhaseDNN to solve the Helmholtz equation with a constant wave number,

$$u'' + \lambda^2 u = \sin \mu x, \quad (32)$$

coupled with boundary conditions  $u(-1) = u(1) = 0$ . For the PhaseDNN, the frequencies  $\{\omega_m\}$  are selected to be  $0, \lambda,$  and  $\mu$ . Each  $A_m, B_m$  is set to be a 1-40-40-40-40-1 DNN. The result for the case of  $\lambda = 205, \mu = 3$  in Fig. 3 shows (right panel) that a normal fully-connected neural network failed to capture the high-frequency oscillation while the PhaseDNN (left panel) gives an accurate approximation to the high-frequency solution. In practice, the frequency for the shift could be chosen as a hyperparameter for the training.

In [80], a dictionary of  $(1, \cos x, \sin x, \cos 2x, \sin 2x, \dots, \cos kx, \sin kx)$  was used to construct a Prior Dictionary based Physics-Informed Neural Networks (PD-PINNs) to the same effect of the PhaseDNN. Also, to avoid using

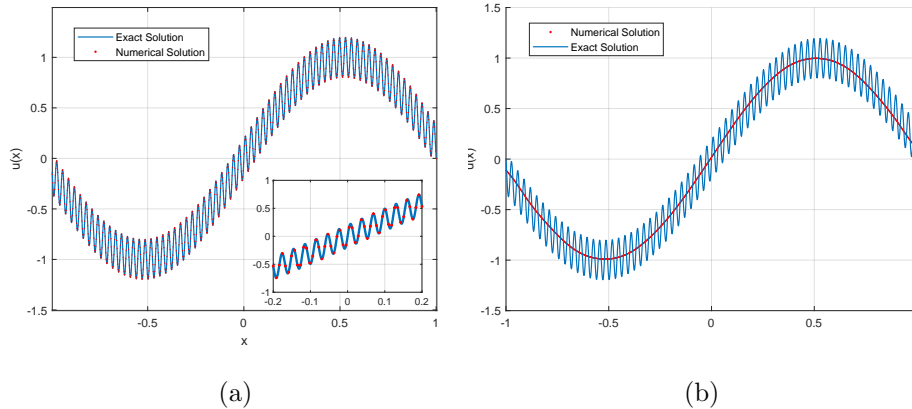


Figure 3: Results demonstrating the efficacy of using PhaseDNN compared to a normal fully-connected DNN. (a) Numerical and exact solution of Helmholtz equation using the PhaseDNN. (b) Failure of a normal fully-connected DNN.

multiple neural networks for  $A_m$ 's and  $B_m$ 's, a network with multiple outputs was used to represent  $A_m$ 's and  $B_m$ 's for the PhaseDNN [81]. Numerical experiments in [79, 80, 81] demonstrate that the PhaseDNN can be very effective in learning low-dimensional high-frequency function. However, as the number of Fourier terms increases with the dimension exponentially, the PhaseDNN suffers from the curse of dimensionality, and will face difficulties for high-dimensional problems.

## 5. Frequency scaling approach

Another approach developed to overcome the spectral bias of DNN is to utilize some type of scaling in the frequency domain so that the higher frequency content will be converted to a lower frequency content before training. One method employing this approach is the multi-scale DNN (MscaleDNN) [12, 82] using radial scaling in the frequency domain (thus equivalently in physical domain). Another related approach is the Fourier feature network [83]. The activation function optimization method and the random feature method [84] have also employed physical scaling to achieve multiscale effects.

### 5.1. Multiscale DNN (MscaleDNN)

The phase shift DNN (PhaseDNN) previously discussed will suffer from the curse of dimensionality for high-dimensional problems due to the amount of phase shifts required along all coordinate directions. As another approach of frequency manipulation, the multiscale deep neural network (MscaleDNN) proposed in [85, 12, 82] utilizes a radial scaling in the frequency domain, instead, to transform the frequency characteristics of the data, as well as activation functions with localized frequency content such as functions with compact support or sin and cos functions. The radial scaling converts the problem of approximating high-frequency contents of target functions or PDEs' solutions to a problem of learning about lower frequency functions, and compact support activation functions or simple sine or cosine functions facilitate the separation of frequency contents of the target function to be approximated by individual DNNs.

#### 5.1.1. The idea of Multiscale DNN

To illustrate how the MscaleDNN arises, again we consider a band-limited function  $f(\mathbf{x})$ ,  $\mathbf{x} \in \mathbb{R}^d$ , whose Fourier transform  $\widehat{f}(\mathbf{k})$  has a compact support, i.e.,

$$\text{Supp}\widehat{f}(\mathbf{k}) \subset B(K_{\max}) = \{\mathbf{k} \in \mathbb{R}^d, |\mathbf{k}| \leq K_{\max}\}. \quad (33)$$

We now partition the domain  $B(K_{\max})$  as union of  $M$  concentric annulus with uniform or non-uniform width, e.g., for the case of uniform  $K_0$ -width

$$A_i = \{\mathbf{k} \in \mathbb{R}^d, (i-1)K_0 \leq |\mathbf{k}| \leq iK_0\}, \quad K_0 = K_{\max}/M, \quad 1 \leq i \leq M \quad (34)$$

so that

$$B(K_{\max}) = \bigcup_{i=1}^M A_i. \quad (35)$$

Now, we can decompose the function  $\widehat{f}(\mathbf{k})$  as before

$$\widehat{f}(\mathbf{k}) = \sum_{i=1}^M \chi_{A_i}(\mathbf{k}) \widehat{f}(\mathbf{k}) \triangleq \sum_{i=1}^M \widehat{f}_i(\mathbf{k}), \quad (36)$$

where  $\chi_{A_i}$  is the indicator function of the set  $A_i$  and

$$\text{Supp}\widehat{f}_i(\mathbf{k}) \subset A_i. \quad (37)$$



The decomposition in the Fourier space give a corresponding one in the physical space

$$f(\mathbf{x}) = \sum_{i=1}^M f_i(\mathbf{x}), \quad (38)$$

where

$$f_i(\mathbf{x}) = \mathcal{F}^{-1}[\widehat{f}_i(\mathbf{k})](\mathbf{x}). \quad (39)$$

From (37), we can apply a simple down-scaling to convert the high frequency region  $A_i$  to a low-frequency one. Namely, we define a scaled version of  $\widehat{f}_i(\mathbf{k})$  as

$$\widehat{f}_i^{(\text{scale})}(\mathbf{k}) = \widehat{f}_i(\alpha_i \mathbf{k}), \quad \alpha_i > 1, \quad (40)$$

and, correspondingly in the physical space

$$f_i^{(\text{scale})}(\mathbf{x}) = \frac{1}{\alpha_i^d} f_i\left(\frac{1}{\alpha_i} \mathbf{x}\right), \quad (41)$$

or

$$f_i(\mathbf{x}) = \alpha_i^d f_i^{(\text{scale})}(\alpha_i \mathbf{x}). \quad (42)$$

We can see the low-frequency spectrum of the scaled function  $\widehat{f}_i^{(\text{scale})}(\mathbf{k})$  if  $\alpha_i$  is chosen large enough, i.e.,

$$\text{Supp} \widehat{f}_i^{(\text{scale})}(\mathbf{k}) \subset \left\{ \mathbf{k} \in \mathbb{R}^d, \frac{(i-1)K_0}{\alpha_i} \leq |\mathbf{k}| \leq \frac{iK_0}{\alpha_i} \right\}. \quad (43)$$

Based on the low-frequency bias of DNNs, with  $iK_0/\alpha_i$  being small, we can train a DNN  $f_{\boldsymbol{\theta}^{n_i}}(\mathbf{x})$ , with  $\boldsymbol{\theta}^{n_i}$  denoting the DNN parameters, to learn  $f_i^{(\text{scale})}(\mathbf{x})$  quickly

$$f_i^{(\text{scale})}(\mathbf{x}) \sim f_{\boldsymbol{\theta}^{n_i}}(\mathbf{x}), \quad (44)$$

giving an approximation to  $f_i(\mathbf{x})$  immediately

$$f_i(\mathbf{x}) \sim \alpha_i^d f_{\boldsymbol{\theta}^{n_i}}(\alpha_i \mathbf{x}) \quad (45)$$

and to  $f(\mathbf{x})$  as well

$$f(\mathbf{x}) \sim \sum_{i=1}^M \alpha_i^d f_{\boldsymbol{\theta}^{n_i}}(\alpha_i \mathbf{x}), \quad (46)$$

which suggests the format of a multiscale DNNs (MscaleDNN) [12, 82]. And the scale factors  $\alpha_i$  can be made as trainable parameters to best fit for the

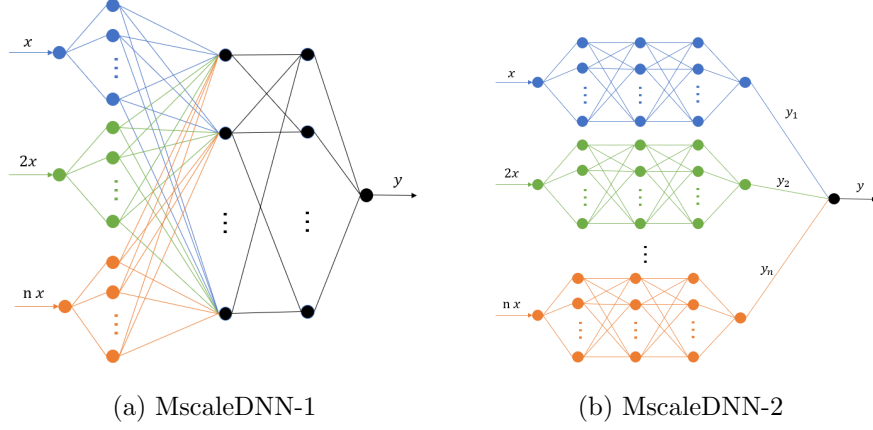


Figure 4: Illustration of two MscaleDNN structures [12].

target functions. Two specific implementations of the MscaleDNNs are given below.

**MscaleDNN-1** For the first kind, the neuron in the first hidden-layer is grouped into to  $N$  parts. The neuron in the  $i$ -th part receives input  $a_i x$ , that is, its output is  $\sigma(a_i \mathbf{W} \cdot \mathbf{x} + \mathbf{b})$ , where  $\mathbf{W}$ ,  $\mathbf{x}$ ,  $\mathbf{b}$  are weight, input, and bias parameters, respectively. A MscaleDNN takes the following form

$$f_{\theta}(\mathbf{x}) = \mathbf{W}^{[L-1]} \sigma \circ (\dots (\mathbf{W}^{[1]} \sigma \circ (\mathbf{K} \odot (\mathbf{W}^{[0]} \mathbf{x}) + \mathbf{b}^{[0]}) + \mathbf{b}^{[1]}) \dots) + \mathbf{b}^{[L-1]}, \quad (47)$$

where  $\mathbf{x} \in \mathbb{R}^d$ ,  $\mathbf{W}^{[l]} \in \mathbb{R}^{m_{l+1} \times m_l}$ ,  $m_l$  is the neuron number of  $l$ -th hidden layer,  $m_0 = d$ ,  $\mathbf{b}^{[l]} \in \mathbb{R}^{m_{l+1}}$ ,  $\sigma$  is a scalar function and “ $\circ$ ” means entry-wise operation,  $\odot$  is the Hadamard product and

$$\mathbf{K} = \left( \underbrace{a_1, a_1, \dots, a_1}_{\text{1st part}}, \underbrace{a_2, \dots, a_{i-1}, a_i, a_i, \dots, a_i}_{\text{ith part}}, \dots, \underbrace{a_N, a_N, \dots, a_N}_{\text{Nth part}} \right)^T, \quad (48)$$

where  $\mathbf{K} \in \mathbb{R}^{m_1}$ , and the scale factor  $a_i = i$  or  $a_i = 2^{i-1}$ . This structure is called Multi-scale DNN-1 (MscaleDNN-1), as depicted in Fig. 4(a).

**MscaleDNN-2** A second kind of multi-scale DNN is given in Fig. 4(b), as a sum of  $N$  subnetworks, in which each scaled input goes through a separate subnetwork. In MscaleDNN-2, weight matrices from  $\mathbf{W}^{[1]}$  to  $\mathbf{W}^{[L-1]}$  are block diagonal. Again, the scale factor  $a_i = i$  or  $a_i = 2^{i-1}$ .

Liu et al [12] has shown that the MscaleDNN can work well for high frequency fitting and PDE solving with a very complex geometry and the

numerical experiments also found that the activation function in the first hidden layer is important for the performance of MscaleDNNs, for example,  $\text{sReLU}(x) = (x)_+(1-x)_+$  and  $\phi(x) = (x-0)_+^2 - 3(x-1)_+^2 + 3(x-2)_+^2 - (x-3)_+^2$  were used, where  $x_+ = \max\{x, 0\} = \text{ReLU}(x)$ . MscaleDNN with various activation functions is also explored, such as the one with sinusoidal activation functions used for PDEs with a point source [86], the one with residual connection and activation  $\sigma(x) = 0.1 * (\sin x + \cos x)$  [87], and the one with the ellipse Gaussian RBFs as activation function [88]. In [89], Li et al. introduced an orthogonality condition between low-frequency and high-frequency output subspaces through a penalty term in the loss function, which is designed to reduce the influences between low and high frequencies in the network, and achieved better error in solving PDEs.

To show the effectiveness of MscaleDNN, we solve a 2-D Poisson equation with an oscillatory solution in  $\Omega = [-1, 1]^2$ ,

$$-\Delta u(x, y) = f(x, y), \quad (49)$$

where the exact solution is  $u(x, y) = \frac{1}{N^2} \sum_{m=1}^N \sum_{n=1}^N e^{\sin(\pi mx)} e^{\cos(\pi ny)}$ . We choose  $N = 20$  and use the variational loss (6) for the training. We compare a MscaleDNN with a vanilla fully-connected DNN (FC-DNN) for PDE solutions with multiple frequencies. The following different network structures are used: (1) a FC-DNN with a size 2-3200-3200-3200-1; (2) a MscaleDNN with scales  $\{1, 2, 4, 8, 16, 32, 64, 128\}$  for eight subnetworks with a size 2-400-400-400-1 each. The results in Fig.5 clearly show the capability of the MscaleDNN in capturing the high-frequency content of the solution while the FC-DNN failed.

MscaleDNNs have been used for solving various scientific problems. Wang et al. [57] apply the MscaleDNN for oscillatory stokes flows in complex domains. Ying et al. [90] use the MscaleDNN-2 structure to construct multi-scale fusion neural network (MSFN) for solving elliptic interface problems. Li et al. [91] utilize MscaleDNN to construct the neural sparse representation to solve Boltzmann equation with BGK and quadratic collision model. To solve the stationary nonlinear Navier-Stokes equation, Liu et al. [92] integrate linearizations of the nonlinear convection term in the Navier-Stokes equation into the training process of MscaleDNN approximations of highly oscillatory Navier-Stokes solutions. Chen et al. [93] applied the MscaleDNN successfully to address the spectral bias in 3-D turbulent wind field reconstruction by leveraging multi-scale PINN, which enhanced the ability to learn

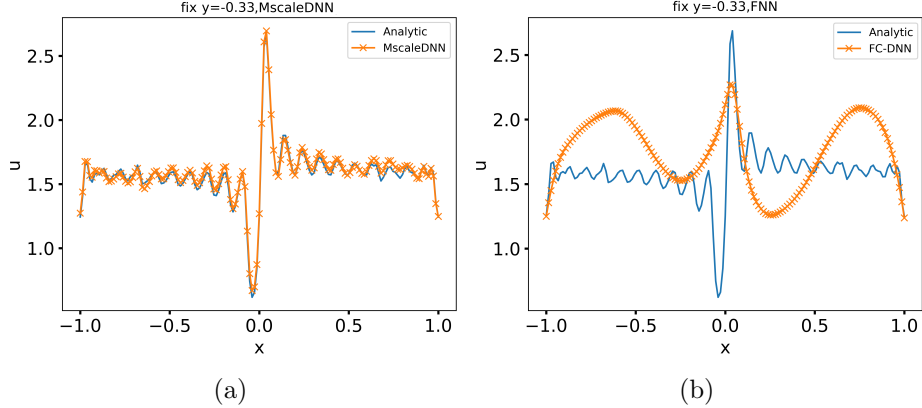


Figure 5: Comparison of a MscaleDNN and a vanilla DNN for an oscillatory solution of the Poisson equation at  $y = -0.33$ . (a) Numerical and exact solution of Poisson equation using MscaleDNN. (b) Failure of a vanilla fully-connected DNN (FC-DNN).

from both low and high-frequency data, thus improving the accuracy of flow field predictions. Also, recently, in [94], the MscaleDNN has been shown to be critical in addressing numerical error due to the failure of normal neural network in capturing high-frequency multiple-scattering field by arrays of random dielectric nanocylinders in a study of isotropic SHU (Stealthy Hyperuniform) optical materials, enabling the inverse retrieval of their effective dielectric profiles in a numerical homogenization. The multiscale network architecture proved to be superior over single-scale PINNs, particularly in scenarios with significant multiple scattering effects and high wave numbers. These results highlight the potential applications of the MscaleDNN in real physical systems.

### 5.1.2. Spectral bias reduction property of the MscaleDNN, and selection of scales.

Based on the training dynamics with the NTK framework (12), Wang et al. [95] derived a diffusion equation model for the error evolution in the frequency domain for a learning algorithm by a multiscale neural network in approximating oscillatory functions, assuming a vanishing learning rate and an infinite wide network in a two-layered network with an activation function  $\sigma(x) = \sin x$ .

The diffusion model is

$$\partial_t \hat{u}^\pm(\boldsymbol{\xi}, t) = \nabla_{\boldsymbol{\xi}} \cdot \left[ A_s^\mp(\boldsymbol{\xi}) \nabla_{\boldsymbol{\xi}} \hat{u}^\pm(\boldsymbol{\xi}, t) \right] - B_s^\pm(\boldsymbol{\xi}) \hat{u}^\pm(\boldsymbol{\xi}, t), \quad \boldsymbol{\xi} \in \mathbb{R}^d, \quad (50)$$

where  $\hat{u}^\pm(\boldsymbol{\xi}, \boldsymbol{\theta}(t))$  are the real and imaginary parts of  $\hat{u}(\boldsymbol{\xi}, \boldsymbol{\theta}(t))$ , respectively, i.e.,  $\hat{u}(\boldsymbol{\xi}, \boldsymbol{\theta}(t)) = \hat{u}^+(\boldsymbol{\xi}, \boldsymbol{\theta}(t)) + i\hat{u}^-(\boldsymbol{\xi}, \boldsymbol{\theta}(t))$ ,  $u(\mathbf{x}, \boldsymbol{\theta})$  is a zero extension of the error of the network defined by

$$u(\mathbf{x}, \boldsymbol{\theta}) = \begin{cases} 0, & \mathbf{x} \notin \Omega, \\ f(\mathbf{x}, \boldsymbol{\theta}) - f(\mathbf{x}), & \mathbf{x} \in \Omega, \end{cases}$$

$f(\mathbf{x}, \boldsymbol{\theta})$  is the multi-scale neural network [82] with one hidden layer, and  $A_s^\pm(\boldsymbol{\xi}) = \frac{1 \pm e^{-2}}{8\pi^2(s+1)} \sum_{j=0}^s \alpha_j^{2(d+1)} \widehat{\mathcal{G}}_j(\boldsymbol{\xi})$ ,  $B_s^\pm(\boldsymbol{\xi}) = \frac{1 \pm e^{-2}}{2(s+1)} \sum_{j=0}^s \alpha_j^{2d} \widehat{\mathcal{G}}_j(\boldsymbol{\xi})$ ,  $\mathcal{G}_p(\mathbf{x}) = e^{-4^p |\mathbf{x}|^2/2}$ ,  $\mathbf{x} \in \mathbb{R}^d$  is a scaled Gaussian function.

This diffusion model matches well with the error decay of practical training of MscaleDNN in approximating oscillatory functions [95].

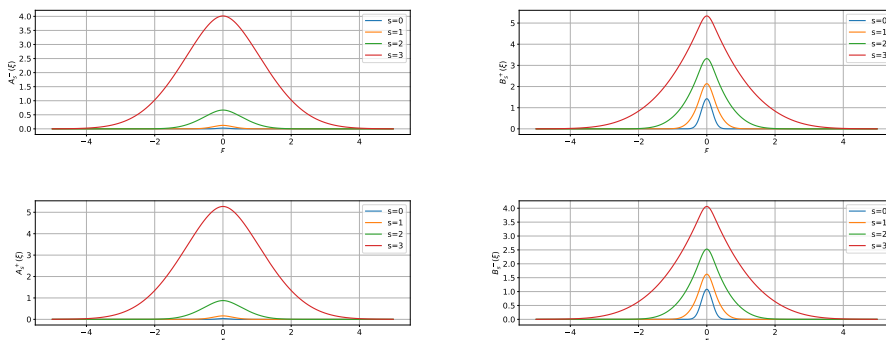


Figure 6: Diffusion coefficients  $A_s^\mp(\boldsymbol{\xi})$  (left) and  $B_s^\pm(\boldsymbol{\xi})$  (right) with  $\alpha_j = 2^j$ ,  $s = 0, 1, 2, 3$ .

A sample of  $\{A_s^\mp(\boldsymbol{\xi}), B_s^\pm(\boldsymbol{\xi})\}$  for a 1-dimensional case are plotted in Fig. 6 for up to 4 scales two-layer MscaleDNNs. We can see that both  $A_s^-(\boldsymbol{\xi})$  and  $B_s^+(\boldsymbol{\xi})$  have larger support in frequencies with increasing of  $s$ , thus providing the mathematical explanation for the capability of the MscaleDNNs in reducing spectral bias as shown by extensive numerical tests. Also for  $s = 0$ , this result confirms the spectral bias of the two-layered fully-connected neural network with a sine-activation function.

The number and specific values of the scales used in the MscaleDNN should depend on the target solution to be learned, therefore, in principle, they could be made as hyper-parameters of the MscaleDNN. Indeed, in a recent work [96], a frequency (i.e., scale) adaptive PINN was investigated, which shows that dynamically selected scales could improve much the DNN solution for PDEs with oscillatory solutions.

### 5.2. Fourier feature networks

The first hidden layer in a neural network can be viewed as a mapping that lifts the input to another space, often of a higher dimension than that of the input space. In the MscaleDNN framework, the first layer lifts the input in  $d$  dimensions to  $m_1$  dimensions. Another lifting operation can be done by setting all trainable weights in the first hidden layer to fixed values. For example, as in [83], the Fourier Feature Networks set all the weights to fixed values and biases to zero with a sinusoidal activation function, i.e.,

$$\gamma(v) = [a_1 \cos(2\pi b_1^T v), a_1 \sin(2\pi b_1^T v), \dots, a_m \cos(2\pi b_m^T v), a_m \sin(2\pi b_m^T v)]^T, \quad (51)$$

where  $v$  is the input, and  $a_k$ 's and  $b_k$ 's are fixed non-trainable parameters. In particular, in the positional encoding of Transformer [97]  $a_k = 1$  for all  $k$  and  $b_k = c^{k/m}$  with a scale  $c$ , similar to the scales  $\alpha_i$  in the MscaleDNN (46). In other cases,  $b_k$  can be sampled from a Gaussian distribution. In [98], a multiscale input structure  $\gamma(v)$  (51) was used with  $a_k = 1, b_k = 2^k$  in representing scenes as neural radiance fields for view synthesis, achieving a great success in computer vision. The resulting network is exactly the MscaleDNN with activation function  $\sigma(x) = \sin x$ , thus the analysis in Section 4.2.1 also applies to the Fourier feature network.

An extension of the Fourier feature networks was given in [99] to solve PDEs by setting the coefficients in (51)  $a_k = 1$  and sampling each  $b_k$  from Gaussian distribution  $N(0, \sigma_k^2)$  with variance  $\sigma_k$ . Another transform was used in multi Fourier feature networks (MFFNN) [100] by setting  $\gamma(\mathbf{x}) = [\cos(2\pi \mathbf{B}\mathbf{x}), \mathbf{x}, \sin(2\pi \mathbf{B}\mathbf{x})]^T$  with entries of  $\mathbf{B} \in R^{m \times d}$  ( $m$  is a given integer) sampled from an isotropic Gaussian distribution  $N(0, \sigma^2)$  and the standard deviation  $\sigma$  treated as a hyper-parameter. Combination of different choices of  $\gamma(v)$  has been used in [101] by using  $\gamma(\mathbf{x}) = [\cos(2\pi \mathbf{B}\mathbf{x}), \alpha * \mathbf{x}, \sin(2\pi \mathbf{B}\mathbf{x})]^T$  for the simulation of MHD problems.

### 5.3. Activation function optimization

The activation functions play an essential part in the learning process of artificial neural networks. There are many choices of activation functions in deep learning, such as  $\text{ReLU}(x)$ ,  $\tanh(x)$ ,  $\text{GeLU}(x)$  [102], rational activation functions [103], and sigmoid type activation functions etc. Empirical works also find that using B-spline [104] or sinusoidal activation function with proper initialization [105] can also mitigate the spectral bias. A survey of activation functions in multi-scale neural networks is referred to Jagtap et al. [106]. To improve the effectiveness and accuracy of deep neural networks (DNNs) in approximating functions with multiple frequencies and to mitigate the impact of spectral bias, one of the strategies involves optimizing the choice of activation functions, pursued in the following works.

#### 5.3.1. Adaptive activation function

Jagtap et al. [107] introduced an adaptive activation function in the form  $\sigma(na\mathbf{x})$  with an integer  $n$  and a trainable hyper-parameter  $a$  in the NN. With a large  $n$ , this effectively increases the sampling range of the weights and the bias in the network structure while the trainable  $a$  introduced another degree of freedom for the training of the network parameters. The resulting adaptive activation function NN will take the following form

$$f_{\theta}(\mathbf{x}) = \mathbf{W}^{[L]} \sigma \circ (na(\cdots (\mathbf{W}^{[1]} \sigma \circ (na(\mathbf{W}^{[0]}(\mathbf{x}) + \mathbf{b}^{[0]})) + \mathbf{b}^{[1]}) \cdots)) + \mathbf{b}^{[L]}. \quad (52)$$

To some extent, this factor  $na$  plays a similar role of the scales in the MscaleDNN, so some reduction of spectral bias is expected and numerical results in [107] validate this approach for both supervised learning and PDE solving up to a certain high frequency, such as  $\sin(12x)$ .

#### 5.3.2. Reproducing activation functions

Another way to improve the performance of the DNN is to enrich the activation function choices using a collection of activation function form so that traditional objects in approximation theory, such as polynomials, Fourier basis, wavelet basis can be reproduced by using enriched activation functions. Liang et al. [108] proposed such an approach with reproducing activation functions (RAFs), which employ several basic activation functions and their learnable linear combinations to construct activation functions for different neurons. For the  $i$ -th neuron of the  $l$ -th layer, the formula of the neuron-wise

RAF is given by

$$\sigma_{i,j}(x) = \sum_{p=1}^P \alpha_{p,i,l} \gamma_p(\beta_{p,i,l} x), \quad (53)$$

where  $A = \{\gamma_1(x), \dots, \gamma_P(x)\}$  is a set of  $P$  basic activation functions,  $\alpha_{p,i,l}$  and  $\beta_{p,i,l}$  are learnable combination coefficient and scaling parameter, respectively. Numerical results have shown better performance with smaller number of parameters for audio and image restoration, PDEs and eigenvalue problems, such as solving a PDE with a frequency of  $\sin(2\pi x_1) \sin(2\pi x_2)$ , compared with traditional NNs. In particular, for solutions with both low and high frequencies, Liang et al. [108] used the poly-sine-Gaussian network with  $A = \{x, x^2, \sin(\beta x), e^{-(\beta x)^2}\}$ , in which  $\beta$  is a trainable hyper-parameter.

### 5.3.3. Multiple activation neural network

Compared with the enrichment of activation function at the neuron level by Liang et al. [108], Jagtap et al. [109] proposed an enrichment of the network using a combination of a collection of activation functions  $\{\phi_k(\mathbf{x})\}$  for each neuron. A two layered of the so-called Kronecker neural networks (KNNs) takes the following form

$$f(\mathbf{x}, \theta) = \sum_{i=1}^m \left[ \sum_{k=1}^K \alpha_k \phi_k(\omega_k(w_i x + b_i)) \right], \quad (54)$$

where each of the activation function  $\phi_k(\mathbf{x})$  shares the same weights and bias and the network parameters are  $\theta = \{c_i, w_i, b_i\}_{i=1}^m \cup \{\alpha_k, \omega_k\}_{k=1}^K$ .

A particular choice of the activation function is to set  $\phi_1$  to be any standard activation function such as ReLU, tanh, ELU, sine, Swish, Softplus etc., and the remaining activation functions are  $\phi_k(\mathbf{x}) = n \sin((k-1)nx)$  or  $n \cos((k-1)nx)$ ,  $2 \leq k \leq K$  resulting in a Rowdy-Net [109]. This choice is similar to the MscaleDNN of (46) in the first hidden layer with a sinusoidal activation function  $\sigma(x) = \sin x$  and scaling factor  $\alpha_k = k$ .

Jagtap et al. [109] also provided a theoretical analysis which reveals that under suitable conditions, the Kronecker network induces a faster decay of the loss than that by the feed-forward network at the beginning of the training.

### 5.4. Random Feature Method

Chen et al. [84] proposed a random feature method (RFM) for solving PDEs, a natural bridge between traditional and machine learning-based



algorithms. It adds several additional components including multi-scale representation and rescaling the weights in the loss function.

Following the random feature model in machine learning, one can construct an approximate solution  $u_M$  of  $u$  by a linear combination of  $M$  network basis functions  $\{\phi_m\}$  over  $\Omega$  as follows

$$u_M(\mathbf{x}) = \sum_{m=1}^M u_m \phi_m(\mathbf{x}). \quad (55)$$

Generally speaking, the basis functions will be chosen as the ones that occur naturally in neural networks, for example:

$$\phi_m(\mathbf{x}) = \sigma(\mathbf{k}_m \cdot \mathbf{x} + b_m), \quad (56)$$

where  $\sigma$  is some scalar nonlinear function,  $\mathbf{k}_m, b_m$  are some randomly selected but fixed parameters.

*Partition of unity and local random feature models.* Random feature functions are globally defined, while the solution of a PDE typically has local variations, possibly at small scales. To accommodate this, RFM constructs many local solutions, each of which corresponds to a random feature model, and piece them together using partition of unity (PoU). The PoU starts with a set of points  $\{\mathbf{x}_n\}_{n=1}^{M_p} \subset \Omega$ , each of which serves as the center for a component in the partition. For each  $n$ , a normalized coordinate is defined

$$\tilde{\mathbf{x}}_n = \frac{1}{r_n}(\mathbf{x} - \mathbf{x}_n), \quad n = 1, \dots, M_p, \quad (57)$$

where  $\{r_n\}$  is pre-selected, and, similar to the scaling in MscaledDNN, the coefficient  $r_n$  in Eq. (57) plays an important role in learning a wider frequency range. Next, for each  $n$ ,  $J_n$  random feature functions are constructed

$$\phi_{n,j}(\mathbf{x}) = \sigma(\mathbf{k}_{n,j} \cdot \tilde{\mathbf{x}}_n + b_{n,j}), \quad j = 1, \dots, J_n, \quad (58)$$

where the feature vectors  $\{(\mathbf{k}_{n,j}, b_{n,j})\}$  often chosen randomly and then fixed. In this way the locally space-dependent information is incorporated into  $M = \sum_{n=1}^{M_p} J_n$  random feature functions. As for the construction of the PoU, one-dimensional PoU can be defined as

$$\psi_n(x) = \mathbb{I}_{-1 \leq \tilde{x}_n \leq 1} \quad (59)$$

or

$$\psi_n(x) = \begin{cases} \frac{1+\sin(2\pi\tilde{x})}{2}, & -\frac{5}{4} \leq \tilde{x}_n < -\frac{3}{4} \\ 1, & -\frac{3}{4} \leq \tilde{x}_n < \frac{3}{4} \\ \frac{1-\sin(2\pi\tilde{x})}{2}, & \frac{3}{4} \leq \tilde{x}_n \\ 0, & \text{otherwise.} \end{cases}$$

where the relation between  $x$  and  $\tilde{x}_n$  is given by Eq. (57). High-dimensional PoU can be constructed using the tensor product of one-dimensional PoU functions, i.e.,  $\psi_n(\mathbf{x}) = \prod_{k=1}^d \psi_n(x_k)$ , where  $\mathbf{x} = (x_1, x_2, \dots, x_d)$  and  $x_k \in \mathbb{R}$  for each  $k = 1, 2, \dots, d$ .

The RFM approximate solution  $u_M$  is given by

$$u_M(\mathbf{x}) = \sum_{n=1}^{M_p} \psi_n(\mathbf{x}) \sum_{j=1}^{J_n} u_{nj} \phi_{nj}(\mathbf{x}), \quad (60)$$

which resulted in a over-determined system to be solved by some QR-based least square solvers.

*Multi-scale basis.* In some situations, (60) alone is less efficient in capturing the large scale features in the solution. Therefore, on top of the PoU-based local basis functions, one can add another global component:

$$u_M(\mathbf{x}) = \sum_{m=1}^M u_m \phi_m(\mathbf{x}) + \sum_{n=1}^{M_p} \psi_n(\mathbf{x}) \sum_{j=1}^{J_n} u_{nj} \phi_{nj}(\mathbf{x}). \quad (61)$$

This RFM is shown to be able to solve a low-dimensional PDE problem with wide frequency range with an almost machine accuracy even in a complex geometry region [84]. Since the RFM utilizes the random feature basis, it is a mesh-less method, therefore, it can handles problems with complex regions. The machine accuracy can be achieved because the problem is not solved by gradient based methods but solved by a matrix inversion method with QR decompositions.

## 6. Hybrid approach

Classic iterative solvers excel at reducing high frequency errors while the normal DNNs are effective in lower frequency ones, therefore, it is natural to combine them to arrive at a solver with uniformly fast error reductions across all frequencies, giving rise to hybrid methods.

### 6.1. Jacobi-DNN

A natural idea of frequency decomposition is to utilize a hybrid approach of using conventional iterative method such as Jacobi or Gauss-Seidel methods for the high-frequency error reduction and a DNN-based method for the fast convergence of lower frequency.

Xu et al. [39] combined DNNs with conventional iterative methods to accelerate the convergence of both low and high frequencies for computational problems. Take Poisson’s equation as an example. First, the DNN was used to solve the Poisson’s equation with a certain optimization steps (or epochs). Then, the Jacobi method with the initial value obtained from the trained DNN is carried out to eliminate the error in high frequencies using the smoothing properties of the iterative method.

Huang et al. [110] developed a similar approach (Int-Deep), applying a deep-learning-initialized iterative method and demonstrated that the Int-Deep method is effective for low-dimensional PDEs, such as semilinear PDEs, linear and nonlinear eigenvalue problems etc. Chen et al. [111] used a Broad Learning System (BLS), which learns low frequency faster similar to DNNs, and found that low and high frequencies can be learned fast for low-dimensional problems by a BLS-Jacobi method.

However, since such hybrid methods use conventional methods in the second stage, it also suffers similar difficulties as conventional methods. For example, such hybrid methods can only be applied to low-dimensional problems.

### 6.2. HINTS

Zhang et al. [112] proposed a hybrid approach of integrating neural operator DeepONet and standard relaxation methods, yielding a hybrid iterative numerical transferable solver (HINTS). Firstly, a DeepONet needs to be trained offline before employing HINTS to approximate the solution operator of a linear differential equation. Then, HINTS starts by discretizing the linear differential equation and alternately adopts the numerical iterator and the DeepONet iterator with ratio  $1 : (n_r - 1)$  (i.e., DeepONet with proportion  $1/n_r$ ), until the iterative solution converges. Instead of existing traditional solvers which adopt a fixed relaxation method in each iteration, HINTS creates a second branch of the iterator using a trained DeepONet. The numerical iterator may be chosen from traditional solvers, such as the Jacobi method, the Gauss-Seidel method et., which are known to be efficient for high-frequency modes but not for low-frequency modes; the DeepONet

solver provides fast approximate solutions for low frequencies but may contain polluted solutions at the high frequencies. A proper combination of these two methods enhances the convergence and enables fast and uniform convergence across all frequencies. Such a combination of solvers incorporates advantages from both worlds, improving the performance beyond individual solvers: classical solvers either are slow or even fail to convergence; an individual deep learning solver may only provide an approximate solution.

HINTS exploits such a bias to tackle low-frequency modes that are otherwise difficult for classical solvers which, essentially, are biased towards high frequencies. By replacing the classical solver with the DeepONet solver for a limited proportion of iterations, HINTS balances the convergence rates across the spectrum of eigenmodes, significantly alleviating the spectral bias and improving computational efficiency.

### 6.3. *MgNet, Multi-level, Multi-grade Nets*

When discretized, a linear PDE is transformed into a linear system of equations, represented as

$$A_\eta u = f. \tag{62}$$

In the context of solving large-scale linear systems, iterative methods are frequently employed as an alternative to direct matrix inversion. As noted before, Jacobi and Gauss-Seidel iterations exhibit slow convergence for low-frequency components. A classical iterative approach for rapid convergence across all frequency regimes is the multi-grid (MG) method [113]. The MG method comprises two distinct stages: smoothing and coarse grid correction. During the smoothing stage, high-frequency errors are eliminated through the application of smoothers, such as Jacobi or Gauss-Seidel iterations.

Crucially, MG methods leverage a coarse grid solution to eliminate low-frequency errors. As demonstrated in a prior study by He et al. [114], the operations involving prolongations, restrictions, and select smoothers within the MG method framework can all be formulated as convolution operations. Subsequently, through the substitution of these operations with convolutional neural networks with trainable parameters, the Multigrid Network (MgNet) framework for solving PDEs has been developed [115]. MgNet is a DNN-based iterative approach, and its loss function is derived from the least squared error of Eq. (62), with a single MG iteration step to approximate the solution  $u$ .

In MgNet, the solution domain is dissected into grids with both fine and coarse resolutions. The neural network embedded within this framework

serves to establish a connection between solutions respectively spanning low and high-frequency ranges. Consequently, MgNet effectively harnesses the strengths of the traditional MG method of frequency uniform convergence, alongside the approximation capabilities intrinsic to neural networks. However, due to the use of grids, the MgNet will be limited to solving PDEs in low dimensions.

Another multi-level learning by neural network is an hierarchical approach proposed in [116], which reduces the residual of the numerical solution using a series of networks recursively. The first neural network is used to learn the target function with a given error loss, such as the least squared error. Once the training stalls, the parameters of the first network are fixed. Then, a second neural network, which takes the same input as the first network, is optimized to reduce the residual error between the target function and the sum of two networks. Recursively, the  $k$ -th network, which takes the same input as previous  $(k-1)$  networks, is optimized to reduce the residual error between the target function and the sum of all  $k$  networks. Therefore, each additional network acts as an incremental correction to previous networks, similar to the Mg-Net approach. To further improve accuracy, [117] assigns appropriate coefficients for each network.

Multi-grade learning model is an alternative approach to realize a multi-stage learning for different frequencies [118], where the  $k$ -th network takes the output of the  $(k-1)$ -th network as input instead of the original input, and the output of the  $k$ -th network is always trained to fit the target output. The training of the  $k$ -th network is based on the already trained  $(k-1)$ -th network. Such multi-grade learning has been empirically shown to be faster than one-grade learning for multi-frequency problems.

## 7. Multi-scale neural operators and diffusion models

As discussed previously, an alternative approach to solve PDEs is to learn the PDE operator, which represents the relation between the solution and some physical quantities such as the material properties (coefficients in the PDEs), forcing, or initial or boundary conditions. For instance, for wave scattering, operator learning could be used to establish the relation between the scattering field and the conductivity or permittivity of the scatterer or the incident waves or both. Once such an operator is learned, as an immediate application, it could be used to quickly predict the scattering field for other un-seen scatterers or incident waves. More importantly, the learned operator

could be used to solve the challenging problem of inverse medium problems for imaging applications in medical and material sciences and geophysics. On the other hand, if a neural network is used directly to parameterize the solution for the problem, we will need to re-train the neural network for different materials or boundary conditions. In training a network to learn the PDE operator, the network takes a function as input and outputs a function. For the input function, it could be represented by its values on fixed grids. For the output function, there are two common ways, one is to output the function values on a group of fixed grids, the other is that the network also takes a position  $x$  as input in addition to the input function and outputs the target function at  $x$ . Such neural network is commonly referred to as neural operator, such as DeepONet [119], FNO [120] etc. Neural operators such as MgNet and UNet-based DNNs [121, 122, 123] use a multi-scale sampling to efficiently solve functions with multiple frequencies.

It has also been observed that a neural network representation of a PDE operator could also suffer from a similar spectral bias for high frequency functions. For example, consider a functional mapping from  $a(x)$  to  $u(x)$  in the following simple form,

$$u(x) = \mathcal{G}[a(x)](x) := \sin(ma(x)). \quad (63)$$

Even the input function  $a(x) = a$  is a constant, for large  $m$ , the mapping  $\mathcal{G}$  will have a high frequency dependence on  $a$  as the latter varies.

To alleviate the spectral bias, multiscale neural operators have been proposed by using scaled input to a normal neural operator such as multiscale FNO (MscaleFNO) [124] and multiscale DeepOnet [125].

For instance, in [124], the 1-D Helmholtz equation for the scattering field with a Dirichlet boundary condition was considered

$$\begin{cases} u'' + (\lambda^2 + c\omega(x))u = f(x) & x \in [-L, L], \\ u(-L) = u(L) = 0, \end{cases} \quad (64)$$

where  $\lambda$  is the wave number of the homogeneous background material and  $c\omega(x)$  corresponds to the variable wave number perturbation of the scatterer,  $u(x)$  is the scattering field and  $f(x)$  is the source term. We are interested in learning the operator

$$\mathcal{G} : \omega(x) \mapsto u(x), \quad (65)$$

for a fixed source term  $f(x)$  in the form of

$$f(x) = \sum_{k=0}^{10} (\lambda^2 - \mu_k^2) \sin(\mu_k x), \quad \mu_k = 300 + 35k. \quad (66)$$

A MscaleFNO of the following form was used [124] to predict the scattering field from  $\omega(x)$  for a fixed source  $f(x)$  for the case  $L = 11, \lambda = 2, c = 0.9\lambda^2 = 3.6$ ,

$$u(x) = \sum_{i=1}^N \gamma_i \text{FNO}_{\theta_m} [c_i x, c_i \omega(x)](x). \quad (67)$$

Fig. 7 shows the predicted scattering field by a normal FNO (left panel, only an average value of the solution is predicted) and a multiscale FNO (right panel) with  $N = 8$  scales with  $c_i = 1, 80, 160, 200, 240, 280, 360$ , and 400 and similar total number of parameters as the normal FNO. Fig. 8 shows the profile of the input function  $\omega(x)$  (left panel) and the comparison of the convergence history of the FNO and the MscaleFNO after 500 epochs of training (right panel).

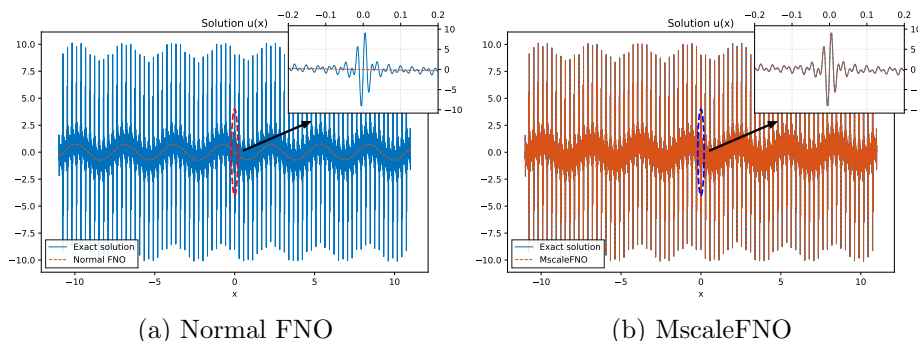


Figure 7: Performance comparison of a Normal FNO and a MscaleFNO in predicting highly oscillatory solution  $u(x)$  of a 1-D Helmholtz equation from its coefficient  $\omega(x)$  (a) Prediction results of the normal FNO compared with the exact solution, with an inset showing details in the region  $[-0.2, 0.2]$  and the failure of FNO to capture the oscillatory behavior of the solution; (b) Prediction results of the MscaleFNO, showing its capability in capturing both global behavior and local details of the solution, as depicted in the zoomed-in insert.

In a recent work [126], an hierarchical attention neural operator was proposed to address the spectral bias issue by leveraging diffusion models in

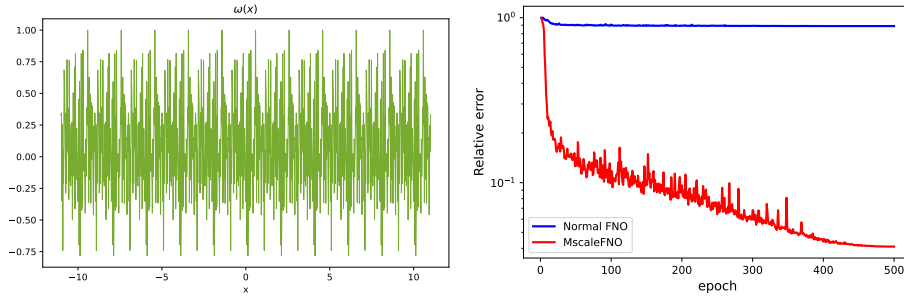


Figure 8: (Left) The input function  $\omega(x)$  with rapid oscillations across the domain; (Right) Convergence comparison of relative errors during 500 training epochs between Normal FNO and MscaleFNO. The MscaleFNO demonstrates superior learning capability with the relative error decreasing to approximately  $4 \times 10^{-2}$ , while Normal FNO remains at a higher error level around  $8 \times 10^{-1}$ .

generative AI. Diffusion models [127] and GANs [128] were developed for various tasks such as image and video generation. In [129], a diffusion model is integrated with various neural operators to improve their spectral representation of turbulent flows. In [130, 129, 131], the authors leveraged generative modeling techniques to learn PDE operator with a wide range of frequencies which can also mitigate the spectral bias typically exhibited by the DNNs.

## 8. Conclusions and open problems

We have provided a survey of the current efforts in developing various methods to overcome the spectral bias of deep neural network in learning PDE solutions with wide frequency contents. This line of research is generating many new results at a fast pace, this survey will surely miss some worth developments. Most of the methods reviewed in this survey in fact can trace their roots in classical approximation theories and traditional numerical methods, such as the multi-resolution idea of wavelets (multiscale DNN, Fourier features, adaptive activation functions), wave-ray MGM [132] (PhaseDNN), domain decomposition (Random feature method), multigrid (Mg-Net), multi-level approaches. The challenge is how to develop effective mathematical tools suitable for the compositional structure of the DNNs to investigate the effects of various constructs of DNN on removing or mitigating the effects of spectral bias, so fast learning is not limited to only lower frequency component of the solution while stalling or even failing at higher frequencies. Thus, it is of great relevance to understanding the dynamics of



the network weights during the training and their relation to the interpolative and expressivity power of the network. The recent results on information bottleneck and condensation of network weights are a good starting point in gaining insights of the mechanism of DNN learning, thus providing researchers new inspirations of developing more sophisticated DNN models as well as analysis tools to best resolve the spectral bias of DNN and to achieve a robust and mature way using DNN as a PDE solver, especially for high dimensional PDE problems or complex geometry problems.

**DNN models and training process** Over-parameterized neural networks have numerous global minima, which may have different generalization performance. The characteristic of the training process of neural network is critical to understand what kind of minima the training will find. For a normal neural network, the spectral bias leads to a minima of low-frequency interpolating capability. There are various phenomena found in the training process, which may give indication if such a spectral bias may occur.

In [133], a phase transition phenomena has been observed in light of information bottleneck theory, which shows two phases of training a neural network, a fitting phase and a diffusion one. In the fitting phase, the training is primarily driven to decrease the loss and the mutual information [134] between the input and output significantly increases. In the diffusion phase, however, noise takes a dominant role in the training and weights evolve chaotically, resembling a diffusion process. This phenomenon is evident in the training of plain PINN [135] and several synthetic examples were also employed in [63] to demonstrate that the fitting process corresponds to the learning of low-frequency content while the diffusion one to the high-frequency content. A future line of work may be to examine various DNN models reviewed in this survey through the information bottleneck theory to see how their phase transitions are possibly related to their effectiveness in overcoming spectral bias.

Another informative phenomenon during the training of the network is the condensation of neurons during the training process, i.e., the weights of hidden neurons in two-layer ReLU neural networks are shown to condense into finite orientations during training in the non-linear regime, particularly with small initialization [74, 136] or with dropout regularization [137, 138]. In the early training stage, neurons in the same layer would condense onto very few directions [139, 140, 141]. Such condensation onto few directions plays a critical role that resets the neural network to a simple state, whose expressivity is small to fit only very low-frequency content. Then, due to

non-zero loss, the network would condense on more and more directions in order to increase the expressivity. Such condensation with more and more directions can ensure the network to fit the target data points but with as low complexity as possible. How new DNN models would affect the condensation process during the training is also an interesting topic for understanding how they alleviate the spectral bias or designing more efficient structure to accelerate the training.

**Frequency domain convergence behaviors of various DNN models.** The convergence and generalization error analysis of various DNNs for solving PDEs have been investigated extensively recently on PINNs [142, 143, 144], DeepRitz type variational DNNs [145, 146, 147], deep Galerkin DNNs [148], and DeepOnet [149, 150], just to list a few. However, not much work is done to illuminate the frequency dependence of the convergences in terms of spectral bias free behaviors. The loss functions for training DNN come from different mathematical formulations of the solution approach for the PDEs, including the residual based PINN, variational energy based DeepRitz, Galerkin methods, stochastic differential equation based methods, etc. The specific form of the loss function will require a specialized tool to analyze the convergence and exam how the spectral bias manifests itself. The convergence analysis for fitting problems from the perspective of frequency domain has been rigorously analyzed in the linear regime of normal fully-connected neural networks [66, 65, 75, 67] or a two layer multiscale networks [95]. Such analysis for PINNs in the linear regime can be anticipated in future works, especially for solving PDEs. Extending Fourier analysis from linear regime to nonlinear regime has intrinsic difficulty due to the composition of functions. Some qualitative analysis can be found in [64]. However, quantitative analysis remains unsolved. Moreover, finding the appropriate tools for the convergence study for DNN using other form of loss functions and training strategies such as the min-max optimization, SDE based DNNs [as well as PDE operator learning](#), remains open problems.

Other challenges in making DNN based algorithms to be free of spectral bias and achieve robust and frequency uniform convergence as a practical simulation tool exist, including how to select various hyper-parameters, including penalty constants in various terms of the loss functions, activation function, and learning rate, etc. Those issues are not addressed in this review. However, with recent progress in reducing the adverse effect of the spectral bias of DNN in approximating PDEs solutions, the opportunities for research in the DNN based computational methods for solving PDEs are

numerous and rewarding, and the fundamental progress on algorithms and mathematical analysis will have far-reaching impact in many application areas.

## Acknowledgement

Z.X. acknowledges the financial support provided by the National Key R&D Program of China Grant No. 2022YFA1008200, the National Natural Science Foundation of China Grant No. 92270001, 12371511, Shanghai Municipal of Science and Technology Major Project No. 2021SHZDZX0102, and the HPC of School of Mathematical Sciences and the Student Innovation Center, and the Siyuan-1 cluster supported by the Center for High Performance Computing at Shanghai Jiao Tong University. W.C. acknowledges the financial support provided by the US National Science Foundation grant DMS-2207449.

## References

- [1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: *Advances in Neural Information Processing Systems*, Vol. 25, 2012, pp. 1106–1114.
- [2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal processing magazine* 29 (6) (2012) 82–97.
- [3] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE* 86 (11) (1998) 2278–2324.
- [4] Y. Bengio, et al., Learning deep architectures for ai, *Foundations and trends<sup>®</sup> in Machine Learning* 2 (1) (2009) 1–127.
- [5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *nature* 521 (7553) (2015) 436–444.
- [6] W. E, Machine learning and computational mathematics, *Communications in Computational Physics* 28 (5) (2020) 1639–1670.

- [7] J. Han, L. Zhang, R. Car, et al., Deep potential: A general representation of a many-body potential energy surface, *Communications in Computational Physics* 23 (3) (2018).
- [8] W. E, J. Han, A. Jentzen, Algorithms for solving high dimensional pdes: From nonlinear monte carlo to machine learning, *Nonlinearity* 35 (1) (2021) 278.
- [9] J. Hermann, Z. Schätzle, F. Noé, Deep-neural-network solution of the electronic schrödinger equation, *Nature Chemistry* 12 (10) (2020) 891–897.
- [10] D. Pfau, J. S. Spencer, A. G. Matthews, W. M. C. Foulkes, Ab initio solution of the many-electron schrödinger equation with deep neural networks, *Physical Review Research* 2 (3) (2020) 033429.
- [11] J. Lim, D. Psaltis, Maxwellnet: Physics-driven deep neural network training based on maxwell’s equations, *Apl Photonics* 7 (1) (2022).
- [12] Z. Liu, W. Cai, Z.-Q. J. Xu, Multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains, *Communications in Computational Physics* 28 (5) (2020) 1970–2001.
- [13] W. Cai, *Computational Methods for Electromagnetic Phenomena: electrostatics in solvation, scattering, and electron transport*, Cambridge University Press, 2013.
- [14] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics* 378 (2019) 686–707.
- [15] X. Jin, S. Cai, H. Li, G. E. Karniadakis, Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations, *Journal of Computational Physics* 426 (2021) 109951.
- [16] M. Yin, E. Zhang, Y. Yu, G. Karniadakis, Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems, *Computer methods in applied mechanics and engineering* 402 (2022) 115027.

- [17] W. E. J. Han, A. 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) (2017) 349–380.
- [18] P. Ciarlet, *The finite element method for elliptic problems*, Society for Industrial and Applied Mathematics, 2002.
- [19] O. Zienkiewicz, R. Taylor, *The finite element method for solid and structural mechanics*, Elsevier, 2005.
- [20] B.-n. Jiang, *The least-squares finite element method: theory and applications in computational fluid dynamics and electromagnetics*, Springer Science & Business Media, 1998.
- [21] P. B. Bochev, M. D. Gunzburger, *Least-squares finite element methods*, Vol. 166, Springer Science & Business Media, 2009.
- [22] J. C. Strikwerda, *Finite difference schemes and partial differential equations*, SIAM, 2004.
- [23] M. Y. Hussaini, T. A. Zang, Spectral methods in fluid dynamics, *Annual review of fluid mechanics* 19 (1) (1987) 339–367.
- [24] J. Shen, T. Tang, L.-L. Wang, *Spectral methods: algorithms, analysis and applications*, Vol. 41, Springer Science & Business Media, 2011.
- [25] W. Hackbusch, *The integral equation method*, Birkhäuser Basel, 1995.
- [26] H. Tennekes, J. Lumley, *A first course in turbulence*, MIT Press, 1972.
- [27] A. J. Chorin, *Vorticity and turbulence*, Springer Science & Business Media, 2013.
- [28] B. E. Saleh, M. C. Teich, *Fundamentals of photonics*, John Wiley & Sons, 2019.
- [29] M. Born, E. Wolf, *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*, Elsevier, 2013.
- [30] L. D. Landau, E. M. Lifshitz, *Quantum mechanics: non-relativistic theory*, Vol. 3, Elsevier, 2013.

- [31] J. Rammer, Quantum transport theory, CRC Press, 2018.
- [32] T. Zhang, Y. Yi, Y. Xu, Z. X. Chen, Y. Zhang, W. E, Z.-Q. J. Xu, A multi-scale sampling method for accurate and robust deep neural network to predict combustion chemical kinetics, *Combustion and Flame* 245 (2022) 112319.
- [33] A. Brandt, Multi-level adaptive solutions to boundary-value problems, *Mathematics of computation* 31 (138) (1977) 333–390.
- [34] J. Xu, Iterative methods by space decomposition and subspace correction, *SIAM review* 34 (4) (1992) 581–613.
- [35] Y. Saad, Iterative methods for sparse linear systems, SIAM, 2003.
- [36] C. E. Shannon, A mathematical theory of communication, *The Bell system technical journal* 27 (3) (1948) 379–423.
- [37] G. Ciaramella, M. Gander, Iterative methods and preconditioners for systems of linear equations, Society for Industrial and Applied Mathematics, 2022.
- [38] S. Olver, A. Townsend, A fast and well-conditioned spectral method, *SIAM review* 55 (3) (2013) 462–489.
- [39] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, Z. Ma, Frequency principle: Fourier analysis sheds light on deep neural networks, *Communications in Computational Physics* 28 (5) (2020) 1746–1767.
- [40] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, in: *International Conference on Machine Learning*, PMLR, 2019, pp. 5301–5310.
- [41] T. Luo, Z. Ma, Z. Wang, Z. J. Xu, Y. Zhang, An upper limit of decaying rate with respect to frequency in linear frequency principle model, in: *Mathematical and Scientific Machine Learning*, PMLR, 2022, pp. 205–214.
- [42] Z.-Q. J. Xu, Y. Zhang, T. Luo, Overview frequency principle/spectral bias in deep learning, *arXiv preprint arXiv:2201.07395* (2022).

- [43] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Mathematics of control, signals and systems* 2 (4) (1989) 303–314.
- [44] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural networks* 2 (5) (1989) 359–366.
- [45] K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural networks* 4 (2) (1991) 251–257.
- [46] M. Leshno, V. Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, *Neural networks* 6 (6) (1993) 861–867.
- [47] W. E. C. Ma, L. Wu, The barron space and the flow-induced function spaces for neural network models, *Constructive Approximation* 55 (1) (2022) 369–406.
- [48] J. Lu, Z. Shen, H. Yang, S. Zhang, Deep network approximation for smooth functions, *SIAM Journal on Mathematical Analysis* 53 (5) (2021) 5465–5506.
- [49] L. Liu, K. Nath, W. Cai, A causality-deeponet for causal responses of linear dynamical systems, *Commun. Comput. Phys.* 35 (5) (2024) 1194–1228.
- [50] M. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, *communications in Numerical Methods in Engineering* 10 (3) (1994) 195–201.
- [51] I. Lagaris, A. Likas, D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE transactions on Neural Networks* 9 (5) (1998) 987–1000.
- [52] I. Lagaris, A. Likas, G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, *IEEE transactions on Neural Networks* 11 (5) (2000) 1041–1049.
- [53] J. Sirignano, K. Spiliopoulos, Dgm: A deep learning algorithm for solving partial differential equations, *Journal of computational physics* 375 (2018) 1339–1364.

- [54] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, *Neurocomputing* 317 (2018) 28–41.
- [55] E. Kharazmi, Z. Zhang, G. E. Karniadakis, hp-vpinns: Variational physics-informed neural networks with domain decomposition, *Computer Methods in Applied Mechanics and Engineering* 374 (1) (2021) 113547.
- [56] M. L. Z. Cai, J. Chen, X. Liu, Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic pdes, *Journal of Computational Physics*. (2020).
- [57] B. Wang, W. Zhang, W. Cai, Multi-scale deep neural network (mscalednn) methods for oscillatory stokes flows in complex domains, *Communications in Computational Physics* 28 (5) (2020) 2139–2157.
- [58] Z. Gao, L. Yan, T. Zhou, Failure-informed adaptive sampling for pinns, *SIAM Journal on Scientific Computing* 45 (4) (2023) A1971–A1994.   
arXiv:<https://doi.org/10.1137/22M1527763>.  
URL <https://doi.org/10.1137/22M1527763>
- [59] Z. Gao, T. Tang, L. Yan, et al., Failure-informed adaptive sampling for pinns, part ii: combining with re-sampling and subset simulation, *Communications on Applied Mathematics and Computation* 6 (3) (2024) 1720–1741.
- [60] W. E, B. Yu, The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, *Communications in Mathematics and Statistics* 6 (1) (2018) 1–12.
- [61] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, Deepxde: A deep learning library for solving differential equations, *SIAM Review* 63 (1) (2021) 208–228.
- [62] C. Ma, L. Wu, et al., Machine learning from a continuous viewpoint, i, *Science China Mathematics* 63 (11) (2020) 2233–2266.
- [63] Z.-Q. J. Xu, Y. Zhang, Y. Xiao, Training behavior of deep neural network in frequency domain, *International Conference on Neural Information Processing* (2019) 264–274.



- [64] T. Luo, Z. Ma, Z.-Q. J. Xu, Y. Zhang, Theory of the frequency principle for general deep neural networks, *CSIAM Transactions on Applied Mathematics* 2 (3) (2021) 484–507. doi:<https://doi.org/10.4208/csiam-am.S0-2020-0005>.
- [65] T. Luo, Z. Ma, Z.-Q. J. Xu, Y. Zhang, On the exact computation of linear frequency principle dynamics and its generalization, *SIAM Journal on Mathematics of Data Science* 4 (4) (2022) 1272–1292.
- [66] Y. Zhang, T. Luo, Z. Ma, Z.-Q. J. Xu, A linear frequency principle model to understand the absence of overfitting in neural networks, *Chinese Physics Letters* 38 (3) (2021) 038701.
- [67] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, Q. Gu, Towards understanding the spectral bias of deep learning, in: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, 2021*, pp. 2205–2211.
- [68] B. Bordelon, A. Canatar, C. Pehlevan, Spectrum dependent learning curves in kernel regression and wide neural networks, in: *International Conference on Machine Learning*, PMLR, 2020, pp. 1024–1034.
- [69] O. Axelsson, *Iterative solution methods*, Cambridge university press, 1996.
- [70] Y. Ma, Z.-Q. J. Xu, J. Zhang, Frequency principle in deep learning beyond gradient-descent-based training, *arXiv preprint arXiv:2101.00747* (2021).
- [71] Z. J. Xu, Understanding training and generalization in deep learning by fourier analysis, *arXiv preprint arXiv:1808.04295* (2018).
- [72] S. Biland, V. C. Azevedo, B. Kim, B. Solenthaler, Frequency-aware reconstruction of fluid simulations with generative networks, *Eurographics* (2020).
- [73] A. Jacot, F. Gabriel, C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks, *Advances in neural information processing systems* 31 (2018) 8580–8589.

- [74] T. Luo, Z.-Q. J. Xu, Z. Ma, Y. Zhang, Phase diagram for two-layer relu neural networks at infinite-width limit, *The Journal of Machine Learning Research* 22 (1) (2021) 3327–3373.
- [75] B. Ronen, D. Jacobs, Y. Kasten, S. Kritchman, The convergence rate of neural networks for learned functions of different frequencies, *Advances in Neural Information Processing Systems* 32 (2019) 4761–4771.
- [76] Y. Zhang, Z.-Q. J. Xu, T. Luo, Z. Ma, Explicitizing an implicit bias of the frequency principle in two-layer neural networks, arXiv preprint arXiv:1905.10264 (2019).
- [77] W. E, C. Ma, L. Wu, Machine learning from a continuous viewpoint, *Science China Mathematics* 63 (11) (2020) 2233–2266.
- [78] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, J. Pennington, Wide neural networks of any depth evolve as linear models under gradient descent, *Advances in neural information processing systems* 32 (2019).
- [79] W. Cai, X. Li, L. Liu, A phase shift deep neural network for high frequency approximation and wave problems, *SIAM Journal on Scientific Computing* 42 (5) (2020) A3285–A3312.
- [80] W. Peng, W. Zhou, J. Zhang, W. Yao, Accelerating physics-informed neural network training with prior dictionaries, arXiv preprint arXiv:2004.08151 (2020).
- [81] Y. Kim, S. Kim, I. Seo, B. Shin, Phase-shifted adversarial training, *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence* 216 (2023) 1068–1077.
- [82] L. Zhang, W. Cai, Z.-Q. J. Xu, A correction and comments on “multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains *cicp*, 28 (5): 1970–2001, 2020”, *Communications in Computational Physics* 33 (5) (2023) 1509–1513.
- [83] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, R. Ng, Fourier features let networks learn high frequency functions in low dimensional domains,

in: *Advances in Neural Information Processing Systems*, Vol. 33, Curran Associates, Inc., 2020, pp. 7537–7547.

- [84] J. Chen, X. Chi, Z. Yang, et al., Bridging traditional and machine learning-based algorithms for solving pdes: The random feature method, *Journal of Machine Learning* (2022).
- [85] W. Cai, Z.-Q. J. Xu, Multi-scale deep neural networks for solving high dimensional pdes, arXiv preprint arXiv:1910.11710 (2019).
- [86] X. Huang, H. Liu, B. Shi, Z. Wang, K. Yang, Y. Li, M. Wang, H. Chu, J. Zhou, F. Yu, B. Dong, L. Chen, A universal pinns method for solving partial differential equations with a point source, in: *International Joint Conference on Artificial Intelligence*, 2022.
- [87] M. Chen, R. Niu, W. Zheng, Adaptive multi-scale neural network with resnet blocks for solving partial differential equations, *Nonlinear Dynamics* 111 (7) (2023) 6499–6518.
- [88] Z. Wang, M. Chen, J. Chen, Solving multiscale elliptic problems by sparse radial basis function neural networks, *Journal of Computational Physics* 492 (2023) 112452.
- [89] X.-A. Li, Z.-Q. J. Xu, L. Zhang, Subspace decomposition based dnn algorithm for elliptic type multi-scale pdes, *Journal of Computational Physics* 488 (2023) 112242.
- [90] J. Ying, J. Liu, J. Chen, S. Cao, M. Hou, Y. Chen, Multi-scale fusion network: A new deep learning structure for elliptic interface problems, *Applied Mathematical Modelling* 114 (2023) 252–269.
- [91] Z. Li, Y. Wang, H. Liu, Z. Wang, B. Dong, Solving boltzmann equation with neural sparse representation, *SIAM Journal on Scientific Computing* (Accepted).
- [92] L. Liu, B. Wang, W. Cai, Linearized learning methods with multi-scale deep neural networks for stationary navier-stokes equations with oscillatory solutions, arXiv preprint arXiv:2102.03293 (2021).
- [93] Y. Chen, D. Wang, D. Feng, G. Tian, V. Gupta, R. Cao, M. Wan, S. Chen, Three-dimensional spatiotemporal wind field reconstruction based on lidar and multi-scale pinn, *Applied Energy* 377 (2025) 124577.

- [94] R. Riganti, Y. Zhu, W. Cai, S. Torquato, L. D. Negro, Effective medium properties of stealthy hyperuniform photonic structures using multiscale physics-informed neural networks, arXiv preprint arXiv:2405.07878 (2024).
- [95] B. Wang, H. Yuan, L. Liu, W. Zhang, W. Cai, On spectral bias reduction of multi-scale neural networks for regression and boundary value problems, arXiv preprint arXiv:2212.03416 (2022).
- [96] J. Huang, R. You, T. Zhou, Frequency-adaptive multi-scale deep neural networks, arXiv preprint arXiv:2410.00053 (2024). To appear in *Computer Methods in Applied Mechanics and Engineering* (2025).
- [97] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, *Advances in neural information processing systems* 30 (2017).
- [98] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis, *Communications of the ACM* 65 (1) (2021) 99–106.
- [99] 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, *Computer Methods in Applied Mechanics and Engineering* 384 (2021) 113938.
- [100] S. Li, Y. Xia, Y. Liu, Q. Liao, A deep domain decomposition method based on fourier features, *Journal of Computational and Applied Mathematics* 423 (2023) 114963.
- [101] X. Guan, B. Hu, S. Mao, X. Wang, Mhdnet: Multi-modes multiscale physics informed neural networks for solving magnetohydrodynamics problems, arXiv preprint arXiv:2305.07940 (2023).
- [102] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), arXiv preprint arXiv:1606.08415 (2016).
- [103] N. Boullé, Y. Nakatsukasa, A. Townsend, Rational neural networks, *Advances in neural information processing systems* 33 (2020) 14243–14253.

- [104] Y. Wang, J. W. Siegel, Z. Liu, T. Y. Hou, On the expressiveness and spectral bias of kans, arXiv preprint arXiv:2410.01803 (2024).
- [105] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, *Advances in neural information processing systems* 33 (2020) 7462–7473.
- [106] A. D. Jagtap, G. E. Karniadakis, How important are activation functions in regression and classification? a survey, performance comparison, and future directions, *Journal of Machine Learning for Modeling and Computing* 4 (1) (2023).
- [107] A. D. Jagtap, K. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, *Journal of Computational Physics* 404 (2020) 109136.
- [108] S. Liang, L. Lyu, C. Wang, H. Yang, Reproducing activation function for deep learning, arXiv preprint arXiv:2101.04844 (2021).
- [109] A. D. Jagtap, Y. Shin, K. Kawaguchi, G. E. Karniadakis, Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions, *Neurocomputing* 468 (2022) 165–180.
- [110] J. Huang, H. Wang, H. Yang, Int-deep: A deep learning initialized iterative method for nonlinear problems, *Journal of Computational Physics* 419 (2020) 109675.
- [111] G.-Y. Chen, Y.-H. Yu, M. Gan, C. Chen, W. Guo, Properties and potential applications of random functional-linked types of neural networks, arXiv preprint arXiv:2304.00957 (2023).
- [112] E. Zhang, A. Kahana, A. Kopanicakova, E. Turkel, R. Ranade, J. Pathak, G. Karniadakis, Blending neural operators and relaxation methods in pde numerical solvers, *Nat Mach Intell* 6 (2024) 1303–1313.
- [113] W. Hackbusch, *Multi-grid methods and applications*, Vol. 4, Springer Science and Business Media, 2013.
- [114] J. He, J. Xu, Mgnet: A unified framework of multigrid and convolutional neural network, *Science china mathematics* 62 (2019) 1331–1354.

- [115] Y. Chen, B. Dong, J. Xu, Meta-mgnet: Meta multigrid networks for solving parameterized partial differential equations, *Journal of computational physics* 455 (2022) 110996.
- [116] J. Han, Y. Lee, Hierarchical learning to solve partial differential equations using physics-informed neural networks, *arXiv preprint arXiv:2112.01254* (2021).
- [117] Z. Aldirany, R. Cottreau, M. Laforest, S. Prudhomme, Multi-level neural networks for accurate solutions of boundary-value problems, *Computer Methods in Applied Mechanics and Engineering* 419 (2024) 116666.
- [118] Y. Xu, Multi-grade deep learning, *arXiv preprint arXiv:2302.00150* (2023).
- [119] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, *Nature Machine Intelligence* 3 (3) (2021) 218–229.
- [120] Z. Li, N. B. Kovachki, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: *International Conference on Learning Representations*, 2020.
- [121] J. K. Gupta, J. Brandstetter, Towards multi-spatiotemporal-scale generalized pde modeling, *arXiv preprint arXiv:2209.15616* (2022).
- [122] M. A. Rahman, Z. E. Ross, K. Azizzadenesheli, U-no: U-shaped neural operators, *arXiv preprint arXiv:2204.11127* (2022).
- [123] O. Ovadia, E. Turkel, A. Kahana, G. E. Karniadakis, Ditto: Diffusion-inspired temporal transformer operator, *arXiv preprint arXiv:2307.09072* (2023).
- [124] Z. You, Z. Xu, W. Cai, Mscalefno - multi-scale fourier neural operator learning for oscillatory function spaces, *arXiv preprint arXiv:2412.20183* (2024).
- [125] L. Liu, W. Cai, Multiscale deeponet for nonlinear operators in oscillatory function spaces for building seismic wave responses, *preprint arXiv:2111.04860* (2021).

- [126] X. Liu, B. Xu, S. Cao, L. Zhang, Mitigating spectral bias for the multi-scale operator learning, *Journal of Computational Physics* 506 112944.
- [127] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: *Advances in Neural Information Processing Systems*, Vol. 33, 2020, pp. 6840–6851.
- [128] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, *Communications of the ACM* 63 (11) (2020) 139–144.
- [129] V. Oommen, A. Bora, Z. Zhang, G. Karniadakis, Integrating neural operators with diffusion models improves spectral representation in turbulence modeling, *arXiv preprint arXiv:2409.08477*. To appear in *Proc Royal Society*.
- [130] R. Molinaro, S. Lanthaler, B. Raonić, T. Rohner, V. Armegioiu, Z. Y. Wan, F. Sha, S. Mishra, L. Zepeda-Núñez, Generative ai for fast and accurate statistical computation of fluids, *arXiv preprint arXiv:2409.18359* (2024).
- [131] H. Wu, K. Zhang, D. Zhou, W.-L. Chen, Z. Han, Y. Cao, High-flexibility reconstruction of small-scale motions in wall turbulence using a generalized zero-shot learning, *Journal of Fluid Mechanics* 990 (2024) R1.
- [132] I. Livshits, A. Brandt, Accuracy properties of the wave-ray multigrid algorithm for helmholtz equations, *SIAM J. Sci. Comput.* 28 (2006) 1228–1251.
- [133] N. Tishby, N. Zaslavsky, Deep learning and the information bottleneck principle, in: *2015 IEEE Information Theory Workshop (ITW)*, IEEE, 2015, pp. 1–5.
- [134] T. E. Duncan, On the calculation of mutual information, *SIAM Journal on Applied Mathematics* 19 (1) (1970) 215–220.
- [135] S. J. Anagnostopoulos, J. D. Toscano, N. Stergiopoulos, G. E. Karniadakis, Residual-based attention and connection to information bottleneck theory in pinns, *arXiv preprint arXiv:2307.00379* (2023).

- [136] H. Zhou, Z. Qixuan, Z. Jin, T. Luo, Y. Zhang, Z.-Q. Xu, Empirical phase diagram for three-layer neural networks with infinite width, *Advances in Neural Information Processing Systems* 35 (2022) 26021–26033.
- [137] Z. Zhang, Z.-Q. J. Xu, Implicit regularization of dropout, *arXiv preprint arXiv:2207.05952* (2022).
- [138] Z. Zhang, Y. Li, T. Luo, Z.-Q. J. Xu, Stochastic modified equations and dynamics of dropout algorithm, *arXiv preprint arXiv:2305.15850* (2023).
- [139] H. Zhou, Q. Zhou, T. Luo, Y. Zhang, Z.-Q. Xu, Towards understanding the condensation of neural networks at initial training, *Advances in Neural Information Processing Systems* 35 (2022) 2184–2196.
- [140] Z. Zhou, H. Zhou, Y. Li, Z.-Q. J. Xu, Understanding the initial condensation of convolutional neural networks, *arXiv preprint arXiv:2305.09947* (2023).
- [141] Z. Chen, Y. Li, T. Luo, Z. Zhou, Z.-Q. J. Xu, Phase diagram of initial condensation for two-layer neural networks, *arXiv preprint arXiv:2303.06561* (2023).
- [142] Y. Shin, J. Darbon, G. Karniadakis, On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes, *COMMUNICATIONS IN COMPUTATIONAL PHYSICS* 28 (5) (2020) 2042–74.
- [143] Y. Shin, Z. Zhang, G. Karniadakis, Error estimates of residual minimization using neural networks for linear pdes, *Journal of Machine Learning for Modeling and Computing* 4 (4) (2023).
- [144] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems, *IMA J. Numer. Anal* 42 (2) (2022) 981–1022.
- [145] J. Lu, Y. Lu, A priori generalization error analysis of two-layer neural networks for solving high dimensional schrödinger eigenvalue problems, *Commun. AMS* 2 (1) (2022) 1–21.



- [146] Y. Lu, J. Lu, M. Wang, A priori generalization analysis of the deep ritz method for solving high dimensional elliptic partial differential equations, *Conference on Learning Theory*, PMLR (2021) 3196–3241.
- [147] P. Dondl, J. Müller, M. Zeinhofer, Uniform convergence guarantees for the deep ritz method for nonlinear problems, *Advances in Continuous and Discrete Models* (2022) 1–9.
- [148] Y. Jiao, Y. Lai, Y. Wang, H. Yang, Y. Yang, Convergence analysis of the deep galerkin method for weak solutions, *arXiv:2302.02405* (2023).
- [149] B. Deng, Y. Shin, L. Lu, Z. Zhang, G. Karniadakis, Convergence rate of deeponets for learning operators arising from advection-diffusion equations, *Neural Networks* 153 (2021) 411–426.
- [150] S. Lanthaler, S. Mishra, G. Karniadakis, Error estimates for deeponets: A deep learning framework in infinite dimensions, *Transactions of Mathematics and Its Applications* 6 (1:tnac001) (2022).