Effects of Data Geometry in Early Deep Learning

Saket Tiwari
Department of Computer Science
Brown University
Providence, RI 02906
saket_tiwari@brown.edu

George Konidaris
Department of Computer Science
Brown University
Providence, RI 02906

Abstract

Deep neural networks can approximate functions on different types of data, from images to graphs, with varied underlying structure. This underlying structure can be viewed as the geometry of the data manifold. By extending recent advances in the theoretical understanding of neural networks, we study how a randomly initialized neural network with piece-wise linear activation splits the data manifold into *regions* where the neural network behaves as a linear function. We derive bounds on the density of boundary of linear regions and the distance to these boundaries on the data manifold. This leads to insights into the expressivity of randomly initialized deep neural networks on non-Euclidean data sets. We empirically corroborate our theoretical results using a toy supervised learning problem. Our experiments demonstrate that number of linear regions varies across manifolds and the results hold with changing neural network architectures. We further demonstrate how the complexity of linear regions is different on the low dimensional manifold of images as compared to the Euclidean space, using the MetFaces dataset.

1 Introduction

The capacity of Deep Neural Networks (DNNs) to approximate arbitrary functions given sufficient training data in the supervised learning setting is well known [Cybenko, 1989, Hornik et al., 1989, Anthony and Bartlett, 1999]. Several different theoretical approaches have emerged that study the effectiveness and pitfalls of deep learning. These studies vary in their treatment of neural networks and the aspects they study range from convergence [Allen-Zhu et al., 2019, Goodfellow and Vinyals, 2015], generalization [Kawaguchi et al., 2017, Zhang et al., 2017, Jacot et al., 2018, Sagun et al., 2018], function complexity [Montúfar et al., 2014, Mhaskar and Poggio, 2016], adversarial attacks [Szegedy et al., 2014, Goodfellow et al., 2015] to representation capacity [Arpit et al., 2017]. Some recent theories have also been shown to closely match empirical observations [Poole et al., 2016, Hanin and Rolnick, 2019b, Kunin et al., 2020].

One approach to studying DNNs is to examine how the underlying structure, or geometry, of the data interacts with learning dynamics. The manifold hypothesis states that high-dimensional real world data typically lies on a low dimensional manifold [Tenenbaum, 1997, Carlsson et al., 2007, Fefferman et al., 2013]. Empirical studies have shown that DNNs are highly effective in deciphering this underlying structure by learning intermediate latent representations [Poole et al., 2016]. The ability of DNNs to "flatten" complex data manifolds, using composition of seemingly simple piece-wise linear functions, appears to be unique [Brahma et al., 2016, Hauser and Ray, 2017].

DNNs with piece-wise linear activations, such as ReLU [Nair and Hinton, 2010], divide the input space into linear regions, wherein the DNN behaves as a linear function [Montúfar et al., 2014]. The density of these linear regions serves as a proxy for the DNN's ability to interpolate a complex data landscape and has been the subject of detailed studies [Montúfar et al., 2014, Telgarsky, 2015, Serra

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

et al., 2018, Raghu et al., 2017]. The work by Hanin and Rolnick [2019a] on this topic stands out because they derive bounds on the average number of linear regions and verify the tightness of these bounds empirically for deep ReLU networks, instead of larger bounds that rarely materialize. Hanin and Rolnick [2019a] conjecture that the number of linear regions correlates to the expressive power of randomly initialized DNNs with piece-wise linear activations. However, they assume that the data is uniformly sampled from the Euclidean space $\mathbb{R}^d$, for some d. By combining the manifold hypothesis with insights from Hanin and Rolnick [2019a], we are able to go further in estimating the number of linear regions and the average distance from *linear boundaries*. We derive bounds on how the geometry of the data manifold affects the aforementioned quantities.

To corroborate our theoretical bounds with empirical results, we design a toy problem where the input data is sampled from two distinct manifolds that can be represented in a closed form. We count the exact number of linear regions and the average distance to the boundaries of linear regions on these two manifolds that a neural network divides the two manifolds into. We demonstrate how the number of linear regions and average distance varies for these two distinct manifolds. These results show that the number of linear regions on the manifold do not grow exponentially with the dimension of input data. Our experiments do not provide estimates for theoretical constants, as in most deep learning theory, but demonstrate that the number of linear regions change as a consequence of these constants. We also study linear regions of deep ReLU networks for high dimensional data that lies on a low dimensional manifold with unknown structure and how the number of linear regions vary on and off this manifold, which is a more realistic setting. To achieve this we present experiments performed on the manifold of natural face images. We sample data from the image manifold using a generative adversarial network (GAN) [Goodfellow et al., 2014] trained on the curated images of paintings. Specifically, we generate images using the pre-trained StyleGAN [Karras et al., 2019, 2020b] trained on the curated MetFaces dataset [Karras et al., 2020a]. We generate *curves* on the image manifold of faces, using StyleGAN, and report how the density of linear regions varies on and off the manifold. These results shed new light on the geometry of deep learning over structured data sets by taking a data intrinsic approach to understanding the expressive power of DNNs.

2 Preliminaries and Background

Our goal is to understand how the underlying structure of real world data matters for deep learning. We first provide the mathematical background required to model this underlying structure as the geometry of data. We then provide a summary of previous work on understanding the approximation capacity of deep ReLU networks via the complexity of linear regions. For the details on how our work fits into one of the two main approaches within the theory of DNNs, from the expressive power perspective or from the learning dynamics perspective, we refer the reader to Appendix C.

2.1 Data Manifold and Definitions

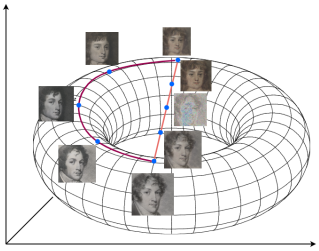


Figure 1: A 2D surface, here represented by a 2-torus, is embedded in a larger input space, $\mathbb{R}^3$. Suppose each point corresponds to an image of a face on this 2-torus. We can chart two curves: one straight line cutting across the 3D space and another curve that stays on the torus. Images corresponding to the points on the torus will have a smoother variation in style and shape whereas there will be images corresponding to points on the straight line that are not faces.

We use the example of the MetFaces dataset [Karras et al., 2020a] to illustrate how data lies on a low dimensional manifold. The images in the dataset are $1028 \times 1028 \times 3$ dimensional. By contrast, the number of *realistic* dimensions along which they vary are limited, e.g. painting style, artist, size and shape of the nose, jaw and eyes, background, clothing style; in fact, very few $1028 \times 1028 \times 3$

dimensional images correspond to realistic faces. We illustrate how this affects the possible variations in the data in Figure 1. A manifold formalises the notion of limited variations in high dimensional data. One can imagine that there exists an unknown function $f : X \to Y$ from a low dimensional space of variations, to a high dimensional space of the actual data points. Such a function $f : X \to Y$, from one open subset $X \subset \mathbb{R}^m$, to another open subset $Y \subset R^k$, is a *diffeomorphism* if f is bijective, and both f and f^{-1} are differentiable (or smooth). Therefore, a manifold is defined as follows.

Definition 2.1. *Let $k, m \in \mathbb{N}_0$. A subset $M \subset \mathbb{R}^k$ is called a smooth m-dimensional submanifold of $\mathbb{R}^k$ (or m-manifold in $\mathbb{R}^k$) iff every point $x \in M$ has an open neighborhood $U \subset \mathbb{R}^k$ such that $U \cap M$ is diffeomorphic to an open subset $\Omega \subset \mathbb{R}^m$. A diffeomorphism (i.e. differentiable mapping),*

$$f : U \cap M \to \Omega$$

is called a coordinate chart of M and the inverse,

$$h := f^{-1} : \Omega \to U \cap M$$

is called a smooth parametrization of $U \cap M$.

For the MetFaces dataset example, suppose there are 10 dimensions along which the images vary. Further assume that each variation can take a value continuously in some interval of $\mathbb{R}$. Then the smooth parametrization would map $f : \Omega \cap \mathbb{R}^{10} \to M \cap \mathbb{R}^{1028\times1028\times3}$. This parametrization and its inverse are unknown in general and computationally very difficult to estimate in practice.

There are similarities in how geometric elements are defined for manifolds and Euclidean spaces. A smooth curve, on a manifold M, $\gamma : I \to M$ is defined from an interval I to the manifold M as a function that is differentiable for all $t \in I$, just as for Euclidean spaces. The shortest such curve between two points on a manifold is no longer a straight line, but is instead a *geodesic*. One recurring geometric element, which is unique to manifolds and stems from the definition of smooth curves, is that of a *tangent space*, defined as follows.

Definition 2.2. *Let M be an m-manifold in $\mathbb{R}^k$ and $x \in M$ be a fixed point. A vector $v \in \mathbb{R}^k$ is called a tangent vector of M at x if there exists a smooth curve $\gamma : I \to M$ such that $\gamma(0) = x, \dot{\gamma}(0) = v$ where $\dot{\gamma}(t)$ is the derivative of γ at t. The set*

$$T_xM := \{\dot{\gamma}(0) | \gamma : \mathbb{R} \to M \text{ is smooth} \gamma(0) = x\}$$

of tangent vectors of M at x is called the tangent space of M at x.

In simpler terms, the plane tangent to the manifold M at point x is called the tangent space and denoted by by T_xM. Consider the upper half of a 2-sphere, $S^2 \subset \mathbb{R}^3$, which is a 2-manifold in $\mathbb{R}^3$. The tangent space at a fixed point $x \in S^2$ is the 2D plane perpendicular to the vector x and tangential to the surface of the sphere that contains the point x. For additional background on manifolds we refer the reader to Appendix B.

2.2 Linear Regions of Deep ReLU Networks

The higher the density of these linear regions the more complex a function a DNN can approximate. For example, a $\sin$ curve in the range $[0, 2\pi]$ is better approximated by 4 piece-wise linear regions as opposed to 2. To clarify this further, with the 4 "optimal" linear regions $[0, \pi/2), [\pi/2, \pi), [\pi, 3\pi/2)$, and $[3\pi/2, 2\pi]$ a function could approximate the $\sin$ curve better than any 2 linear regions. In other words, higher density of linear regions allows a DNN to approximate the variation in the curve better. We define the notion of boundary of a linear regions in this section and provide an overview of previous results.

We consider a neural network, F, which is a composition of activation functions. Inputs at each layer are multiplied by a matrix, referred to as the weight matrix, with an additional bias vector that is added to this product. We limit our study to ReLU activation function [Nair and Hinton, 2010], which is piece-wise linear and one of the most popular activation functions being applied to various learning tasks on different types of data like text, images, signals etc. We further consider DNNs that map inputs, of dimension n_{in}, to scalar values. Therefore, $F : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}$ is defined as,

$$F(x) = W_L\sigma(B_{L-1} + W_{L-1}\sigma(...\sigma(B_1 + W_1x))), \tag{1}$$

where $W_l \in \mathbb{M}^{n_l \times n_{l-1}}$ is the weight matrix for the lth hidden layer, n_l is the number of neurons in the lth hidden layer, $B_l \in \mathbb{R}^{n_l}$ is the vector of biases for the lth hidden layer, $n_0 = n_{\text{in}}$ and $\sigma : \mathbb{R} \to \mathbb{R}$

is the activation function. For a neuron z in the lth layer we denote the *pre-activation* of this neuron, for given input $x \in \mathbb{R}^{n_{\text{in}}}$, as $z_l(x)$. For a neuron z in the layer l we have

$$z(x) = W_{l-1,z}\sigma(...\sigma(B_1 + W_1 x)), \tag{2}$$

for $l > 1$ (for the base case $l = 1$ we have $z(x) = W_{1,z}x$) where $W_{l-1,z}$ is the row of weights, in the weight matrix of the lth layer, W_l, corresponding to the neuron z. We use W_z to denote the weight vector for brevity, omitting the layer index l in the subscript. We also use b_z to denote the bias term for the neuron z.

Neural networks with piece-wise linear activations are piece-wise linear on the input space [Montúfar et al., 2014]. Suppose for some fixed $y \in \mathbb{R}^{n_{\text{in}}}$ as $x \to y$ if we have $z(x) \to -b_z$ then we observe a discontinuity in the gradient $\nabla_x \sigma(b_z + W_z z(x))$ at y. Intuitively, this is because x is approaching the boundary of the linear region of the function defined by the output of z. Therefore, the boundary of linear regions, for a feed forward neural network F, is defined as:

$$\mathcal{B}_F = \{x | \nabla F(x) \text{ is not continuous at } x\}.$$

Hanin and Rolnick [2019a] argue that an important generalization for the approximation capacity of a neural network F is the $(n_{\text{in}} - 1)-$dimensional volume density of linear regions defined as $\text{vol}_{n_{\text{in}}-1}(\mathcal{B}_F \cap K)/\text{vol}_{n_{\text{in}}}(K)$, for a bounded set $K \subset \mathbb{R}^{n_{\text{in}}}$. This quantity serves as a proxy for density of linear regions and therefore the expressive capacity of DNNs. Intuitively, higher density of linear boundaries means higher capacity of the DNN to approximate complex non-linear functions. The quantity is applied to lower bound the distance between a point $x \in K$ and the set $\mathcal{B}_F$, which is

$$\text{distance}(x, \mathcal{B}_F) = \min_{\text{neurons } z} |z(x) - b_z| / ||\nabla z(x)||,$$

which measures the sensitivity over neurons at a given input. The above quantity measures how "far" the input is from flipping any neuron from inactive to active or vice-versa.

Informally, Hanin and Rolnick [2019a] provide two main results for a randomly initialized DNN F, with a reasonable initialisation. Firstly, they show that

$$\mathbb{E}\Big[\frac{\text{vol}_{n_{\text{in}}-1}(\mathcal{B}_F \cap K)}{\text{vol}_{n_{\text{in}}}(K)}\Big] \approx \#\{\text{ neurons}\},$$

meaning the density of linear regions is bound above and below by some constant times the number of neurons. Secondly, for $x \in [0, 1]^{n_{\text{in}}}$,

$$\mathbb{E}\Big[\text{distance}(x, \mathcal{B}_F)\Big] \geq C\#\{\text{ neurons}\}^{-1},$$

where $C > 0$ depends on the distribution of biases and weights, in addition to other factors. In other words, the distance to the nearest boundary is bounded above and below by a constant times the inverse of the number of neurons. These results stand in contrast to earlier worst case bounds that are exponential in the number of neurons. Hanin and Rolnick [2019a] also verify these results empirically to note that the constants lie in the vicinity of 1 throughout training.

3 Linear Regions on the Data Manifold

One important assumption in the results presented by Hanin and Rolnick [2019a] is that the input, x, lies in a compact set $K \subset \mathbb{R}^{n_{\text{in}}}$ and that $\text{vol}_{n_{\text{in}}}(K)$ is greater than 0. Also, the theorem pertaining to the lower bound on average distance of x to linear boundaries the input assumes the input uniformly distributed in $[0, 1]^{n_{\text{in}}}$. As noted earlier, high-dimensional real world datasets, like images, lie on low dimensional manifolds, therefore both these assumptions are false in practice. This motivates us to study the case where the data lies on some $m-$dimensional submanifold of $\mathbb{R}^{n_{\text{in}}}$, i.e. $M \subset \mathbb{R}^{n_{\text{in}}}$ where $m \ll n_{\text{in}}$. We illustrate how this constraint effects the study of linear regions in Figure 2.

As introduced by Hanin and Rolnick [2019a], we denote the "$(n_{\text{in}} - k)-$dimensional piece" of $\mathcal{B}_F$ as $\mathcal{B}_{F,k}$. More precisely, $\mathcal{B}_{F,0} = \emptyset$ and $\mathcal{B}_{F,k}$ is recursively defined to be the set of points $x \in \mathcal{B}_F \setminus \{\mathcal{B}_{F,0} \cup ... \cup \mathcal{B}_{F,k-1}\}$ with the added condition that in a neighbourhood of x the set $\mathcal{B}_{F,k}$ coincides with hyperplane of dimension $n_{\text{in}} - k$. We provide a detailed and formal definition for $\mathcal{B}_{F,k}$ with intuition in Appendix E. In our setting, where the data lies on a manifold M, we define $\mathcal{B}'_{F,k}$

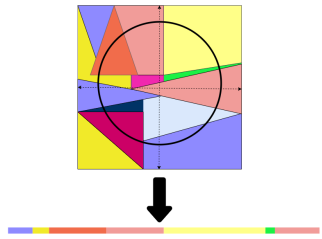


Figure 2: A circle is an example of a 1D manifold in a 2D Euclidean space. The effective number of linear regions on the manifold, the upper half of the circle, are the number of linear regions on the arc from $-\pi$ to π. In the diagram above, each color in the 2D space corresponds to a linear region. When the upper half of the circle is flattened into a 1D space we obtain a line. Each color on the line corresponds to a linear region of the 2D space.

as $\mathcal{B}_{F,k} \cap M$, and note that $\dim(\mathcal{B}'_{F,k}) = m - k$ (Appendix E Proposition E.4). For example, the *transverse* intersection (see Definition E.3) of a plane in 3D with the 2D manifold S^2 is a 1D curve in S^2 and therefore has dimension 1. Therefore, $\mathcal{B}'_{F,k}$ is a submanifold of dimension $3 - 2 = 1$. This imposes the restriction $k \leq m$, for the intersection $\mathcal{B}_{F,k} \cap M$ to have a well defined volume.

We first note that the definition of the determinant of the Jacobian, for a collection of neurons $z_1, ..., z_k$, is different in the case when the data lies on a manifold M as opposed to in a compact set of dimension n_{in} in $\mathbb{R}^{n_{\text{in}}}$. Since the determinant of the Jacobian is the quantity we utilise in our proofs and theorems repeatedly we will use the term Jacobian to refer to it for succinctness. Intuitively, this follows from the Jacobian of a function being defined differently in the ambient space $\mathbb{R}^{n_{\text{in}}}$ as opposed to the manifold M. In case of the former it is the volume of the paralellepiped determined by the vectors corresponding to the directions with steepest ascent along each one of the n_{in} axes. In case of the latter it is more complex and defined below. Let $\mathcal{H}^m$ be the $m-$dimensional Hausdorff measure (we refer the reader to the Appendix B for background on Hausdorff measure). The Jacobian of a function on manifold M, as defined by Krantz and Parks [2008] (Chapter 5), is as follows.

Definition 3.1. *The (determinant of) Jacobian of a function $H : M \to \mathbb{R}^k$, where $k \leq \dim(M) = m$, is defined as*

$$J^M_{k,H}(x) = \sup \left\{ \frac{\mathcal{H}^k(D_M H(P))}{\mathcal{H}^k(P)} \middle| P \textit{ is a } k\textit{-dimensional parallelepiped contained in } T_x M. \right\},$$

where $D_M : T_x M \to \mathbb{R}^k$ is the differential map (see Appendix B) and we use $D_M H(P)$ to denote the mapping of the set P in $T_x M$, which is a parallelepiped, to $\mathbb{R}^k$. The supremum is taken over all parallelepipeds P.

We also say that neurons $z_1, ..., z_k$ are good at x if there exists a path of neurons from z to the output in the computational graph of F so that each neuron is activated along the path. Our three main results that hold under the assumptions listed in Appendix A, each of which extend and improve upon the theoretical results by Hanin and Rolnick [2019a], are:

Theorem 3.2. *Given F a feed-forward ReLU network with input dimension n_{in}, output dimension 1, and random weights and biases. Then for any bounded measurable submanifold $M \subset \mathbb{R}^{n_{in}}$ and any $k = 1, ..., m$ the average $(m-k)-$dimensional volume of $\mathcal{B}_{F,k}$ inside M,*

$$\mathbb{E}[vol_{m-k}(\mathcal{B}_{F,k} \cap M)] = \sum_{\textit{distinct neurons } z_1,...,z_k \textit{ in } F} \int_M \mathbb{E}[Y_{z_1,...,z_k}] dvol_m(x), \tag{3}$$

where $Y_{z_1,...,z_k}$ is $J^M_{m,H_k}(x)\rho_{b_1,...,b_k}(z_1(x), ..., z_k(x))$, times the indicator function of the event that z_j is good at x for each $j = 1, ..., k$. Here the function $\rho_{b_{z_1},...,b_{z_k}}$ is the density of the joint distribution of the biases $b_{z_1}, ..., b_{z_k}$.

This change in the formula, from Theorem 3.4 by Hanin and Rolnick [2019a], is a result of the fact that $z(x)$ has a different direction of steepest ascent when it is restricted to the data manifold M, for any j. The proof is presented in Appendix E. Formula 3 also makes explicit the fact that the data manifold has dimension $m \leq n_{\text{in}}$ and therefore the $m - k$-dimensional volume is a more representative measure of the linear boundaries. Equipped with Theorem 3.2, we provide a result for the density of boundary regions on manifold M.

Theorem 3.3. *For data sampled uniformly from a compact and measurable m dimensional manifold M we have the following result for all $k \leq m$:*

$$\frac{vol_{m-k}(\mathcal{B}_{F,k} \cap M)}{vol_m(M)} \leq \binom{\#\ neurons}{k} (2C_{grad}C_{bias}C_M)^k,$$

where C_{grad} depends on $||\nabla z(x)||$ and the DNN's architecture, C_M depends on the geometry of M, and C_{bias} on the distribution of biases ρ_b.

The constant C_M is the supremum over the matrix norm of projection matrices onto the tangent space, T_xM, at any point $x \in M$. For the Euclidean space C_M is always equal to 1 and therefore the term does not appear in the work by Hanin and Rolnick [2019a], but we cannot say the same for our setting. We refer the reader to Appendix F for the proof, further details, and interpretation. Finally, under the added assumptions that the diameter of the manifold M is finite and M has polynomial volume growth we provide a lower bound on the average distance to the linear boundary for points on the manifold and how it depends on the geometry and dimensionality of the manifold.

Theorem 3.4. *For any point, x, chosen randomly from M, we have:*

$$\mathbb{E}[distance_M(x, \mathcal{B}_F \cap M)] \geq \frac{C_{M,\kappa}}{C_{grad}C_{bias}C_M \#neurons},$$

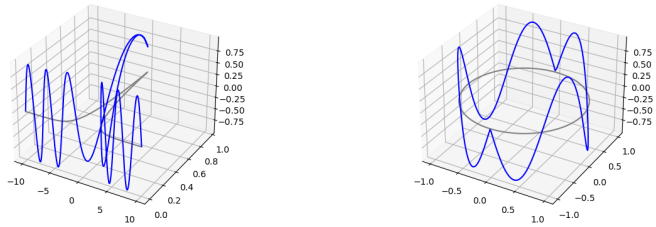
where $C_{M,\kappa}$ depends on the scalar curvature, the input dimension and the dimensionality of the manifold M. The function $distance_M$ is the distance on the manifold M.

This result gives us intuition on how the density of linear regions around a point depends on the geometry of the manifold. The constant $C_{M,\kappa}$ captures how volumes are distorted on the manifold M as compared to the Euclidean space, for the exact definition we refer the reader to the proof in Appendix G. For a manifold which has higher volume of a unit ball, on average, in comparison to the Euclidean space the constant $C_{M,\kappa}$ is higher and lower when the volume of unit ball, on average, is lower than the volume of the Euclidean space. For background on curvature of manifolds and a proof sketch we refer the reader to the Appendices B and D, respectively. Note that the constant C_M is the same as in Theorem 3.3. Another difference to note is that we derive a lower bound on the geodesic distance on the manifold M and not the Euclidean distance in $\mathbb{R}^k$ as done by Hanin and Rolnick [2019a]. This distance better captures the distance between data points on a manifold while incorporating the underlying structure. In other words, this distance can be understood as how much a data point should change to reach a linear boundary while ensuring that all the individual points on the curve, tracing this change, are "valid" data points.

3.1 Intuition For Theoretical Results

One of the key ingredients of the proofs by Hanin and Rolnick [2019a] is the *co-area formula* [Krantz and Parks, 2008]. The co-area formula is applied to get a closed form representation of the $k-$dimensional volume of the region where any set of k neurons, $z_1, z_2, ..., z_k$ is "good" in terms of the expectation over the Jacobian, in the Euclidean space. Instead of the co-area formula we use the *smooth co-area formula* [Krantz and Parks, 2008] to get a closed form representation of the $m-k-$dimensional volume of the region intersected with manifold, M, in terms of the Jacobian defined on a manifold (Definition 3.1). The key difference between the two formulas is that in the smooth co-area formula the Jacobian (of a function from the manifold M) is restricted to the tangent plane. While the determinant of the "vanilla" Jacobian measures the distortion of volume around a point in Euclidean space the determinant of the Jacobian defined as above (Definition 3.1) measures the distortion of volume on the manifold instead for the function with the same domain, the function that is 1 if the set of neurons are good and 0 otherwise.

The value of the Jacobian as defined in Definition 3.1 has the same volume as the projection of the parallelepiped defined by the gradients $\nabla z(x)$ onto the tangent space (see Proposition F.1 in Appendix). This introduces the constant C_M, defined above. Essentially, the constant captures how the magnitude of the gradients, $\nabla z(x)$, are modified upon being projected to the tangent plane. Certain manifolds "shrink" vectors upon projection to the tangent plane more than others, on an average, which is a function of their geometry. We illustrate how two distinct manifolds "shrink" the gradients differently upon projection to the tangent plane as reflected in the number of linear regions on the manifolds (see Figure 11 in the appendix) for 1D manifolds. We provide intuition



(a) (b)

Figure 3: The tractrix (a) and circle (b) are plotted in grey and the target function is in blue. This is for illustration purposes and does not match the actual function or domains used in our experiments.

for the curvature of a manifold in Appendix B, due to space constraints, which is used in the lower bound for the average distance in Theorem 3.4. The constant $C_{M,\kappa}$ depends on the curvature as the supremum of a polynomial whose coefficients depend on the curvature, with order at most n_{in} and at least $n_{\text{in}} - m$. Note that despite this dependence on the ambient dimension, there are other geometric constants in this polynomial (see Appendix G). Finally, we also provide a simple example as to how this constant varies with n_{in} and m, for a simple and contrived example, in Appendix G.1.

4 Experiments

4.1 Linear Regions on a 1D Curve

To empirically corroborate our theoretical results, we calculate the number of linear regions and average distance to the linear boundary on 1D curves for regression tasks in two settings. The first is for 1D manifolds embedded in 2D and higher dimensions and the second is for the high-dimensional data using the MetFaces dataset. We use the same algorithm, for the toy problem and the high-dimensional dataset, to find linear regions on 1D curves. We calculate the exact number of linear regions for a 1D curve in the input space, $x : I \to \mathbb{R}^{n_{\text{in}}}$ where I is an interval in real numbers, by finding the points where $z(x(t)) = b_z$ for every neuron z. The solutions thus obtained gives us the boundaries for neurons on the curve x. We obtain these solutions by using the programmatic activation of every neuron and using the sequential least squares programming (SLSQP) algorithm [Kraft, 1988] to solve for $|z(x(t)) - b_z| = 0$ for $t \in I$. In order to obtain the programmatic activation of a neuron we construct a Deep ReLU network as defined in Equation 2. We do so for all the neurons for a given DNN with fixed weights.

4.2 Supervised Learning on Toy Dataset

We define two similar regression tasks where the data is sampled from two different manifolds with different geometries. We parameterize the first task, a unit circle without its north and south poles, by $\psi_{\text{circle}} : (-\pi, \pi) \to \mathbb{R}^2$ where $\psi_{\text{circle}}(\theta) = (\cos\theta, \sin\theta)$ and θ is the angle made by the vector from the origin to the point with respect to the x-axis. We set the target function for regression task to be a periodic function in θ. The target is defined as $z(\theta) = a\sin(\nu\theta)$ where a is the amplitude and ν is the frequency (Figure 3). DNNs have difficulty learning periodic functions [Ziyin et al., 2020]. The motivation behind this is to present the DNN with a challenging task where it has to learn the underlying structure of the data. Moreover the DNN will have to split the circle into linear regions. For the second regression task, a tractrix is parametrized by $\psi_{\text{tractrix}} : \mathbb{R}^1 \to \mathbb{R}^2$ where $\psi_{\text{tractrix}}(y) = (y - \tanh y, \operatorname{sech} y)$ (see Figure 3). We assign a target function $z(t) = a\sin(\nu t)$. For the purposes of our study we restrict the domain of ψ_{tractrix} to $(-3, 3)$. We choose ν so as to ensure that the number of peaks and troughs, 6, in the periodic target function are the same for both the manifolds. This ensures that the domains of both the problems have length close to 6.28. Further experimental details are in Appendix H.

The results, averaged over 20 runs, are presented in Figures 4 and 5. We note that C_M is smaller for Sphere (based on Figure 4) and the curvature is positive whilst C_M is larger for tractrix and the

curvature is negative. Both of these constants (curvature and C_M) contribute to the lower bound in Theorem 3.4. Similarly, we show results of number of linear regions divided by the number of neurons upon changing architectures, consequently the number of neurons, for the two manifolds in Figure 8, averaged over 30 runs. Note that this experiment observes the effect of $C_M \times C_{\text{grad}}$, since changing the architecture also changes C_{grad} and the variation in C_{grad} is quite low in magnitude as observed empirically by Hanin and Rolnick [2019a]. The empirical observations are consistent with our theoretical results. We observe that the number of linear regions starts off close to #neurons and remains close throughout the training process for both the manifolds. This supports our theoretical results (Theorem 3.3) that the constant C_M, which is distinct across the two manifolds, affects the number of linear regions throughout training. The tractrix has a higher value of C_M and that is reflected in both Figures 4 and 5. Note that its relationship is inverse to the average distance to the boundary region, as per Theorem 3.4, and it is reflected as training progresses in Figure 5. This is due to different "shrinking" of vectors upon being projected to the tangent space (Section 3.1).

4.3 Varying n_{in}

To empirically corroborate the results of Theorems 2 and 3 we vary the dimension n_{in} while keeping m constant. We achieve this by counting the number of linear regions and the average distance to boundary region on the 1D circle as we vary the input dimension in steps of 5. We draw samples of 1D circles in $\mathbb{R}^{n_{\text{in}}}$ by randomly choosing two perpendicular basis vectors. We then train a network with the same architecture as the previous section on the periodic target function ($a\sin(\nu\theta)$) as defined above. The results in Figure 6 shows that the quantities stay proportional to $\#neurons$, and do not vary as n_{in} is increased, as predicted by our theoretical results. Our empirical study asserts how the relevant upper and lower bounds, for the setting where data lies on a low-dimensional manifold, does not grow exponentially with n_{in} for the density of linear regions in a compact set of $\mathbb{R}^{n_{\text{in}}}$ but instead depend on the intrinsic dimension. Further details are in Appendix H.

4.4 MetFaces: High Dimensional Dataset

Our goal with this experiment is to study how the density of linear regions varies across a low dimensional manifold and the input space. To discover latent low dimensional underlying structure of data we employ a GAN. Adversarial training of GANs can be effectively applied to learn a mapping from a low dimensional latent space to high dimensional data [Goodfellow et al., 2014]. The generator is a neural network that maps $g : \mathbb{R}^k \to \mathbb{R}^{n_{\text{in}}}$. We train a deep ReLU network on the MetFaces dataset with random labels (chosen from $0, 1$) with cross entropy loss. As noted by Zhang et al. [2017], training with random labels can lead to the DNN memorizing the entire dataset.

We compare the log density of number of linear regions on a curve on the manifold with a straight line off the manifold. We generate these curves using the data sampled by the StyleGAN by [Karras et al., 2020a]. Specifically, for each curve we sample a random pair of latent vectors: $z_1, z_2 \in \mathbb{R}^k$, this gives us the start and end point of the curve using the generator $g(z_1)$ and $g(z_2)$. We then generate 100 images to approximate a curve connecting the two images on the image manifold in a piece-wise manner. We do so by taking 100 points on the line connecting z_1 and z_2 in the latent space that are evenly spaced and generate an image from each one of them. Therefore, the ith image is generated as: $z'_i = g(((100 - i) \times z_1 + i \times z_2)/100)$, using the StyleGAN generator g. We qualitatively verify the images to ensure that they lie on the manifold of images of faces. The straight line, with two fixed points $g(z_1)$ and $g(z_2)$, is defined as $x(t) = (1 - t)g(z_1) + tg(z_2)$ with $t \in [0, 1]$. The approximated curve on the manifold is defined as $x'(t) = (1 - t)g(z'_i) + tg(z'_{i+1})$ where $i = \texttt{floor}(100t)$. We then apply the method from Section 4.1 to obtain the number of linear regions on these curves.

The results are presented in Figure 9. This leads us to the key observation: the density of linear regions is significantly lower on the data manifold and devising methods to "concentrate" these linear regions on the manifold is a promising research direction. That could lead to increased expressivity for the same number of parameters. We provide further experimental details in Appendix I.

5 Discussion and Conclusions

There is significant work in both supervised and unsupervised learning settings for non-Euclidean data [Bronstein et al., 2017]. Despite these empirical results most theoretical analysis is agnostic

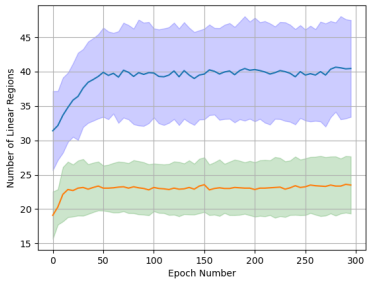


Figure 4: Graph of number of linear regions for tractrix (blue) and sphere (orange). The shaded regions represent one standard deviation. Note that the number of neurons is 26 and the number of linear regions are comparable to 26 but different for both the manifolds throughout training.

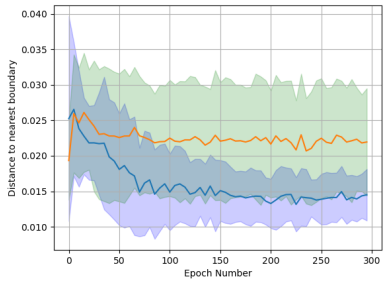


Figure 5: Graph of distance to linear regions for tractrix (blue) and sphere (orange). The distances are normalized by the maximum distance on the range, for both tractrix and sphere. The shaded regions represent one standard deviation.

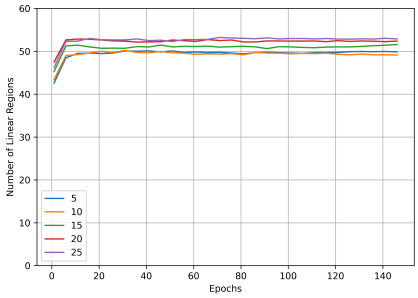


Figure 6: We observe that as the dimension n_{in} is increased, while keeping the manifold dimension constant, the number of linear regions remains proportional to number of neurons (26).

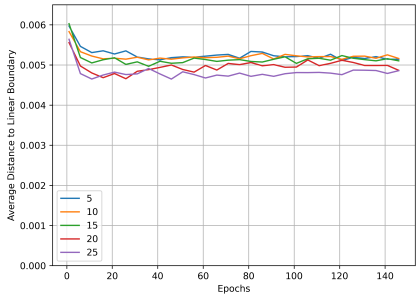


Figure 7: We observe that as the dimension n_{in} is increased, while keeping the manifold dimension constant, the average distance varies very little.

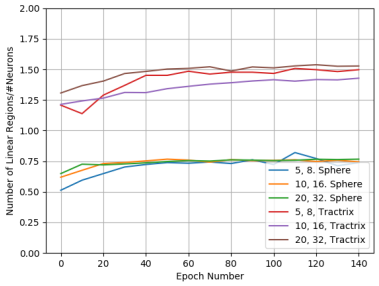


Figure 8: The effects of changing the architecture on the number of linear regions. We observe that the value of C_M effects the number of linear regions proportionally. The number of hidden units for three layer networks are in the legend along with the data manifold.

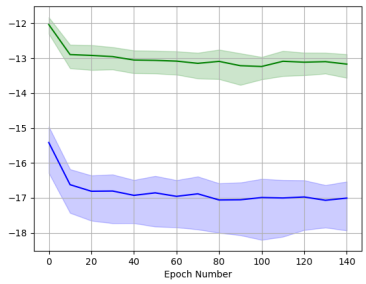


Figure 9: We observe that the log density of number of linear regions is lower on the manifold (blue) as compared to off the manifold (green). This is for the MetFaces dataset.

to data geometry, with a few prominent exceptions [Cloninger and Klock, 2020, Shaham et al., 2015, Schmidt-Hieber, 2019]. We incorporate the idea of data geometry into measuring the effective approximation capacity of DNNs, deriving average bounds on the density of boundary regions and distance from the boundary when the data is sampled from a low dimensional manifold. Our experimental results corroborate our theoretical results. We also present insights into expressivity of DNNs on low dimensional manfiolds for the case of high dimensional datasets. Estimating the geometry, dimensionality and curvature, of these image manifolds accurately is a problem that remains largely unsolved [Brehmer and Cranmer, 2020, Perraul-Joncas and Meila, 2013], which limits our inferences on high dimensional dataset to observations that guide future research. We note that proving a lower bound on the number of linear regions, as done by Hanin and Rolnick [2019a], for the manifold setting remains open. Our work opens up avenues for further research that combines model geometry and data geometry and can lead to empirical research geared towards developing DNN architectures for high dimensional datasets that lie on a low dimensional manifold.

6 Acknowledgements

This work was funded by L2M (DARPA Lifelong Learning Machines program under grant number FA8750-18-2-0117), the Penn MURI (ONR under the PERISCOPE MURI Contract N00014- 17-1-2699), and the ONR Swarm (the ONR under grant number N00014-21-1-2200). This research was conducted using computational resources and services at the Center for Computation and Visualization, Brown University.

We would like to thank Sam Lobel, Rafael Rodriguez Sanchez, and Akhil Bagaria for refining our work, multiple technical discussions, and their helpful feedback on the implementation details. We also thank Tejas Kotwal for assistance on deriving the mathematical details related to the 1D Tractrix and sources for various citations. We thank Professor Pedro Lopes de Almeida, Nihal Nayak, Cameron Allen and Aarushi Kalra for their valuable comments on writing and presentation of our work. We thank all the members of the Brown robotics lab for their guidance and support at various stages of our work. Finally, we are indebted to, and graciously thank, the numerous anonymous reviewers for their time and labor as their valuable feedback and thoughtful engagement have shaped and vastly refine our work.

References

Zeyuan Allen-Zhu, Y. Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *ArXiv*, abs/1811.03962, 2019.

M. Anthony and P. Bartlett. Neural network learning - theoretical foundations. 1999.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *ArXiv*, abs/1802.05296, 2018.

Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *ArXiv*, abs/1810.02281, 2019a.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *NeurIPS*, 2019b.

D. Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. *ArXiv*, abs/1706.05394, 2017.

Peter L. Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear vc-dimension bounds for piecewise polynomial networks. *Neural Computation*, 10:2159–2173, 1998.

P. P. Brahma, Dapeng Oliver Wu, and Y. She. Why deep learning works: A manifold disentanglement perspective. *IEEE Transactions on Neural Networks and Learning Systems*, 27:1997–2008, 2016.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *ArXiv*, abs/2003.13913, 2020.

Richard P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *Comput. J.*, 14:422–425, 1971.

M. Bronstein, Joan Bruna, Y. LeCun, Arthur Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34:18–42, 2017.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velivckovi'c. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *ArXiv*, abs/2104.13478, 2021.

Sam Buchanan, Dar Gilboa, and John Wright. Deep networks and the multiple manifold problem. *ArXiv*, abs/2008.11245, 2021.

G. Carlsson, T. Ishkhanov, V. D. Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76:1–12, 2007.

Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. Efficient approximation of deep relu networks for functions on low dimensional manifolds. *ArXiv*, abs/1908.01842, 2019.

Alexander Cloninger and Timo Klock. Relu nets adapt to intrinsic dimensionality beyond the target domain. *ArXiv*, abs/2008.02545, 2020.

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

Simon Shaolei Du, Wei Hu, and J. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *NeurIPS*, 2018.

C. Fefferman, S. Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *arXiv: Statistics Theory*, 2013.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *ArXiv*, abs/1805.09112, 2018.

Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modelling the influence of data structure on learning in neural networks. *ArXiv*, abs/1909.11500, 2020.

I. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, S. Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Ian J. Goodfellow and Oriol Vinyals. Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544, 2015.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

Alfred Gray. The volume of a small geodesic ball of a riemannian manifold. *Michigan Mathematical Journal*, 20:329–344, 1974.

Victor Guillemin and Alan Pollack. *Differential Topology*. Prentice-Hall, 1974.

B. Hanin and M. Nica. Products of many large random matrices and gradients in deep neural networks. *Communications in Mathematical Physics*, 376:287–322, 2018.

B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. *ArXiv*, abs/1901.09021, 2019a.

B. Hanin and D. Rolnick. Deep relu networks have surprisingly few activation patterns. In *NeurIPS*, 2019b.

Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *ArXiv*, abs/1708.02691, 2019.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. *ArXiv*, abs/1909.05989, 2020.

M. Hauser and A. Ray. Principles of riemannian geometry in neural networks. In *NIPS*, 2017.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *ArXiv*, abs/1506.05163, 2015.

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

Arthur Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

Tero Karras, S. Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019.

Tero Karras, Miika Aittala, Janne Hellsten, S. Laine, J. Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *ArXiv*, abs/2006.06676, 2020a.

Tero Karras, S. Laine, Miika Aittala, Janne Hellsten, J. Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020b.

Kenji Kawaguchi, L. Kaelbling, and Yoshua Bengio. Generalization in deep learning. *ArXiv*, abs/1710.05468, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017.

Dieter Kraft. A software package for sequential quadratic programming. *Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center — Institute for Flight Mechanics*, 1988.

S. Krantz and Harold R. Parks. Geometric integration theory. 2008.

Daniel Kunin, Javier Sagastuy-Breña, S. Ganguli, Daniel L. K. Yamins, and H. Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. *ArXiv*, abs/2012.04728, 2020.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jascha Sohl-Dickstein. Wide neural networks of any depth evolve as linear models under gradient descent. *ArXiv*, abs/1902.06720, 2019.

Tengyuan Liang, Tomaso A. Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *ArXiv*, abs/1711.01530, 2019.

L. Loveridge. Physical and geometric interpretations of the riemann tensor, ricci tensor, and scalar curvature. 2004.

H. Mhaskar and T. Poggio. Deep vs. shallow networks : An approximation theory perspective. *ArXiv*, abs/1608.03287, 2016.

Federico Monti, D. Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5425–5434, 2017.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

V. Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Behnam Neyshabur, Srinadh Bhojanapalli, David A. McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *ArXiv*, abs/1707.09564, 2018.

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

Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021, 2020.

Dominique Perraul-Joncas and Marina Meila. Non-linear dimensionality reduction: Riemannian metric estimation and the problem of geometric discovery. *arXiv: Machine Learning*, 2013.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *NIPS*, 2016.

C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

M. Raghu, Ben Poole, J. Kleinberg, S. Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *ArXiv*, abs/1606.05336, 2017.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Dräxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron C. Courville. On the spectral bias of neural networks. In *ICML*, 2019.

Joel W. Robbin, Uw Madison, and Dietmar A. Salamon. *INTRODUCTION TO DIFFERENTIAL GEOMETRY*. Preprint, 2011.

Levent Sagun, Utku Evci, V. U. Güney, Yann Dauphin, and L. Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *ArXiv*, abs/1706.04454, 2018.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, abs/1312.6120, 2014.

Johannes Schmidt-Hieber. Deep relu network approximation of functions on a manifold. *ArXiv*, abs/1908.00695, 2019.

Thiago Serra, Christian Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks. In *ICML*, 2018.

Uri Shaham, Alexander Cloninger, and Ronald R. Coifman. Provable approximation properties for deep neural networks. *ArXiv*, abs/1509.07385, 2015.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. *ArXiv*, abs/1710.06451, 2018.

Weijie J. Su, Stephen P. Boyd, and Emmanuel J. Candès. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *J. Mach. Learn. Res.*, 2016.

Christian Szegedy, W. Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

Matus Telgarsky. Representation benefits of deep feedforward networks. *ArXiv*, abs/1509.08101, 2015.

Joshua B. Tenenbaum. Mapping a manifold of perceptual observations. In *NIPS*, 1997.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Z. Wan. Geometric interpretations of curvature. 2016.

Tingran Wang, Sam Buchanan, Dar Gilboa, and John Wright. Deep networks provably classify data on curves. *ArXiv*, abs/2107.14324, 2021.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2019.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ArXiv*, abs/1611.03530, 2017.

Liu Ziyin, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. *ArXiv*, abs/2006.08195, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our work is primarily theoretical with few toy experiments we do not see its applicability
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix A for a list
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix J
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See experimental sections in the Appendix and main body
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Except for the cases where there are multiple graphs that are overlapping (Figure 6,7, 8) because it would make interpreting them difficult.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Appendix J
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Assumptions

We first make explicit the assumptions on the distribution of weights and biases.

A1: The conditional distribution of any set of biases $b_{z_1}, ..., b_{z_k}$ given all other weights and biases has a density $\rho_{z_1,...,z_k}(b_1, ..., b_k)$ with respect to Lebesgue measure on $\mathbb{R}^k$.

A2: The joint distribution of all weights has a density with respect to Lebesgue measure on $\mathbb{R}^{\#\text{weights}}$.

A3: The data manifold M is smooth.

A4: (Only needed for Theorem 3) the diameter of M defined by $d_M = \sup_{x,y \in M} \text{distance}_M(x, y)$ is finite.

A5: (Only needed for Theorem 3) a geodesic ball in manifold M has polynomial volume growth of order m.

B Additional Background on Manifolds

We provide further background on the theory of manifolds. In this section we first provide the background, definition and an interpretation for the **scalar curvature** of a manifold at a point. Every smooth manifold is also equipped with a *Riemannian metric tensor* (or metric tensor in short). Given any two vectors, v and w, in the tangent space of a point x on a manifold M, the metric tensor defines a parallel to the dot product in Euclidean spaces. The metric tensor, at a point x, is defined by the smooth functions $g_{ij} : M \to \mathbb{R}, i, j \in \{1, ..., k\}$. Where the matrix defined by

$$G_x = [g_{ij}(x)] = \begin{bmatrix} g_{11}(x) & \dots & g_{1n}(x) \\ \vdots & \ddots & \vdots \\ g_{n1}(x) & \dots & g_{nn}(x) \end{bmatrix}$$

is symmetric and invertible. The inner product of $u, v \in T_x M$ is then defined by $\langle u, v \rangle_M = u^T G_x v$. the inner product is symmetric, non-degenerate, and bilinear, i.e.

$$\begin{aligned} \langle ku, v \rangle_M =& k\langle u, v \rangle_M = \langle u, kv \rangle_M, \\ \langle u + w, v \rangle_M =& \langle u, v \rangle_M + \langle w, v \rangle_M, \\ \langle u, v \rangle_M =& \langle v, u \rangle_M. \end{aligned}$$

As can be seen, these properties also hold for the Euclidean inner product (with $G_x = I$ for all x). Let the inverse of $G = [g_{ij}(x)]$ be denoted by $[g^{ij}(x)]$. Building on this definition of the metric tensor the Ricci curvature tensor is defined as

$$\begin{aligned} R_{ij} =& -\frac{1}{2} \sum_{a,b=1}^{n} \Big(\frac{\partial^2 g_{ij}}{\partial x_a \partial x_b} + \frac{\partial^2 g_{ab}}{\partial x_i \partial x_j} - \frac{\partial^2 g_{ib}}{\partial x_j \partial x_a} - \frac{\partial^2 g_{jb}}{\partial x_i \partial x_a} \Big) g^{ab} \\ &+ \sum_{a,b,c,d=1}^{n} \Big(\frac{1}{2} \frac{\partial g_{ac}}{\partial x_i} \frac{\partial g_{bd}}{\partial x_j} + \frac{\partial g_{ic}}{\partial x_a} \frac{\partial g_{jd}}{\partial x_b} - \frac{\partial g_{ic}}{\partial x_a} \frac{\partial g_{jb}}{\partial x_d} \Big) g^{ab} g^{cd} \\ &- \frac{1}{4} \sum_{a,b,c,d=1}^{n} \Big(\frac{\partial g_{jc}}{\partial x_i} + \frac{\partial g_{ic}}{\partial x_j} - \frac{\partial g_{ij}}{\partial x_c} \Big) g^{ab} g^{cd}. \end{aligned}$$

For geometric interpretations of the above tensors we refer the reader to the work by Loveridge [2004].

Another quantity, from the theory of manifolds, which we utilise in our proofs and theorems, is scalar curvature (or Ricci curvature). The curvature is a measure how much the volume of a geodisic ball on the manifold M, e.g. S^2, deviates from a $d - 1$ sphere in the flat space, e.g. $\mathbb{R}^3$. The volume on the manifold deviates by an amount proportional to the curvature. We illustrate this idea in figure 10. We refer the reader to works by Gray [1974] and Wan [2016] for further technical details. Since our main theorems relate to the volume of linear regions the scalar curvature plays an important role.

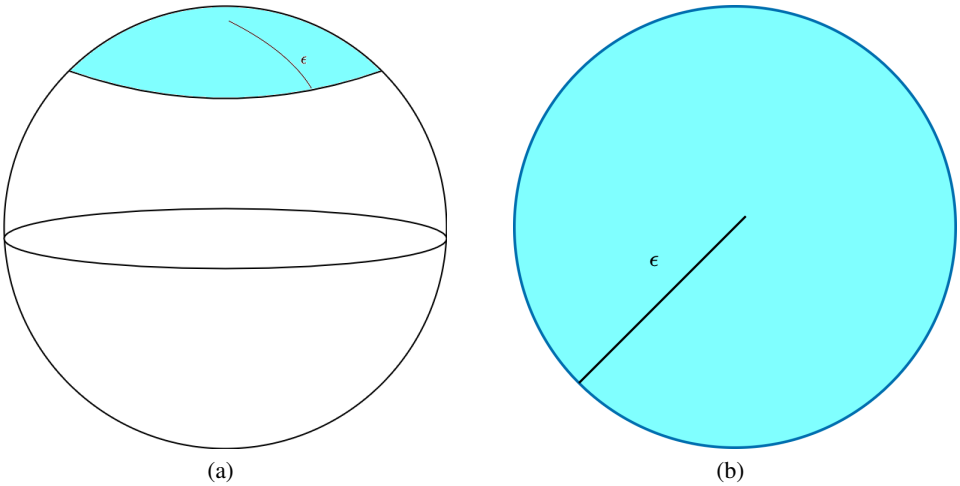


Figure 10: The geodesic circle on S^2 (blue region in (a)) does not have the same area as the flat circle (b), both of radius ϵ. One can imagine cutting the blue top off the sphere's surface and trying to "flatten" it. Such an effort will lead to failure, if the material of the sphere does not "stretch", since the geodesic ball, on S^2, cannot be mapped to a circle in $\mathbb{R}^2$ in a distance preserving manner. Thus, the area of the two blue regions in (a) and (b) vary. This deviation in the area spanned by the two spheres, despite their radii being the same, is proportional to the scalar curvature.

Formally, the scalar curvature of a manifold M at a point x with metric tensor $[g_{ij}]$ and Ricci tensor $[R_{ij}]$ is defined as

$$C = \sum_{i,j=1}^{n} g^{ij} R_{ij}.$$

Another important concept is that of **Hausdorff measure**. Since the volumes are "distorted" on a manifold it requires careful consideration when defining a measure and integrating using it on a manifold. The $m-$dimensional Hausdorff measure, of a set S, is defined as

$$H^m(S) := \sup_{\delta>0} \inf \Big\{ \sum_{i=1}^{\infty} (\text{diam } U_i)^d | S \subseteq \cup_{i=1}^{\infty} U_i, \text{diam } U_i < \delta \Big\}.$$

Next we introduce the definition of the **differential map** that is used in Definition 3.1, for the determinant of the Jacobian. The differential map of a smooth function H from a manifold M to a manifold S at a point $x \in M$ is the smooth map $dH : T_xM \to T_xS$ such that the tangent vector corresponding to any smooth curve $\gamma : I \to M$ at x, $\gamma'(0) \in T_xM$, maps to the tangent vector of $H \circ \gamma$ in $T_{H(x)}N$. This is the analog of the total derivative of "vanilla calculus". More intuitively, the differential map captures how the function changes along different directions on N as its input changes along different directions on M, this also has an analog to how rows of the Jacobian matrix are viewed in calculus. In Definition 3.1 we use the specific case where the function H maps from manifold M to the Euclidean space $\mathbb{R}^k$ and the tangent space of a Euclidean space is the Euclidean space itself. Finally, a paralellepiped's, P in T_xM, mapping via the differential map gives us the points in $\mathbb{R}^k$ that correspond to this set P.

C Related Work

There have been various approaches to explain the efficacy of DNNs in approximating arbitrarily complex functions. We briefly touch upon two such promising approaches. Broadly, the theory of DNNs can be viewed from two lenses: expressive power [Hornik et al., 1989, Bartlett et al., 1998, Poole et al., 2016, Raghu et al., 2017, Kawaguchi et al., 2017, Neyshabur et al., 2018, Hanin, 2019] and learning dynamics [Saxe et al., 2014, Su et al., 2016, Smith and Le, 2018, Jacot et al., 2018, Lee et al., 2019, Arora et al., 2019a,b]. These approaches are not independent of one another but

complementary. For example, Kawaguchi et al. [2017] argue theoretically how the family of DNNs generalize well despite the large capacity of the function class. Neyshabur et al. [2018] provide PAC-Bayes generalization bounds which are improved upon by Arora et al. [2018]. Hanin [2019] shows that Deep ReLU networks of finite width can approximate any continuous, convex or smooth functions on a unit cube. These works look at DNNs from the lens of expressive power. More recently, there has been a surge in explaining how various algorithms arrive at these almost accurate function approximations by applying different theoretical models of DNNs. Jacot et al. [2018] provide results for convergence and generalization of DNNs in the infinite width limit by introducing a the neural tangent kernel (NTK). Hanin and Nica [2020] provide finite depth and width corrections for the NTK. Another line of work within the learning dynamics literature looks at implicit regularization that emerge from the learning algorithm and over-parametrised DNNs [Arora et al., 2019a,b, Du et al., 2018, Liang et al., 2019].

Researchers have begun to incorporate data geometry into the theoretical analyses of DNNs by applying the assumption that the data lies on a general manifold. First we note the works looking at DNNs from the lens of expressive power combined with the idea of data geometry. Shaham et al. [2015] demonstrate that the size of the neural network depends on the curvature of the data manifold and the complexity of the function, whilst depending weakly on the input data dimension, for their construction of sparsely-connected 4-layer neural networks. Cloninger and Klock [2020] show that their construction of deep ReLU nets achieve near optimal approximation rates which depend only on the intrinsic dimensionality of the data. Chen et al. [2019] exploit the low dimensional structure of data to enhance the function approximation capacity of Deep ReLU networks by means of theoretical guarantees. Schmidt-Hieber [2019] shows that sparsely connected deep ReLU networks can approximate a Holder function on a low dimensional manifold embedded in a high dimensional space. Simultaneously, researchers have incorporated data geometry into the learning dynamics line of work [Goldt et al., 2020, Paccolat et al., 2020, Buchanan et al., 2021, Wang et al., 2021]. Buchanan et al. [2021] apply the NTK model to study how DNNs can separate two curves, representing the data manifolds of two separate classes, on the unit sphere. Goldt et al. [2020] introduce the Hidden Manifold Model for structured data sets to capture the dynamics of two-layer neural networks trained with stochastic gradient descent. Rahaman et al. [2019] provide empirical results on which data manifolds are learned faster. Finally, the work by Novak et al. [2018] comes the closes in studying the number of linear regions on the data manifold. They study the change in input output Jacobian, and as a consequence the number of linear regions, for DNNs with piece-wise linearities. They provide empirical studies by counting the number of linear regions along lines connecting data points as a proxy for number of linear regions on the data manifold.

Our work fits into the study of expressive power of DNNs. The number of linear regions is a good proxy for the *practical* expressive power or approximation capacity of Deep ReLU networks [Montúfar et al., 2014]. The results surrounding the density of linear regions make the fewest simplifying assumptions both on the data and the architecture of the DNN. The results by Hanin and Rolnick [2019a] bound the number of linear regions orders of magnitude tighter than previous results by deriving bounds for the average case and not the worst case. Moreover, they demonstrate the validity empirically in a setting with very few simplifying assumptions. We introduce the manifold hypothesis to this setting in order to obtain tighter bounds for the first time. This introduces a toolbox of ideas from differential geometry to analyse the approximation capacity of deep ReLU networks.

In addition to the theoretical works listed above, there has been significant empirical work that applies DNNs to non-Euclidean data [Bronstein et al., 2017, 2021]. Here the data is assumed to be sampled from manifolds with certain geometric properties. For example, Ganea et al. [2018] design DNNs for data sampled from Hyperbolic spaces of arbitrary dimensionality and modify the forward and backward passes accordingly. There have been numerous applications of modified DNNs, namely graph convolutional networks, to graph data that incorporate the idea that graphs are discrete samples from a smooth manifold [Henaff et al., 2015, Monti et al., 2017, Kipf and Welling, 2017], see the survey by Wu et al. [2019] for a comprehensive review. Graph convolutional networks have also been applied to point cloud data for applications in graphics [Qi et al., 2017, Wang et al., 2019].

D Proof Sketch

In this section we provide an overview of how the three main theorems are proved. Theorem 3.2 provides an equality for measuring the volume of $m - k$ dimensional boundary regions on the

manifold. To this effect, we introduce the idea of viewing boundary regions as submanifolds on the data manifold instead of hyperplanes (Proposition 6). We then prove an equality between the volume of boundary regions and the Jacobian of the neurons over the manifold. We utilise the smooth coarea formula that, intuitively, is applied to integrate a function using level sets on a manifold. This completes the proof for Theorem 3.2.

To prove Theorem 3.3 we first prove that the Jacobian of a function on a manifold can be denoted using the volume of paralellepiped of vectors in the ambient space subject to a linear transform (Proposition 8). Using this result and combining it with Theorem 3.2 we can then give an inequality for the density of linear regions. As can be expected this volume depends on the aforementioned projection, which in turn is related to the geometry of the manifold.

Finally, for proving Theorem 3.4 we first provide an inequality over the tubular neighbourhood of the boundary region. We then use this result to lower bound the geodesic distance between the boundary region and any random point on the manifold. The proof strategy follows that of Hanin and Rolnick [2019a] but there are major deviations when it comes to accounting for the geometry of the data manifold. To the best of our knowledge, we are utilising elements of differential topology that are unique to machine learning when it comes to developing a theoretical understanding of DNNs.

E Proof of Theorem 3.2

We follow the proof strategy used by Hanin and Rolnick [2019a] but deviate from it to account for our setting where $x \in M$. Let S_z be the set of values at which the neuron z has a discontinuity in the differential of its output (or the neuron switches between the two linear regions of the piece-wise linear activation σ),

$$S_z := \{x \in \mathbb{R}^{n_{\text{in}}} | z(x) - b_z = 0\}.$$

We also have

$$\mathcal{O} := \Big\{x \in \mathbb{R}^{n_{\text{in}}} | \forall j = 1, ..., L \; \exists \text{ neuron } z \text{ with } l(z) = j \text{ s.t. } \sigma'(z(x) - b_z) \neq 0\Big\}.$$

Further,

$$\widetilde{S_z} := S_z \cap \mathcal{O}.$$

We state propositions 9 and 10 by Hanin and Rolnick [2019a] as we apply them to prove Theorem 3.2, relabeling them as needed.

Proposition E.1. ***(Proposition 9 by Hanin and Rolnick [2019a])*** *Under assumptions A1 and A2, we have, with probability 1,*

$$B_F = \bigcup_{\textit{neurons } z} \widetilde{S_z}.$$

By extending the notion of S_z to multiple neurons we have

$$\widetilde{S}_{z_1,...,z_k} := \bigcap_{j=1}^{k} \widetilde{S}_{z_j},$$

meaning that the set $\widetilde{S}_{z_1,...,z_k}$ is, intuitively, the collection of inputs in $\mathbb{R}^{\text{in}}$ where the neurons $z_j, j = 1, ..., k$, switch between linear regions for σ and at which the output of F is affected by the outputs of these neurons. We refer the reader to section B of the appendix in the work by Hanin and Rolnick [2019a] for an intuitive explanation of proposition E.1. Before proceeding we provide a formal definition and intuition for the set $\mathcal{B}_{F,k}$,

$$B_{F,k} = \{x | x \in B_F \setminus \{\mathcal{B}_{F,0} \cup ... \cup \mathcal{B}_{F,k-1}\} = \mathcal{B}_{F,-k} \text{ and for any ball of radius } \epsilon > 0,$$
$$B(x, \epsilon) \cap \mathcal{B}_{F,-k} \text{ is subset to a } n - k \text{ dimensional hyperplane}\}.$$

Following the explanation provided by Hanin and Rolnick [2019a], $\mathcal{B}_{F,k}$ is the $n_{\text{in}} - k$ dimensional piece of $\mathcal{B}_F$. Suppose the boundaries of linear regions for $n_{\text{in}} = 2$ are unions of polygon boundaries, as depicted in Figure 2 of the main body of the paper, then $\mathcal{B}_{F,1}$ are all the open line segments of these polygons and $\mathcal{B}_{F,2}$ are the end points. Next we state Proposition 10 by Hanin and Rolnick [2019a].

Proposition E.2. ***(Prosposition 10 by Hanin and Rolnick [2019a])*** *Fix $k = 1, ..., n_{in}$, and k distinct neurons $z_1, ..., z_k$ in F. Then, with probability 1, for every $x \in B_{F,k}$ there exists a neighbourhood in which $B_{F,k}$ coincides with a $n_{in-k}-$dimensional hyperplane.*

We now present Proposition E.4, and its proof, which incorporates the additional constraint that $x \in M$, which is an m-dimensional manifold in $\mathbb{R}^{n_{\text{in}}}$. To prove the proposition we need the definition of tranversal intersection of two manifolds [Guillemin and Pollack, 1974].

Definition E.3. *Two submanifolds, M_1 and M_2, of S are said to intersect transversally if at every point of intersection their tangent spaces, at that point, together generate the tangent space of the manifold, S, by means of linear combinations. Formally, for all $x \in M_1 \cap M_2$*

$$T_x S = T_x M_1 + T_x M_2,$$

if and only if M_1 and M_2 intersect transversally.

For example, given a 2D hyperplane, P, and the surface of a 3D sphere, S^2, intersect in the ambient space $\mathbb{R}^3$. We have that this intersection is transverse if and only if P is not tangent to S_2. For the case where a 2D hyperplane, $\bar{P}$, intersects with S^2 at a point p but does not intersect tranversally it coincides exactly with the tangent plane of S^2 at point $\{p\} = S^2 \cap P$, i.e. $T_p S = P$. Note that in either case the tangent space of the 2D hyperplane P at any point of intersection is the plane itself.

Proposition E.4. *Fix $k = 1, ..., m$ and k distinct neurons $z_1, ..., z_k$ in F. Then, with probability 1, for every $x \in B_{F,k} \cap M$ there exists a neighbourhood in which $B_{F,k}$ coincides with an $m - k$ dimensional submanifold in $\mathbb{R}^{in}$.*

Proof. From Proposition E.2 we already know that $B_{F,k}$ is a $n_{\text{in}} - k$-dimensional hyperplane in some neighbourhood of x, with probability 1, for any $x \in B_{F,k} \cap M$. Let this hyperplane be denoted by P_k. This is an $n - k$ dimensional submanifold of $\mathbb{R}^{n_{\text{in}}}$. The tangent space of this hyperplane at x is the hyperplane itself. Therefore, from assumptions A1 and A2 we have that the probability that this hyperplane intersects the manifold M transversally with probability 1. In other words the probability that this plane P_k contains or is contained in $T_x M$ is 0. Finally, we have the intersection, $M \cap H_k$, has dimension $\dim(M) + \dim(H_k) - n_{\text{in}}$ [Guillemin and Pollack, 1974], which is equal to $m - k$. □

One implication of Proposition E.4 is that for any $k \leq m$ the $m - (k + 1)$ dimensional volume of $B_{F,k} \cap M$ is 0. In addition to that, Proposition E.4 implies that, with probability 1,

$$\text{vol}_{m-k}(\mathcal{B}_{F,k}) = \sum_{\text{distinct neurons } z_1,\ldots,z_k} \text{vol}_{m-k}(\widetilde{S}_{z_1,\ldots,z_k} \cap M). \tag{4}$$

The final step in the proof of Theorem 3.2 is to prove the following result.

Proposition E.5. *Let $z_1, ..., z_k$ be distinct neurons in F and $k \leq m$. Then for a bounded $m-$Hausdorff measurable manifold M embedded in $\mathbb{R}^{n_{in}}$,*

$$\mathbb{E}\Big[vol_{m-k}\Big(\widetilde{S}_{z_1,\ldots,z_k} \cap M\Big)\Big] = \int_M \mathbb{E}\Big[Y_{z_1,\ldots,z_k}(x)\Big] dx,$$

where $Y_{z_1,\ldots,z_k}(x)$ equals

$$J^M_{m,H_k}(x)\rho_{b_1,\ldots,b_k}(z_1(x), ..., z_k(x)),$$

times the indicator function of the event that z_j, for $j = 1, ..., k$, is good at x for every j and $H_k : \mathbb{R}^{n_{in}} \to \mathbb{R}^k$ is such that $H_k(x) = [z_1(x), ..., z_k(x)]^T$. The expectation is over the distribution of weights and biases.

Proof. Let $z_1, ..., z_k$ be distinct neurons in F and M be an $m-$dimensional compact Haudorff measurable manifold. We seek to compute the mean of $\text{vol}_{m-k}(\widetilde{S}_{z_1,\ldots,z_k} \cap M)$ over the distribution of weights and biases. We can rewrite this expression as

$$\int_{S_{z_1,\ldots,z_k} \cap M} \mathbf{1}_{z_j \text{ is good at } x} d\text{vol}_{m-k}(x). \tag{5}$$

The map H_k is Lipschitz and C^1 almost everywhere. We first note the smooth coarea formula (theorem 5.3.9 by Krantz and Parks [2008]) in context of our notation. Suppose $m \geq k$ and $H_k : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}^k$ is C^1 and $M \subseteq \mathbb{R}^{n_{\text{in}}}$ is an $m-$dimensional C^1 manifold in $\mathbb{R}^{n_{\text{in}}}$, then

$$\int_M g(x) J^M_{k,H_k}(x) d\text{vol}_m(x) = \int_{\mathbb{R}^k} \int_{M \cap H_k^{-1}(y)} g(y) d\text{vol}_{m-k}(y) d\text{vol}_k(x), \tag{6}$$

for every $\mathcal{H}^m$-measurable function g where J^M_{k,H_k} is as defined in Definition 3.1.

We denote preactivations and biases of neurons as $\mathbf{z}(x) = [z_1(x), ..., z_k(x)]^T$ and $\mathbf{b_z} = [b_{z_1}, ..., b_{z_k}]^T$. From the notation in A1, we have that

$$\rho_{\mathbf{b_z}} = \rho_{b_{z_1},...,b_{z_k}},$$

is the joint conditional density of $b_{z_1}, ..., b_{z_k}$ given all other weights and biases. The mean of the term in equation 5 over the conditional distribution of $b_{z_1}, ..., b_{z_k}$, $\rho_{\mathbf{b_z}}$, is therefore

$$\int_{\mathbb{R}^k} \mathbf{b} d\text{vol}_k(\mathbf{b}) \int_{\{\mathbf{z}=\mathbf{b}\} \cap M} \mathbf{1}_{z_j \text{ is good at } x} d\text{vol}_{m-k}(x), \tag{7}$$

where we denote $[b_1, ..., b_k]^T$ as $\mathbf{b}$. Thus applying the smooth co-area formula (Equation 6) to the expression in 7 shows that the average 5 is equal to

$$\int_M Y_{z_1,...,z_k}(x) dx.$$

Finally, we take the average over the remaining weights and biases and commute the expectation with the dx integral. We can do this since the integrand is non-negative. This gives us the result:

$$\mathbb{E}\Big[\text{vol}_{m-k}\Big(\widetilde{S}_{z_1,...,z_k} \cap M\Big)\Big] = \int_M \mathbb{E}\Big[Y_{z_1,...,z_k}(x)\Big] dx, \tag{8}$$

as required. □

Finally, taking the summation over all possible sets of distinct neurons $z_1, ..., z_k$ and combining equation 4 with Proposition E.5 completes the proof for Theorem 3.2.

F Proof of Theorem 3.3

To prove the upper bound in Theorem 3.3 we first show that the (determinant of) Jacobian for the function $H_k : M \to \mathbb{R}^k$, $H_k(x) = [z_1(x), ..., z_k(x)]^T$, as defined in 3.1 is equal to the volume of the parallelopiped defined by the vectors $\phi_{H_k}(\nabla z_j(x))$, for $j = 1, ..., k$, where $\phi_{H_k} : \mathbb{R}^k \to T_x M$ is an orthogonal projection onto the orthogonal complement of the kernel of the differential $D_M H_k$. Intuitively, this shows that with the added assumption $x \in M$ in Theorem 3.3 how exactly we can incorporate the geometry of the data manifold M into the upper bound provided by Hanin and Rolnick [2019a] in corollary 7.

Proposition F.1. *Given $H_k : M \to \mathbb{R}^k$ such that $H_k(x) = [z_1(x), ..., z_k(x)]^T$ and the differential $D_M H_k$ is surjective at x then*

$$J^M_{k,H_k}(x) = \sqrt{\det(\textit{Gram}(\phi_{H_k}(\nabla z_1(x)), ..., \phi_{H_k}(\nabla z_k(x))))}, \tag{9}$$

where $\phi_{H_k} : \mathbb{R}^n \to \mathbb{R}^k$ is a linear map and Gram denotes the Gramian matrix.

Proof. We first define the orthogonal complement of the kernel of the differential $D_M H_k$. For a manifold $M \subset \mathbb{R}^n$ and a fixed point x we have that $T_x M$ is a $m-$dimensional hyperplane. If we choose an orthonormal basis $e_1, ..., e_n$ of $\mathbb{R}^n$ such that $e_1, ..., e_m$ spans $T_x M$ for a fixed x we can denote all vectors in $T_x M$ using m coordinates corresponding to this basis. Therefore, for any vector $y \in \mathbb{R}^k$ we can get the orthogonal projection of y onto $T_x M$ using a $m \times n$ matrix which we denote as P_x, where $P_x y$ (matrix multiplied by a vector) represents a vector in $T_x M$ corresponding to the basis $e_1, ..., e_m$. For any manifold M in R^n and function $H_k : M \to \mathbb{R}^k$ we have that $D_M H_k : T_x M \to \mathbb{R}^k$ at a fixed point x is linear function. Therefore we can write $D_M H_k(v) = Av$

where $v \in T_x M$ is denoted using the aforementioned basis of $T_x M$. This implies that A is a $k \times m$ matrix. Therefore, the kernel of $D_M H_k$ for a fixed point $x \in M$ is

$$\ker(D_M H_k) = \Big\{ z | Az = 0 \text{ and } z \in T_x M \Big\}.$$

Since we can create a canonical basis for the space $\ker(D_M H_k)$ starting from the basis $e_1, ..., e_m$ in R^n using the Gram-Schmidt process given the matrix A we have that for any $y \in R^n$ we can project it orthogonally onto $\ker(D_M H_k)$. The orthogonal complement of $\ker(D_M H_k)$ is therefore defined by

$$\ker(D_M H_k)^{\perp} = \Big\{ a | a \cdot z = 0 \text{ for all } z \in \ker(D_M H_k) \text{ and } a \in T_x M \Big\}.$$

Similar to the previous argument, we construct a canonical basis starting from $e_1, ..., e_m$ for $\ker(D_M H_k)^{\perp}$ and therefore we can denote the orthogonal projection onto $\ker(D_M H_k)^{\perp}$ as a linear transformation. We denote this linear projection for fixed x using ϕ_k.

We denote the basis vectors $e_1, ..., e_m$ as a $m \times n$ matrix E where each row i corresponds to the vector e_i. Therefore, the orthogonal projection of any vector $y \in \mathbb{R}^n$ is Ey. Now we can get the matrix A using $E \nabla z_j(x)$ corresponding to each row j for $j = 1, ..., m$. This uses the fact that the direction of steepest ascent on $z_j(x)$ restricted to the tangent space $T_x M$ of the manifold M is an orthogonal projection of the direction of steepest ascent in $\mathbb{R}^n$.

Finally, from lemma 5.3.5 by Guillemin and Pollack [1974] we have that

$$J^M_{k,H_k}(x) = \mathcal{H}^k(D_M H_k(P))/\mathcal{H}^k(P),$$

for any parallelepiped P contained in $(\ker(D_M H_k))^{\perp}$. Arguing similar to the proof of lemma 5.3.5 by Guillemin and Pollack [1974] we get that

$$J^M_{k,H_k}(x) = \sqrt{\det((A)^T A)} = \sqrt{\det \text{Gram}(E \nabla z_1(x), ..., E \nabla z_k(x))},$$

thereby showing that $\phi_{H_k}(y) = Ey$ is a linear mapping. □

Although we state Proposition F.1 for neurons $z_j(x), j = 1, ..., k$ in the proof, it applies to any function that satisfy the conditions laid out in the proposition. Equipped with Proposition F.1 we prove Theorem 3.3. When the weights and biases of F are independent obtain an upper bound on $\rho_{b_{z_1}, ..., b_{z_k}}(b_1, ..., b_k)$ as

$$\Pi_{j=1}^k \rho_{b_{z_j}}(b_1, ..., b_k) \leq \Big(\sup_{\text{neurons } z} \rho_{b_z}(b) \Big)^k = C^k_{\text{bias}}.$$

Hence,

$$Y_{z_1, ..., z_k} \leq C^k_{\text{bias}} J^M_{k,H_k}.$$

From Proposition 9 we have that J^M_{k,H_k} is equal to the k-dimensional volume of the paralellopiped spanned by $\phi_x(\nabla z_j(x))$ for $j = 1, ..., k$. Therefore, we have

$$J^M_{k,H_k} \leq \Pi_{j=1}^k ||E \nabla z_j(x)|| \leq ||E||^k \Pi_{j=1}^k ||\nabla z_j(x)||, \tag{10}$$

where $||E||$ denotes the matrix norm which is defined as

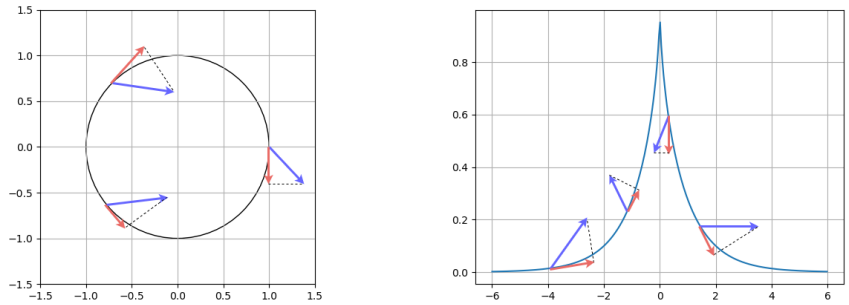
$$||E|| = \sup \Big\{ ||Ey|| \Big| y \in \mathbb{R}^k, ||y|| = 1 \Big\}.$$

Note that E does not depend on F (or $z_1, ..., z_k$) but only on $T_x M$ or more generally the geometry of M at any point x. From Theorem 3.2 by Hanin and Nica [2018] we have, for any fixed x,

$$\mathbb{E}\Big[\Pi_{j=1}^k ||\nabla z_j(x)|| \Big] \leq \Big(C_{\text{grad}} \Big)^k, \tag{11}$$

where,

$$C_{\text{grad}} = \sup_z \sup_{x \in \mathbb{R}^{n_{\text{in}}}} \mathbb{E}[||\nabla z(x)||^{2k}]^{1/k} \leq C e^{C \sum_{j=1}^d \frac{1}{n_j}},$$



(a) (b)

Figure 11: We illustrate how vectors project differently on tangent planes of two different manifolds: circle (a) and tractrix (b). In case of the tractrix the tangents (and the projection of vectors onto them) are on the inside of the tractrix whereas for the sphere the tangents are always on the outside of the sphere. Since the projections of vectors onto the tangent space are an essential aspect of our proof we end up with the term C_M, which quantifies the "shrinking" of these vectors upon projection, in the inequalities for Theorems 3.3 and 3.4.

wherein $C > 0$ depends only on μ and not on the architecture of F and n_j is the width of the hidden layer j. Let C_M be defined as

$$C_M := \sup \Big\{ C | \text{ there exists a set, S, of non zero } m-k\text{-dimensional Hausdorff measure} \\ \text{such that } ||E_x|| \geq C \forall x \in S \Big\}$$

Therefore, combining equations 11, 10 and result from Theorem 3.2 we have

$$\frac{\mathbb{E}[\text{vol}_{m-k}(\mathcal{B}_{F,k} \cap M)]}{\text{vol}_m(M)} \leq \binom{\text{number of neurons}}{k} (2C_{\text{grad}} C_{\text{bias}} C_M)^k,$$

where the expectation is over the distribution of weights and biases.

G Proof of Theorem 3.4

We first prove the following proposition

Proposition G.1. *For a compact m-dimensional submanifold M in $\mathbb{R}^n$, $m, n \geq 1$ and $m < n$ let $S \subseteq \mathbb{R}^n$ be a compact fixed continuous piece-wise linear submanifold with finitely many pieces and given any $U > 0$. Let $S_0 = \emptyset$ and let S_k be the union of the interiors of all k-dimensional pieces of $S \setminus (S_0 \cup ... \cup S_{k-1})$. Denote by T_ϵ the ϵ-tubuluar neighbourhood of any $X \subset M$ such that*

$$T_\epsilon(X) = \Big\{ y | d_M(y, X) < \epsilon \text{ and } y \in M \Big\},$$

where $\epsilon \in (0, U)$, d_M is the geodesic distance between the point y and set X on the manifold M, we have

$$vol_m(T_\epsilon(S)) \leq \sum_{k=n-m}^{d} vol_k(S_k \cap M) \omega_{n-k} \epsilon^{n-k} C_{k,\kappa,U},$$

where $C_{k,\kappa,U} > 0$ is a constant that depends on the average scalar curvature $\kappa_{(S_k \cap M)^\perp}$ and U, and ω_{n-k} is the volume of the unit ball in $\mathbb{R}^{n-k}$.

Proof. Define d to be the maximal dimension of linear pieces in S. Let $x \in T_\epsilon(X \cap M)$. Suppose $x \notin T_\epsilon(X \cap M)$ for all $k = n-m, ..., d-1$. Then the intersection of a geodesic ball of radius ϵ around s with S is a ball inside $S_d \cap M$. Using the convexity of this ball, with respect to the manifold M [Robbin et al., 2011], there exists a point y in $S_d \cap M$ such that the geodesic $\gamma : [0, 1] \to M$ with

$\gamma(0) = y$ and $\gamma(1) = x$ is perpendicular to $S_d \cap M$ at y. Formally, $T_{S_d \cap M} M$ at y is perpendicular to $\dot{\gamma(0)} \in T_M$ at y. Let $B_\epsilon(N^*(S_d \cap M))$ be the union of all the ϵ balls along the fiber of the submanifold $S_d \cap M$. Therefore, we have

$$\text{vol}_m(T_\epsilon(S \cap M) \leq \text{vol}_m(B_\epsilon(N^*(S_d \cap M)) + \text{vol}_m(T_\epsilon(S_{\leq d-1} \cap M)), \tag{12}$$

where $S_{\leq d-1} := \cup_{k=0}^{d-1} S_k$. We also note that

$$\text{vol}_m(B_\epsilon(N^*(S_d \cap M)) = \text{vol}_{m+d-n}(S_d \cap M)\text{vol}_{n-d}(B_\epsilon((M \cap S_d)^\perp)),$$

where $B_\epsilon((M \cap S_d)^\perp)$ is the average volume of an ϵ ball in the submanifold of M orthogonal to $M \cap S_d$. This volume depends on the average scalar curvature, $\kappa_{(M \cap S_d)^\perp}$ of the submanifold $(M \cap S_d)^\perp$. As shown by Wan [2016], for a fixed point $x \in (M \cap S_d)^\perp$

$$\text{vol}_{n-d}(B_\epsilon(x, (M \cap S_d)^\perp)) = \omega_{n-d}\epsilon^{n-d}\Big(1 - \frac{\kappa(x)_{(M \cap S_d)^\perp}}{n-d+2}\epsilon^2 + O(\epsilon^4)\Big),$$

where ω_{n-d} is the volume of the unit ball of dimension $n-d$, $B_\epsilon(x, (M \cap S_d)^\perp)$ is the geodesic ball of radius ϵ in the manifold $(M \cap S_d)^\perp$ centered at x and $\kappa_{(M \cap S_d)^\perp}(x)$ denotes the scalar curvature at point x. Gray [1974] provides the second order expansion of the formula above. Given that $\epsilon \in (0, U)$, for all $k \in \{n-m, n-m+1, ..., d\}$, then we have a smallest $C_{k,\kappa,U}$ such that

$$\text{vol}_k(B_\epsilon(x, (M \cap S_k)^\perp)) \leq C_{k,\kappa,U}\epsilon^k. \tag{13}$$

The above inequality follows from assumption A5. Using the above inequalities 12, 13 and repeating the argument $d-1-n+m$ times we get the result of the proposition. □

We also note that $C_{k,\kappa,U}$ increases monotonically with U, this also follows from the volume being monotonically increasing and positive for $\epsilon > 0$. Finally, we can now prove Theorem 3.4. Let $x \in M$ be uniformly chosen. Then, for all $\epsilon \in (0, U)$, using Markov's inequality and Proposition G.1, we have

$$\begin{aligned}
\mathbb{E}[\text{distance}_M(x, B_f \cap M)] &\geq \epsilon \Pr(\text{distance}_M(x, B_F \cap M) > \epsilon) \\
&= \epsilon(1 - \Pr(\text{distance}_M(x, B_F \cap M) <= \epsilon)) \\
&\geq \epsilon(1 - \sum_{k=n_{\text{in}}-m}^{n_{\text{in}}} \text{vol}_k(S_k \cap M)\omega_{n-k}\epsilon^{n_{\text{in}}-k}C_{n_{\text{in}}-k,\kappa,U}\Big) \\
&\geq \epsilon(1 - \sum_{k=n_{\text{in}}-m}^{n_{\text{in}}} C_{n_{\text{in}}-k,\kappa,U}(C_{\text{grad}}C_{\text{bias}}C_M\epsilon\{\#\text{neurons}\})^k\Big).
\end{aligned}$$

Note that as we increase U the constants $C_{n-k,\kappa,U}$ increase, although not strictly, for all k. To find the supremum of the expression on the right hand side, of the last inequality, in $\epsilon \in (0, U)$ we multiply and divide the expression by $C_{\text{grad}}C_{\text{bias}}C_M\#\text{neurons}$ to get the polynomial

$$p_U(\zeta) = \zeta\Big(1 - \sum_{k=n_{\text{in}}-m}^{n_{\text{in}}} C_{n_{\text{in}}-k,\kappa,U}\zeta^k\Big),$$

where $\zeta = \epsilon C_{\text{grad}}C_{\text{bias}}C_M\#\text{neurons}$ and $\zeta \in (0, U')$ where $U' = UC_{\text{grad}}C_{\text{bias}}C_M\#\text{neurons}$. Let d_M be the diameter of the manifold M, defined by $d_M = \sup_{x,y \in M} \text{distance}_M(x, y)$. We assume that d_M is finite. Taking the supremum over all $U \in (0, d_M]$ or $U' \in (0, d'_M]$, where $d'_M = d_M C_{\text{grad}}C_{\text{bias}}C_M\#\text{neurons}$, gives us the constant $C_{M,\kappa}$

$$C_{M,\kappa} = \sup_{U' \in (0, d'_M]} \{ \sup_{\zeta \in (0, U')} \{p_U(\zeta)\}\}.$$

Since d_M is finite the constant above exists and is finite. We make a note on the existence of this constant $C_{M,\kappa}$ in the absence of the constraint that the diameter of manifold M is finite. As U increases the constants $C_{n_{\text{in}}-k,\kappa,U}$ also increase and are all positive. The solution for $p'_U(\zeta) = 0, \zeta > 0$, which we denote by ζ_U, is unique and keeps decreasing as U increases. The uniqueness of the solution follows from the fact that the coefficients $C_{n_{\text{in}}-k,\kappa,U}$ are all positive. We also note

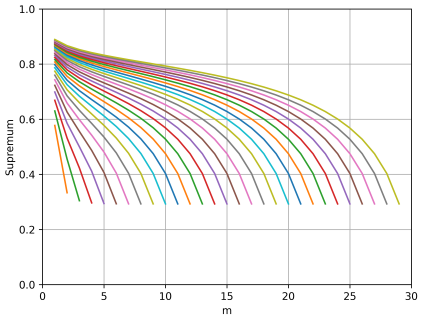


Figure 12: We plot the optima for a simplified polynomial as described in Section G.1. The individual plots correspond to n_{in} increasing from $n_{\text{in}} = 2$ to $n_{\text{in}} = 30$ (left to right) with m varying from 1 to $n_{\text{in}} - 1$ on the x-axis.

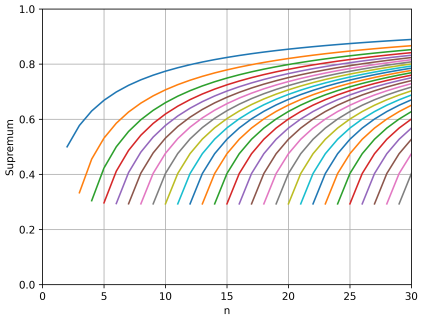


Figure 13: We plot the optima for a simplified polynomial as described in Section G.1. The individual plots correspond to m increasing from $m = 1$ to $m = 29$ (left to right) with n_{in} varying from $m + 1$ to 30 on the x-axis.

that $p_U(\zeta_U)$ need not be equal to $\sup_{\zeta \in (0,U')}\{p_U(\zeta)\}$ because ζ_U need not lie in $(0, U')$. In all such cases $\sup_{\zeta \in (0,U')}\{p_U(\zeta)\} = p_U(U')$. Given the polynomial $p_U(\zeta)$ above if we can assert that there exists a C_U, and the corresponding $C_{U'}$, such that for all $U > C_U$, and corresponding $U' > C_{U'}$, we have $\sup_{\zeta \in (0,U')}\{p_U(\zeta)\} = p_U(\zeta_U) < \infty$ and for all $0 < U \leq C_U$ we have $\sup_{\zeta \in (0,U')}\{p_U(\zeta)\} = p_U(U') < \infty$. Therefore, $C_{M,\kappa}$ exists and is finite if the previous assertion holds, proving this assertion is beyond the scope of our current work and particularly challenging.

Finally, taking the average over distribution of weights gives us the inequality

$$\mathbb{E}[\text{distance}_M(x, B_f \cap M)] \geq \frac{C_{M,\kappa}}{C_{\text{grad}} C_{\text{bias}} C_M \#\text{neurons}},$$

where $C_{M,\kappa}$ is a constant which depends on the average scalar curvature of the manifold M. This completes the proof of Theorem 3.4.

G.1 Variations in Supremum of p_U

We illustrate the dependence of the the constant $C_{M,\kappa}$ on varying values of n_{in}, m using a simple example. We fix the coefficient of the polynomial $p(\zeta)$ to be all 1, this not always the case but we do so to illustrate the relationship between the optima and the exponents for simplest such polynomial:

$$p_{\text{simplified}}(\zeta) = \zeta\Big(1 - \sum_{k=n_{\text{in}}-m}^{n_{\text{in}}} \zeta^k\Big)$$

We plot the supremums of this simplified polcynomial $C_{\text{simplified}} = \sup_{\zeta \in (0,1)} p_{\text{simplified}}(\zeta)$ for each n_{in} from the $\{2, ..., 30\}$ and varying m in Figure 12. Similarly, we vary n_{in} with fixed m and report the supremums $C_{\text{simplified}}$ in Figure 13. We notice that for a fixed n_{in} the supremum decreases with m and for a fixed m the supremum increases with n_{in}.

We programatically calculate the supremum being reported by restricting the domain of $p_{\text{simplified}}$ to $(0, 1)$. We solve for the supremum by using the `fminbound` method from the `scipy` package [Virtanen et al., 2020]. The function uses Brent's method [Brent, 1971] to find the supremum.

H Toy Supervised Learning Problems

For the two supervised learning tasks with different geometries (tractrix and sphere), we uniformly sample 1000 data points from each 1D manifold to come up with samples of (x_i, y_i) pairs. We then add Gaussian noise to y. We train a DNN with 2 hidden layers, with 10 and 16 neurons in each layer and a single linear output neuron, for a total of 26 neurons with piece-wise linearity, using the

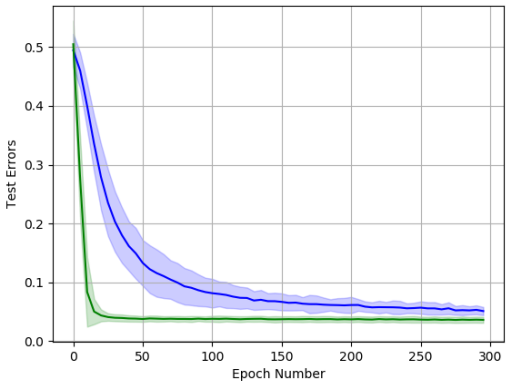


Figure 14: The test errors for the cases where data is sampled from the tractrix (blue) and the circle (green). We see that the tractrix converges slower but the magnitude of the errors remains comparable as training progresses across the two manifolds.

PyTorch library. The optimization is performed using the Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.01. We ensure a reasonable fit of the model by reducing the test time mean squared error (see Figure 14). We then calculate the exact number of linear regions on the respective domains by finding the points where $z(x(t)) = b_z$ for every neuron z and x is on the 1D manifold. We do this by adding neurons, z, one by one at every layer and using the SLSQP [Kraft, 1988] to solve for $|z(x(t)) - b_z| = 0$ in t for tractrix and $|z(x(\theta)) - b_z| = 0$ in θ for the circle. Note that this methodology can be extended to solve for linear regions of a deep ReLU network for any 1D curve $x(.)$ in any dimension. We then split a linear region depending on where this solution lies compared to previous layers. For every epoch, we then uniformly randomly sample points from the 1D manifold, by sampling directly from θ and t, to measure average distance to the nearest linear boundaries. The experiment was run on CPUs, from training to counting of number of linear regions. The intel cpus had access to 4 GB memory per core. A total of, approximately, 24 cpu hours were required for all the experiments in this section. This was run on an on demand cloud instance. All implementations are in PyTorch, except for SLQSP for which we used sklearn.

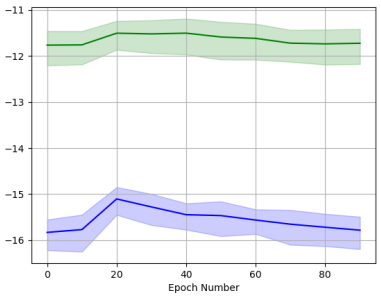
H.1 Varying n_{in}

The experimental setup, hyperparameters, network architecture, target function and methods are all the same as described for the toy supervised learning problem for the case where the geometry is a sphere. The only difference is that the input dimension varies, n_{in}.

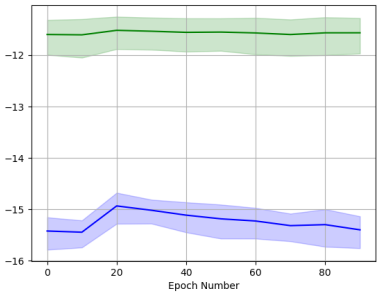
I High Dimensional Dataset

We utilise the official implementation of pretrained StyleGAN generator to generate curves of images that lie on the manifold of face images. Specifically, for each curve we sample a random pair of latent vectors: $z_1, z_2 \in \mathbb{R}^k$, this gives us the start and end point of the curve using the generator $g(z_1)$ and $g(z_2)$. We then generate 100 images to approximate a curve connecting the two images on the image manifold in a piece-wise manner. We do so by taking 100 points on the line connecting z_1 and z_2 in the latent space that are evenly spaced and generate an image from each one of them. Therefore, the ith image is generated as: $x_i = g(((100 - i) \times z_1 + i \times z_2)/100)$, using the StyleGAN generator g. We qualitatively verify the images to ensure that they lie on the manifold of images of faces. 4 examples of these curves, sampled as above, are illustrated in the video here: `https://drive.google.com/file/d/1p9B8ATVQGQYoiMh3Q22D-jSaI0USsoNx/view?usp=sharing`.

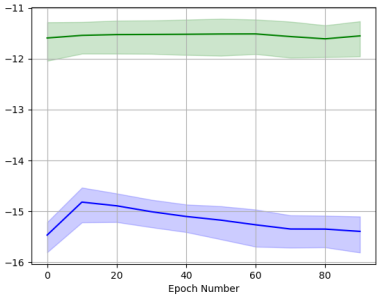
These two constructions allow us to formulate two curves in the high-dimensional setting. The straight line, with two fixed points $g(z_1)$ and $g(z_2)$, is defined as $x(t) = (1 - t)g(z_1) + tg(z_2)$ with $t \in [0, 1]$. The approximated curve on the manifold is defined as $x'(t) = (1 - t)g(z_i) + tg(z_{i+1})$



(a) LR: 0.025, momentum: 0.5, BS: 64



(b) LR: 0.005, momentum: 0.75, BS: 64



(c) LR: 0.01, momentum: 0.75, BS: 128

Figure 15: We report the log density of linear regions for various hyperparameters. Lr refers to the learning rate and BS is the batch size.

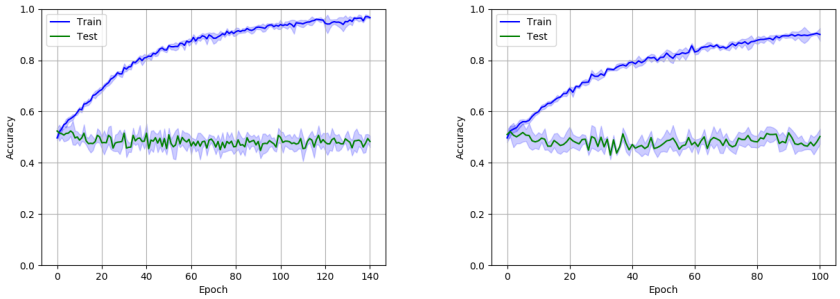
where $i = \texttt{floor}(100t)$. This once again gives us two curves and we solve for the zeros of $|z(x(t)) - b_z| = 0$ and $|z(x'(t)) - b_z| = 0$ for $t \in [0, 1]$ using SLQSP as described in Appendix H.

The neural network, used for classification in our MetFaces experiment, is feed forward with ReLU activation. There are two hidden layers with 256 and 64 neurons in the first and second layers respectively. We downsample the images to $128 \times 128 \times 3$. We augment the dataset using random horizontal flips of the images. All inputs are normalized. We use a batch size of 32. The neural network is trained using SGD. The learning rate is 0.01 and the momentum is 0.5. The total time required, for these experiments on MetFaces dataset, was approximately 36 GPU hours on a Titan RTX GPU that has 24 GB memory. This was run on an on demand cloud instance. We chose hyperparameters by trial and error, targeting a better fit for the training data for the results reported in Figure 9 of the main body of the paper.

We report further results for density of linear regions with varying hyperparameters in Figure 15. We also report the training and testing accuracy for the various sets of hyperparameters in Figure 16. Note that Figure 16(a) corresponds to the test and train accuracies on MetFaces reported in the main body of the paper (Figure 9). Note all of these results are for the same architecture as described above.

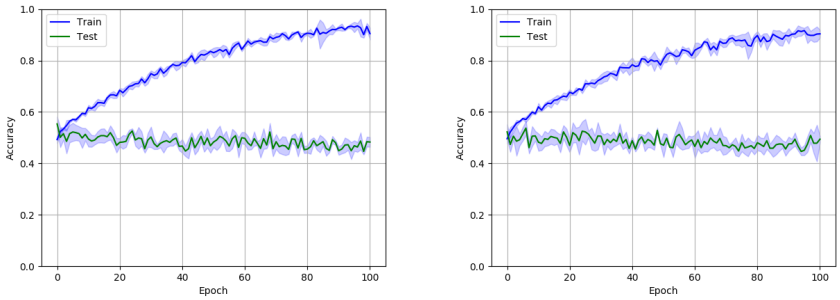
J Code, Data and Licenses

All the code used for our experiments (except the StyleGAN2 code) is enclosed in the folder `exp/`. The instructions to run all the experiments are enclosed in `exp/readme.txt`. We plan on releasing the code as an open github repository under the MIT License (`https://opensource.org/licenses/MIT`). The files changed on the github repository for the official implementation of StyleGAN2 (`https://github.com/NVlabs/stylegan2-ada-pytorch`) are enclosed in the folder `stylegan2-ada-pytorch`. The instructions to run the experiments are documented in `stylegan2-ada-pytorch/readme.txt`.



(a) LR: 0.01, momentum: 0.5, BS: 32

(b) LR: 0.025, momentum: 0.5, BS: 64



(c) LR: 0.005, momentum: 0.75, BS: 64

(d) LR: 0.01, momentum: 0.75, BS: 128

Figure 16: We report the test and train accuracies across 5 random seeds above.

Finally, the images we used to sample linear regions on a curve's piece-wise approximation on the manifold of face images, for the MetFaces experiment, are in the zip file `https://drive.google.com/file/d/1x5t-sc9NlW5N_ZBXUM0WcfX-toUXa85L/view?usp=sharing`.