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

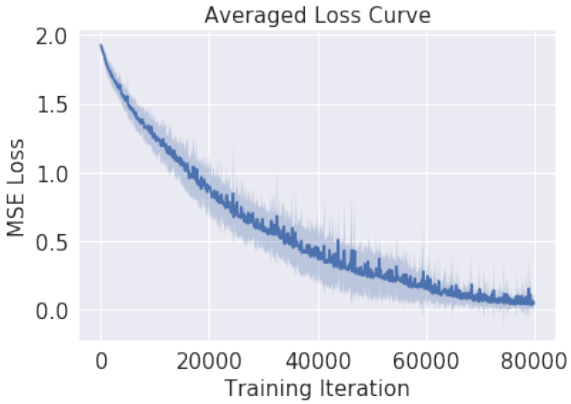
$$\lambda : \mathcal{D} \to \mathbb{R},\ z \mapsto \lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i)$$

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

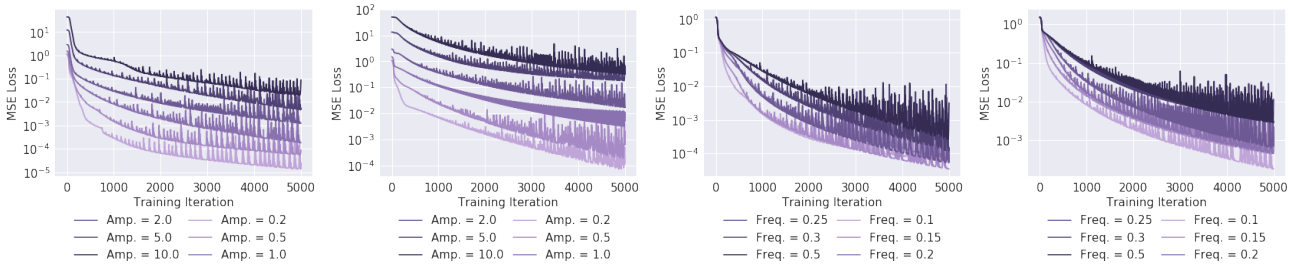
In Figure 11, we show the training curves corresponding to Figure 4.

(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

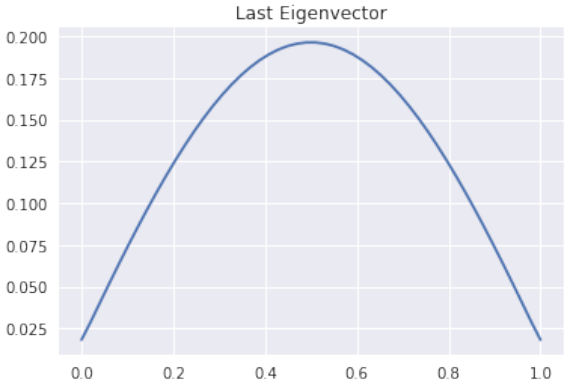
A.5. Qualitative Ablation over Architectures

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.

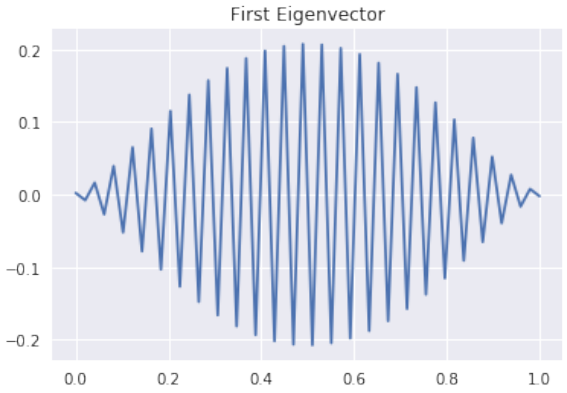
Experiment 7. In this experiment, we fit various networks to the δ-function at $x = 0.5$ (see Fig 14a). Its spectrum is constant for all frequencies (Fig 14b), which makes it particularly useful for testing how well a given network can fit large frequencies. Fig 17 shows the ablation over weight clip (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.

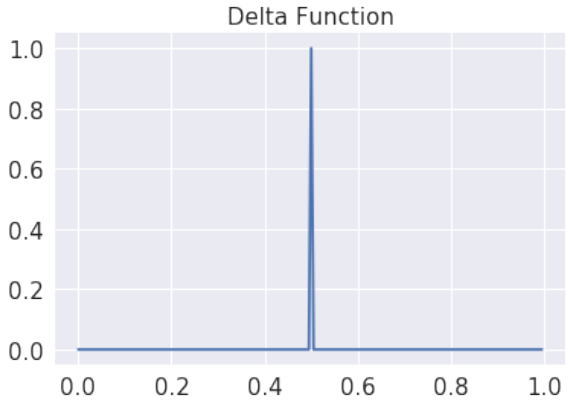
A.6. MNIST: A Proof of Concept

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.

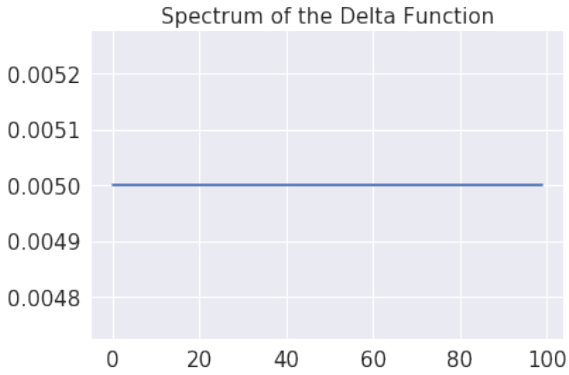
Experiment 8. In this experiment, we investigate if it is easier to learn a signal on a more realistic data-manifold like that of MNIST (assuming the manifold hypothesis is true), and compare with a flat manifold of the same dimension. To that end, we use the 64-dimensional feature-space $\mathcal{E}$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

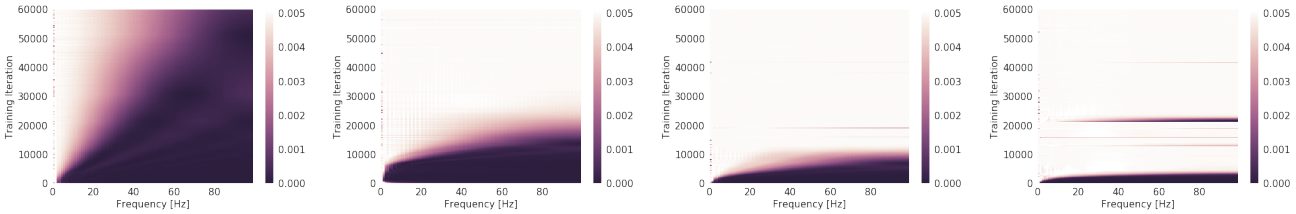
Experiment 9. Using AutoNEB (Kolsbjerg et al., 2016), we construct paths between (adversarial) Cifar-10 images that are classified by a ResNet20 to be all of the same target class. AutoNEB bends a linear path between points in some space $\mathbb{R}^m$ so that some maximum energy along the path is minimal.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

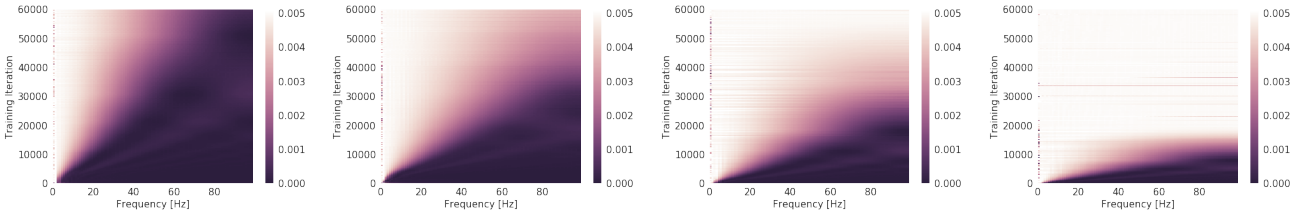
The AutoNEB parameters are chosen as follows: We run four NEB iterations with 10 steps of SGD with learning rate

in $\mathbb{R}^3$.



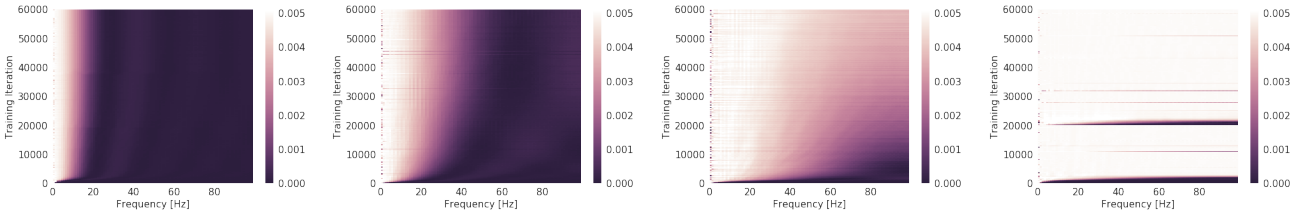
(a) Depth $= 3$. (b) Depth $= 4$. (c) Depth $= 5$. (d) Depth $= 6$.

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.



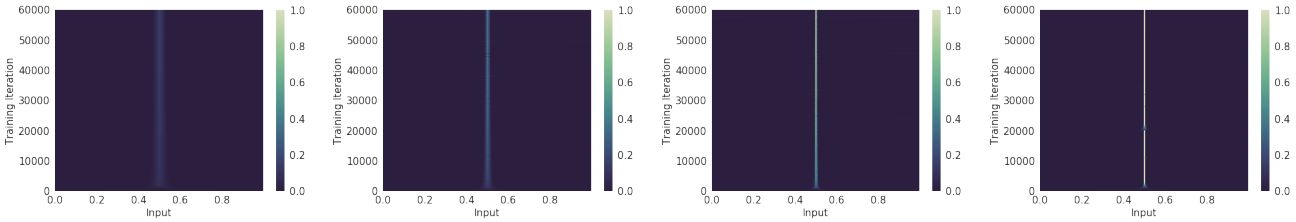
(a) Width $= 16$. (b) Width $= 32$. (c) Width $= 64$. (d) Width $= 128$.

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 $x = 0.5$.

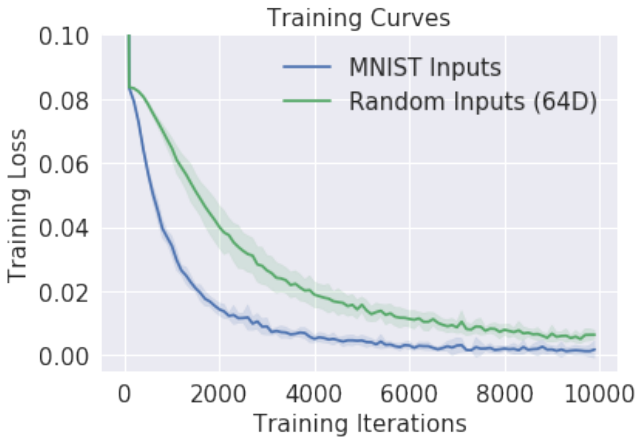


Figure 19. Loss curves of two identical networks trained to regress white-noise under identical conditions, one on MNIST reconstructions from a DAE with 64 encoder features (blue), and the other on 64-dimensional random vectors (green).

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:

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

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

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

[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.

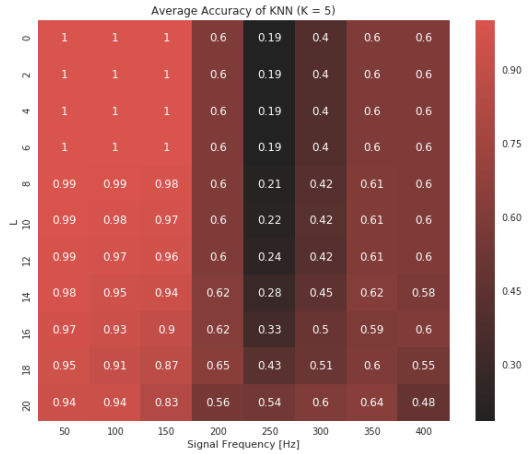
E.2. K-NN Classifier vs. DNN classifier

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

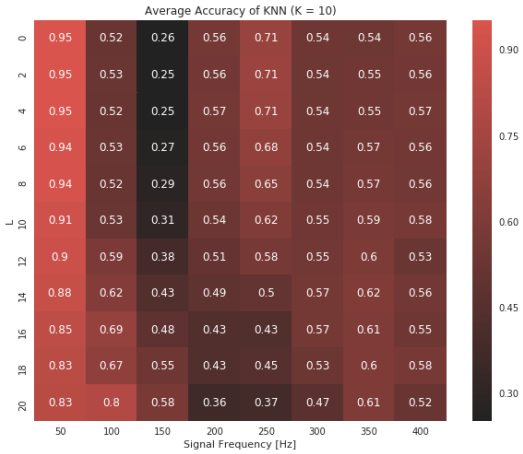
$K \to \infty$ and $K/N \to 0$, where N is the number of training samples. However, because it is a memory based algorithm, it is prohibitively slow for large datasets. Since the smoothness of a KNN prediction function is not well studied, we compare the smoothness between KNN and DNN. For various values 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 $\zeta(k)$:

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

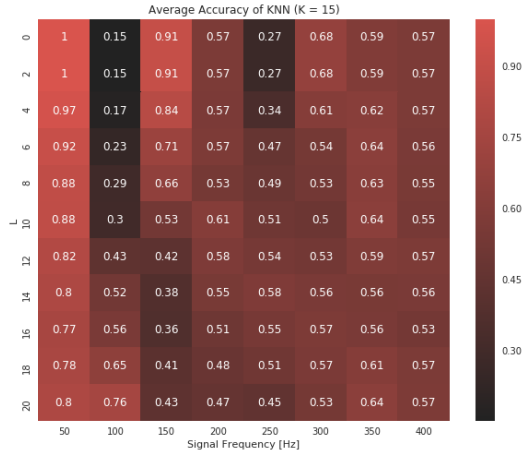
Finally, we plot $\zeta(k)$ for various K in figure 22e. Furthermore, we train a DNN on the very same dataset and overlay the radial spectrum of the resulting probability map on the same plot. We find that while DNN's are as expressive as a $K = 1$ KNN classifier at lower (radial) frequencies, the frequency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



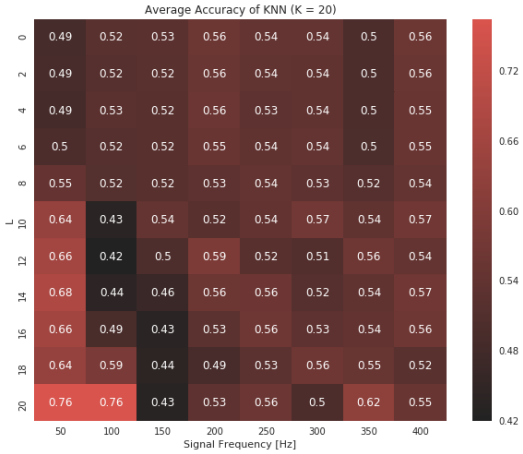
(a) $K = 5$.



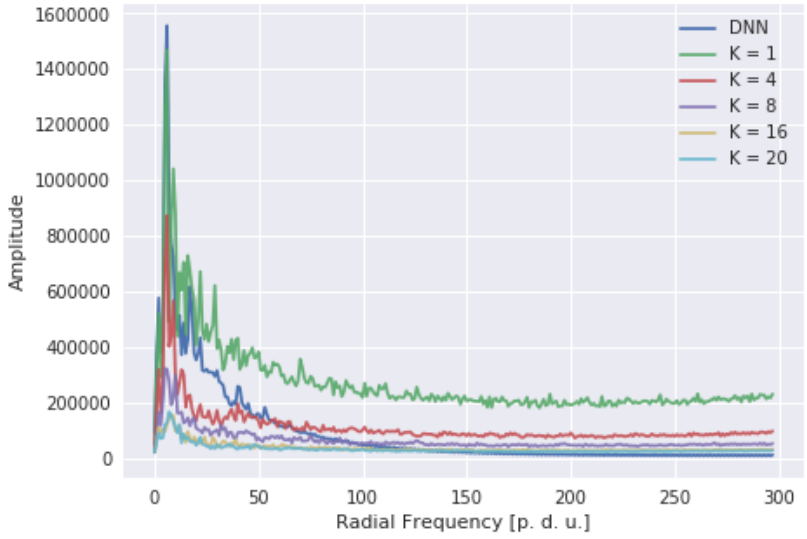
(b) $K = 10$.



(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.