A Computable Definition of the Spectral Bias

Jonas Kiessling,[1,2] **Filip Thor** [1,2 *]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
[2] H-AI AB, Stockholm, Sweden
jonas.kiessling@h-ai.se, filip.thor@it.uu.se

Abstract

Neural networks have a bias towards low frequency functions. This spectral bias has been the subject of several previous studies, both empirical and theoretical. Here we present a computable definition of the spectral bias based on a decomposition of the reconstruction error into a low and a high frequency component. The distinction between low and high frequencies is made in a way that allows for easy interpretation of the spectral bias. Furthermore, we present two methods for estimating the spectral bias. Method 1 relies on the use of the discrete Fourier transform to explicitly estimate the Fourier spectrum of the prediction residual, and Method 2 uses convolution to extract the low frequency components, where the convolution integral is estimated by Monte Carlo methods. The spectral bias depends on the distribution of the data, which is approximated with kernel density estimation when unknown. We devise a set of numerical experiments that confirm that low frequencies are learned first, a behavior quantified by our definition.

Introduction

Neural networks (NN) have been observed to perform exceptionally well on a large set of machine learning problems. However, all aspects of their convergence are not completely understood. One example of this knowledge gap resides in the frequency domain. The NNs have been seen to earlier in training better approximate low frequency functions than high frequency ones (Basri et al. 2019). In other words, the NNs seem to learn the low frequency components of the target function before it finds the high frequencies, which is sometimes referred to as the spectral bias of neural networks (Rahaman et al. 2019). With a computable definition of the spectral bias the phenomenon can be measured quantitatively, and we hope to gain insights on the behavior of network optimization.

Our Contribution

Our main contribution is a computable definition of the spectral bias in function reconstruction problems. The frequency domain is split into *high* and *low* frequencies by a cut-off frequency ω_0, and we denote the corresponding NN reconstruction error contributions as $\mathcal{E}_{low}$ and $\mathcal{E}_{high}$ and refer to them as the frequency errors. The spectral bias is defined as the quotient $SB = (\mathcal{E}_{high} - \mathcal{E}_{low})/(\mathcal{E}_{high} + \mathcal{E}_{low})$. The cut-off frequency is defined such that half the variance of the target function is comprised of low frequencies.

Two methods for computing the spectral bias are presented. In Method 1 we estimate the Fourier transform of the difference between the target and the output of the NN with the discrete Fourier transform. The frequency errors are then estimated with a simple quadrature rule. Method 2 avoids computation of the Fourier transform. It relies on convolution and Monte Carlo quadrature for estimation of the frequency errors.

To compute the spectral bias we need an expression for the density of the data. In cases where the density is not known explicitly, we estimate it with kernel density estimation. We compute the spectral bias in a number of experiments, and our results are in line with previous work in that low frequencies are learned first.

Related Work

Rahaman et al. (2019) use Fourier analysis on ReLU NNs and perform experiments with both synthetic and real-world data to show that NNs have a spectral bias. They highlight the spectral bias by for example studying how different frequencies of noise affect the accuracy on the MNIST classification problem, and conclude that the networks' performance is more sensitive to low frequency noise than high frequency noise.

Basri et al. (2019) use harmonic analysis to show that neural networks learn high frequency functions at a lower rate than low frequency functions. Cao et al. (2021) show, using the neural tangent kernel (NTK), that the lower spherical harmonics are more easily captured when learning data with uniform distributions. Basri et al. (2020) also study the NTK, and show that the convergence rate depends on the density of the data. Zhang, Xiong, and Wu (2021) discuss the relation between the spectral bias, generalization, and memorization of NNs. To study the spectrum of the NN for high dimensional targets they introduce a metric based on computing local low dimensional discrete Fourier transforms, instead of the expensive d- dimensional Fourier transform.

A sequence of publications Xu (2018, 2020); Xu, Zhang,

*Current affiliation is Department of Information Technology, Uppsala University.

and Xiao (2019); Luo et al. (2019); Xu and Zhou (2021) discuss the spectral bias while referring to it as the *Frequency Principle*. In Xu, Zhang, and Xiao (2019); Xu (2018) the errors of the network prediction on certain frequency indices is compared, showing that higher frequencies have slower convergence compared to lower frequencies. Xu (2020) introduces ways to measure the effect of the frequency principle, using one method based on projection onto basis functions, and one using convolution. (Xu and Zhou 2021) studies the frequency principle in deep NNs, where they find that deeper networks may be more biased to learning low frequencies. Luo et al. (2019) develop convergence bounds for low and high frequencies at different stages of training. The idea behind their way to compute errors for low frequency components of a NN prediction using convolution is similar to Method 2 presented in our work. The main differences are that: We give a definition of the spectral bias in terms of the error decomposition that is described by a single value. Our definition of the spectral bias depends on the distribution of the data, thus connecting the definition of the bias with the loss function. Finally we present a method for choosing the cutoff frequency that enables a comparison of the frequency errors across different examples.

Shallow neural networks with trigonometric activation are an interesting case to study in this context since their potential to approximate certain frequencies can directly be inferred through the distribution of the network weights. Such NNs have been studied by Kammonen et al. (2020). Their main contribution is a novel training algorithm of shallow neural networks based on an adaptive Metropolis sampling algorithm. The authors also show that in training shallow trigonometric networks on high frequency target functions with standard optimization techniques like SGD, many more iterations are required to learn the high frequency content, in comparison to the low frequency content, of the target function.

Notation and Definitions

Problem Setting

We focus on function reconstruction, a form of supervised learning. We denote the neural network by β, which we train on a training set $\mathcal{T} = \{(x_i, y_i)\}_{i=0}^{N}$, consisting of samples $(x, y) \in \mathbb{R}^d \times \mathbb{R}^m$, where x are inputs with corresponding outputs y. The data points x_i are assumed to be i.i.d. drawn from a probability density $p(x)$, and the outputs are generated as evaluations of an unknown target function, $y = f(x)$. For clarity of exposition, y is assumed without noise. This report will exclusively study fully connected feed-forward networks.

Fourier Transform

The Fourier transform plays a central role in this work. It is defined as $\hat{f}(\omega) = \int_{\mathbb{R}^d} f(x) e^{-i\omega\cdot x} dx$, with the corresponding inverse transform $f(x) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{f}(\omega) e^{i\omega\cdot x} d\omega$.

Quality of Fit

We use fraction of variance unexplained, FVU, to measure the quality of fit. It is defined as the variance of the residual $r(x) = f(x) - \beta(x)$ of the true target value $f(x)$ and the networks approximation $\beta(x)$, divided by the variance of $f(x)$:

$$\text{FVU} = \frac{\text{Var}(r(x))}{\text{Var}(f(x))} = \frac{\mathbb{E}_x[(r(x) - \mathbb{E}_x[r(x)])^2]}{\mathbb{E}_x[(f(x) - \mathbb{E}_x[f(x)])^2]}, \tag{1}$$

where $\mathbb{E}_x[f(x)] = \int_{\mathbb{R}^d} f(x)p(x)dx$.

Variance

For a function f and a random variable $x \in \mathbb{R}^d$, we introduce the notation

$$f_p(x) = \sqrt{p(x)}\left(f(x) - \mathbb{E}_x[f(x)]\right), \tag{2}$$

where p is the density of x. With (2) and Plancherel's theorem the variance of f can be expressed in the following ways

$$\text{Var}(f(x)) = \int_{\mathbb{R}^d} p(x)(f(x) - \mathbb{E}_x[f(x)])^2 dx \tag{3}$$

$$= \int_{\mathbb{R}^d} f_p(x)^2 dx \tag{4}$$

$$= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\hat{f}_p(\omega)|^2 d\omega. \tag{5}$$

Given samples $f(x_i)$, $i = 1, \ldots, N$, the variance can be approximated by Monte Carlo integration as

$$\text{Var}(f(x)) \approx \frac{1}{N} \sum_{i=1}^{N} \left(f(x_i) - \frac{1}{N} \sum_{j=1}^{N} f(x_j) \right)^2. \tag{6}$$

Definition of the Spectral Bias

This section presents the way we define the spectral bias. In order to evaluate the NNs performance in the Fourier domain, the idea, similar to the approach taken in e.g. (Xu and Zhou 2021; Xu 2020), is to decompose the Fourier domain into two parts, corresponding to high and low frequencies respectively.

To get a mathematical description of the errors in the frequency domain, we take the FVU as the starting point. Using the integral definition of the expected value, the notation introduced in (2), and Plancherel's theorem results in (7)

$$\text{FVU} = \frac{\int_{\mathbb{R}^d} r_p(x)^2 dx}{\text{Var}(f(x))} = \frac{\int_{\mathbb{R}^d} |\hat{r}_p(\omega)|^2 d\omega}{(2\pi)^d \text{Var}(f(x))}. \tag{7}$$

To compare how different frequencies contribute to the error in the spatial domain, the frequency domain is split into low and high frequencies, Ω_{low} and Ω_{high}, by a cutoff frequency ω_0. The split is done via the max norm $|\cdot|_\infty$ as:

$$\begin{aligned} \Omega_{low} &= \{\omega \in \mathbb{R}^d : |\omega|_\infty \leq \omega_0\}, \\ \Omega_{high} &= \{\omega \in \mathbb{R}^d : |\omega|_\infty > \omega_0\}. \end{aligned} \tag{8}$$

In formulas, the error contributions are defined as

$$\text{FVU} = \underbrace{\frac{\int_{\Omega_{low}} |\hat{r}_p(\omega)|^2 d\omega}{(2\pi)^d \text{Var}(f(x))}}_{\text{Low frequency error, } \mathcal{E}_{low}} + \underbrace{\frac{\int_{\Omega_{high}} |\hat{r}_p(\omega)|^2 d\omega}{(2\pi)^d \text{Var}(f(x))}}_{\text{High Frequency error, } \mathcal{E}_{high}}. \tag{9}$$

The quantities $\mathcal{E}_{low}$ and $\mathcal{E}_{high}$ are referred to as the frequency errors. The cutoff frequency ω_0 is defined such that Ω_{low} and Ω_{high} contribute equal amounts to the total variance, i.e.,

$$\int_{\Omega_{low}} |\hat{f}_p(\omega)|^2 d\omega = \int_{\Omega_{high}} |\hat{f}_p(\omega)|^2 d\omega. \tag{10}$$

With the cutoff frequency and frequency errors introduced, we now propose a definition for the spectral bias.

Definition 1 *The spectral bias is defined as the quotient*

$$SB = \frac{\mathcal{E}_{high} - \mathcal{E}_{low}}{\mathcal{E}_{low} + \mathcal{E}_{high}} = \frac{\mathcal{E}_{high} - \mathcal{E}_{low}}{FVU}, \tag{11}$$

with the quantities $\mathcal{E}_{low}$ and $\mathcal{E}_{high}$ as defined in (9) *computed using ω_0 such that* (10) *holds.*

The spectral bias is zero if the neural network performs as well for low frequencies as it does for high frequencies. If that is the case, we say that the NN is spectrally unbiased. If the neural network prediction has a small error for the low frequency components in comparison to the high frequency components, i.e., $\mathcal{E}_{low} \ll \mathcal{E}_{high}$, then $SB \approx 1$, we say that the neural network has large spectral bias.

The spectral bias has in earlier work mostly been studied in the context of neural networks. However, the spectral bias as defined in Definition 1 can be used as a measure in any function reconstruction problem.

Computing the Spectral Bias

In this section we present two methods for estimating the spectral bias.

Method 1

Method 1 relies on the discrete Fourier transform (DFT) to estimate the spectral bias as defined in (11). To compute the spectral bias with Method 1 we need samples of the target function on an equidistant grid. This section presents Method 1 for dimension $d = 1$, and later comments on the use in higher dimension.

First, we need to establish how the DFT relates to the Fourier transform. Assume we are given N equidistant samples $\{g[n]\}_{n=0}^{N-1}$ of a function $g = g(x)$, with $g[n] = g(x_n)$, where $x_n = \Delta x(n - N/2)$ for some spatial increment Δx, the DFT of $\{g[n]\}_{n=0}^{N-1}$ for $k = -N/2, \dots, N/2 - 1$ is defined by

$$\text{DFT}(g)[k] = \sum_{n=0}^{N-1} g[n] e^{-i2\pi kn/N}, \tag{12}$$

with frequency resolution $\Delta\omega = \frac{2\pi}{N\Delta x}$.

The Fourier transform of $g(x)$ evaluated at $k\Delta\omega$, for $k = -N/2, \dots, N/2-1$, is approximated by a left Riemann sum as

$$\hat{g}(k\Delta\omega) = \int_{\mathbb{R}} g(x) e^{-ik\Delta\omega x} dx \tag{13}$$

$$\approx \sum_{n=0}^{N-1} g[n] e^{-ik2\pi(n-N/2)/N} \Delta x. \tag{14}$$

Comparing (12) and (14), the DFT scaled by Δx is a phase shifted first order approximation of the Fourier transform. That is, the Fourier transform is approximated at equidistant frequencies by the DFT as

$$\hat{g}(k\Delta\omega) \approx \text{DFT}(g)[k] e^{ik\pi} \Delta x. \tag{15}$$

Recall ω_0 defined by (10). With (15) we get $\hat{f}_p(k\Delta\omega) \approx \text{DFT}(f_p)[k] e^{ik\pi}\Delta x$. Defining $\Omega = [-\frac{N}{2}, \dots, (\frac{N}{2} - 1)\Delta\omega]$, $\Omega'_{low} = \Omega \cap \Omega_{low}$ and $\Omega'_{high} = \Omega \cap \Omega_{high}$, we approximate (10) with Riemann sums, and find ω_0 by solving

$$\min_{\substack{\omega_0 \in \Omega, \\ \omega_0 \geq 0}} \left| \sum_{\Omega'_{low}} |F_p[k]\Delta x|^2 \Delta\omega - \sum_{\Omega'_{high}} |F_p[k]\Delta x|^2 \Delta\omega \right|, \tag{16}$$

where $F_p[k] = \text{DFT}(f_p)[k] e^{ik\pi}$. The low frequency error defined in (9) is estimated through

$$\mathcal{E}_{low} \approx \frac{\sum_{\Omega'_{low}} |R_p[k]\Delta x|^2 \Delta\omega}{\sum_{\Omega} |F_p[k]\Delta x|^2 \Delta\omega}, \tag{17}$$

where $R_p[k] = \text{DFT}(r_p)[k] e^{ik\pi}$, and the variance as in (5) has been approximated by a Riemann sum.

Method 1 of estimating the spectral bias is defined as computing (11) using ω_0 from (16), $\mathcal{E}_{low}$ from (17) and $\mathcal{E}_{high} = \text{FVU} - \mathcal{E}_{low}$. Using a fast Fourier transform (FFT) enables cheap computation of the DFT and is the reason we require an equidistant grid. This results in a computational cost of $\mathcal{O}(N \log(N))$. In higher dimensions keeping the same resolution in each direction results in a cost of $\mathcal{O}(N^d \log(N^d))$, which quickly becomes insurmountable for large d. Two severe drawbacks with Method 1 are the requirement of data availability on an equidistant grid which is seldom the case, and the large computational cost for high dimensional targets. With the aim to alleviate these two shortcomings Method 2 is now presented.

Method 2

Method 2 extracts the low frequency error in (9) by convolution with the sinc function, which in the Fourier domain corresponds to multiplication with the indicator function $\mathbb{1}_{\Omega_{low}} = \mathbb{1}_{\Omega_{low}}(\omega)$. Integrals are approximated with Monte Carlo integration. This enables estimation of the spectral bias without explicitly computing the frequency spectrum, keeping all computations in the spatial domain and on arbitrary densities. We now present how to compute the spectral bias with Method 2.

Fix some ω_0. As in (9), the variance of a function f can in the Fourier domain be decomposed into

$$\begin{aligned} \text{Var}(f) = &\frac{1}{(2\pi)^d} \int_{\Omega_{low}} |\hat{f}_p(\omega)|^2 d\omega \\ &+ \frac{1}{(2\pi)^d} \int_{\Omega_{high}} |\hat{f}_p(\omega)|^2 d\omega. \end{aligned} \tag{18}$$

The first integral can be expressed as

$$\frac{1}{(2\pi)^d} \int_{\Omega_{low}} |\hat{f}_p(\omega)|^2 d\omega = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \mathbb{1}_{\Omega_{low}} |\hat{f}_p(\omega)|^2 dw \tag{19}$$

$$= \int_{\mathbb{R}^d} (f_p(x) * \varphi(x))^2 dx, \tag{20}$$

where $*$ denotes the convolution operator, and φ is the inverse Fourier transform of the indicator function. The convolution in (20) is approximated by Monte Carlo integration:

$$(f_p(x) * \varphi(x))^2 = \left(\int_{\mathbb{R}^d} f_p(y)\varphi(x-y)dy \right)^2 \tag{21}$$

$$\approx \left(\frac{1}{N} \sum_i \frac{f_p(x_i)}{p(x_i)} \varphi(x - x_i) \right)^2 \tag{22}$$

$$= \frac{1}{N^2} \sum_{i,j} \frac{f_p(x_i) f_p(x_j)}{p(x_i)p(x_j)} \varphi(x-x_i)\varphi(x-x_j). \tag{23}$$

Inserting (23) into (20), we obtain

$$\int_{\mathbb{R}^d} (f_p(x) * \varphi(x))^2 dx \approx \frac{1}{N^2} \sum_{i,j} \frac{f_p(x_i) f_p(x_j)}{p(x_i)p(x_j)} \int_{\mathbb{R}^d} \varphi(x-x_i)\varphi(x-x_j)dx. \tag{24}$$

Using the time shift property, and recalling that φ is the sinc function, we have

$$\int_{\mathbb{R}^d} \varphi(x-x_i)\varphi(x-x_j)dx = \frac{\omega_0^d}{\pi^d} \prod_{k=1}^d \text{sinc}(\omega_0(x_j^k - x_i^k)),$$

where the sinc function is defined as

$$\text{sinc}(x) = \begin{cases} \frac{\sin(x)}{x} & \text{if } x \neq 0, \\ 1 & \text{if } x = 0. \end{cases}$$

Thus, (19) is approximated as

$$\frac{1}{(2\pi)^d} \int_{\Omega_{low}} |\hat{f}_p(\omega)|^2 d\omega \approx \frac{1}{N^2} \sum_{i,j} \frac{f_p(x_i) f_p(x_j)}{p(x_i)p(x_j)} \frac{\omega_0^d}{\pi^d} \prod_{k=1}^d \text{sinc}(\omega_0(x_j^k - x_i^k)). \tag{25}$$

Recalling the cutoff frequency as defined in (10), we seek ω_0 such that

$$\text{Var}(f) = \frac{2}{(2\pi)^d} \int_{\Omega_{low}} |\hat{f}_p(\omega)|^2 d\omega. \tag{26}$$

From the approximations in (25) and (6), we solve for ω_0 such that

$$\frac{2}{N^2} \sum_{i,j} \frac{f_p(x_i) f_p(x_j)}{p(x_i)p(x_j)} \left(\frac{\omega_0}{\pi}\right)^d \prod_{k=1}^d \text{sinc}(\omega_0(x_j^k - x_i^k)) - \frac{1}{N} \sum_{i=1}^N \left(f(x_i) - \frac{1}{N} \sum_{j=1}^N f(x_j) \right)^2 = 0. \tag{27}$$

With the definition of the frequency errors in (9) the approximations (25) and (6) with ω_0 from (27) enables the estimation of $\mathcal{E}_{low}$ as

$$\mathcal{E}_{low} \approx \frac{\sum_{i,j} \frac{r_p(x_i) r_p(x_j)}{p(x_i)p(x_j)} \left(\frac{\omega_0}{\pi}\right)^d \prod_{k=1}^d \text{sinc}(\omega_0(x_j^k - x_i^k))}{N \sum_{i=1}^N \left(f(x_i) - \frac{1}{N} \sum_{j=1}^N f(x_j) \right)^2}. \tag{28}$$

The corresponding high frequency error is estimated as

$$\mathcal{E}_{high} = \text{FVU} - \mathcal{E}_{low}. \tag{29}$$

We define Method 2 of estimating the spectral bias as computing (11) using ω_0 from (27), and $\mathcal{E}_{low}$ and $\mathcal{E}_{high}$ from (28) and (29). Compared with Method 1, Method 2 is more efficient in high dimension. The use of Monte Carlo integration results in a complexity that only scales linearly with dimension, when estimating (28). It does however have a high base complexity, the double sum in (28) yields a computational cost of $\mathcal{O}(N^2)$.

Estimation of Data Density

Recall that both methods need access to the density function p of the data to compute the spectral bias. In the coming synthetic experiments, the exact form of the density is known. This is not the case in general and for cases where the density is unknown it needs to be estimated. In one dimension we use kernel density estimation (KDE). We use a Gaussian kernel, which introduces the kernel bandwidth h as a hyperparameter. The optimal bandwidth is in general nontrivial to find, but for one dimensional normal distributions with unknown parameters we follow Scott (1992) and estimate the optimal bandwidth as $h = 1.06\sigma N^{-1/5}$, where σ is the standard deviation of the data and N the number of data points.

Numerical Experiments

This section presents numerical experiments on function reconstruction using neural network. Its purpose is threefold:

- to show examples of the spectral bias,
- show that the spectral bias as defined in Definition 1 can quantify the behavior observed in the experiments,
- validate that the two methods produce similar results.

Implementation Details

The numerical experiments are done in Python 3.8.6, and all neural networks used in this section are implemented in Tensorflow 2.5.0, and are densely connected feed forward networks with ReLU activation. All hidden layers in the model have the same number of nodes. The weights are initialized with He-initialization (He et al. 2015). The models are all trained to minimize the mean square error loss. Before the models are trained, both the input x and output y of the training data are transformed such that they are component wise demeaned and have unit variance.

Method 1 uses the FFT from the NumPy 1.19.5 library. We use the secant method with initial guess (1,2) to solve (27) for ω_0 in Method 2. When the data has been sampled from the standard normal distribution we have $p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, and using KDE to estimate the density is done via the KernelDensity function from the ScikitLearn library. The experiments are performed on a Windows 10 Home desktop with an Intel i7-10700K CPU @ 3.8 GHz, 48 GB of memory, and an Nvidia GeForce RTX 2070 GPU. The code that reproduces the experiments can be found in the accompanying code appendix.

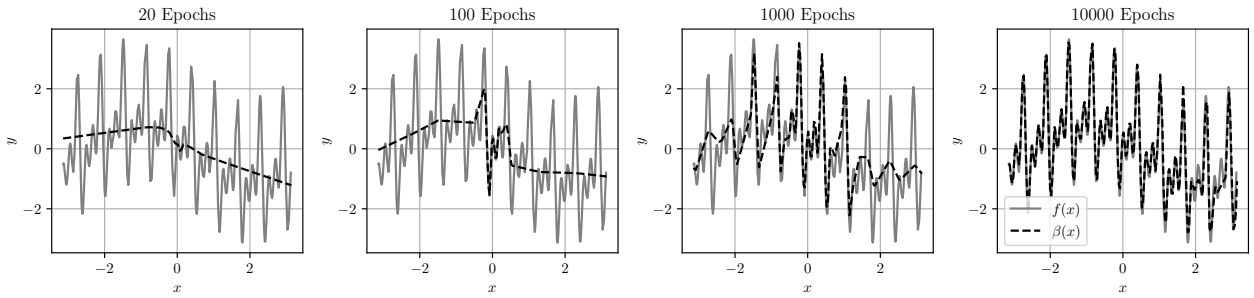

Figure 1: Experiment 1: Reconstructions of $f_1(x)$ made by a NN for an increasing number of epochs. The target function is displayed in gray, and the network approximation $\beta(x)$ in dashed black. The faster oscillations are learned later than the low frequency components.

Experiment 1: Superposition of Sine Functions

Following (Basri et al. 2019) and (Rahaman et al. 2019) we study a target function that is a sum of sine functions with varying frequencies, $f_1(x) = \sin(x) + \sum_{k=1}^{3} \sin(10kx + b_k)$. The training set is given by $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i are 512 i.i.d. samples from $U(-\pi, \pi)$ and $y = f(x)$. The phase shifts b_k are sampled uniformly as $b_k \sim U(0, 2\pi)$. The NN has 5 layers with 64 nodes in each, trained with the Adam optimizer (Kingma and Ba 2015), a batch size of 32, and learning rate of 0.0005. The resulting NN prediction is plotted after different number of epochs in Figure 1.

Experiment 2: High Frequency Target

While the target function used in Experiment 1 is good for the purpose of visualizing the spectral bias, a target function with a wide support in the Fourier domain is better suited for quantitative analysis. The target function in the following experiment is given by $f_2(x) = e^{-x^2/2} Si(ax)$. Here $Si(ax) = \int_0^{ax} \frac{sin(t)}{t} dt$ denotes the sine integral. The support of the Fourier transform of f_2 is determined by a. In this experiment we let $a = 100$. We draw 2^{12} i.i.d. points from $\mathcal{N}(0, 1)$ to use as training data, and another 2^{12} points used as validation data and for estimating the spectral bias with Method 2. Estimating the spectral bias with Method 1 requires an equidistant grid fine enough to resolve the fast oscillations of the target, and wide enough to have a small sampling frequency. We use 2^{14} points on $[-25\pi, 25\pi]$. The network has 5 layers with 64 nodes, is trained with the Adam optimizer, a learning rate of 10^{-3}, and a batch size of 32. Figure 2 shows the FVU and the spectral bias estimated with both methods. Both Method 1 and Method 2 produce almost indistinguishable estimates of the spectral bias, which is observed to be $SB \approx 0.7$ throughout training.

Experiment 3: Image Regression

Another machine learning task where the spectral bias can be visualized is image regression. The problem setup follows Tancik et al. (2020). A color image can be represented with a 3-dimensional value corresponding to the image's RGB values at each pixel coordinate. The goal is for the neural network to predict the color in previously unseen coordinates. Thus, this problem is a function reconstruction problem where we want to learn a function $f : \mathbb{R}^2 \mapsto \mathbb{R}^3$. Sharp contrasts in the image correspond in the Fourier domain to high frequencies. Thus, to attain a good approximation of the image that retains the fine details, the NN must learn the high frequency components of the image. We expect this to be a hard task, given the observed results in Experiment 1 and 2. The image used in this experiment comes from the DIV2K data set (Agustsson and Timofte 2017) used in the NTIRE 2017 challenge on the SISR problem (Timofte et al. 2017). To generate the data given an image, a centered crop of 512×512 pixels is extracted. The training data is chosen as every other pixel in the cropped image, and $x_i = [x_i^1, x_i^2]^\top \in [0, 1] \times [0, 1]$ are the coordinates of pixel number i, and $y_i \in [0, 1]^3$ is a vector containing the corresponding RGB values of the pixel.

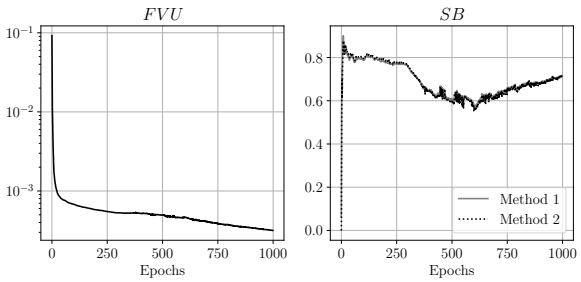

Figure 2: Experiment 2: Resulting FVU and spectral bias for a ReLU network trained to regress the target $f_2(x)$ using Method 1 and Method 2 respectively. Both methods produce similar results.

The neural network has 8 layers, 128 nodes in each layer. The network is trained with the Adam optimizer, a learning rate of 10^{-3}, and a batch size of one tenth of the training set. The resulting network predictions are shown for an increasing number of epochs in Figure 3. In this experiment Method 1 is used to compute the spectral bias since the data is given on an equidistant grid. Figure 4 shows the FVU and spectral bias as a function of epochs. We observe an increasing spectral bias, above 0.8 for the better part of the training.

Figure 3: Experiment 3: Resulting model predictions of a densely connected ReLU network when applied to the task of image regression. The leftmost image shows the target image used for generating the training data, and the subsequent images are the prediction for an increasing amount of training epochs, the NN learns the full RGB representation, but is plotted in grayscale. The spectral bias is visualized by the fact that sharp contrasts in the image are only found later in training.

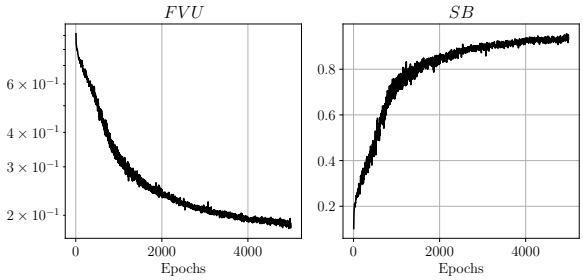

Figure 4: Experiment 3: Computed FVU and spectral bias in image regression, both measures are computed on the grayscale representation of the prediction. A moving average has been applied to the error quantities in post to produce easier visualized plots.

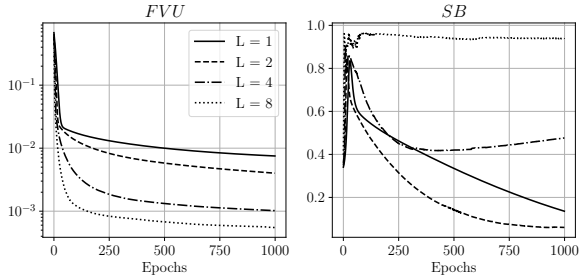

Figure 5: Experiment 4: Resulting FVU and spectral bias for NNs with varying number of layers $L = 1, 2, 4, 8$ when learning $f_2(x)$.

Experiment 4: Network Depth

Deeper neural networks have been observed to in general perform better than shallow ones in terms of overall quality of fit. With the developed definition of the spectral bias we can investigate how this improved performance is exhibited in the Fourier domain. Four ReLU networks with $L = 1, 2, 4, 8$ layers are trained on same data set as in Experiment 2. In attempting to give a fair comparison, the total number of nodes, i.e., the number of layers times the number of nodes per layer is fixed to 1024. All NNs are trained with SGD, a learning rate of 10^{-3}, and batch size of 32. The resulting FVU and spectral bias computed with Method 2 are presented in Figure 5.

Experiment 5: Density Estimation

To show that we do not require the exact density $p(x)$, we compare the computed spectral bias when using the true density $p(x)$, and when the density is approximated by KDE. This experiment uses the same neural network and data set

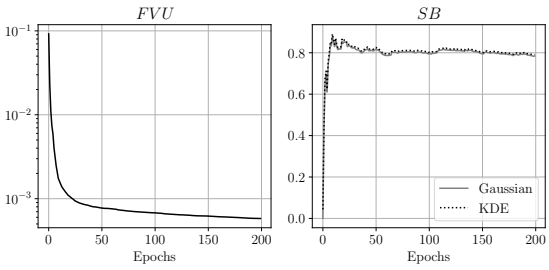

Figure 6: Experiment 5: FVU and spectral bias computed using Method 2 for a NN learning the target $f_2(x)$. The spectral bias is computed using both the true density function (Gaussian), and approximating it via kernel density estimation (KDE).

as in Experiment 2, and the NN is trained with SGD with a learning rate of 10^{-3}. The KDE uses the Gaussian Kernel with a bandwidth of $h = 1.06\sigma N^{-1/5} \approx 0.20$. Figure 6 shows the resulting spectral bias computed with Method 2.

Discussion

We have in this work given a computable definition of the spectral bias in function reconstruction problems, and two methods for estimating the proposed definition. For clarity of exposition, the data was assumed without noise. Neither the definition of spectral bias nor the computational methods presented depend on the no-noise assumption. Five experiments are performed to validate the computational methods, and to investigate the spectral bias.

The spectral bias indicates how well the model predicts low frequency components of the target function in comparison to high frequency components, with a value in $[-1, 1]$. A spectral bias of 0 indicates that the neural network predicts low and high frequencies with equal accuracy. A positive spectral bias implies that the error for the low frequencies is smaller than for the high frequencies. For example, $SB = 0.8$ means that $\mathcal{E}_{low}$ amounts to 10% of the total FVU, with $\mathcal{E}_{high}$ comprising the remaining 90%. Note that the definition of spectral bias does not directly consider the spectrum of the reconstructed function, it only depends on the relative size of the error of the high and low frequency components. All performed experiments show a positive spectral bias throughout training, i.e., $\mathcal{E}_{high}$ is the dominant term, which is in line with the results shown by previous work where it has been concluded that lower frequencies are learned first (Basri et al. 2019; Cao et al. 2021; Rahaman et al. 2019; Xu 2020, 2018; Xu, Zhang, and Xiao 2019; Xu and Zhou 2021; Luo et al. 2019).

Both proposed estimation methods focus on computing the low frequency error. Method 1 by directly approximating the Fourier transform via an FFT, and Riemann sums to estimate $\mathcal{E}_{low}$. Method 2 uses convolution with the Fourier transform of the indicator function, and Monte Carlo integration to estimate $\mathcal{E}_{low}$. Both methods have their respective strengths and weaknesses. Directly computing the FFT as in Method 1 makes the computational complexity increase exponentially with the dimension as $\mathcal{O}(N^d \log(N^d))$ for a d-dimensional cube with equal resolution $\Delta\omega$ in each direction. This makes Method 1 efficient for low dimensional problems, but unfeasible for large d. Method 1 assumes access to the target function on an equidistant grid, which is not always available. Future work may study the use of non-uniform DFTs. The integral estimation in Method 1 uses truncation and Riemann quadrature which has an error proportional to $\Delta\omega$.

Method 2 overcomes the need for an equidistant grid, and the spectral bias can be computed on the training or validation data regardless of density. It does not need to tune the sampling frequency to attain a desired accuracy, or to prevent aliasing errors. The complexity of Method 2 is $\mathcal{O}(dN^2)$, which in dimension 1 is substantially larger than for Method 1. However, because Method 2 uses Monte Carlo integration, the complexity only scales linearly with d, avoiding the curse of dimensionality observed for Method 1. The Monte Carlo integration also introduces an approximation error proportional to $N^{-1/2}$.

Both methods use the density $p(x)$ explicitly. In cases where $p(x)$ is unknown it needs to be estimated, which in general is a hard task. In this work we use KDE, which may be insufficient for complicated densities. In such cases it is a weakness that our analysis depends on estimation of p.

Qualitative illustrations of the spectral bias are given in Experiment 1 and 3, both showing that the neural network initially captures a smoothed-out version of the target function and requires many epochs before it learns the high frequency components of the target. With our definition the spectral bias can also be observed quantitatively. This is done both for synthetic experiments on the sine integral target in Experiments 2,4, and 5, and on real world data in the form of image regression in Experiment 3. Experiment 2 shows that the proposed methods of computing the spectral bias produce almost identical results even when computed on different sets of data, confirming that the two methods estimate the same quantity. In Figure 2 we see a spectral bias of $SB = 0.7$, which means that $\mathcal{E}_{high}$ contributes to 85% of the total FVU. Figure 4 shows that after 2000 epochs the spectral bias for Experiment 3 surpasses 0.8, which can be interpreted as $\mathcal{E}_{low}$ being approximately one order of magnitude smaller than $\mathcal{E}_{high}$. That is, the large spectral bias indicates that the NN is more proficient in learning low frequency components than high frequency components. A result that correlates well with Figure 3, where the details in the image need many training iterations to be resolved. Experiment 4 indicates that while the spectral bias is larger for deeper neural networks, which is in line with conclusions drawn by e.g. Xu and Zhou (2021), the ability to capture high frequency content in an absolute sense is increased with deeper networks signified by a smaller FVU for the deeper networks. Experiment 5 shows only a marginal difference when estimating the spectral bias with the true density, and when using KDE to estimate the density. We conclude that when the density is well behaved density, it is possible to estimate the spectral bias without direct access to the density.

The cutoff frequency is determined by letting the low frequency components comprise half of the variance of the target. Reducing only one out of $\mathcal{E}_{low}$ and $\mathcal{E}_{high}$ will not yield a good quality of fit. The numerical experiments show that $\mathcal{E}_{high}$ is the dominating component in every case that has been investigated. In other words, the NNs become biased to learning low frequency content. This is observed in the plots of the spectral bias, which is always greater than 0. The spectral bias need not tend to 0 in order for the NN to learn high frequencies. Indeed, in our Experiments it typically stabilizes for large epochs. A frequency is *high* if it is larger than ω_0, defined in (10) and otherwise *low*. A frequency being *high* or *low* thus depends on both the target function and on the distribution of the data.

Regularization of network weights is a technique used to prevent overfitting, but will also produce more smooth predictions. One possible use of this work could be to shed light on parameter regularization. A large spectral bias may indicate that the regularization is too strict.

Acknowledgments

This work was partially supported by the KAUST Office of Sponsored Research (OSR) under Award numbers OSR-2019-CRG8-4033.2.

References

Agustsson, E.; and Timofte, R. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1122–1131.

Basri, R.; Galun, M.; Geifman, A.; Jacobs, D.; Kasten, Y.; and Kritchman, S. 2020. Frequency Bias in Neural Networks for Input of Non-Uniform Density. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 685–694. PMLR.

Basri, R.; Jacobs, D.; Kasten, Y.; and Kritchman, S. 2019. The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Cao, Y.; Fang, Z.; Wu, Y.; Zhou, D.-X.; and Gu, Q. 2021. Towards Understanding the Spectral Bias of Deep Learning. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2205–2211. International Joint Conferences on Artificial Intelligence Organization. Main Track.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–1034.

Kammonen, A.; Kiessling, J.; Plecháč, P.; Sandberg, M.; and Szepessy, A. 2020. Adaptive random Fourier features with Metropolis sampling. *Foundations of Data Science*, 2(3): 309–332.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Luo, T.; Ma, Z.; Xu, Z.-Q. J.; and Zhang, Y. 2019. Theory of the Frequency Principle for General Deep Neural Networks. arXiv:1906.09235.

Rahaman, N.; Baratin, A.; Arpit, D.; Draxler, F.; Lin, M.; Hamprecht, F.; Bengio, Y.; and Courville, A. 2019. On the Spectral Bias of Neural Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 5301–5310. PMLR.

Scott, D. W. 1992. *Multivariate density estimation : theory, practice, and visualization*. Wiley series in probability and mathematical statistics. New York: Wiley. ISBN 0471547700.

Tancik, M.; Srinivasan, P. P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J. T.; and Ng, R. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Timofte, R.; Agustsson, E.; Van Gool, L.; Yang, M.-H.; Zhang, L.; Lim, B.; et al. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Xu, Z. J. 2018. Understanding training and generalization in deep learning by Fourier analysis. arXiv:1808.04295.

Xu, Z. J.; and Zhou, H. 2021. Deep Frequency Principle Towards Understanding Why Deeper Learning Is Faster. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35: 10541–10550.

Xu, Z.-Q. J. 2020. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. *Communications in Computational Physics*, 28(5): 1746–1767.

Xu, Z.-Q. J.; Zhang, Y.; and Xiao, Y. 2019. Training Behavior of Deep Neural Network in Frequency Domain. In Gedeon, T.; Wong, K. W.; and Lee, M., eds., *Neural Information Processing*, 264–274. Cham: Springer International Publishing. ISBN 978-3-030-36708-4.

Zhang, X.; Xiong, H.; and Wu, D. 2021. Rethink the Connections among Generalization, Memorization, and the Spectral Bias of DNNs. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 3392–3398. International Joint Conferences on Artificial Intelligence Organization. Main Track.