On the Spectral Bias of Neural Networks

Nasim Rahaman [*1 2] **Aristide Baratin** [*1] **Devansh Arpit** [1] **Felix Draxler** [2] **Min Lin** [1] **Fred A. Hamprecht** [2]
Yoshua Bengio [1] **Aaron Courville** [1]

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, we highlight a learning bias of deep networks towards low frequency functions – i.e. functions that vary globally without local fluctuations – which manifests itself as a frequency-dependent learning speed. Intuitively, this property is in line with the observation that over-parameterized networks prioritize learning simple patterns that generalize across data samples. We also investigate the role of the shape of the data manifold by presenting empirical and theoretical evidence that, somewhat counter-intuitively, learning higher frequencies gets *easier* with increasing manifold complexity.

1. Introduction

The remarkable success of deep neural networks at generalizing to natural data 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). This has prompted recent investigations of possible implicit regularization mechanisms inherent in the learning process which induce a bias towards low complexity solutions (Neyshabur et al., 2014; Soudry et al., 2017; Poggio et al., 2018; Neyshabur et al., 2017).

In this work, we take a slightly shifted view on implicit regularization by suggesting that low-complexity functions are *learned faster* during training by gradient descent. We expose this bias by taking a closer look at neural networks through the lens of Fourier analysis. While they can approximate arbitrary functions, we find that these networks prioritize learning the low frequency modes, a phenomenon we call the *spectral bias*. This bias manifests itself not just in the process of learning, but also in the parameterization of the model itself: in fact, we show that the lower frequency components of trained networks are more robust to random parameter perturbations. Finally, we also expose and analyze the rather intricate interplay between the spectral bias and the geometry of the data manifold by showing that high frequencies get easier to learn when the data lies on a lower-dimensional manifold of complex shape embedded in the input space of the model. We focus the discussion on networks with rectified linear unit (ReLU) activations, whose continuous piece-wise linear structure enables an analytic treatment.

Contributions[1]

1. We exploit the continuous piecewise-linear structure of ReLU networks to evaluate its Fourier spectrum (Section 2).

2. We find empirical evidence of a *spectral bias*: i.e. lower frequencies are learned first. We also show that lower frequencies are more robust to random perturbations of the network parameters (Section 3).

3. We study the role of the shape of the data manifold: we show how complex manifold shapes can facilitate the learning of higher frequencies and develop a theoretical understanding of this behavior (Section 4).

2. Fourier analysis of ReLU networks

2.1. Preliminaries

Throughout the paper we call 'ReLU network' a scalar function $f : \mathbb{R}^d \mapsto \mathbb{R}$ defined by a neural network with L hidden layers of widths $d_1, \cdots d_L$ and a single output neuron:

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

[*]Equal contribution [1]Mila, Quebec, Canada [2]Image Analysis and Learning Lab, Ruprecht-Karls-Universität Heidelberg, Germany. Correspondence to: Nasim Rahaman <nasim.rahaman@live.com>, Aristide Baratin <aristide.baratin@umontreal.ca>, Devansh Arpit <devansharpit@gmail.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]Code: https://github.com/nasimrahaman/SpectralBias

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 $\sigma(\mathbf{u})_i = \max(0, u_i)$ denotes the ReLU activation function acting elementwise on a vector $\mathbf{u} = (u_1, \cdots u_n)$. In the standard basis, $T^{(k)}(\mathbf{x}) = W^{(k)}\mathbf{x} + \mathbf{b}^{(k)}$ for some weight matrix $W^{(k)}$ and bias vector $\mathbf{b}^{(k)}$.

ReLU networks are known to be continuous piece-wise linear (CPWL) functions, where the linear regions are convex polytopes (Raghu et al., 2016; Montufar et al., 2014; Zhang et al., 2018; Arora et al., 2018). Remarkably, the converse also holds: every CPWL function can be represented by a ReLU network (Arora et al., 2018, Theorem 2.1), which in turn endows ReLU networks with universal approximation properties. Given the ReLU network f from Eqn. 1, we can make the piecewise linearity explicit by writing,

$$f(\mathbf{x}) = \sum_{\epsilon} 1_{P_\epsilon}(\mathbf{x}) \, (W_\epsilon \mathbf{x} + \mathbf{b}_\epsilon) \tag{2}$$

where ϵ is an index for the linear regions P_ϵ and 1_{P_ϵ} is the indicator function on P_ϵ. As shown in Appendix B in more detail, each region corresponds to an *activation pattern*[2] of all hidden neurons of the network, which is a binary vector with components conditioned on the sign of the input of the respective neuron. The $1 \times d$ matrix W_ϵ is given by

$$W_\epsilon = W^{(L+1)} W_\epsilon^{(L)} \cdots W_\epsilon^{(1)} \tag{3}$$

where $W_\epsilon^{(k)}$ is obtained from the original weight $W^{(k)}$ by setting its j^{th} column to zero whenever the neuron j of the k^{th} layer is inactive.

2.2. Fourier Spectrum

In the following, we study the structure of ReLU networks in terms of their Fourier representation, $f(\mathbf{x}) := (2\pi)^{d/2} \int \tilde{f}(\mathbf{k}) \, e^{i\mathbf{k}\cdot\mathbf{x}} \mathbf{dk}$, where $\tilde{f}(\mathbf{k}) := \int f(\mathbf{x}) \, e^{-i\mathbf{k}\cdot\mathbf{x}} \mathbf{dx}$ is the Fourier transform[3]. Lemmas 1 and 2 yield the explicit form of the Fourier components (we refer to Appendix C for the proofs and technical details).

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

$$\tilde{f}(\mathbf{k}) = i \sum_{\epsilon} \frac{W_\epsilon \mathbf{k}}{k^2} \, \tilde{1}_{P_\epsilon}(\mathbf{k}) \tag{4}$$

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

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

[3] Note that general ReLU networks need not be squared integrable: for instance, the class of two-layer ReLU networks represent an arrangement of hyperplanes (Montufar et al., 2014) and hence grow linearly as $x \to \infty$. In such cases, the Fourier transform is to be understood in the sense of tempered distributions acting on rapidly decaying smooth functions ϕ as $\langle \tilde{f}, \phi \rangle = \langle f, \tilde{\phi} \rangle$. See Appendix C for a formal treatment.

The Fourier transform of the indicator over linear regions appearing in Eqn. 4 are fairly intricate mathematical objects. Diaz et al. (2016) develop an elegant procedure for evaluating it in arbitrary dimensions via a recursive application of Stokes theorem. We describe this procedure in detail[4] in Appendix C.2, and present here its main corollary.

Lemma 2. *Let P be a full dimensional polytope in $\mathbb{R}^d$. Its Fourier spectrum takes the form:*

$$\tilde{1}_P(\mathbf{k}) = \sum_{n=0}^{d} \frac{D_n(\mathbf{k}) 1_{G_n}(\mathbf{k})}{k^n} \tag{5}$$

where G_n is the union of n-dimensional subspaces that are orthogonal to some n-codimensional face of P, $D_n : \mathbb{R}^d \to \mathbb{C}$ is in $\Theta(1)$ $(k \to \infty)$ and 1_{G_n} the indicator over G_n.

Lemmas 1, 2 together yield the main result of this section.

Theorem 1. *The Fourier components of the ReLU network f_θ with parameters θ is given by the rational function:*

$$\tilde{f}_\theta(\mathbf{k}) = \sum_{n=0}^{d} \frac{C_n(\theta, \mathbf{k}) 1_{H_n^\theta}(\mathbf{k})}{k^{n+1}} \tag{6}$$

where H_n^θ is the union of n-dimensional subspaces that are orthogonal to some n-codimensional faces of some polytope P_ϵ and $C_n(\cdot, \theta) : \mathbb{R}^d \to \mathbb{C}$ is $\Theta(1)$ $(k \to \infty)$.

Note that Eqn 6 applies to general ReLU networks with arbitrary width and depth[5].

Discussion. We make the following two observations. First, the spectral decay of ReLU networks is highly anisotropic in large dimensions. In almost all directions of $\mathbb{R}^d$, we have a k^{-d-1} decay. However, the decay can be as slow as k^{-2} in specific directions orthogonal to the $d-1$ dimensional faces bounding the linear regions[6].

Second, the numerator in Eqn 6 is bounded by $N_f L_f$ (cf. Appendix C.3), where N_f is the number of linear regions and $L_f = \max_\epsilon \|W_\epsilon\|$ is the Lipschitz constant of the network. Further, the Lipschitz constant L_f can be bounded as (cf. Appendix C.6):

$$L_f \leq \prod_{k=1}^{L+1} \|W^{(k)}\| \leq \|\theta\|_\infty^{L+1} \sqrt{d} \prod_{k=1}^{L} d_k \tag{7}$$

where $\|\cdot\|$ is the spectral norm and $\|\cdot\|_\infty$ the max norm, and d_k is the number of units in the k-th layer. This makes the

[4] We also generalize the construction to tempered distributions.

[5] Symmetries that might arise due to additional assumptions can be used to further develop Eqn 6, see e.g. Eldan & Shamir (2016) for 2-layer networks.

[6] 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.

bound on L_f scale exponentially in depth and polynomial in width. As for the number N_f of linear regions, Montufar et al. (2014) and Raghu et al. (2016) obtain tight bounds that exhibit the same scaling behaviour (Raghu et al., 2016, Theorem 1). In Appendix A.5, we qualitatively ablate over the depth and width of the network to expose how this reflects on the Fourier spectrum of the network.

3. Lower Frequencies are Learned First

We now present experiments showing that networks tend to fit *lower frequencies first* during training. We refer to this phenomenon as the *spectral bias*, and discuss it in light of the results of Section 2.

3.1. Synthetic Experiments

Experiment 1. The setup is as follows[7]: 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

$$\lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i). \tag{8}$$

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 $A_{10} = 1$. Figure 1 shows the normalized magnitudes $|\tilde{f}_\theta(k_i)|/A_i$ at various frequencies, as training progresses with full-batch gradient descent. Further, Figure 2 shows the learned function at intermediate training iterations. The result is that lower frequencies (i.e. smaller k_i's) are regressed first, regardless of their amplitudes.

Experiment 2. Our goal here is to illustrate a phenomenon that complements the one highlighted above: lower frequencies are more *robust* to parameter perturbations. The set up is the same as in Experiment 1. The network is trained to regress a target function with frequencies $\kappa = (10, 15, 20, ..., 45, 50)$ and amplitudes $A_i = 1 \, \forall \, i$. After convergence to θ^*, we consider random (isotropic) perturbations $\theta = \theta^* + \delta\hat{\theta}$ of given magnitude δ, where $\hat{\theta}$ is a random unit vector in parameter space. We evaluate the network function f_θ at the perturbed parameters, and compute the magnitude of its discrete Fourier transform at frequencies k_i to obtain $|\tilde{f}_\theta(k_i)|$. We also average over 100 samples of $\hat{\theta}$ to obtain $|\tilde{f}_{\mathbb{E}\theta}(k_i)|$, which we normalize by $|\tilde{f}_{\theta*}(k_i)|$. Finally, we average over the phases ϕ (see Eqn 8).

[7] More experimental details and additional plots are provided in Appendix A.1.

The result, shown in Figure 3, demonstrates that higher frequencies are significantly less robust than the lower ones, guiding the intuition that expressing higher frequencies requires the parameters to be finely-tuned to work together. In other words, parameters that contribute towards expressing high-frequency components occupy a small volume in the parameter space. We formalize this in Appendix D.

Discussion . Multiple theoretical aspects may underlie these observations. First, for a fixed architecture, recall that the numerator in Theorem 1 is[8] $\mathcal{O}(L_f)$ (where L_f is the Lipschitz constant of the function). However, L_f is bounded by the parameter norm, which can only increase gradually during training by gradient descent. This leads to the higher frequencies being learned[9] 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.

Second (cf. Appendix C.4), the exact form of the Fourier spectrum yields that for a fixed direction $\hat{\mathbf{k}}$, the spectral decay rate of the parameter gradient $\partial\tilde{f}/\partial\theta$ is at most one exponent of k lower than that of $\tilde{f}$. If for a fixed $\hat{\mathbf{k}}$ we have $\tilde{f} = \mathcal{O}(k^{-\Delta-1})$ where $1 \leq \Delta \leq d$, we obtain for the residual $h = f - \lambda$ and (continuous) training step t:

$$\left|\frac{d\tilde{h}(\mathbf{k})}{dt}\right| = \left|\frac{d\tilde{f}(\mathbf{k})}{dt}\right| = \underbrace{\left|\frac{d\tilde{f}(\mathbf{k})}{d\theta}\right|}_{\mathcal{O}(k^{-\Delta})} \overbrace{\left|\frac{d\theta}{dt}\right|}^{|\eta \cdot d\mathcal{L}/d\theta|} = \mathcal{O}(k^{-\Delta}) \tag{9}$$

where we use the fact that $d\theta/dt$ is just the learning rate times the parameter gradient of the loss which is independent[10] of k, and assume that the target function λ is fixed. Eqn 9 shows that the rate of change of the residual decays with increasing frequency, which is what we find in Experiment 1.

3.2. Real-Data Experiments

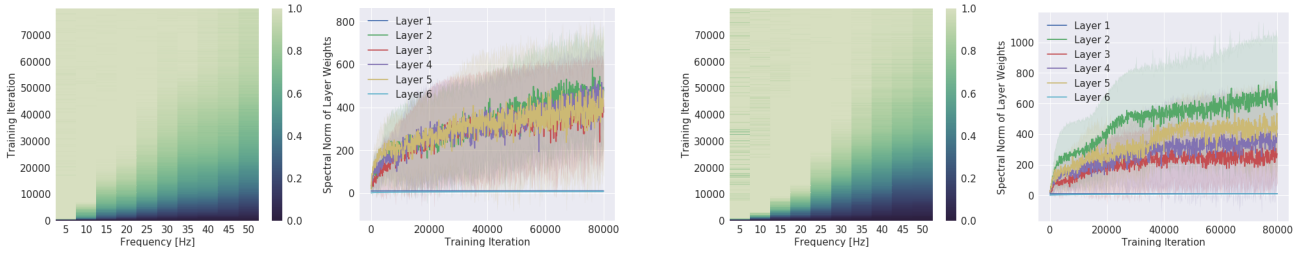
While Experiments 1 and 2 establish the spectral bias by explicitly evaluating the Fourier coefficients, doing so becomes prohibitively expensive for larger d (e.g. on MNIST). To tackle this, we propose the following set of experiments to measure the effect of spectral bias indirectly on MNIST.

Experiment 3. In this experiment, we investigate how the validation performance dependent on the frequency of noise

[8] The tightness of this bound is verified empirically in appendix A.5.

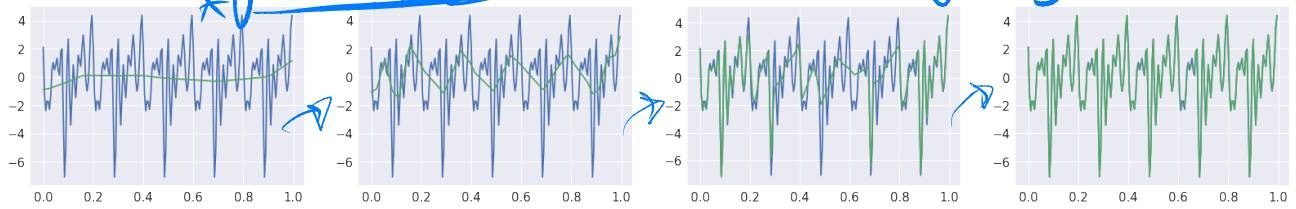
[9] This assumes that the Lipschitz constant of the (noisy) target function is larger than that of the network at initialization.

[10] Note however that the loss term might involve a sum or an integral over all frequencies, but the summation is over a different variable.



(a) Equal Amplitudes (b) Increasing Amplitudes

Figure 1. Left (a, b): 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) Iteration 100 (b) Iteration 1000 (c) Iteration 10000 (d) Iteration 80000

Figure 2. 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.

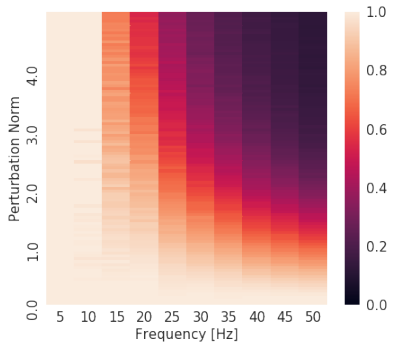


Figure 3. Normalized spectrum of the model (x-axis for frequency, colorbar for magnitude) with perturbed parameters as a function of parameter perturbation (y-axis). The colormap is clipped between 0 and 1. We observe that the lower frequencies are more robust to parameter perturbations than the higher frequencies.

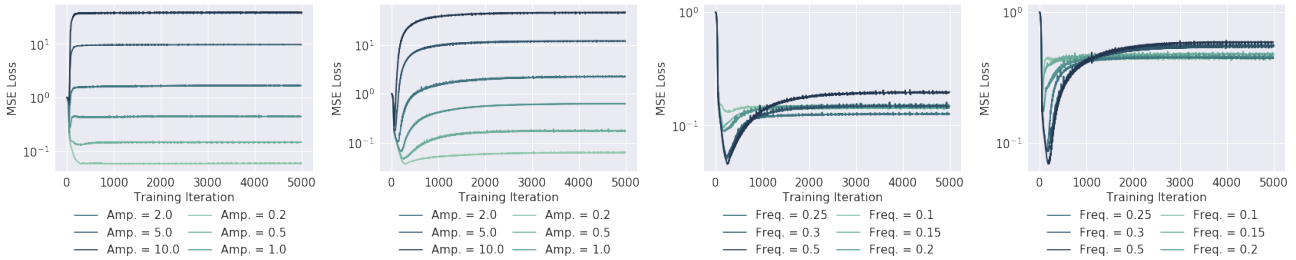
added to the training target. We find that the best validation performance on MNIST is particularly insensitive to the magnitude of high-frequency noise, yet it is adversely affected by low-frequency noise. We consider a target (binary) function $\tau_0 : X \to \{0, 1\}$ defined on the space $X = [0, 1]^{784}$ of MNIST inputs. Samples $\{\mathbf{x}_i, \tau_0(\mathbf{x}_i)\}_i$ form a subset of the MNIST dataset comprising samples $\mathbf{x}_i$ belonging to two classes. Let $\psi_k(\mathbf{x})$ be a *noise function*:

$$\psi_k(\mathbf{x}) = \sin(k\|\mathbf{x}\|) \tag{10}$$

corresponding to a *radial wave* defined on the 784-dimensional input space[11]. The final target function τ_k is then given by $\tau_k = \tau_0 + \beta\psi_k$, where β is the effective amplitude of the noise. We fit the same network as in Experiment 1 to the target τ_k with the MSE loss. In the first set of experiments, we ablate over k for a pair of fixed βs, while in the second set we ablate over β for a pair of fixed ks. In Figure 4, we show the respective validation loss curves, where the validation set is obtained by evaluating τ_0 on a separate subset of the data, i.e. $\{\mathbf{x}_j, \tau_0(\mathbf{x}_j)\}_j$. Figure 11 (in appendix A.3) shows the respective training curves.

Discussion. The profile of the loss curves varies significantly with the frequency of noise added to the target. In Figure 4a, we see that the validation performance is adversely affected by the amplitude of the low-frequency noise, whereas Figure 4b shows that the amplitude of high-

[11]The rationale behind using a radial wave is that it induces oscillations (simultaneously) along all spatial directions. Another viable option is to induce oscillations along the principle axes of the data: we have verified that the key trends of interest are preserved.



(a) $k = 0.1$ (b) $k = 1$ (c) $\beta = 0.5$ (d) $\beta = 1.$

Figure 4. (a,b,c,d): Validation curves for various settings of noise amplitude β and frequency k. Corresponding training curves can be found in Figure 11 in appendix A.3. **Gist**: Low frequency noise affects the network more than their high-frequency counterparts. Further, for high-frequency noise, one finds that the validation loss dips early in the training. Both these observations are explained by the fact that network readily fit lower frequencies, but learn higher frequencies later in the training.

frequency noise does not significantly affect the best validation score. This is explained by the fact that the network readily fits the noise signal if it is low frequency, whereas the higher frequency noise is only fit later in the training. In the latter case, the dip in validation score early in the training is when the network has learned the low frequency true target function τ_0; the remainder of the training is spent learning the higher-frequencies in the training target τ, as we shall see in the next experiment. Figures 4c and 4d confirm that the dip in validation score exacerbates for increasing frequency of the noise. Further, we observe that for higher frequencies (e.g. $k = 0.5$), increasing the amplitude β does not significantly degrade the best performance at the dip, confirming that the network is fairly robust to the amplitude of high-frequency noise.

Finally, we note that the dip in validation score was also observed by Arpit et al. (2017) with i.i.d. noise[12] in a classification setting.

Experiment 4. To investigate the dip observed in Experiment 3, we now take a more direct approach by considering a generalized notion of frequency. To that end, we project the network function to the space spanned by the orthonormal eigenfunctions φ_n of the Gaussian RBF kernel (Braun et al., 2006). These eigenfunctions φ_n (sorted by decreasing eigenvalues) resemble sinusoids (Fasshauer, 2011), and the index n can be thought of as being a proxy for the frequency, as can be seen from Figure 6 (see Appendix A.4 for additional details and supporting plots). While we will call $\tilde{f}[n]$ as the spectrum of the function f, it should be understood as $\tilde{f}[n] = \langle f_{\mathcal{H}}, \varphi_n \rangle_{\mathcal{H}}$, where $f_{\mathcal{H}} \in \text{span}\{\varphi_n\}_n$ and $f_{\mathcal{H}}(\mathbf{x}_i) = f(\mathbf{x}_i)$ on the MNIST samples $\mathbf{x}_i \in X$. This allows us to define a noise function as:

$$\psi_\gamma(\mathbf{x}) = \sum_n^N \left(\frac{n}{N}\right)^\gamma \varphi_n(\mathbf{x}) \tag{11}$$

[12]Recall that i.i.d. noise is white-noise, which has a constant Fourier spectrum magnitude in expectation, i.e. it also contains high-frequency components.

where N is the number of available samples and $\gamma = 2$. Like in Experiment 3, the target function is given by $\tau = \tau_0 + \beta\psi$, and the same network is trained to regress τ. Figure 5 shows the (generalized) spectrum τ and τ_0, and that of f as training progresses. Figure 13 (in appendix) shows the corresponding dip in validation loss, where the validation set is same as the training set but with true target function τ_0 instead of the noised target τ.

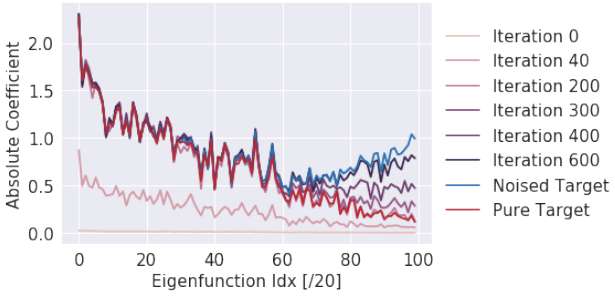


Figure 5. Spectrum of the network as it is trained on MNIST target with high-frequency noise (*Noised Target*). We see that the network fits the true target at around the 200th iteration, which is when the validation score dips (Figure 13 in appendix).

Discussion. From Figure 5, we learn that the drop in validation score observed in Figure 4 is exactly when the higher-frequencies of the noise signal are yet to be learned. As the network gradually learns the higher frequency eigenfunctions, the validation loss increases while the training loss continues to decrease. Thus these experiments show that the phenomenon of spectral bias persists on non-synthetic data and in high dimensional input spaces.

4. Not all Manifolds are Learned Equal

In this section, we investigate subtleties that arise when the data lies on a lower dimensional manifold embedded in the higher dimensional input space of the model. We find

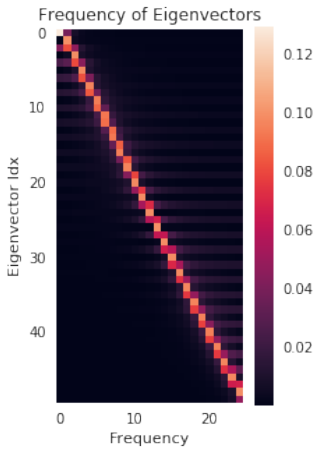


Figure 6. Spectrum (x-axis for frequency, colorbar for magnitude) of the n-th (y-axis) eigenvector of the Gaussian RBF kernel matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, where the sample set is $\{x_i \in [0,1]\}_{i=1}^{50}$ is $N = 50$ uniformly spaced points between 0 and 1 and k is the Gaussian RBF kernel function. **Gist:** The eigenfunctions with increasing n roughly correspond to sinusoids of increasing frequency. Refer to Appendix A.4 for more details.

that the *shape* of the data-manifold impacts the learnability of high frequencies in a non-trivial way. As we shall see, this is because low frequency functions in the input space may have high frequency components when restricted to lower dimensional manifolds of complex shapes. We demonstrate results in an illustrative minimal setting[13], free from unwanted confounding factors, and present a theoretical analysis of the phenomenon.

Manifold hypothesis. We consider the case where the data lies on a lower dimensional *data manifold* $\mathcal{M} \subset \mathbb{R}^d$ embedded in input space (Goodfellow et al., 2016), which we assume to be the image $\gamma([0,1]^m)$ of some injective mapping $\gamma : [0,1]^m \to \mathbb{R}^d$ defined on a lower dimensional latent space $[0,1]^m$. Under this hypothesis and in the context of the standard regression problem, a target function $\tau : \mathcal{M} \to \mathbb{R}$ defined on the data manifold can be identified with a function $\lambda = \tau \circ \gamma$ defined on the latent space. Regressing τ is therefore equivalent to finding $f : \mathbb{R}^d \to \mathbb{R}$ such that $f \circ \gamma$ matches λ. Further, assuming that the data probability distribution μ supported on $\mathcal{M}$ is induced by γ from the uniform distribution U in the latent space $[0,1]^m$, the mean square error can be expressed as:

$$\begin{aligned} &\text{MSE}_{\mu}^{(\mathbf{x})}[f, \tau] = \mathbb{E}_{\mathbf{x}\sim\mu}|f(\mathbf{x}) - \tau(\mathbf{x})|^2 = \\ &\mathbb{E}_{\mathbf{z}\sim U}|(f(\gamma(\mathbf{z})) - \lambda(\mathbf{z})|^2 = \text{MSE}_{U}^{(\mathbf{z})}[f \circ \gamma, \lambda] \end{aligned} \tag{12}$$

Observe that there is a vast space of degenerate solutions f that minimize the mean squared error – namely all functions on $\mathbb{R}^d$ that yield the same function when restricted to the data manifold $\mathcal{M}$.

[13] We include additional experiments on MNIST and CIFAR-10 in appendices A.6 and A.7.

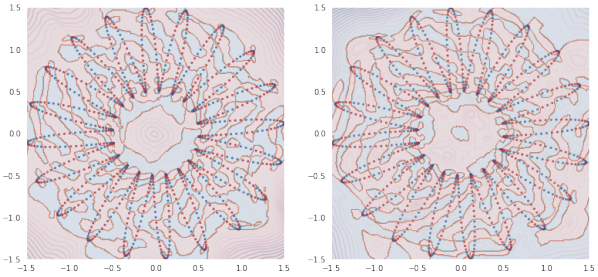


Figure 7. Functions learned by two identical networks (up to initialization) to classify the binarized value of a sine wave of frequency $k = 200$ defined on a $\gamma_{L=20}$ manifold. Both yield close to perfect accuracy for the samples defined on the manifold (scatter plot), yet they differ significantly elsewhere. The shaded regions show the predicted class (Red or Blue) whereas contours show the confidence (absolute value of logits).

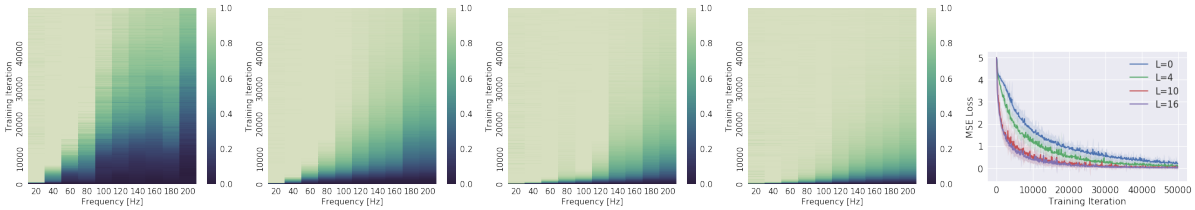
Our findings from the previous section suggest that neural networks are biased towards expressing a particular subset of such solutions, namely those that are low frequency. It is also worth noting that there exist methods that restrict the space of solutions: notably adversarial training (Goodfellow et al., 2014) and Mixup (Zhang et al., 2017b).

Experimental set up. 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 $\gamma_L : [0,1] \to \mathbb{R}^2$ given by

$$\begin{aligned} \gamma_L(z) =& R_L(z)(\cos(2\pi z), \sin(2\pi z)) \\ \text{where } R_L(z) =& 1 + \frac{1}{2}\sin(2\pi L z) \end{aligned} \tag{13}$$

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 7). 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 5. The set-up is similar to that of Experiment 1, and λ is as defined in Eqn. 8 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 is shown in Fig 8 for various L. Fig 8e shows the corresponding mean squared error curves. More experimental details in appendix A.2.



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

Figure 8. (a,b,c,d): 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. (e): Corresponding learning curves. **Gist**: Some manifolds (here with larger L) make it easier for the network to learn higher frequencies than others.

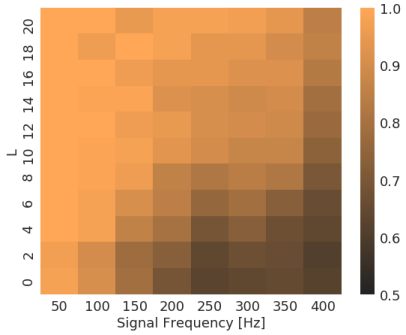


Figure 9. Heatmap of training accuracies of a network trained to predict the binarized value of a sine wave of given frequency (x-axis) defined on γ_L for various L (y-axis).

The results demonstrate a clear attenuation of the spectral bias as L grows. Moreover, Fig 8e suggests that the larger the L, the easier the learning task.

Experiment 6. Here, we adapt the setting of Experiment 5 to binary classification by simply thresholding the function λ at 0.5 to obtain a binary target signal. To simplify visualization, we only use signals with a single frequency mode k, such that $\lambda(z) = \sin(2\pi kz + \varphi)$. We train the same network on the resulting classification task with cross-entropy loss[14] for $k \in \{50, 100, ..., 350, 400\}$ and $L \in \{0, 2, ..., 18, 20\}$. The heatmap in Fig 9 shows the classification accuracy for each (k, L) pair. Fig 7 shows visualizations of the functions learned by the same network, trained on $(k, L) = (200, 20)$ under identical conditions up to random initialization.

Observe that increasing L (i.e. going up a column in Fig 9) results in better (classification) performance for the same target signal. This is the same behaviour as we observed in Experiment 5 (Fig 8a-d), but now with binary cross-entropy loss instead of the MSE.

[14]We use Pytorch's `BCEWithLogitsLoss`. Internally, it takes a sigmoid of the network's output (the logits) before evaluating the cross-entropy.

Discussion. These experiments hint towards a rich interaction between the shape of the manifold and the effective difficulty of the learning task. The key mechanism underlying this phenomenon (as we formalize below) is that the relationship between frequency spectrum of the network f and that of the fit $f \circ \gamma_L$ is mediated by the embedding map γ_L. In particular, we 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.

To understand this mathematically, we address the following questions: given a target function λ, how small can the frequencies of a solution f be such that $f \circ \gamma = \lambda$? And further, how does this relate to the geometry of the data-manifold $\mathcal{M}$ induced by γ? To find out, we write the Fourier transform of the composite function,

$$\widetilde{(f \circ \gamma)}(\mathbf{l}) = \int \mathbf{dk} \tilde{f}(\mathbf{k}) P_\gamma(\mathbf{l}, \mathbf{k}) \tag{14}$$

$$\text{where } P_\gamma(\mathbf{l}, \mathbf{k}) = \int_{[0,1]^m} \mathbf{dz}\, e^{i(\mathbf{k}\cdot\gamma(\mathbf{z}) - \mathbf{l}\cdot\mathbf{z})}$$

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 a 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

$$\mathbf{l} = J_\gamma(\bar{\mathbf{z}})\, \mathbf{k} \tag{15}$$

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 15 has a solution. Further, given that the components of γ (i.e. its coordinate functions) are defined

on an interval $[0, 1]^m$, one can use their Fourier series representation together with Eqn 15 to obtain a condition on their frequencies (shown in appendix C.7). More precisely, we find that the i-th component of the RHS in Eqn 15 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[15] enough for large $\mathbf{p}$, even when k_i is fixed.

This is precisely what Experiments 5 and 6 demonstrate in a minimal setting. From Eqn 13, 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.

5. Related Work

A number of works have focused on showing that neural networks are capable of approximating arbitrarily complex functions. Hornik et al. (1989); Cybenko (1989); Leshno et al. (1993) have shown that neural networks can be universal approximators when given sufficient width; more recently, Lu et al. (2017) proved that this property holds also for width-bounded networks. Montufar et al. (2014) showed that the number of linear regions of deep ReLU networks grows polynomially with width and exponentially with depth; Raghu et al. (2016) generalized this result and provided asymptotically tight bounds. There have been various results of the benefits of depth for efficient approximation (Poole et al., 2016; Telgarsky, 2016; Eldan & Shamir, 2016). These analysis on the expressive power of deep neural networks can in part explain why over-parameterized networks can perfectly learn random input-output mappings (Zhang et al., 2017a).

Our work more directly follows the line of research on implicit regularization in neural networks trained by gradient descent (Neyshabur et al., 2014; Soudry et al., 2017; Poggio et al., 2018; Neyshabur et al., 2017). In fact, while our Fourier analysis of deep ReLU networks also reflects the width and depth dependence of their expressivity, we focused on showing a learning bias of these networks towards simple functions with dominant lower frequency components. We view our results as a first step towards formalizing the findings of Arpit et al. (2017), where it is empirically shown that deep networks prioritize learning simple patterns of the data during training.

A few other works studied neural networks through the lens of harmonic analysis. For example, Candès (1999) used the ridgelet transform to build constructive procedures for approximating a given function by neural networks, in the case of oscillatory activation functions. This approach has been recently generalized to unbounded activation functions by Sonoda & Murata (2017). Eldan & Shamir (2016) use insights on the support of the Fourier spectrum of two-layer networks to derive a worse-case depth-separation result. Barron (1993) makes use of Fourier space properties of the target function to derive an architecture-dependent approximation bound. In a concurrent and independent work, Xu et al. (2018) make the same observation that lower frequencies are learned first. The subsequent work by Xu (2018) proposes a theoretical analysis of the phenomenon in the case of 2-layer networks with sigmoid activation, based on the spectrum of the sigmoid function.

In light of our findings, it is worth comparing the case of neural networks and other popular algorithms such that kernel machines (KM) and K-nearest neighbor classifiers. We refer to the Appendix E for a detailed discussion and references. In summary, our discussion there 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.

6. Conclusion

We studied deep ReLU networks through the lens of Fourier analysis. Several conclusions can be drawn from our analysis. While neural networks can approximate arbitrary functions, we find that they favour *low frequency* ones – hence they exhibit a bias towards smooth functions – a phenomenon that we called *spectral bias*. We also illustrated how the geometry of the data manifold impacts expressivity in a non-trivial way, as high frequency functions defined on complex manifolds can be expressed by lower frequency network functions defined in input space.

We view future work that explore the properties of neural networks in Fourier domain as promising. For example, the Fourier transform affords a natural way of measuring how fast a function can change within a small neighborhood in its input domain; as such, it is a strong candidate for quantifying and analyzing the *sensitivity* of a model – which in turn provides a natural measure of complexity (Novak et al., 2018). We hope to encourage more research in this direction.

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

what solution though?

Number of linear regions of deep NN with ReLU
* grows exponentially with depth of the NN
* grows polynomially with width

Acknowledgements

The authors would like to thank Joan Bruna, Rémi Le Priol, Vikram Voleti, Ullrich Köthe, Steffen Wolf, Lorenzo Cerrone, Sebastian Damrich, as well as the anonymous reviewers for their valuable feedback.

References

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

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Barron, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

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

Bergner, S., Möller, T., Weiskopf, D., and Muraki, D. J. A spectral analysis of function concatenations and its implications for sampling in direct volume visualization.

Braun, M. L., Lange, T., and Buhmann, J. M. Model selection in kernel methods based on a spectral analysis of label information. In *Joint Pattern Recognition Symposium*, pp. 344–353. Springer, 2006.

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

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

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

Diaz, R., Le, Q.-N., and Robins, S. Fourier transforms of polytopes, solid angle sums, and discrete volume. *arXiv preprint arXiv:1602.08593*, 2016.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2018.

Eldan, R. and Shamir, O. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pp. 907–940, 2016.

Fasshauer, G. E. Positive definite kernels: past, present and future. *Dolomite Research Notes on Approximation*, 4: 21–63, 2011.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

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

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

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pp. 6231–6239, 2017.

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

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

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5949–5958, 2017.

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

Poggio, T., Kawaguchi, K., Liao, Q., Miranda, B., Rosasco, L., Boix, X., Hidary, J., and Mhaskar, H. Theory of deep learning iii: the non-overfitting puzzle. Technical report, Technical report, CBMM memo 073, 2018.

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

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

Rasmussen, C. E. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pp. 63–71. Springer, 2004.

Serov, V. *Fourier series, Fourier transform and their applications to mathematical physics*. Springer, 2017.

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

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *arXiv preprint arXiv:1710.10345*, 2017.

Spivak, M. *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus*. CRC press, 2018.

Telgarsky, M. Benefits of depth in neural networks. *Conference on Learning Theory (COLT), 2016*, 2016.

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

Xu, Z.-Q. J., Zhang, Y., and Xiao, Y. Training behavior of deep neural network in frequency domain. *arXiv preprint arXiv:1807.01251*, 2018.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations (ICLR)*, 2017a.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017b.

Zhang, L., Naitzat, G., and Lim, L.-H. Tropical geometry of deep neural networks. *arXiv preprint arXiv:1805.07091*, 2018.

A. Experimental Details

A.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 2, we show an example target function and the predictions of the network trained on it (over the iterations), and in figure 10 we plot the loss curves.

A.2. Experiment 5

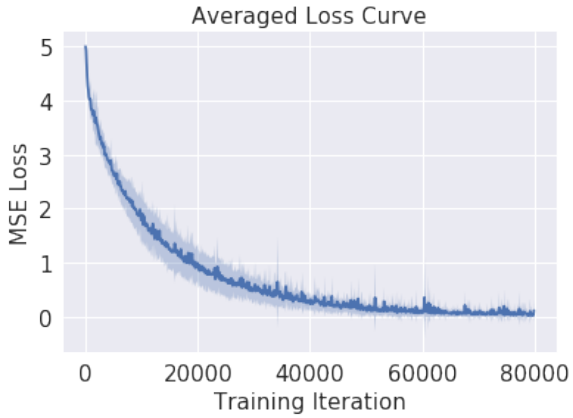
We use the same 6-layer deep 256-unit wide network and define the target function

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

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 8 shows the evolution of the spectrum as training progresses for $L = 0, 4, 10, 16$, and Fig 8e shows the corresponding loss curves.

A.3. Experiment 3

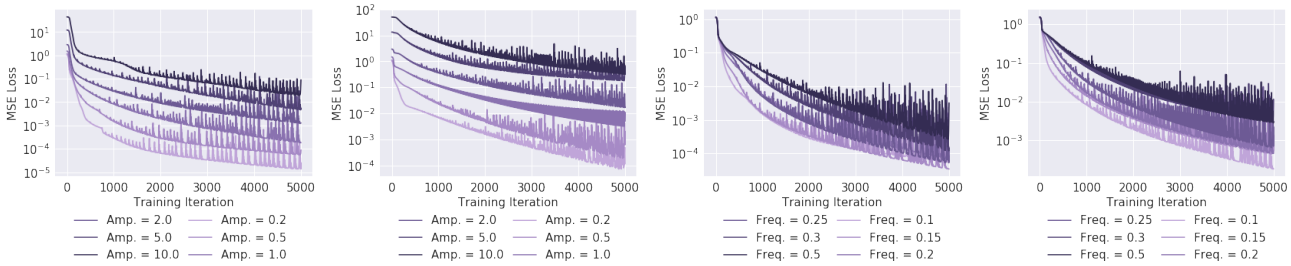
In Figure 11, we show the training curves corresponding to Figure 4.



(a) Equal Amplitudes.

(b) Increasing Amplitudes.

Figure 10. Loss curves averaged over multiple runs. (cf. Experiment 1)



(a) $k = 0.1$ (b) $k = 1$ (c) $\beta = 0.5$ (d) $\beta = 1.$

Figure 11. (a,b,c,d): Training curves for various settings of noise amplitude β and frequency k corresponding to Figure 4.

A.4. Experiment 4

Consider the Gaussian Radial Basis Kernel, given by:

$$k : X \times X \to \mathbb{R},\ k_\sigma(\mathbf{x}, \mathbf{y}) \mapsto \exp\left(\frac{\|\mathbf{x} - \mathbf{y}\|}{\sigma^2}\right) \tag{16}$$

where X is a compact subset of $\mathbb{R}^d$ and $\sigma \in \mathbb{R}_+$ is defined as the width of the kernel[16]. Since k is positive definite (Fasshauer, 2011), Mercer's Theorem can be invoked to express it as:

$$k(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{\infty} \lambda_i \varphi_n(\mathbf{x}) \varphi_n(\mathbf{y}) \tag{17}$$

where φ_n is the eigenfunction of k satisfying:

$$\int k(\mathbf{x}, \mathbf{y}) \varphi_n(\mathbf{y}) \mathbf{dy} = \langle k(\mathbf{x}, \cdot), \varphi_n \rangle = \lambda_n \varphi_n(\mathbf{x}) \tag{18}$$

Due to positive definiteness of the kernel, the eigenvalues λ_i are non-negative and the eigenfunctions φ_n form an orthogonal basis of $L^2(X)$, i.e. $\langle \varphi_i, \varphi_j \rangle = \delta_{ij}$. The analogy to the final case is easily seen: let $X = \mathbf{x}_i{}_{i=1}^N$ be the set of samples, $f : X \to \mathbb{R}$ a function. One obtains (cf. Chapter 4 (Rasmussen, 2004)):

$$\langle k(\mathbf{x}, \cdot), f \rangle = \sum_{i=1}^{N} k(\mathbf{x}, \mathbf{x}_i) f_i \tag{19}$$

where $f_i = f(\mathbf{x}_i)$. Now, defining K as the positive definite kernel matrix with elements $K_{ij} = k(\mathbf{x}_i \mathbf{x}_j)$, we consider it's eigendecomposition $V \Lambda V^T$ where Λ is the diagonal matrix of (w.l.o.g sorted) eigenvalues $\lambda_1 \leq ... \leq \lambda_N$ and the columns of V are the corresponding eigenvectors. This yields:

$$k(\mathbf{x}_i, \mathbf{x}_j) = K_{ij} = (V \Lambda V^T)_{ij} = \sum_{n=1}^{N} \lambda_n v_{ni} v_{nj}$$

$$= \sum_{n=1}^{N} \lambda_n \varphi_n(\mathbf{x}_i) \varphi_n(\mathbf{x}_j) \implies \varphi_n(\mathbf{x}_i) = v_{ni} \tag{20}$$

[16] We drop the subscript σ to simplify the notation.

Like in (Braun et al., 2006), we define the *spectrum* $\tilde{f}[n]$ of the function f as:

$$\tilde{f}[n] = \langle f, \varphi_n \rangle = \mathbf{f} \cdot \mathbf{v}_n \tag{21}$$

where $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_N))$. The value n can be thought of a generalized notion of *frequency*. Indeed, it is known (Fasshauer, 2011; Rasmussen, 2004), for instance, that the eigenfunctions φ_n resemble sinusoids with increasing frequencies (for increasing n or decreasing λ_n). In Figure 6, we plot the eigenvectors $\mathbf{v}_0$ and $\mathbf{v}_N$ for $\{\mathbf{x}_i\}_{i=1}^{50}$ uniformly spaced between $[0, 1]$. Further, in Figure ? we evaluate the discrete Fourier transform of all $N = 50$ eigenvectors, and find that the eigenfunction index n does indeed coincide with frequency k. Finally, we remark that the link between signal complexity and the spectrum is extensively studied in (Braun et al., 2006).

A.4.1. Loss Curves Accompanying Figure 5

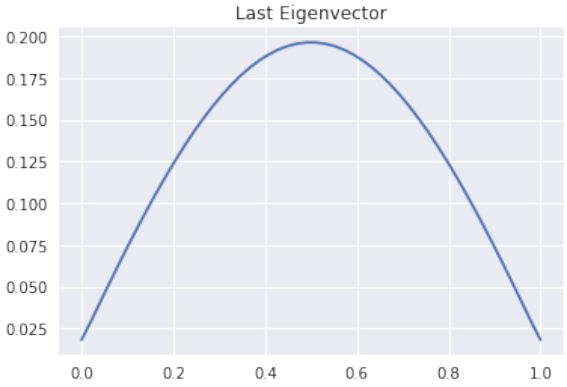
A.5. Qualitative Ablation over Architectures

Theorem 1 exposes the relationship between the fourier spectrum of a network and its depth, width and max-norm of parameters. The following experiment is a qualitative ablation study over these variables.

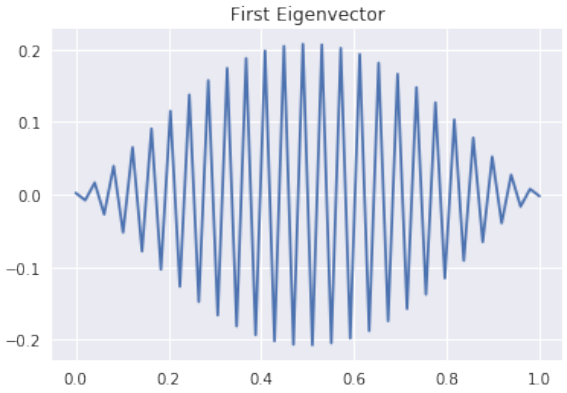
Experiment 7. In this experiment, we fit various networks to the δ-function at $x = 0.5$ (see Fig 14a). Its spectrum is constant for all frequencies (Fig 14b), which makes it particularly useful for testing how well a given network can fit large frequencies. Fig 17 shows the ablation over weight clip (i.e. max parameter max-norm), Fig 15 over depth and Fig 16 over width. Fig 18 exemplarily shows how the network prediction evolves with training iterations. All networks are trained for 60K iterations of full-batch gradient descent under identical conditions (Adam optimizer with $lr = 0.0003$, no weight decay).

We make the following observations.

(a) Fig 15 shows that increasing the depth (for fixed width) significantly improves the network's ability to fit higher frequencies (note that the depth increases linearly).



(a) Eigenvector with the largest eigenvalue ($n = 1$).



(b) Eigenvector with the smallest eigenvalue ($n = 50$).

Figure 12. Two extreme eigenvectors of the Gaussian RBF kernel for 50 uniformly spaced samples between 0 and 1.

Figure 13. Loss curves for the Figure 5. We find that the validation loss dips at around the 200th iteration.

(b) Fig 16 shows that increasing the width (for fixed depth) also helps, but the effect is considerably weaker (note that the width increases exponentially).

(c) Fig 17 shows that increasing the weight clip (or the max parameter max-norm) also helps the network fit higher frequencies.

The above observations are all consistent with Theorem 1, and further show that lower frequencies are learned first (i.e. the spectral bias, cf. Experiment 1). Further, Figure 17 shows that constraining the Lipschitz constant (weight clip) prevents the network from learning higher frequencies, furnishing evidence that the $\mathcal{O}(L_f)$ bound can be tight.

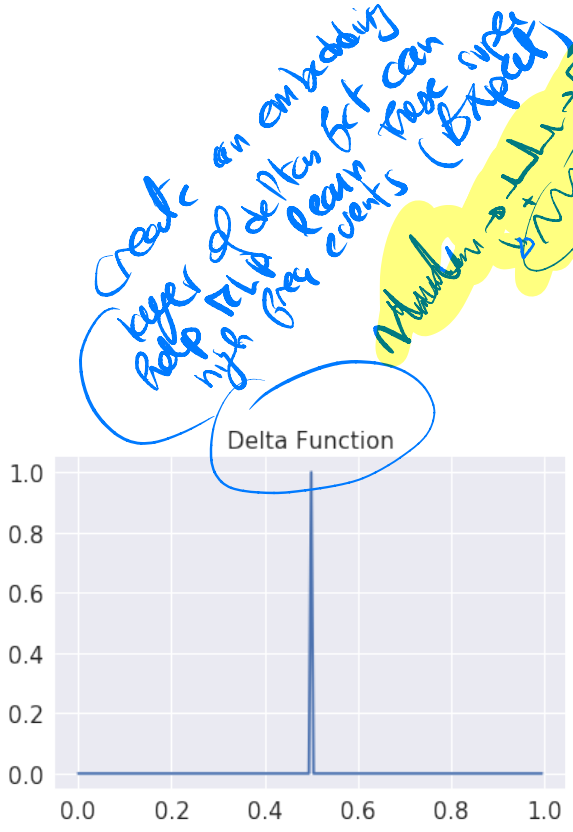
A.6. MNIST: A Proof of Concept

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.

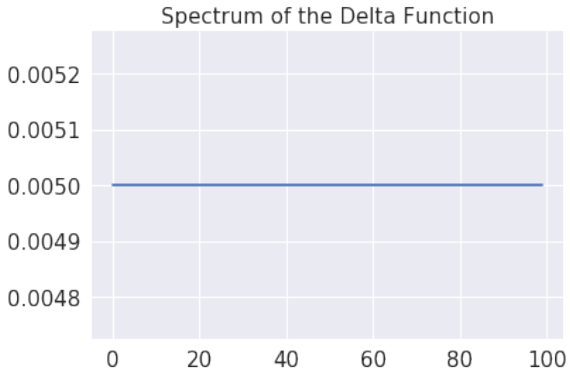
Experiment 8. In this experiment, we investigate if it is easier to learn a signal on a more realistic data-manifold like that of MNIST (assuming the manifold hypothesis is true), and compare with a flat manifold of the same dimension. To that end, we use the 64-dimensional feature-space $\mathcal{E}$ of a denoising[17] 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[18] of a 64-dimensional hyperplane in X.

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

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



(a) Sampled δ-function at $x = 0.5$.



(b) Constant Spectrum of the δ-function.

Figure 14. The target function used in Experiment 7.

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 19 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).

A.7. 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.

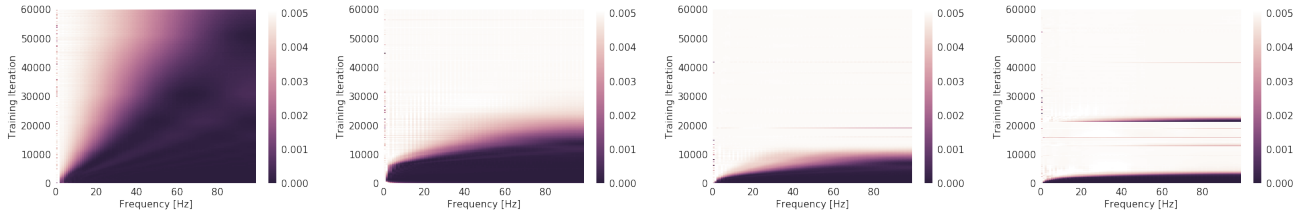
Experiment 9. Using AutoNEB (Kolsbjerg et al., 2016), we construct paths between (adversarial) Cifar-10 images that are classified by a ResNet20 to be all of the same target class. AutoNEB bends a linear path between points in some space $\mathbb{R}^m$ so that some maximum energy along the path is minimal. 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 and generate adversarial samples from the latter. Then, paths between all pairs from the whole set of images are computed.

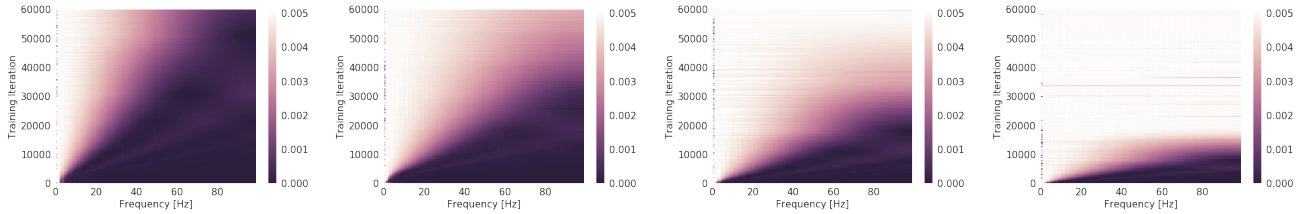
The AutoNEB parameters are chosen as follows: We run four NEB iterations with 10 steps of SGD with learning rate

in $\mathbb{R}^3$.



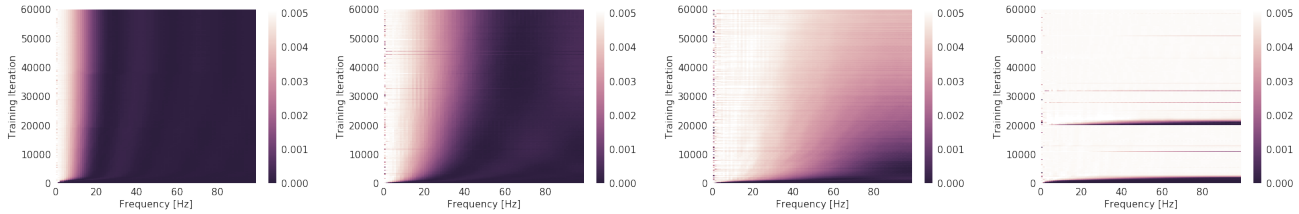
(a) Depth $= 3$. (b) Depth $= 4$. (c) Depth $= 5$. (d) Depth $= 6$.

Figure 15. Evolution with training iterations (y-axis) of the Fourier spectrum (x-axis for frequency, and colormap for magnitude) for a network with **varying depth**, width $= 16$ and weight clip $= 10$. The spectrum of the target function is a constant 0.005 for all frequencies.



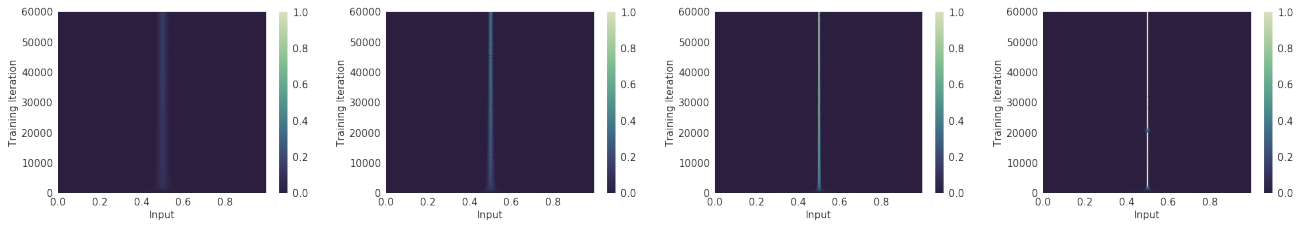
(a) Width $= 16$. (b) Width $= 32$. (c) Width $= 64$. (d) Width $= 128$.

Figure 16. Evolution with training iterations (y-axis) of the Fourier spectrum (x-axis for frequency, and colormap for magnitude) for a network with **varying width**, depth $= 3$ and weight clip $= 10$. The spectrum of the target function is a constant 0.005 for all frequencies.



(a) Weight Clip $= 0.1$. (b) Weight Clip $= 0.15$. (c) Weight Clip $= 0.2$. (d) Weight Clip $= 2$.

Figure 17. Evolution with training iterations (y-axis) of the Fourier spectrum (x-axis for frequency, and colormap for magnitude) for a network with **varying weight clip**, depth $= 6$ and width $= 64$. The spectrum of the target function is a constant 0.005 for all frequencies.



(a) Weight Clip $= 0.1$. (b) Weight Clip $= 0.15$. (c) Weight Clip $= 0.2$. (d) Weight Clip $= 2$.

Figure 18. Evolution with training iterations (y-axis) of the network prediction (x-axis for input, and colormap for predicted value) for a network with **varying weight clip**, depth $= 6$ and width $= 64$. The target function is a δ peak at $x = 0.5$.

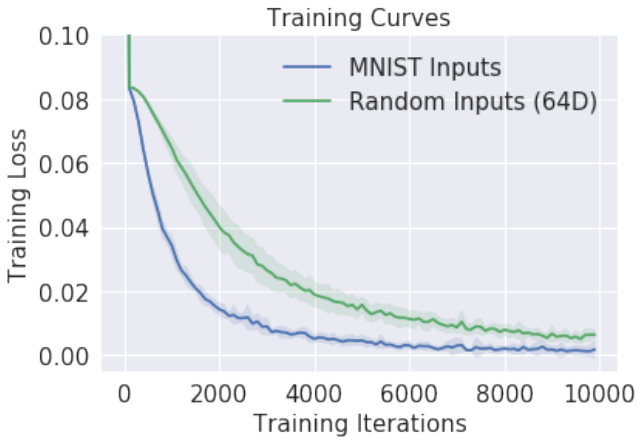


Figure 19. Loss curves of two identical networks trained to regress white-noise under identical conditions, one on MNIST reconstructions from a DAE with 64 encoder features (blue), and the other on 64-dimensional random vectors (green).

Figure 20. Path between CIFAR-10 adversarial examples (e.g. "frog" and "automobile", such that all images are classified as "airplane").

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 21 shows connecting training to adversarial images and Figure 20 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.

B. The Continuous Piecewise Linear Structure of Deep ReLU Networks

We consider the class of ReLU network functions $f : \mathbb{R}^d \mapsto \mathbb{R}$ defined by Eqn. 1. Following the terminology of (Raghu et al., 2016; Montufar et al., 2014), each linear region of the network then corresponds to a unique *activation pattern*, wherein each hidden neuron is assigned an activation variable $\epsilon \in \{-1, 1\}$, conditioned on whether its input is positive or negative. ReLU networks can be explictly expressed as a sum over all possible activation patterns, as in

Figure 21. 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.

the following lemma.

Lemma 3. *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, as defined in Eqn. 1, can be expressed as*

$$f(\mathbf{x}) = \sum_{\epsilon^{(1)},\cdots\epsilon^{(L)}} 1_{P_{f,\epsilon}}(\mathbf{x}) \left(T^{(L+1)} \circ T^{(L)}_{\epsilon^{(L)}} \circ \cdots \circ T^{(1)}_{\epsilon^{(1)}}\right)(\mathbf{x}) \tag{22}$$

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 $k = 1, \cdots, L$):

$$(\epsilon_k)_i \left(T^{(k)} \circ T^{(k-1)}_{\epsilon^{(k-1)}} \circ \cdots \circ T^{(1)}_{\epsilon^{(1)}}\right)(\mathbf{x})_i \geq 0, \quad i = 1, \cdots d_k \tag{23}$$

f is therefore affine on each of the polytopes $P_{f,\epsilon}$, which finitely partition the input space $\mathbb{R}^d$ to convex polytopes. Remarkably, the correspondence between ReLU networks and CPWL functions goes both ways: Arora et al. (2018) show that every CPWL function can 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 $f_\epsilon(x) = W_\epsilon x + b_\epsilon$, where in particular

$$W_\epsilon = W^{(L+1)} W^{(L)}_{\epsilon_L} \cdots W^{(1)}_{\epsilon_1} \in \mathbb{R}^{1 \times d}, \tag{24}$$

where $W^{(k)}_\epsilon$ is obtained from $W^{(k)}$ by setting its jth column to zero whenever $(\epsilon_k)_j = -1$.

C. Fourier Analysis of ReLU Networks

C.1. Proof of Lemma 1

Proof. **Case 1: The function f has compact support.** 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 (see e.g Spivak (2018)), the integral of its divergence is a pure boundary term. Since we restricted to functions with compact support, the theorem yields

$$\int \nabla_{\mathbf{x}} \cdot \left[\mathbf{k} f(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}}\right] \mathbf{dx} = 0 \tag{25}$$

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,

$$\hat{f}(\mathbf{k}) = \frac{1}{-ik^2} \mathbf{k} \cdot \int (\nabla_{\mathbf{x}} f)(\mathbf{x})\, e^{-i\mathbf{k}\cdot\mathbf{x}} \tag{26}$$

Now, within each polytope of the decomposition (22), f is affine so its gradient is a constant vector, $\nabla_{\mathbf{x}} f_\epsilon = W_\epsilon^T$, which gives the desired result (1).

Case 2: The function f does not have compact support. Without the assumption of compact support, the function f is not squared-integrable. The Fourier transform therefore only exists in the sense of distributions, as defined below.

Let $\mathcal{S}$ be the Schwartz space over $\mathbb{R}^d$ of rapidly decaying test functions which together with its derivatives decay to zero as $x \to \infty$ faster than any power of x. A tempered distribution is a continuous linear functional on $\mathcal{S}$. A function f that doesn't grow faster than a polynomial at infinity can be identified with a tempered distribution T_f as:

$$T_f : \mathcal{S} \to \mathbb{R},\ \varphi \mapsto \langle f, \varphi \rangle = \int_{\mathbb{R}^d} f(\mathbf{x})\varphi(\mathbf{x})\mathbf{dx} \tag{27}$$

In the following, we shall identify T_f with f. The Fourier transform $\tilde{f}$ of the tempered distribution is defined as:

$$\langle \tilde{f}, \varphi \rangle := \langle f, \tilde{\varphi} \rangle \tag{28}$$

where $\tilde{\varphi}$ is the Fourier transform of φ. In this sense, the standard notion of the Fourier transform is generalized to functions that are not squared-integrable.

Consider the continuous piecewise-linear ReLU network $f : \mathbb{R}^d \to \mathbb{R}$. Since it can grow at most linearly, we interpret it as a tempered distribution on $\mathbb{R}^d$. Recall that the linear regions P_ϵ are enumerated by ϵ. Let f_ϵ be the restriction of f to P_ϵ, making $f_\epsilon(\mathbf{x}) = W_\epsilon^T \mathbf{x}$. The distributional derivative of f is given by:

$$\nabla_{\mathbf{x}} f = \sum_\epsilon \nabla_{\mathbf{x}} f_\epsilon \cdot 1_{P_\epsilon} = \sum_\epsilon W_\epsilon^T 1_{P_\epsilon} \tag{29}$$

where 1_{P_ϵ} is the indicator over P_ϵ and we used $\nabla_{\mathbf{x}} f_\epsilon = W_\epsilon^T$. It then follows from elementary properties of Schwartz spaces (see e.g. Chapter 16 of Serov (2017)) that:

$$\widetilde{[\nabla_{\mathbf{x}} f]}(\mathbf{k}) = -i\mathbf{k}\tilde{f}(\mathbf{k}) \tag{30}$$

$$\implies \tilde{f}(\mathbf{k}) = \frac{1}{-ik^2}\mathbf{k} \cdot \widetilde{[\nabla_{\mathbf{x}} f]}(\mathbf{k}) \tag{31}$$

Together with Eqn 29 and linearity of the Fourier transform, this 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 $\mathbb{R}^d$, such that $1 \leq m \leq d$. Denote by $\mathbf{k} \in \mathbb{R}^d$ 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:

$$\tilde{F} = \text{vol}_F(F)e^{i\Phi_{\mathbf{k}}} \tag{32}$$

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

$$\tilde{F} = i \sum_{G \in \partial F} \frac{\text{Proj}_F(\mathbf{k}) \cdot \mathbf{N}_F(G)}{\|\text{Proj}_F(\mathbf{k})\|^2} \tilde{G}(\mathbf{k}) \tag{33}$$

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 directed acyclic graph (DAG) 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:

$$W_{F,G} = i\frac{\text{Proj}_F(\mathbf{k}) \cdot \mathbf{N}_F(G)}{\|\text{Proj}_F(\mathbf{k})\|^2} \tag{34}$$

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 $T : F_0 = P \to F_1 \to ... \to F_{|T|}$ where each $F_{i+1} \in \partial F_i$ has one dimension less than F_i. For a given path and $\mathbf{k}$, the terminal node for this path, F_{n_T}, is the first polytope for which $\text{Proj}_{F_{n_T}}(\mathbf{k}) = 0$. The final Fourier transform is obtained by multiplying the weights along each path and summing over all tree paths:

$$\tilde{1}_P(\mathbf{k}) = \sum_T \prod_{i=0}^{|T|-1} W_{F_i,F_{i+1}} \text{vol}_{F_{|T|}}(F_{|T|}) e^{i\Phi_{\mathbf{k}}} \tag{35}$$

where $\Phi^{(T)} = \mathbf{k} \cdot \mathbf{x}_0^T$ for an arbitrary point $\mathbf{x}_0^T$ in $F_{|T|}$.

To write this as a weighted sum of indicator functions, as in Lemma 2, let $\mathcal{T}_n$ denote the set of all tree paths T of length n, i.e. $|T| = n$. For a tree path T, let $S(T)$ be the orthogonal to the terminal node F_n, i.e the vectors $\mathbf{k}$ such

that $\text{Proj}_{F_n}(\mathbf{k}) = 0$. The sum over T in Eqn (35) can be split as:

$$\tilde{1}_P = \sum_{n=0}^{d} \frac{1_{G_n}}{k^n} \sum_{T \in \mathcal{T}_n} 1_{S(T)} \prod_{i=0}^{n-1} \bar{W}_{F_i^T, F_{i+1}^T} \text{vol}_{F_n^T}(F_n^T) e^{i\Phi_{\mathbf{k}}^{(T)}} \tag{36}$$

where $\bar{W}_{F,G} = kW_{F,G}$ and $G_n = \bigcup_{T \in \mathcal{T}_n} S(T)$. In words, G_n is the set of all vectors $\mathbf{k}$ that are orthogonal to some n-codimensional face of the polytope. We identify:

$$D_q = \sum_{T \in \mathcal{T}_n} 1_{S(T)} \prod_{i=0}^{n-1} \bar{W}_{F_i^T, F_{i+1}^T} \text{vol}_{F_n^T}(F_n^T) e^{i\Phi_{\mathbf{k}}^{(T)}} \tag{37}$$

and $D_0(\mathbf{k}) = \text{vol}(P)$ to obtain Lemma 2. Observe that D_n depends on k only via the phase term $e^{i\Phi_{\mathbf{k}}^{(T)}}$, implying that $D_n = \Theta(1)\,(k \to \infty)$.

Informally, for a generic vector $\mathbf{k}$, all paths terminate at the zero-dimensional vertices of the original polytope, i.e. $\dim(F_n) = 0$, implying the length of the path n equals the number of dimensions d, yielding a $\mathcal{O}(k^{-d})$ spectrum. The exceptions occur if a path terminates prematurely, because $\mathbf{k}$ happens to lie orthogonal to some $d-r$-dimensional face F_r in the path, in which case we are left with a $\mathcal{O}(k^{-r})$ term (with $r < d$) which dominates asymptotically. Note that all vectors orthogonal to the $d-r$ dimensional face F_r lie on a r-dimensional subspace of $\mathbb{R}^d$. 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 $\mathcal{O}(k^{-d})$) lies on a finite union of closed subspaces of dimension r (with $r < d$). 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 $\mathcal{O}(k^{-d})$ for *almost all* $\mathbf{k}$'s.

C.3. On Theorem 1

Equation 6 can be obtained by swapping the (finite) sum over ϵ in Lemma 1 with that over the paths T in Eqn 36. In particular, we have:

$$\tilde{f} = \sum_{n=0}^{d} \frac{1_{H_n}}{k^{n+1}} \sum_{\epsilon} W_\epsilon D_n^\epsilon 1_{G_n^\epsilon} \tag{38}$$

Now, the sum $\sum_\epsilon W_\epsilon D_n^\epsilon(\hat{\mathbf{k}}) I_{G_n^\epsilon}(\mathbf{k})$ is supported on the union:

$$H_n = \bigcup_{\epsilon} G_n^\epsilon \tag{39}$$

Identifying:

$$C_n(\cdot, \theta) = \sum_{\epsilon} W_\epsilon D_n^\epsilon 1_{G_n^\epsilon} \tag{40}$$

where $C_n(\cdot, \theta) = \mathcal{O}(1)\,(k \to \infty)$, we obtain Theorem 1. Further, if N_f is the number of linear regions of the network and $L_f = \max_\epsilon \|W_\epsilon\|$, we see that $C_n = \mathcal{O}(L_f N_f)$. Indeed, in Appendix A.5, we empirically find that relaxing the constraint on the weight clip (which can be identified with L_f) enabled the network to fit higher frequencies, implying that the $\mathcal{O}(L_f)$ bound can be tight.

C.4. Spectral Decay Rate of the Parameter Gradient

Proposition 1. *Let θ be a generic parameter of the network function f. The spectral decay rate of $\partial\tilde{f}/\partial\theta$ is $\mathcal{O}(k\tilde{f})$.*

Proof. For a fixed $\hat{\mathbf{k}}$, observe from Eqn 38 and Eqn 37 that the only terms dependent on k are the pure powers k^{-n-1} and the phase terms $e^{i\Phi_{\mathbf{k}}^{(T)}}$, where $\Phi_{\mathbf{k}}^{(T)} = k\hat{\mathbf{k}} \cdot \mathbf{x}_0^{q(T)}$. However, the term $\mathbf{x}_0^{q(T)}$ is in general a function of θ, and consequently the partial derivative of $e^{i\Phi_{\mathbf{k}}^{(T)}}$ w.r.t θ yields a term that is proportional to k. This term now dominates the asymptotic behaviour as $k \to \infty$, adding an extra power of k to the total spectral decay rate of $\tilde{f}$. □

Therefore, if $f = \mathcal{O}(k^{-\Delta-1})$ where Δ is the codimension of the highest dimensional polytope $\hat{\mathbf{k}}$ is orthogonal to, we have that $\partial f/\partial\theta = \mathcal{O}(k^{-\Delta})$.

C.5. Convergence Rate of a Network Trained on Pure-Frequency Targets

In this section, we derive an asymptotic bound on the convergence rate under the assumption that the target function has only one frequency component.

Proposition 2. *Let $\lambda : [0, 1] \to \mathbb{R}$ be a target function sampled in its domain at N uniformly spaced points. Suppose that its Fourier transform after sampling takes the form: $\tilde{\lambda}(k) = A_0\delta_{k,k_0}$, where δ is the Kronecker delta. Let f be a neural network trained with full-batch gradient descent with learning rate η on the Mean Squared Error, and denote by f_t the state of the network at time t. Let $h(\cdot, t) = f_t - \lambda$ be the residual at time t. We have that:*

$$\left|\frac{\partial\tilde{h}(k_0, t)}{\partial t}\right| = \mathcal{O}(k_0^{-1}) \tag{41}$$

Proof. Consider that:

$$\left|\frac{\partial\tilde{h}(k_0)}{\partial t}\right| = \left|\frac{\partial\tilde{f}(k_0)}{\partial\theta}\right| \left|\frac{\partial\theta}{\partial t}\right| \tag{42}$$

$$= \left|\eta\frac{\partial\tilde{f}}{\partial\theta}\right| \left|\frac{\partial\mathcal{L}[\tilde{f}, \tilde{\lambda}]}{\partial\theta}\right| \tag{43}$$

where $\mathcal{L}$ is the sampled MSE loss and the first term is $\mathcal{O}(k_0^{-1})$ as can be seen from Proposition 1. With Parce-

val's Theorem, we obtain:

$$\mathcal{L}[f,\lambda] = \sum_{x=0}^{N-1} |f(x)-\lambda(x)|^2 = \sum_{k=-N/2}^{N/2-1} |\tilde{f}(k)-\tilde{\lambda}(k)|^2 = \mathcal{L}[\tilde{f},\tilde{\lambda}] \tag{44}$$

For the magnitude of parameter gradient, we obtain:

$$\begin{aligned}\left|\frac{\partial\mathcal{L}[\tilde{f},\tilde{\lambda}]}{\partial\theta}\right| &= 2\left|\sum_{k=-N/2}^{N/2-1} \mathrm{Re}[\tilde{f}(k)-\tilde{\lambda}(k)]\frac{\partial\tilde{f}(k)}{\partial\theta}\right| \\ &\leq 2\sum_{k=-N/2}^{N/2-1} |\tilde{f}(k)-\tilde{\lambda}(k)|\left|\frac{\partial\tilde{f}(k)}{\partial\theta}\right| \\ &\leq 2\left|A_0\frac{\partial\tilde{f}(k_0)}{\partial\theta}\right| + 2\sum_{k=-N/2}^{N/2-1}\left|\tilde{f}(k)\frac{\partial\tilde{f}(k)}{\partial\theta}\right|\end{aligned} \tag{45}$$

where in the last line we used that $\tilde{\lambda}$ is a Kronecker-δ in the Fourier domain. Now, the second summand does not depend on k_0, but the first summand is again $\mathcal{O}(k_0^{-1})$. □

C.6. Proof of the Lipschtiz bound

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

$$\|W_\epsilon\| \leq L_f \leq \prod_{k=1}^{L+1}\|W^{(k)}\| \leq \|\theta\|_\infty^{L+1}\sqrt{d}\prod_{k=1}^{L} d_k \tag{46}$$

Proof. 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[19], 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:

$$\|W\| = \max_{\|\mathbf{h}\|=1}\|W\mathbf{h}\| \tag{47}$$

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:

$$\|W\| \leq \sqrt{IJ}\|\theta\|_\infty \tag{48}$$

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. □

C.7. The Fourier Transform of a Function Composition

Consider Equation 14. 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 $\partial_\nu = \partial/\partial x_\nu$):

$$l_\nu = k_\mu\partial_\nu\gamma_\mu(\mathbf{z}) \tag{49}$$

Now, we impose periodic boundary conditions[20] on the components of γ, and without loss of generality we let the period be 2π. Further, we require that the manifold be contained in a box[21] of some size in $\mathbb{R}^d$. The μ-th component γ_μ can now be expressed as a Fourier series:

$$\begin{aligned}\gamma_\mu(\mathbf{z}) &= \sum_{\mathbf{p}\in\mathbb{Z}^m}\tilde{\gamma}_\mu[\mathbf{p}]e^{-ip_\rho z_\rho} \\ \partial_\nu\gamma_\mu(\mathbf{z}) &= \sum_{\mathbf{p}\in\mathbb{Z}^m} -ip_\nu\tilde{\gamma}_\mu[\mathbf{p}]e^{-ip_\rho z_\rho}\end{aligned} \tag{50}$$

Equation 50 can be substituted in equation 49 to obtain:

$$l\hat{l}_\nu = -ik\sum_{\mathbf{p}\in\mathbb{Z}^m} p_\nu\hat{k}_\mu\tilde{\gamma}_\mu[\mathbf{p}]e^{-ip_\rho z_\rho} \tag{51}$$

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 $\hat{k}_\nu$).

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

[20] This is possible whenever γ is defined on a bounded domain, e.g. on $[0,1]^m$.

[21] This is equivalent to assuming that the data lies in a bounded set.

D. Volume of *High-Frequency Parameters* in Parameter Space

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 4. *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[22] $|\tilde{f}_\theta(k)| = \mathcal{O}(k^{-\Delta-1})$ for an integer $1 \leq \Delta \leq d$. In the worse case where $\Delta = 1$, 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 $|\tilde{f}_\theta(k)| \geq 0$.*

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

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:

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

□

[22] Note from Theorem 1 that Δ implicitly depends only on the unit vector $\hat{\mathbf{k}}$.

E. Kernel Machines and KNNs

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

E.1. Kernel Machines vs DNNs

Given that 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.

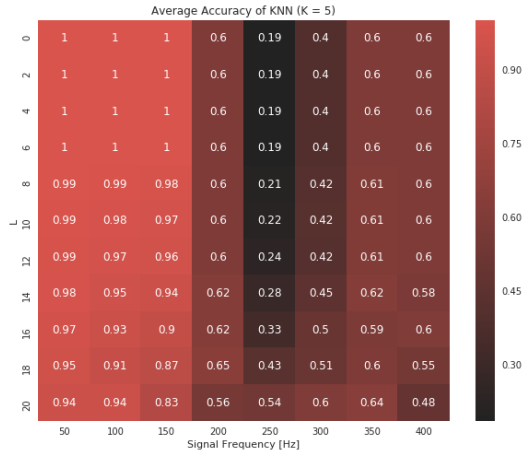
E.2. K-NN Classifier vs. DNN classifier

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

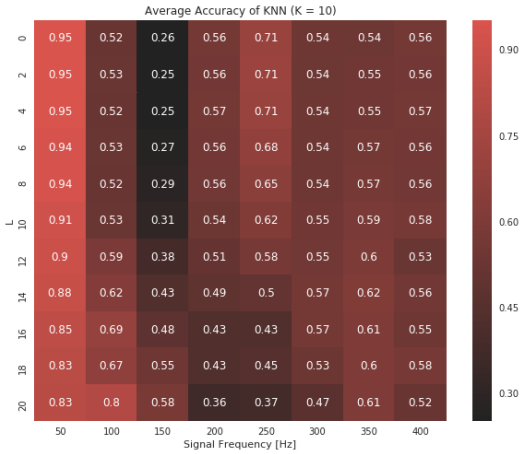
$K \to \infty$ and $K/N \to 0$, where N is the number of training samples. However, because it is a memory based algorithm, it is prohibitively slow for large datasets. Since the smoothness of a KNN prediction function is not well studied, we compare the smoothness between KNN and DNN. For various values 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 this interval, we evaluate the 2D FFT and integrate out the angular components (where the angle is parameterized by φ) to obtain $\zeta(k)$:

$$\zeta(k) = \frac{d}{dk} \int_0^k dk' k' \int_0^{2\pi} d\varphi |\tilde{f}(k', \varphi)| \tag{52}$$

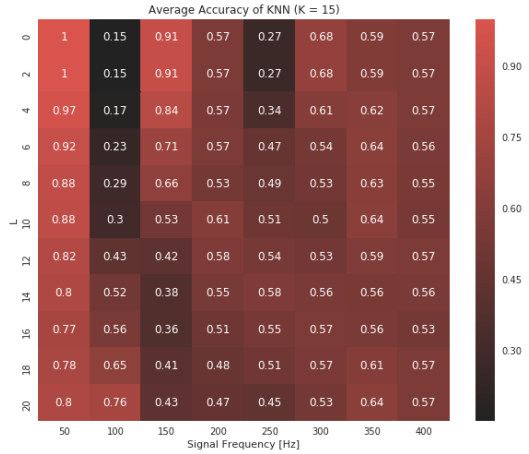
Finally, we plot $\zeta(k)$ for various K in figure 22e. Furthermore, we train a DNN on the very same dataset and overlay the radial spectrum of the resulting probability map on the same plot. We find that while DNN's are as expressive as a $K = 1$ KNN classifier at lower (radial) frequencies, the frequency 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 Fig. 9 with KNNs (see Fig. 22) 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.



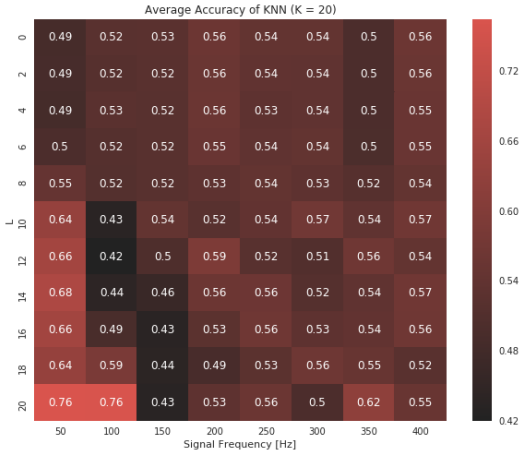
(a) $K = 5$.



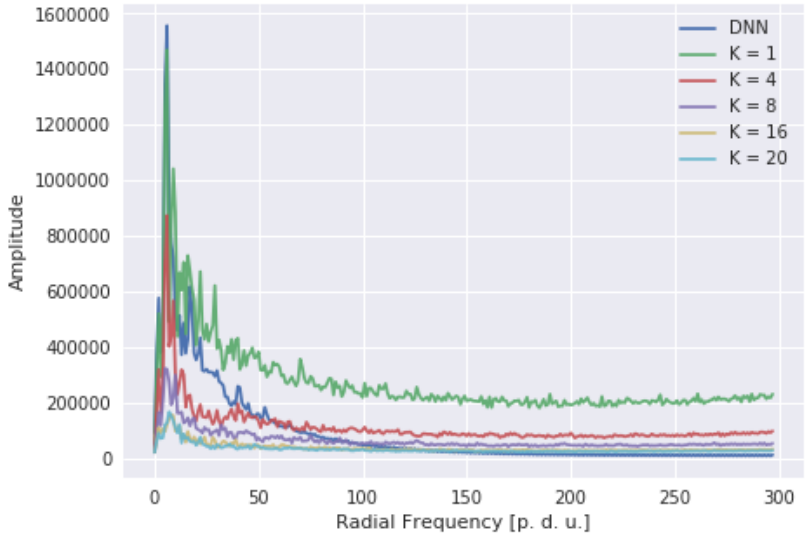
(b) $K = 10$.



(c) $K = 15$.



(d) $K = 20$.



(e) Frequency spectrum

Figure 22. (a,b,c,d): Heatmaps of training accuracies (L-vs-k) of KNNs for various K. When comparing with figure 9, note that the y-axis is flipped. (e): 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.