ON THE SPECTRAL BIAS OF NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks are known to be a class of highly expressive functions able to fit even random input-output mappings with 100% accuracy. In this work we present properties of neural networks that complement this aspect of expressivity. By using tools from Fourier analysis to study neural networks, we show that deep ReLU networks are biased towards low frequency functions, meaning that they cannot have local fluctuations without affecting their global behavior. Intuitively, this property is in line with the observation that over-parameterized networks find simple patterns that generalize across data samples. We also investigate how the shape of the data manifold affects this spectral bias by showing evidence that the learning gets *easier* with increasing manifold complexity, and present a theoretical understanding of this behavior. Finally, we study the robustness of the frequency components with respect to parameter perturbation, to develop the intuition that the parameters must be finely tuned to express high frequency functions.

1 INTRODUCTION

While the universal function approximation property of neural networks is known since the early 90s (Hornik et al., 1989; Cybenko, 1989; Leshno et al., 1993), recent research has shed light on the underlying mechanisms which endow neural networks with such expressivity (Montufar et al., 2014; Raghu et al., 2016; Poole et al., 2016). At the same time, deep neural networks, despite being massively over-parameterized, have been remarkably successful at generalizing to natural data. This fact is at odds with the traditional notions of model complexity and their empirically demonstrated ability to fit arbitrary random data to perfect accuracy (Zhang et al., 2017a; Arpit et al., 2017). Therefore, it highlights the need of asking not just *if* a neural network can express a given function, but also *how* it comes to expressing the said function.

In this work, we connect the *if* to the *how* by utilizing a potent mathematical toolbox that has seen wide application in the natural sciences and beyond – Fourier analysis[1]. The Fourier transform affords a natural way of measuring how fast a function can change within a small neighborhood in its input domain - and as such, it is a strong candidate for quantifying and analyzing the *sensitivity* of a model. Moreover, its mathematical structure makes it intimately compatible with the Jacobian, which in turn has been empirically found to be a good measure of complexity (Novak et al., 2018). The conclusions we draw from our analysis are two-fold: while neural networks can approximate arbitrary continuous functions, they exhibit a certain bias towards those that are *smooth*. We find that this bias manifests not just in the process of learning, but also in the parameterization of the model itself. At the same time, we discover that the effect of the said bias is rather intricate when the data lies on a lower dimensional manifold embedded in a higher dimensional space, as it tends to be the case for natural data.

Contributions. In section 2, we exploit the well known piecewise-linear structure of a deep ReLU network to evaluate and bound it's Fourier spectrum. In the sections that follow, we demonstrate the peculiar behaviour of neural networks in illustrative and minimal settings. In section 3, we demonstrate that lower frequencies are always regressed first, even when the higher frequencies dominate in the power spectrum of the target function. We call this phenomenon the the *spectral bias* and place it in the context of the analysis in section 2. In section 4, we find that the manifold hypothesis adds a layer of subtlety, and demonstrate how the shape of the data-manifold interacts with the spectral bias. We then go on to formalize this interaction and derive conditions on the shape

[1] See appendix A for a brief recap of Fourier analysis.

of the manifold that facilitate learning. Finally in section 5, we investigate yet another implication of our analysis, but now on the parameterization of a trained network. We find that parameters that contribute towards expressing high frequencies are less robust than their low frequency counterparts.

2 FOURIER ANALYSIS OF RELU NETWORKS

In this section, we express the family of ReLU networks in the Fourier domain. Throughout the paper, we will denote a vector with $\mathbf{a}$, its magnitude with a, its direction with $\hat{\mathbf{a}}$, and matrices as A.

2.1 PRELIMINARIES

The continous piecewise linear (CPWL) structure of ReLU networks has been extensively studied (Raghu et al., 2016; Montufar et al., 2014; Zhang et al., 2018; Arora et al., 2018a). Not only are ReLU networks CPWL, but remarkably, every CPWL function can be represented by a ReLU network (Arora et al., 2018a). In equation 1, we express a ReLU network in a way that exposes the piecewise linearity.

Consider a ReLU network f with d inputs (i.e. $\mathbf{x} \in \mathbb{R}^d$) and one output. It can be expressed as:

$$f(\mathbf{x}) = \sum_{\epsilon} 1_{P_\epsilon}(\mathbf{x}) \left(W_\epsilon \mathbf{x} + \mathbf{b}_\epsilon\right) \tag{1}$$

where ϵ is an index over the linear regions of the network, which are known to be convex polytopes P_ϵ. 1_{P_ϵ} is the indicator function on P_ϵ, and W_ϵ a $1 \times d$ matrix characterizing the *slope* of the network in the corresponding linear region (its exact form is explicitly presented in appendix B). We let the network be $L+1$ layers deep, and denote by $W^{(k)}$ the weights and $\mathbf{b}^{(k)}$ the biases of the k-th layer. $W^{(k)}$ is a $d_{k-1} \times d_k$ matrix and $\mathbf{b}^{(k)}$ is a d_k vector. We will assume that the input data lies in a bounded domain of $\mathbb{R}^d$, say $X = [-A, A]^d$ for some $A > 0$. We shall thus restrict ourselves to ReLU networks with bounded support[2].

2.2 FOURIER TRANSFORM

We want to study the structure of ReLU networks in the Fourier domain,

$$f(\mathbf{x}) = (2\pi)^{d/2} \int \tilde{f}(\mathbf{k})\, e^{i\mathbf{k}\cdot\mathbf{x}} \mathbf{dk}, \qquad \tilde{f}(\mathbf{k}) := \int f(\mathbf{x})\, e^{-i\mathbf{k}\cdot\mathbf{x}} \mathbf{dx} \tag{2}$$

where $\mathbf{dx}, \mathbf{dk}$ are the uniform Lebesgue measure on $\mathbb{R}^d$ and $\tilde{f}$ denotes the Fourier transform of f (a short recap of the Fourier transform is included in appendix A). Lemmas 1 and 2 (proved in appendix C) yield the explicit form of the Fourier components.

Lemma 1. *The Fourier transform of ReLU networks decomposes as,*

$$\tilde{f}(\mathbf{k}) = i \sum_{\epsilon} \frac{W_\epsilon \mathbf{k}}{k^2}\, \tilde{1}_{P_\epsilon}(\mathbf{k}) \tag{3}$$

where $\tilde{1}_P(\mathbf{k}) = \int_P e^{-i\mathbf{k}\cdot\mathbf{x}} \mathbf{dx}$ *is the Fourier transform of the indicator function of* $P \subset \mathbb{R}^d$.

The fourier transform of a polytope (appearing in equation 1) is a fairly intricate mathematical object; Diaz et al. (2016) develop an elegant procedure for evaluating it in arbitrary number of dimensions via a recursive application of Stokes theorem (briefly described in appendix C.2). One obtains:

Lemma 2. *The Fourier transform of a polytope P is a rational function of k of degree $\Delta^{(P)}_{\hat{\mathbf{k}}} - d$, where $\Delta^{(P)}_{\hat{\mathbf{k}}}$ is the maximum integer $0 \leq j < d$ such that $\hat{\mathbf{k}}$ is orthogonal to (the tangent space of) a j-dimensional face of P.*

[2]Note that there is no loss of generality here: given any ReLU network f, its restriction $f_{|X}$ to X can always be extended to a continuous piece-wise linear function on $\mathbb{R}^d$ with bounded support; this function, in turn, is representable by a ReLU network. This argument also shows that ReLU networks with bounded support are universal approximators on X.

Note that, given a face of P that is not a point, the set of vectors orthogonal to its tangent face is a strict subspace of $\mathbb{R}^d$ with zero Lebesgue measure. Hence $\Delta_{\hat{\mathbf{k}}}^{(P)} = 0$ for almost every $\mathbf{k}$ in $\mathbb{R}^d$. Finally, we have the following bound for the Lipschitz constant of a deep ReLU network:

Lemma 3. *The Lipschitz constant L_f of the ReLU network f is bound as follows (for all ϵ):*

$$\|W_\epsilon\| \leq L_f \leq \prod_{k=1}^{L+1} \|W^{(k)}\| \leq \|\theta\|_\infty^{L+1} \sqrt{d} \prod_{k=1}^{L} d_k \tag{4}$$

where $\|\cdot\|$ is the spectral norm, θ is the ravelled parameter vector of the network and $\|\cdot\|_\infty$ is the max-norm.

Lemmas 1, 2 and 3 taken together yield the main result of this section. Given a ReLU network f and the polytopes P_ϵ defined by its linear regions, we denote by $\mathcal{F}$ the set of faces of all such polytopes. For $\mathbf{k} \in \mathbb{R}^d$, let $\Delta_{\hat{\mathbf{k}}}$ be the maximum integer $0 \leq j < d$ such that $\hat{\mathbf{k}}$ is orthogonal to the tangent space of some j-dimensional face $F_j \in \mathcal{F}$:

$$\Delta_{\hat{\mathbf{k}}} = \max\{0 \leq j < d \mid \hat{\mathbf{k}} \perp F_j,\, F_j \in \mathcal{F}, \dim F_j = j\} \tag{5}$$

Theorem 1. *The Fourier components of the ReLU network f admit the following bound:*

$$|\tilde{f}(\mathbf{k})| \leq C \frac{N_f L_f}{k^{d-\Delta_{\hat{\mathbf{k}}}+1}} \tag{6}$$

where N_f is the number of linear regions (polytopes $P_{f,\epsilon}$), $L_f = \max_\epsilon \|W_\epsilon\|_2$ is the Lipschitz constant, and C is a universal constant.

Several remarks are in order:

(a) The spectral decay of ReLU networks is highly anisotropic in large dimension. We have $\Delta_k = 0$, hence a k^{-d-1} decay, in almost all directions $\hat{\mathbf{k}}$ of $\mathbb{R}^d$. It can decay as slow as k^{-2} in specific directions orthogonal to the facets bounding linear regions[3].

(b) Montufar et al. (2014) and Raghu et al. (2016) obtain tight bounds for the number N_f of linear regions, which also scale as w^{dL} for a network with L layers of width w (cf. theorem 1 of Raghu et al. (2016)). This makes the overall bound – and with it the ability to express larger frequencies – scale exponentially in depth and polynomial in width (see Lemma 3). This result complements the well known universal approximation property[4] by explicitly incorporating a control on the capacity[5] of the network, namely the width, depth and the norm of parameters.

(c) For a given architecture (i.e. fixed width and depth), the high frequency contributions of the network can be increased by increasing the norm of the parameters. Assuming the weight norm increases with iteration time in the learning process, that suggests that the training of ReLu networks is biased towards low frequencies. We investigate this fact empirically in the next section.

3 Lower Frequencies are Learned First

In this section and the following ones, we present experiments that illustrate the peculiar behaviour of deep ReLU networks in the Fourier domain. With the following first experiment, the goal is to demonstrate that networks tend to fit *lower frequencies first* during training, a phenomenon we refer to as the *spectral bias*.

Experiment 1. The setup is as follows[6] : Given frequencies $\kappa = (k_1, k_2, ...)$ with corresponding amplitudes $\alpha = (A_1, A_2, ...)$, and phases $\phi = (\varphi_1, \varphi_2, ...)$, we consider the mapping $\lambda : [0, 1] \to \mathbb{R}$ given by

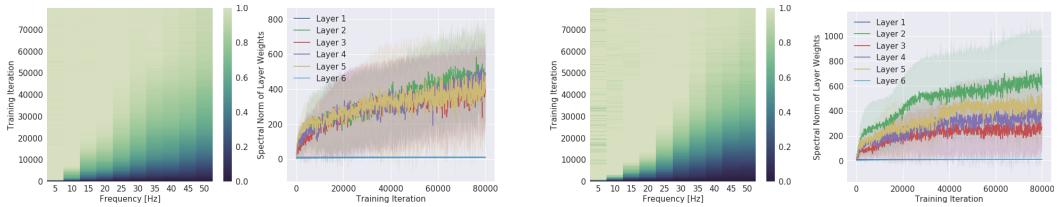
$$\lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i). \tag{7}$$

[3]Note that such a rate is *not* guaranteed by piecewise smoothness alone. For instance, the function $\sqrt{|x|}$ is continuous and smooth everywhere except at $x = 0$, yet it decays as $k^{-1.5}$ in the Fourier domain.

[4]The universal approximation property of MLPs with non-polynomial activations is known since the 90s, see e.g. Leshno et al. (1993).

[5]Note that the bound is relaxed with increasing capacity. In the limit of capacity to infinity, any constraint on universal approximation is dissolved.

[6]More experimental details and additional plots are provided in Appendix D.1.



(a) Equal Amplitudes (b) Increasing Amplitudes

Figure 1: Left (a, b): Plot showing the evolution of the spectrum (x-axis for frequency) during training (y-axis). The colors show the measured amplitude of the network spectrum at the corresponding frequency, normalized by the target amplitude at the same frequency (i.e. $|\tilde{f}_{k_i}|/A_i$) and the colorbar is clipped between 0 and 1. Right (a, b): Evolution of the spectral norm (y-axis) of each layer during training (x-axis). Figure-set (a) shows the setting where all frequency components in the target function have the same amplitude, and (b) where higher frequencies have larger amplitudes. **Gist**: We find that even when higher frequencies have larger amplitudes, the model prioritizes learning lower frequencies first. We also find that the spectral norm of weights increases as the model fits higher frequency, which is what we expect from theorem 1.

A 6-layer deep 256-unit wide ReLU network f_θ is trained to regress λ with $\kappa = (5, 10, ..., 45, 50)$ and $N = 200$ input samples spaced equally over $[0, 1]$; its spectrum $\tilde{f}_\theta(k)$ in expectation over $\varphi_i \sim U(0, 2\pi)$ is monitored as training progresses. In the first setting, we set equal amplitude $A_i = 1$ for all frequencies and in the second setting, the amplitude increases from $A_1 = 0.1$ to $A_{10} = 1$. Fig 1 shows the normalized magnitudes $|\tilde{f}_\theta(k_i)|/A_i$ at various frequencies, as training progresses. The result is that lower frequencies (i.e. smaller k_i's) are regressed first, regardless of their amplitudes.

The interpretation is as follows: for a fixed architecture, the bound in Theorem 1 allows for larger fourier coefficients at higher frequencies if the parameter norm is large. However, the parameter norm can increase only gradually during training by gradient descent, which leads to the higher frequencies being learned late in the optimization process. To confirm that the bound indeed increases as the model fits higher frequencies, we plot in Fig 1 the spectral norm of weights of each layer during training for both cases of constant and increasing amplitudes.

We stress that this bias does not originate from the Mean Squared Error. This can be seen by writing the mean squared training error $\text{MSE}[f_\theta, \lambda]$ in terms of the Fourier components: letting $z_i = i/N$ be the training sample points,

$$\text{MSE}[f_\theta, \lambda] = \frac{1}{N} \sum_{i=0}^{N-1} |f_\theta(z_i) - \lambda(z_i)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |\tilde{f}_\theta(k) - \tilde{\lambda}(k)|^2 = \text{MSE}[\tilde{f}_\theta, \tilde{\lambda}] \tag{8}$$

where the second equality follows from Plancherel theorem. This shows that square error in input space translates into square error in Fourier domain, with no bias towards any particular frequency component (note that the finite range in frequencies is due to sampling).

4 Not all Manifolds are Learned Equal

In this section, we investigate the subtleties that arise when the data lies on a lower dimensional manifold embedded in the higher dimensional input space of the model (Goodfellow et al., 2016). To systematically investigate how the *shape* of such data-manifolds interacts with the spectral bias, we first demonstrate results in an illustrative minimal setting[7]. Finally, we present a mathematical exposition coupling the Fourier spectrum of the network with that of the target function defined on the manifold and geometry of the manifold itself, and in the process justify our choice of the experimental setting.

The experimental setting is designed to afford control over both the shape of the data manifold and the target function defined on it. We will consider the family of curves in $\mathbb{R}^2$ generated by mappings

[7]We include an experiments on MNIST and CIFAR-10 in appendices D.3 and D.4.

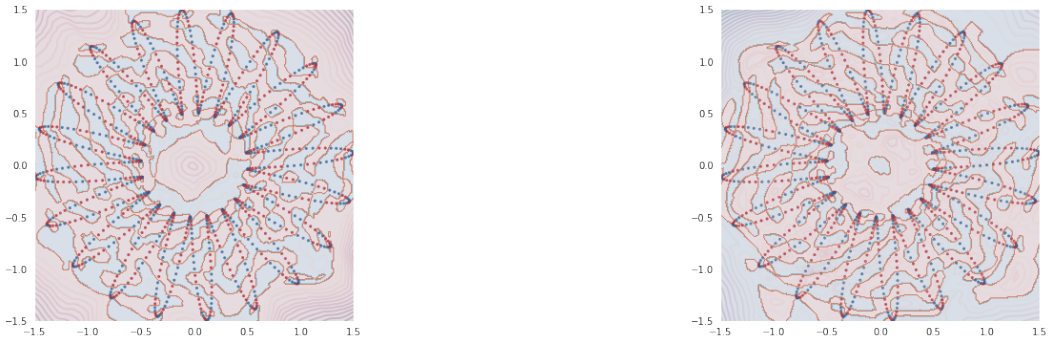


Figure 2: Functions learned by two identical networks (identical upto initialization) to classify the binarized value of a sine wave of frequency $k = 200$ defined on a $\gamma_{L=20}$ manifold. Both functions yield close to perfect accuracy for the samples defined on the manifold (scatter plot), yet they differ significantly elsewhere. The shaded regions show the predicted class (Red or Blue) whereas contours show the confidence (absolute value of logits).

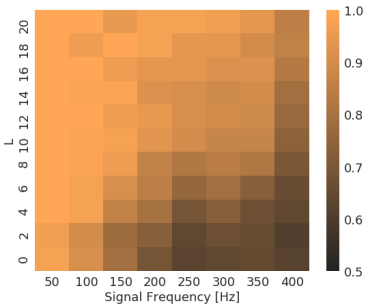


Figure 3: Heatmap of training accuracies of a network trained to predict the binarized value of a sine wave of given frequency (x-axis) defined on γ_L for various L (y-axis).

$\gamma_L : [0, 1] \to \mathbb{R}^2$ given by[8]

$$\gamma_L(z) = R_L(z)(\cos(2\pi z), \sin(2\pi z)) \text{ where } R_L(z) = 1 + \frac{1}{2}\sin(2\pi L z) \tag{9}$$

Here, $\gamma_L([0, 1])$ defines the data-manifold and corresponds to a flower-shaped curve with L petals, or a unit circle when $L = 0$ (see e.g. Fig 2). Given a signal $\lambda : [0, 1] \to \mathbb{R}$ defined on the latent space $[0, 1]$, the task entails learning a network $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f \circ \gamma_L$ matches the signal λ.

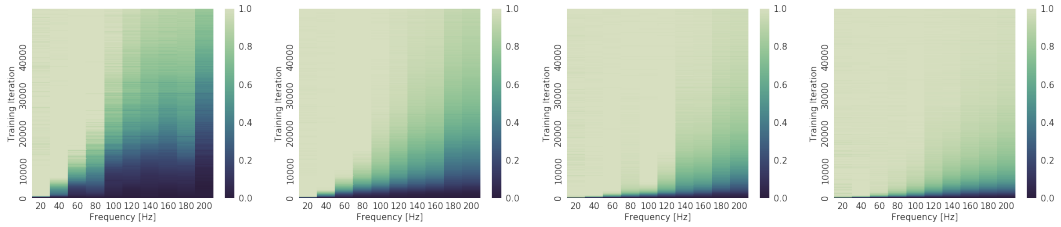
Experiment 2. The set-up is similar to that of Experiment 1, and λ is as defined in Eqn. 7 with frequencies $\kappa = (20, 40, ..., 180, 200)$, and amplitudes $A_i = 1 \, \forall \, i$. The model f is trained on the dataset $\{\gamma_L(z_i), \lambda(z_i)\}_{i=1}^N$ with $N = 1000$ uniformly spaced samples z_i between 0 and 1. The spectrum of $f \circ \gamma_L$ in expectation over $\varphi_i \sim U(0, 2\pi)$ is monitored as training progresses, and the result shown in Fig 4 for $L = 0, 4, 10, 16$. Fig 5 shows the corresponding mean squared error curves. More experimental details in appendix D.2.

The results demonstrate a clear attenuation of the spectral bias as L grows. Moreover, Fig 5 suggests that the larger the L, the easier the learning task.

Experiment 3. Here we adapt the setting of Experiment 2 to binary classification by simply thresholding the function λ at 0.5. To simplify visualization, we only use signals with a single frequency mode k, such that $\lambda(z) = \sin(2\pi k z + \varphi)$. We train the same network on the resulting classification task for $k \in \{50, 100, ..., 350, 400$ and $L \in \{0, 2, ..., 18, 20\}$. The heatmap in Fig 3 shows the classification accuracy for each (k, L) pair. Fig 2 shows visualizations of the functions learned by the same network, trained on $(k, L) = (200, 20)$ under identical conditions up to random initialization.

The results confirm the behaviour observed in Experiment 2, but in the case of classification tasks with categorical cross-entropy loss. Increasing L effectively reduces the spectral bias of the network.

[8]The interval $[0, 1]$ can be thought of as the latent space of a generative process characterized by γ_L.



(a) $L = 0$ (b) $L = 4$ (c) $L = 10$ (d) $L = 16$

Figure 4: Plots showing the evolution of the network spectrum (x-axis for frequency, colorbar for magnitude) during training (y-axis) for the same target functions defined on manifolds γ_L for various L. Since the target function has amplitudes $A_i = 1$ for all frequencies k_i plotted, the colorbar is clipped between 0 and 1. See more details in experiment 2. **Gist**: Some manifolds (here with larger L) make it easier for the network to learn higher frequencies than others.

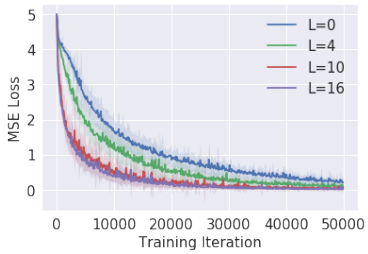


Figure 5: Loss curves for the model trained on target defined on γ_L for various L's.

Discussion. These experiments hint at a rich interaction between the shape of the manifold and the effective difficulty of the learning task. The key technical reason underlying this phenomenon (formalized later in the section) is that the relationship between frequency spectrum of the network f that of the fit $f \circ \gamma_L$ is mediated by the embedding map γ_L. In particular, we will argue that a given signal defined on the manifold is easier to fit when the coordinate functions of the manifold embedding itself has high frequency components. Thus, in our experimental setting, the same signal embedded in a flower with more petals can be captured with lower frequencies of the network.

In our experimental setting where we assumed explicit control over the embedding γ, we had defined the target function λ directly on the latent space $[0, 1]$. In doing so we modeled the manifold hypothesis, which states that the data lies on a lower dimensional manifold in a higher dimensional space, and that only a small number of latent factors contain all the relevant information about the data. This can be translated to a degeneracy of the data probability distribution μ in the input space induced under the mapping γ by (say) a uniform distribution in the latent space. In this case, the standard regression problem $\min_f \mathbb{E}_\mu |f - \tau|^2$ becomes minimizing the expected square error for some $\tau : \mathbb{R}^d \to \mathbb{R}$ defined on the input space such that $\tau \circ \gamma = \lambda$. But the fact that the support of μ lies on a lower dimensional data manifold (i.e. the image of $[0, 1]$ under γ) leads to a vast space Λ of degenerate functions $\tau \in \Lambda$ which can be identified with λ when restricted to the data-manifold (see Fig 2). This connects with the spectral bias we found in the previous section, in the sense that neural networks are biased towards expressing a particular subset of solutions in Λ, namely those that are low frequency. It is also worth noting that there exist methods that restrict the space of functions Λ: notably adversarial training (Goodfellow et al., 2014) and Mixup (Zhang et al., 2017b).

In the following, we formally relate the frequency spectrum of f with that of $f \circ \gamma$ to find that the lowest frequency a solution in Λ can have depends intimately with the shape of the manifold (i.e. the support of μ). We start by formulating the expected loss for an arbitrary $\tau \in \Lambda$. Recall that μ is induced by a uniform distribution in the latent space $[0, 1]^m$:

$$\mathbb{E}_\mu |f - \tau|^2 = \int_{[0,1]^m} \mathbf{dz} |f(\gamma(\mathbf{z})) - \lambda(\mathbf{z}))|^2 = \int \mathbf{dl} |\widetilde{(f \circ \gamma)}(\mathbf{l}) - \tilde{\lambda}(\mathbf{l}))|^2 \tag{10}$$

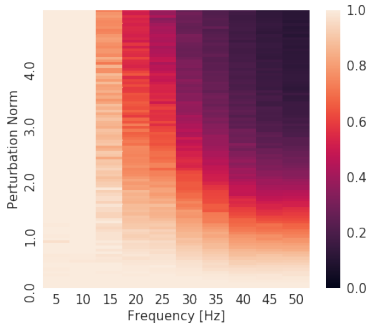


Figure 6: Plot showing the normalized spectrum of the model (x-axis for frequency, colorbar for magnitude) with perturbed parameters as a function of parameter perturbation (y-axis). The colormap is clipped between 0 and 1. Observe that the lower frequencies are more robust to parameter perturbations than the higher frequencies.

where the second equality arises from Parceval's theorem. Now the Fourier transform of the function composition can be expressed as:

$$\widetilde{(f \circ \gamma)}(\mathbf{l}) = \int \mathbf{dk} \tilde{f}(\mathbf{k}) P_\gamma(\mathbf{l}, \mathbf{k}) \quad \text{where} \quad P_\gamma(\mathbf{l}, \mathbf{k}) = \int_{[0.1]^m} \mathbf{dz}\, e^{i(\mathbf{k}\cdot\gamma(\mathbf{z}) - \mathbf{l}\cdot\mathbf{z})} \tag{11}$$

The kernel P_γ depends on only γ and elegantly encodes the correspondence between frequencies $\mathbf{k} \in \mathbb{R}^d$ in input space and frequencies $\mathbf{l} \in \mathbb{R}^m$ in the latent space $[0, 1]^m$. Following the procedure from Bergner et al., we can further investigate the behaviour of the kernel in the regime where the stationary phase approximation is applicable, i.e. when $l^2 + k^2 \to \infty$ (cf. section 3.2. of Bergner et al.). In this regime, the integral P_γ is dominated by critical points $\bar{\mathbf{z}}$ of its phase, which satisfy

$$\mathbf{l} = J_\gamma(\bar{\mathbf{z}})\, \mathbf{k} \tag{12}$$

where $J_\gamma(\mathbf{z})_{ij} = \nabla_i \gamma_j(\mathbf{z})$ is the $m \times d$ Jacobian matrix of γ. Non-zero values of the kernel correspond to pairs $(\mathbf{l}, \mathbf{k})$ such that Eqn 12 has a solution. Further, given that the components of γ (i.e. its coordinate functions) are defined on an interval (namely $[0, 1]^m$), one can use their Fourier series representation together with Eqn 12 to obtain a condition on their frequencies (shown in appendix E). More precisely, we find that the i-th component of the RHS in Eqn 12 is proportional to $\mathbf{p}\tilde{\gamma}_i[\mathbf{p}]k_i$ where $\mathbf{p} \in \mathbb{Z}^m$ is the frequency of the coordinate function γ_i. This yields that we can get arbitrarily large frequencies l_i if $\tilde{\gamma}_i[\mathbf{p}]$ is large[9] enough for large $\mathbf{p}$, even when k_i is fixed.

This is precisely what experiments 2 and 3 demonstrate in a minimal setting. From Eqn 9, observe that the coordinate functions have a frequency mode at L. For increasing L, it is apparent that the frequency magnitudes l (in the latent space) that can be expressed with the same frequency k (in the input space) increases with increasing L. This allows the remarkable interpretation that the neural network function can express large frequencies on a manifold (l) with smaller frequencies w.r.t its input domain (k), provided that the coordinate functions of the data manifold embedding itself has high-frequency components[10].

5 Lower frequencies are more robust

The goal of this section is to show that lower frequencies of trained networks are more *robust* than larger ones, with respect to perturbations in parameter space. More precisely, we observe that in the neighbourhood of a solution in parameter space, the large frequency components decay faster than the low frequency ones. This property does not directly depend on the training process, but rather on the parametrization of the trained model. We present empirical evidence and a theoretical explanation of the phenomenon.

Experiment 4. The set up is the same as in Experiment 1, where λ is given by Eqn. 7. Training is performed for the frequencies $\kappa = (10, 15, 20, ..., 45, 50)$ and amplitudes $A_i = 1 \,\forall\, i$. After

[9] Consider that the data-domain is bounded, implying that $\tilde{\gamma}$ cannot be arbitrarily scaled.

[10] Informally, we allow ourselves the intuition that *it's easy for neural networks to fit wiggly functions if they are defined on wiggly manifolds*.

convergence to θ^*, we consider random (isotropic) perturbations $\theta = \theta^* + \delta\hat{\theta}$ of given magnitude δ, where $\hat{\theta} \sim U(\mathcal{S}^{\dim(\theta^*)})$ is a unit vector. We evaluate the network function f_θ at the perturbed parameters, and compute the magnitude of its discrete Fourier transform at frequencies k_i, $|\tilde{f}_\theta(k_i)|$. We also average over 100 samples of $\hat{\theta}$ to obtain $|\tilde{f}_{\mathbb{E}\theta}(k_i)|$, which we normalize by $|\tilde{f}_{\theta*}(k_i)|$. The result, shown in Figure 6, demonstrate that higher frequencies are significantly less robust than the lower ones.

The interpretation is as follows: parameters that contribute towards expressing high-frequency components occupy a small volume in the parameter space. To formalize this intuition, given a bounded domain Θ of parameter space, let us define,

$$\Xi_\epsilon(k) = \{\theta \in \Theta | \exists \mathbf{k}', k' > k, \, |\tilde{f}_\theta(\mathbf{k}')| > \epsilon\}$$

to be the set of parameters such that f_θ has Fourier components larger than ϵ for some $\mathbf{k}'$ with larger norm than k. Then the following Proposition holds (proved in appendix F).

Proposition 1. *The volume ratio,*

$$R(k) = \frac{Vol(\Xi_\epsilon(k))}{Vol(\Theta)} \tag{13}$$

inherits the spectral decay rate of $|\tilde{f}_\theta(\mathbf{k})|$*, given by Theorem 1.*

Intuitively, expressing larger frequencies requires the parameters to be finely-tuned to work together.

6 Related Work

Recently a number of papers derive modern generalization bounds (Dziugaite & Roy, 2017; Neyshabur et al., 2017; Achille & Soatto, 2017; Smith & Le, 2017; Arora et al., 2018b) for deep networks that are non-vacuous in certain cases compared with the traditional ones (Eg. VC-dimensions Vapnik (2013), Rademacher complexity Bartlett et al. (2005)). A main novel contribution of our work that complements the aforementioned line of work is that we establish the implicit bias of ReLU networks towards learning functions that suppress local irregularities over the input space, thus representing globally smooth functions. While on the other hand generalization bounds provide conditions under which the generalization gap will be small, but do not reveal such a bias. Interestingly, a commonality between our work and most of these modern generalization bounds is that of the role of parameter norm. For many of these bounds, a larger parameter norm makes the generalization gap loose, and in a similar spirit, our results show that a larger norm allows the network to represent more complex functions. On the analysis side, Sonoda & Murata (2017) reprove Leshno et al.'s with ridgelet transforms, and Candès (1999) use harmonic analysis to express neural networks with oscillatory activation as a superposition of ridge functions.

In light of our findings, we also contrast between kernel machines (KM) and Knearest neighbor classifiers in appendix G. In summary, our discussion suggests that 1. DNNs strike a good balance between function smoothness and expressivity/parameter-efficiency compared with KM; 2. DNNs learn a smoother function compared with KNNs since the spectrum of the DNN decays faster compared with KNNs in the experiments shown there.

7 Conclusion

We expressed deep ReLU networks in the Fourier domain, and used our analysis to explain a peculiar behaviour of such networks – that they tend to learn lower frequencies first. We called this behaviour the *spectral bias* and exposed the subtleties that arise when the training data lies on a manifold. Finally, we found that the parameters contributing towards expressing lower frequencies are more robust to random perturbations than their higher frequency counterparts.

References

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentangling in deep representations. *arXiv preprint arXiv:1706.01350*, 2017.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=B1J_rgWRW.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018b.

Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Peter L Bartlett, Olivier Bousquet, Shahar Mendelson, et al. Local rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Steven Bergner, Torsten Möller, Daniel Weiskopf, and David J Muraki. A spectral analysis of function concatenations and its implications for sampling in direct volume visualization.

Emmanuel J Candès. Harmonic analysis of neural networks. *Applied and Computational Harmonic Analysis*, 6(2):197–218, 1999.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

Luc Devroye, László Györfi, and Gábor Lugosi. Consistency of the k-nearest neighbor rule. In *A Probabilistic Theory of Pattern Recognition*, pp. 169–185. Springer, 1996.

Ricardo Diaz, Quang-Nhat Le, and Sinai Robins. Fourier transforms of polytopes, solid angle sums, and discrete volume. *arXiv preprint arXiv:1602.08593*, 2016.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2018.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Barbara Hammer and Kai Gersmann. A note on the universal approximation capability of support vector machines. *Neural Processing Letters*, 17(1):43–53, 2003.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Esben L Kolsbjerg, Michael N Groves, and Bjørk Hammer. An automated nudged elastic band method. *The Journal of chemical physics*, 145(9):094107, 2016.

Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Siyuan Ma and Mikhail Belkin. Diving into the shallows: a computational perspective on large-scale shallow learning. In *Advances in Neural Information Processing Systems*, pp. 3781–3790, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5949–5958, 2017.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJC2SzZCW.

Ben Poole, Subhaneil Lahiri, Maithreyi Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3360–3368. Curran Associates, Inc., 2016.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

Samuel L Smith and Quoc V Le. Understanding generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations (ICLR)*, 2017a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017b.

Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. *arXiv preprint arXiv:1805.07091*, 2018.

A Brief Recapitulation of Fourier Analysis

The Fourier transform is a powerful mathematical tool used to represent functions as a weighted sum of oscillating functions, given that the function satisfies certain conditions. In the realm of signal processing and beyond, it is used to represent a time (space) domain signal f as a sum of sinusoids of various (spatial) frequencies $\mathbf{k}$, where the weights are referred to as the Fourier coefficients $\tilde{f}(\mathbf{k})$.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a squared-integrable function[11], i.e. $\int_{\mathbf{x}\in\mathbb{R}^n} |f(\mathbf{x})|^2 d^n\mathbf{x}$ is finite, or $f \in L^2(\mathbb{R}^n)$. With the Fourier inversion theorem, it holds:

$$f(\mathbf{x}) = \frac{1}{2\pi}\int_{\mathbf{k}\in\mathbb{R}^n} \tilde{f}(\mathbf{k}) e^{i\mathbf{k}\cdot\mathbf{x}} d\mathbf{k} \qquad (14) \qquad\qquad \tilde{f}(\mathbf{k}) = \int_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}} d\mathbf{x} \qquad (15)$$

[11] On a formal note, the squared integrability is only required for the inverse Fourier transform to exist; for the forward transform, integrability is enough. Moreover, the Fourier transform can be generalized to tempered distributions, which allow for evaluating the fourier coefficients of non-integrable e.g. non-zero constant functions.

Informally, equation 14 expresses the function $f(\mathbf{x})$ as a weighted sum (the integral) of plane waves $e^{\pm i\mathbf{k}\cdot\mathbf{x}}$ of the angular wavenumber $\mathbf{k}$, where the unit vector $\hat{\mathbf{k}}$ gives the direction of the corresponding wave in n-D space and the magnitude k is inversely proportional to the wavelength. Equation 15 gives the expression for $\tilde{f}(\mathbf{k})$, which is called the Fourier transform or the *Fourier spectrum* or simply the spectrum of f. The $\frac{1}{2\pi}$ coefficient and sign in the exponential functions are matters of convention.

The asymptotic behaviour of $\tilde{f}$ for $k \to \infty$ is a measure of smoothness of f. In Bachmann-Landau or *asymptotic* notation[12], we say $\tilde{f} = \mathcal{O}(k^{-1})$ if for $k \to \infty$, the function $\tilde{f}$ decays at least as fast as $\frac{1}{k}$. A function whose spectrum is $\mathcal{O}(k^{-2})$ is in a sense smoother than one whose spectrum is $\mathcal{O}(k^{-1})$, while the spectrum of an infinitely differentiable (or *smooth*) function must decay faster than any rational function of k, assuming the function is integrable, i.e. the integral of its absolute value over its domain is finite (or the function is L^1). Intuitively, the higher-frequency oscillations in a smoother function must vanish faster. Formally, this is a straightforward consequence of the Riemann-Lebesgue lemma, stating that spectrum of any L^1 function must vanish at infinity (potentially arbitrarily slowly), taken together with the well known property of the Fourier transform that it diagonalizes the differential operator i.e. $[\widetilde{\nabla_{\mathbf{x}} f}](\mathbf{k}) = \mathbf{k}\tilde{f}(\mathbf{k})$.

B THE CONTINUOUS PIECEWISE LINEAR STRUCTURE OF DEEP RELU NETWORKS

We consider the class of scalar functions $f : \mathbb{R}^d \mapsto \mathbb{R}$ defined by a neural network with L hidden layers of widths $d_1, \cdots d_L$ and a single output neuron, defined by:

$$f(\mathbf{x}) = \left(T^{(L+1)} \circ \sigma \circ T^{(L)} \circ \cdots \circ \sigma \circ T^{(1)}\right)(\mathbf{x}) \tag{16}$$

where each $T_k : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ is an affine function ($d_0 = d$ and $d_{L+1} = 1$) and $\sigma = \max(0, u_i)$ denotes the ReLU activation function acting elementwise on a vector $\mathbf{u} = (u_1, \cdots u_n)$.

ReLU networks are known to be continuous piece-wise linear (CPWL) functions, where the linear regions are convex polytopes (Raghu et al., 2016; Zhang et al., 2018). To expose the piecewise linear structure more explicitly, we associate a binary activation variable $\epsilon \in \{-1, 1\}$ to each hidden neuron, conditioned on whether its input is positive or negative. Each linear region of the network then corresponds to a unique *activation pattern*[13], wherein each neuron is assigned an activation variable. With that, the ReLU network from Eqn. 16 can be explictly expressed as a sum over all possible activation patterns, as in the following lemma.

Lemma 4. *Given L binary vectors $\epsilon_1, \cdots \epsilon_L$ with $\epsilon_k \in \{-1, 1\}^{d_k}$, let $T^{(k)}_{\epsilon_k} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ the affine function defined by $T^{(k)}_{\epsilon_k}(\mathbf{u})_i = (T^{(k)}(\mathbf{u}))_i$ if $(\epsilon_k)_i = 1$, and 0 otherwise. ReLU network functions, defined in Eqn. 16, can be expressed as*

$$f(\mathbf{x}) = \sum_{\epsilon_1, \cdots \epsilon_L} 1_{P_{f,\epsilon}}(\mathbf{x}) \left(T^{(L+1)} \circ T^{(L)}_{\epsilon_L} \circ \cdots \circ T^{(1)}_{\epsilon_1}\right)(\mathbf{x}) \tag{17}$$

where 1_P denotes the indicator function of the subset $P \subset \mathbb{R}^d$, and $P_{f,\epsilon}$ is the polytope defined as the set of solutions of the following linear inequalities (for all $k = 1, \cdots, L$):

$$(\epsilon_k)_i \, (T^{(k)} \circ T^{(k-1)}_{\epsilon_{k-1}} \circ \cdots \circ T^{(1)}_{\epsilon_1})(\mathbf{x})_i \geq 0, \quad i = 1, \cdots d_k \tag{18}$$

f is therefore affine on the polytopes $P_{f,\epsilon}$, which finitely partition the input space $\mathbb{R}^d$ to convex polytopes. But remarkably, the correspondence between ReLU networks and CPWL functions goes both ways: Arora et al. (2018a) show that every CPWL function is be represented by a ReLU network, which in turn endows ReLU networks with the universal approximation property.

Finally, in the standard basis, each affine map $T^{(k)} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ is specified by a weight matrix $W^{(k)} \in \mathbb{R}^{d_{k-1}} \times \mathbb{R}^{d_k}$ and a bias vector $b^{(k)} \in \mathbb{R}^{d_k}$. In the linear region $P_{f,\epsilon}$, f can be expressed as

[12] Formally, $f = \mathcal{O}(g) \implies \limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$.

[13] We adopt the terminology of Raghu et al. (2016); Montufar et al. (2014).

$f_\epsilon(x) = W_\epsilon x + b_\epsilon$, where in particular

$$W_\epsilon = W^{(L+1)} W^{(L)}_{\epsilon_L} \cdots W^{(1)}_{\epsilon_1} \in \mathbb{R}^{1\times d}, \tag{19}$$

where $W^{(k)}_\epsilon$ is obtained from $W^{(k)}$ by setting its jth column to zero whenever $(\epsilon_k)_j = -1$. In what follows, we will assume that the input data lies in a bounded domain of $\mathbb{R}^d$, say $X = [-A, A]^d$ for some $A > 0$. We will thus restrict to ReLU networks with bounded support. Note that there is no loss of generality here: given any ReLU network f, its restriction $f_{|X}$ to X can always be extended to a continuous piece-wise linear function on $\mathbb{R}^d$ with bounded support; this function, in turn, is representable by a ReLU network. This argument also shows that ReLU networks with bounded support are universal approximators on X.

C Fourier Analysis of ReLU Networks

C.1 Proof of Lemma 1

Proof. The vector-valued function $\mathbf{k} f(\mathbf{x}) e^{i\mathbf{k}\cdot\mathbf{x}}$ is continuous everywhere and has well-defined and continuous gradients almost everywhere. So by Stokes' theorem [cite], the integral of its divergence is a pure boundary term. Since f vanishes outside its bounded support, the theorem yields

$$\int \nabla_{\mathbf{x}} \cdot \left[\mathbf{k} f(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}}\right] \mathbf{dx} = 0 \tag{20}$$

The integrand is $(\mathbf{k} \cdot (\nabla_{\mathbf{x}} f)(\mathbf{x}) - ik^2 f(\mathbf{x})) e^{-i\mathbf{k}\cdot\mathbf{x}}$, so we deduce,

$$\hat{f}(\mathbf{k}) = \frac{1}{-ik^2} \mathbf{k} \cdot \int (\nabla_{\mathbf{x}} f)(\mathbf{x})\, e^{-i\mathbf{k}\cdot\mathbf{x}} \tag{21}$$

Now, within each polytope of the decomposition (17), f is affine so its gradient is a constant vector, $\nabla_{\mathbf{x}} f_\epsilon = W_\epsilon^T$, which gives the desired result (1). □

C.2 Fourier Transform of Polytopes

C.2.1 Theorem 1 of Diaz et al. (2016)

Let F be a m dimensional polytope in $\mathbb{R}^n$, such that $1 \le m \le n$. Denote by $\mathbf{k} \in \mathbb{R}^n$ a vector in the fourier space, by $\phi_{\mathbf{k}}(x) = -\mathbf{k}\cdot\mathbf{x}$ the linear phase function, by $\tilde{F}$ the fourier transform of the indicator function on F, by ∂F the boundary of F and by vol_m the m-dimensional (Hausdorff) measure. Let $\text{Proj}_F(\mathbf{k})$ be the orthogonal projection of $\mathbf{k}$ on to F (obtained by removing all components of $\mathbf{k}$ orthogonal to F). Given a $m-1$ dimensional facet G of F, let $\mathbf{N}_F(G)$ be the unit normal vector to G that points out of F. It then holds:

1. If $\text{Proj}_F(\mathbf{k}) = 0$, then $\phi_{\mathbf{k}}(x) = \Phi_{\mathbf{k}}$ is constant on F, and we have:

$$\tilde{F} = \text{vol}_F(F) e^{i\Phi_{\mathbf{k}}}$$

2. But if $\text{Proj}_F(\mathbf{k}) \neq 0$, then:

$$\tilde{F} = i \sum_{G\in\partial F} \frac{\text{Proj}_F(\mathbf{k}) \cdot \mathbf{N}_F(G)}{\|\text{Proj}_F(\mathbf{k})\|^2} \tilde{G}(\mathbf{k})$$

C.2.2 Discussion

The above theorem provides a recursive relation for computing the fourier transform of an arbitrary polytope. More precisely, the Fourier transform of a m-dimensional polytope is expressed as a sum of fourier transforms over the $m-1$ dimensional boundaries of the said polytope (which are themselves polytopes) times a $\mathcal{O}(k^{-1})$ *weight* term (with $k = \|\mathbf{k}\|$). The recursion terminates if $\text{Proj}_F(\mathbf{k}) = 0$, which then yields a constant.

To structure this computation, Diaz et al. (2016) introduce a book-keeping device called the *face poset* of the polytope. It can be understood as a weighted tree diagram with polytopes of various dimensions as its nodes. We start at the root node which is the full dimensional polytope P (i.e. we initially set $m = n$). For all of the codimension-one boundary faces F of P, we then draw an edge from the root P to node F and weight it with a term given by:

$$W_{F,G} = i\frac{\text{Proj}_F(\mathbf{k}) \cdot \mathbf{N}_F(G)}{\|\text{Proj}_F(\mathbf{k})\|^2}\tilde{G}(\mathbf{k})$$

and repeat the process iteratively for each F. Note that the weight term is $\mathcal{O}(k^{-1})$ where $\text{Proj}_F(\mathbf{k}) \neq 0$. This process yields tree paths $P \to F_1 \to ... \to F_q$ where each F_{i+1} has one dimension less than F_i. For a given path and $\mathbf{k}$, the terminal node for this path, F_q, is the first polytope for which $\text{Proj}_{F_q}(\mathbf{k}) = 0$. The final fourier transform is obtained by multiplying the weights along each path and summing over all paths.

It is then shown that for a generic vector $\mathbf{k}$, all paths terminate at the zero-dimensional vertices of the original polytope, i.e. $\dim(F_q) = 0$, implying the length of the path q equals the number of dimensions n, yielding a $\mathcal{O}(k^{-n})$ spectrum. The exceptions occur if a path terminates prematurely, because $\mathbf{k}$ happens to lie orthogonal to some $n-r$-dimensional face F_r in the path, in which case we are left with a $\mathcal{O}(k^{-r})$ term (with $r < n$) which dominates asymptotically. Note that all vectors orthogonal to the $n-r$ dimensional face F_r lie on a r-dimensional subspace of $\mathbb{R}^n$. Since a polytope has a finite number of faces (of any dimension), the $\mathbf{k}$'s for which the Fourier transform is $\mathcal{O}(k^{-r})$ (instead of $\mathcal{O}(k^{-n})$) lies on a finite union of closed subspaces of dimension r (with $r < n$). The Lebesgue measure of all such lower dimensional subspaces for all such r is 0, leading us to the conclusion that the spectrum decays as $\mathcal{O}(k^{-n})$ for *almost all* $\mathbf{k}$'s. We formalize this in the following corollary.

Corollary 1. *Let P be a full dimensional polytope in $\mathbb{R}^n$. The Fourier spectrum of its indicator function 1_P satisfies the following:*

$$|\tilde{1}_P(\mathbf{k})| = \mathcal{O}\left(\frac{1}{k^{\Delta_\mathbf{k}}}\right)$$

where $1 \leq \Delta_\mathbf{k} \leq n$, and $\Delta_\mathbf{k} = j$ for $\mathbf{k}$ on a finite union of j-dimensional subspaces of $\mathbb{R}^n$.

C.3 Proof of Lemma 3

The first equality is simply the fact that $L_f = \max_\epsilon \|W_\epsilon\|$, and the second inequality follows trivially from the parameterization of a ReLU network as a chain of function compositions[14], together with the fact that the Lipschitz constant of the ReLU function is 1 (cf. Miyato et al. (2018), equation 7). To see the third inequality, consider the definition of the spectral norm of a $I \times J$ matrix W:

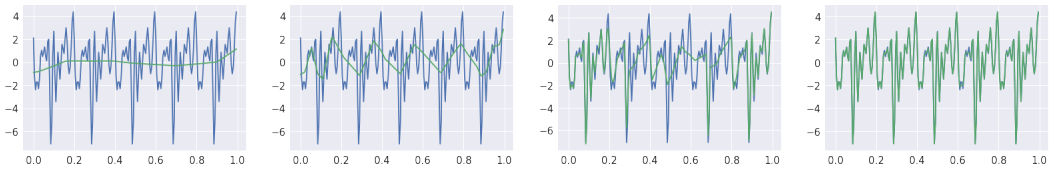
$$\|W\| = \max_{\|\mathbf{h}\|=1} \|W\mathbf{h}\| \tag{22}$$

Now, $\|W\mathbf{h}\| = \sqrt{\sum_i |\mathbf{w}_i \cdot \mathbf{h}|}$, where $\mathbf{w}_i$ is the i-th row of the weight matrix W and $i = 1, ..., I$. Further, if $\|\mathbf{h}\| = 1$, we have $|\mathbf{w}_i \cdot \mathbf{h}| \leq \|\mathbf{w}_i\|\|\mathbf{h}\| = \|\mathbf{w}_i\|$. Since $\|\mathbf{w}_i\| = \sqrt{\sum_j |w_{ij}|}$ (with $j = 1, ..., J$) and $|w_{ij}| \leq \|\theta\|_\infty$, we find that $\|\mathbf{w}_i\| \leq \sqrt{J}\|\theta\|_\infty$. Consequently, $\sqrt{\sum_i |\mathbf{w}_i \cdot \mathbf{h}|} \leq \sqrt{IJ}\|\theta\|_\infty$ and we obtain:

$$\|W\| \leq \sqrt{IJ}\|\theta\|_\infty \tag{23}$$

Now for $W = W^{(k)}$, we have $I = d_{k-1}$ and $J = d_k$. In the product over k, every d_k except the first and the last occur in pairs, which cancels the square root. For $k = 1$, $d_{k-1} = d$ (for the d input neurons) and for $k = L+1$, $d_k = 1$ (for a single output neuron). The final inequality now follows.

[14] Recall that the Lipschitz constant of a composition of two or more functions is the product of their respective Lipschtiz constants.



(a) Iteration 100. (b) Iteration 1000. (c) Iteration 10000. (d) Iteration 80000.

Figure 7: The learnt function (green) overlayed on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, ..., 45, 50)$, equal amplitudes and randomly sampled phases.

D EXPERIMENTAL DETAILS

D.1 EXPERIMENT 1

We fit a 6 layer ReLU network with 256 units per layer f_θ to the target function λ, which is a superposition of sine waves with increasing frequencies:

$$\lambda : [0, 1] \to \mathbb{R},\ \lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i)$$

where $k_i = (5, 10, 15, ..., 50)$, and φ_i is sampled from the uniform distribution $U(0, 2\pi)$. In the first setting, we set equal amplitude for all frequencies, i.e. $A_i = 1\,\forall\, i$, while in the second setting we assign larger amplitudes to the higher frequencies, i.e. $A_i = (0.1, 0.2, ..., 1)$. We sample λ on 200 uniformly spaced points in $[0, 1]$ and train the network for 80000 steps of full-batch gradient descent with Adam (Kingma & Ba, 2014). Note that we do not use stochastic gradient descent to avoid the stochasticity in parameter updates as a confounding factor. We evaluate the network on the same 200 point grid every 100 training steps and compute the magnitude of its (single-sided) discrete fourier transform at frequencies k_i which we denote with $|\tilde{f}_{k_i}|$. Finally, we plot in figure 1 the normalized magnitudes $\frac{|\tilde{f}_{k_i}|}{A_i}$ averaged over 10 runs (with different sets of sampled phases φ_i). We also record the spectral norms of the weights at each layer as the training progresses, which we plot in figure 1 for both settings (the spectral norm is evaluated with 10 power iterations). In figure 7, we show an example target function and the predictions of the network trained on it (over the iterations).

D.2 EXPERIMENT 2

We use the same 6-layer deep 256-unit wide network and define the target function

$$\lambda : \mathcal{D} \to \mathbb{R},\ z \mapsto \lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i)$$

where $k_i = (20, 40, ..., 180, 200)$, $A_i = 1\,\forall\, i$ and $\varphi \sim U(0, 2\pi)$. We sample ϕ on a grid with 1000 uniformly spaced points between 0 and 1 and map it to the input domain via γ_L to obtain a dataset $\{(\gamma_L(z_j), \lambda(z_j))\}_{j=0}^{999}$, on which we train the network with 50000 full-batch gradient descent steps of Adam. On the same 1000-point grid, we evaluate the magnitude of the (single-sided) discrete Fourier transform of $f_\theta \circ \gamma_L$ every 100 training steps at frequencies k_i and average over 10 runs (each with a different set of sampled z_i's). Fig 4 shows the evolution of the spectrum as training progresses for $L = 0, 4, 10, 16$, and Fig 5 shows the corresponding loss curves.

D.3 MNIST: A PROOF OF CONCEPT

In the following experiment, we show that given two manifolds of the same dimension – one flat and the other not – the task of learning random labels is harder to solve if the input samples lie on the same manifold. We demonstrate on MNIST under the assumption that the manifold hypothesis is true, and use the fact that the spectrum of the target function we use (white noise) is constant in expectation, and therefore independent of the underlying coordinate system when defined on the manifold.

Experiment 5. In this experiment, we investigate if it is easier to learn a signal on a more realistic data-manifold like that of MNIST (assuming the manifold hypothesis is true), and compare with

a flat manifold of the same dimension. To that end, we use the 64-dimensional feature-space $\mathcal{E}$ of a denoising[15] autoencoder as a proxy for the real data-manifold of unknown number of dimensions. The decoder functions as an embedding of $\mathcal{E}$ in the input space $X = \mathbb{R}^{784}$, which effectively amounts to training a network on the reconstructions of the autoencoder. For comparision, we use an injective embedding[16] of a 64-dimensional hyperplane in X. The latter is equivalent to sampling 784-dimensional vectors from $U([0, 1])$ and setting all but the first 64 components to zero. The target function is white-noise, sampled as scalars from the uniform distribution $U([0, 1])$. Two identical networks are trained under identical conditions, and Fig 8 shows the resulting loss curves, each averaged over 10 runs.

This result complements the findings of Arpit et al. (2017) and Zhang et al. (2017a), which show that it's easier to fit random labels to random inputs if the latter is defined on the full dimensional input space (i.e. the dimension of the flat manifold is the same as that of the input space, and not that of the underlying data-manifold being used for comparison).

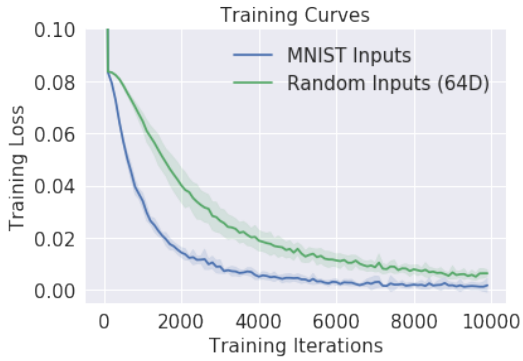


Figure 8: Loss curves of two identical networks trained to regress white-noise under identical conditions, one on MNIST reconstructions from a DAE with 64 encoder features (blue), and the other on 64-dimensional random vectors (green).

D.4 CIFAR-10: IT'S ALL CONNECTED

We have seen that deep neural networks are biased towards learning low frequency functions. This should have as a consequence that isolated *bubbles* of constant prediction are rare. This in turn implies that given any two points in the input space and a network function that predicts the same class for the said points, there should be a path connecting them such that the network prediction does not change along the path. In the following, we present an experiment where we use a path finding method to find such a path between all Cifar-10 input samples indeed exist.

Experiment 6. Using AutoNEB Kolsbjerg et al. (2016), we construct paths between (adversarial) Cifar-10 images that are classified by a ResNet20 to be all of the same target class. AutoNEB bends a linear path between points in some space $\mathbb{R}^m$ so that some maximum energy along the path is minimal. Here, the space is the input space of the neural network, i.e. the space of $32 \times 32 \times 3$ images and the logit output of the ResNet20 for a given class is minimized. We construct paths between the following points in image space:

- From one training image to another,
- from a training image to an adversarial,
- from one adversarial to another.

We only consider pairs of images that belong to the same class c (or, for adversarials, that originate from another class $\neq c$, but that the model classifies to be of the specified class c). For each class, we randomly select 50 training images and select a total of 50 random images from all other classes

[15]This experiment yields the same result if variational autoencoders are used instead.

[16]The xy-plane is $\mathbb{R}^3$ an injective embedding of a subset of $\mathbb{R}^2$ in $\mathbb{R}^3$.

Figure 9: Path between CIFAR-10 adversarial examples (e.g. "frog" and "automobile", such that all images are classified as "airplane").

and generate adversarial samples from the latter. Then, paths between all pairs from the whole set of images are computed.

The AutoNEB parameters are chosen as follows: We run four NEB iterations with 10 steps of SGD with learning rate 0.001 and momentum 0.9. This computational budget is similar to that required to compute the adversarial samples. The gradient for each NEB step is computed to maximize the logit output of the ResNet-20 for the specified target class c. We use the formulation of NEB without springs Draxler et al. (2018).

The result is very clear: We can find paths between *all* pairs of images for all CIFAR10 labels that do not cross a single decision boundary. This means that all paths belong to the same connected component regarding the output of the DNN. This holds for all possible combinations of images in the above list. Figure 10 shows connecting training to adversarial images and Figure 9 paths between pairs of adversarial images. Paths between training images are not shown, they provide no further insight. Note that the paths are strikingly simple: Visually, they are hard to distinguish from the linear interpolation. Quantitatively, they are essentially (but not exactly) linear, with an average length $(3.0 \pm 0.3)\%$ longer than the linear connection.

E THE FOURIER TRANSFORM OF A FUNCTION COMPOSITION

Consider equation 11. The general idea is to investigate the behaviour of $P_\gamma(\mathbf{l}, \mathbf{k})$ for large frequencies $\mathbf{l}$ on manifold but smaller frequencies $\mathbf{k}$ in the input domain. In particular, we are interested in the regime where the stationary phase approximation is applicable to P_γ, i.e. when $l^2 + k^2 \to \infty$ (cf. section 3.2. of Bergner et al.). In this regime, the integrand in $P_\gamma(\mathbf{k}, \mathbf{l})$ oscillates fast enough such that the only constructive contribution originates from where the phase term $u(\mathbf{z}) = \mathbf{k} \cdot \gamma(\mathbf{z}) - \mathbf{l} \cdot \mathbf{z}$ does not change with changing $\mathbf{z}$. This yields the condition that $\nabla_{\mathbf{z}} u(\mathbf{z}) = 0$, which translates to the condition (with Einstein summation convention implied and $\partial_\nu = \partial / \partial x_\nu$):

$$l_\nu = k_\mu \partial_\nu \gamma_\mu(\mathbf{z}) \tag{24}$$

Now, let the components of γ be periodic, and without loss of generality we let the period be 2π. Further, we require that the manifold be contained in a box of some size in $\mathbb{R}^d$. The μ-th component γ_μ can now be expressed as a Fourier series:

$$\gamma_\mu(\mathbf{z}) = \sum_{\mathbf{p} \in \mathbb{Z}^m} \tilde{\gamma}_\mu[\mathbf{p}] e^{-i p_\rho z_\rho} \tag{25} \qquad \partial_\nu \gamma_\mu(\mathbf{z}) = \sum_{\mathbf{p} \in \mathbb{Z}^m} -i p_\nu \tilde{\gamma}_\mu[\mathbf{p}] e^{-i p_\rho z_\rho} \tag{26}$$

Equation 26 can be substituted in equation 24 to obtain:

$$l \hat{l}_\nu = -ik \sum_{\mathbf{p} \in \mathbb{Z}^m} p_\nu \hat{k}_\mu \tilde{\gamma}_\mu[\mathbf{p}] e^{-i p_\rho z_\rho} \tag{27}$$

Figure 10: Each row is a path through the image space from an adversarial sample (right) to a true training image (left). All images are classified by a ResNet-20 to be of the class of the training sample on the right with at least 95% softmax certainty. This experiment shows we can find a path from adversarial examples (right, Eg. "(cat)") that are classified as a particular class ("airplane") are connected to actual training samples from that class (left, "airplane") such that all samples along that path are also predicted by the network to be of the same class.

where we have split k_μ and l_ν in to their magnitudes k and l and directions $\hat{k}_\nu$ and $\hat{l}_\mu$ (respectively). We are now interested in the conditions on γ under which the RHS can be large in magnitude, even when k is fixed. Recall that γ is constrained to a box – consequently, we can not arbitrarily scale up $\tilde{\gamma}_\mu$. However, if $\tilde{\gamma}_\mu[\mathbf{p}]$ decays slowly enough with increasing $\mathbf{p}$, the RHS can be made arbitrarily large (for certain conditions on $\mathbf{z}$, $\hat{l}_\mu$ and $\hat{k}_\nu$).

F Volume in Parameter Space and Proof of Proposition 1

For a given neural network, we now show that the volume of the parameter space containing parameters that contribute ϵ-non-negligibly to frequency components of magnitude k' above a certain cut-off k decays with increasing k. For notational simplicity and without loss of generality, we absorb the direction $\hat{\mathbf{k}}$ of $\mathbf{k}$ in the respective mappings and only deal with the magnitude k.

Definition 1. *Given a ReLU network f_θ of fixed depth, width and weight clip K with parameter vector θ, an $\epsilon > 0$ and $\Theta = B_K^\infty(0)$ a L^∞ ball around 0, we define:*

$$\Xi_\epsilon(k) = \{\theta \in \Theta | \exists k' > k, |\tilde{f}_\theta(k')| > \epsilon\}$$

as the set of all parameters vectors $\theta \in \Xi_\epsilon(k)$ that contribute more than an ϵ in expressing one or more frequencies k' above a cut-off frequency k.

Remark 1. *If $k_2 \geq k_1$, we have $\Xi_\epsilon(k_2) \subseteq \Xi_\epsilon(k_1)$ and consequently vol$(\Xi_\epsilon(k_2)) \leq$ vol$(\Xi_\epsilon(k_1))$, where vol is the Lebesgue measure.*

Lemma 5. *Let $1_k^\epsilon(\theta)$ be the indicator function on $\Xi_\epsilon(k)$. Then:*

$$\exists\, \kappa > 0 : \forall k \geq \kappa, 1_k^\epsilon(\theta) = 0$$

Proof. From theorem 1, we know that[17] $|\tilde{f}_\theta(k)| = \mathcal{O}(k^{-d+\Delta-1})$ for an integer $\Delta \leq d-1$. In the worse case where $\Delta = d-1$, we have that $\exists M < \infty : |\tilde{f}_\theta(k)| < \frac{M}{k^2}$. Now, simply select a $\kappa > \sqrt{\frac{M}{\epsilon}}$ such that $\frac{M}{\kappa^2} < \epsilon$. This yields that $|\tilde{f}_\theta(\kappa)| < \frac{M}{\kappa^2} < \epsilon$, and given that $\frac{M}{\kappa^2} \leq \frac{M}{k^2}\, \forall\, k \geq \kappa$, we find $|\tilde{f}_\theta(k)| < \epsilon\, \forall\, k \geq \kappa$. Now by definition 1, $\theta \notin \Xi_\epsilon(\kappa)$, and since $\Xi_\epsilon(k) \subseteq \Xi_\epsilon(\kappa)$ (see remark 1), we have $\theta \notin \Xi_\epsilon(k)$, implying $1_k^\epsilon(\theta) = 0\, \forall\, k \geq \kappa$. □

Remark 2. *We have $1_k^\epsilon(\theta) \leq |\tilde{f}_\theta(k)|$ for large enough k (i.e. for $k \geq \kappa$), since $|\tilde{f}_\theta(k)| \geq 0$.*

[17] Note that in theorem 1, $\Delta_{\hat{\mathbf{k}}}$ depends only on the direction of $\mathbf{k}$, which we absorb in the definition of Δ.

Theorem 2. *The relative volume of* $\Xi_\epsilon(k)$ *w.r.t.* Θ *is* $\mathcal{O}(k^{-d+\Delta-1})$ *where* $0 \leq \Delta \leq d-1$.

Proof. The volume is given by the integral over the indicator function, i.e.

$$\text{vol}(\Xi_\epsilon(k)) = \int_{\theta \in \Theta} 1_k^\epsilon(\theta) d\theta$$

For a large enough k, we have from remark 2, the monotonicity of the Lebesgue integral and theorem 1 that:

$$\text{vol}(\Xi_\epsilon(k)) = \int_{\theta \in \Theta} 1_k^\epsilon(\theta) d\theta \leq \int_{\theta \in \Theta} |\tilde{f}_\theta(k)| d\theta = \mathcal{O}(k^{-d+\Delta-1}) \text{vol}(\Theta)$$
$$\implies \frac{\text{vol}(\Xi_\epsilon(k))}{\text{vol}(\Theta)} = \mathcal{O}(k^{-d+\Delta-1})$$

□

G KERNEL MACHINES AND KNNS

In this section we contrast DNNs with K-nearest neighbor (k-NN) classifier and kernel machines which are also popular learning algorithms, but in contrast to DNNs are better understood theoretically.

G.1 KERNEL MACHINES VS DNNS

Given we study why DNNs are biased towards learning smooth functions, we note that kernel machines (KM) are also highly Lipschitz smooth (Eg. for Gaussian kernels all derivatives are bounded). However there are crutial differences between the two. While kernel machines can approximate any target function in principal (Hammer & Gersmann, 2003), the number of Gaussian kernels needed scales linearly with the number of sign changes in the target function (Bengio et al., 2009). Ma & Belkin (2017) have further shown that for smooth kernels, a target function cannot be approximated within ϵ precision in any polynomial of $1/\epsilon$ steps by gradient descent.

Deep networks on the other hand are also capable of approximating any target function (as shown by the universal approximation theorems Hornik et al. (1989); Cybenko (1989)), but they are also parameter efficient in contrast to KM. For instance, we have seen that deep ReLU networks separate the input space into number of linear regions that grow polynomially in width of layers and exponentially in the depth of the network (Montufar et al., 2014; Raghu et al., 2016). A similar result on the exponentially growing expressive power of networks in terms of their depth is also shown in (Poole et al., 2016). In this paper we have further shown that DNNs are inherently biased towards lower frequency (smooth) functions over a finite parameter space. This suggests that DNNs strike a good balance between function smoothness and expressibility/parameter-efficiency compared with KM.

G.2 K-NN CLASSIFIER VS. DNN CLASSIFIER

K-nearest neighbor (KNN) also has a historical importance as a classification algorithm due to its simplicity. It has been shown to be a consistent approximator Devroye et al. (1996), i.e., asymptotically its empirical risk goes to zero as $K \to \infty$ and $K/N \to 0$, where N is the number of training samples. However, because it is a memory based algorithm, it is prohibitively slow for large datasets. Since the smoothness of a KNN prediction function is not well studied, we compare the smoothness between KNN and DNN. For various values of K, we train a KNN classifier on a $k = 150$ frequency signal (which is binarized) defined on the $L = 20$ manifold (see section 4), and extract probability predictions on a box interval in $\mathbb{R}^2$. On

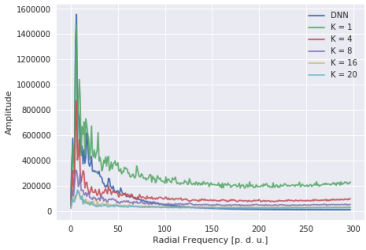
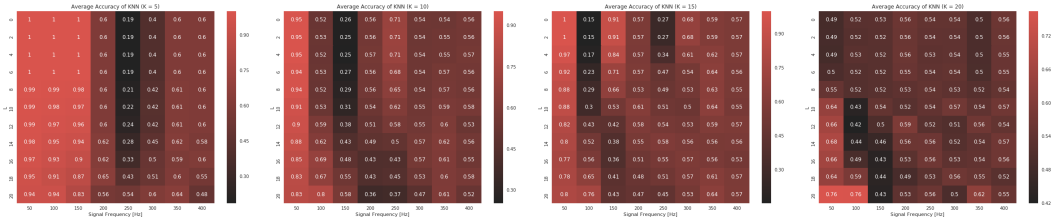


Figure 11: The frequency spectrum of KNNs with different values of K, and a DNN. The DNN learns a smoother function compared with the KNNs considered since the spectrum of the DNN decays faster compared with KNNs.



(a) $K = 5$. (b) $K = 10$. (c) $K = 15$. (d) $K = 20$.

Figure 12: Heatmaps of training accuracies (L-vs-k) of KNNs for various K. When comparing with figure 3, note that the y-axis is flipped.

this interval, we evaluate the 2D FFT and integrate out the angular components to obtain $\zeta(k)$:

$$\zeta(k) = \frac{d}{dk} \int_0^k dk' k' \int_0^{2\pi} d\varphi |\tilde{f}(k', \varphi)| \qquad (28)$$

Finally, we plot $\zeta(k)$ for various K in figure 11. Furthermore, we train a DNN on the very same dataset and overlay the radial spectrum of the resulting probability map on the same plot. We find that while DNN's are as expressive as a $K = 1$ KNN classifier at lower (radial) frequencies, the frequency spectrum of DNNs decay faster than KNN classifier for all values of K considered, indicating that the DNN is smoother than the KNNs considered. We also repeat the experiment corresponding to figure 3 with KNNs (figure 12) for various K's, to find that unlike DNNs, KNNs do not necessarily perform better for larger L's, suggesting that KNNs do not exploit the geometry of the manifold like DNNs do.