Appendix: Effects of Data Geometry in Early Deep Learning

Anonymous Author(s)
Affiliation
Address
email

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

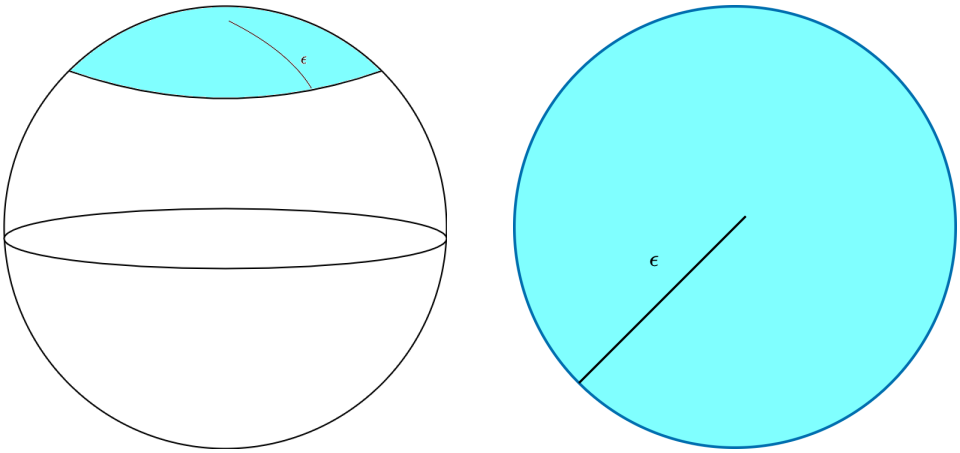
$$G_x = [g_{ij}(x)] = \begin{bmatrix} g_{11}(x) & \dots & g_{1n}(x) \\ \vdots & \ddots & \vdots \\ g_{n1}(x) & \dots & g_{nn}(x) \end{bmatrix}$$

is symmetric and invertible. The inner product of $u, v \in T_xM$ 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

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) (b)

Figure 1: 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.

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 1. 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. 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 [2019] 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 [2019] 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 [2019] 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 [2019] as we apply them to prove Theorem 3.2, relabeling them as needed.

Proposition E.1. ***(Proposition 9 by Hanin and Rolnick [2019])*** *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 [2019] 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 [2019], $\mathcal{B}_{F,k}$ 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 [2019].

Proposition E.2. ***(Prosposition 10 by Hanin and Rolnick [2019])*** *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,

$$\text{vol}_{m-k}(\mathcal{B}_{F,k}) = \sum_{\text{distinct neurons } z_1,...,z_k} \text{vol}_{m-k}(\widetilde{S}_{z_1,...,z_k} \cap M). \tag{1}$$

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 $\mathbb{R}^{n_{in}}$,*

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

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

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

times the indicator function of the event that z_j, 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 $\text{vol}_{m-k}(\widetilde{S}_{z_1,...,z_k} \cap M)$ over the distribution of weights and biases. We can rewrite this expression as

$$\int_{S_{z_1,...,z_k} \cap M} \mathbf{1}_{z_j \text{ is good at } x} d\text{vol}_{m-k}(x). \tag{2}$$

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

$$\int_M g(x) J^M_{k,H_k}(x) d\text{vol}_m(x) = \int_{\mathbb{R}^k} \int_{M \cap H_k^{-1}(y)} g(y) d\text{vol}_{m-k}(y) d\text{vol}_k(x), \tag{3}$$

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 2 over the conditional distribution of $b_{z_1}, ..., b_{z_k}$, $\rho_{\mathbf{b_z}}$, is therefore

$$\int_{\mathbb{R}^k} \mathbf{b} d\text{vol}_k(\mathbf{b}) \int_{\{\mathbf{z}=\mathbf{b}\} \cap M} \mathbf{1}_{z_j \text{ is good at } x} d\text{vol}_{m-k}(x), \tag{4}$$

where we denote $[b_1, ..., b_k]^T$ as $\mathbf{b}$. Thus applying the smooth co-area formula (Equation 3) to the expression in 4 shows that the average 2 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:

$$\mathbb{E}\Big[\text{vol}_{m-k}\Big(\widetilde{S}_{z_1,...,z_k} \cap M\Big)\Big] = \int_M \mathbb{E}\Big[Y_{z_1,...,z_k}(x)\Big]dx, \tag{5}$$

as required. □

Finally, taking the summation over all possible sets of distinct neurons $z_1, ..., z_k$ and combining equation 1 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 [2019] 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*

$$J^M_{k,H_k}(x) = \sqrt{\det(\textit{Gram}(\phi_{H_k}(\nabla z_1(x)), ..., \phi_{H_k}(\nabla z_k(x))))}, \tag{6}$$

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

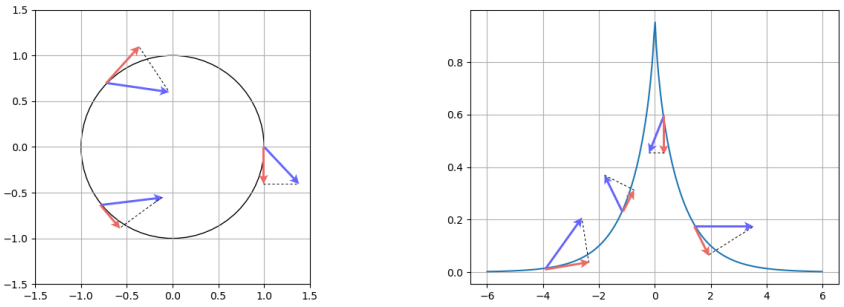
$$\Pi^k_{j=1} \rho_{b_{z_j}}(b_1, ..., b_k) \leq \Big(\sup_{\text{neurons } z} \rho_{b_z}(b) \Big)^k = C^k_{\text{bias}}.$$

Hence,

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

From Proposition 6 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

$$J^M_{k,H_k} \leq \Pi^k_{j=1} ||E \nabla z_j(x)|| \leq ||E||^k \Pi^k_{j=1} ||\nabla z_j(x)||, \tag{7}$$



(a) (b)

Figure 2: 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.

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_xM 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,

$$\mathbb{E}\Big[\Pi_{j=1}^k ||\nabla z_j(x)||\Big] \leq \Big(C_{\text{grad}}\Big)^k, \tag{8}$$

where,

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

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

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

Therefore, combining equations 8, 7 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

$$\text{vol}_m(T_\epsilon(S \cap M) \leq \text{vol}_m(B_\epsilon(N^*(S_d \cap M)) + \text{vol}_m(T_\epsilon(S_{\leq d-1} \cap M)), \tag{9}$$

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

$$\text{vol}_k(B_\epsilon(x, (M \cap S_k)^\perp)) \leq C_{k,\kappa,U}\epsilon^k. \tag{10}$$

The above inequality follows from assumption A5. Using the above inequalities 9, 10 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

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

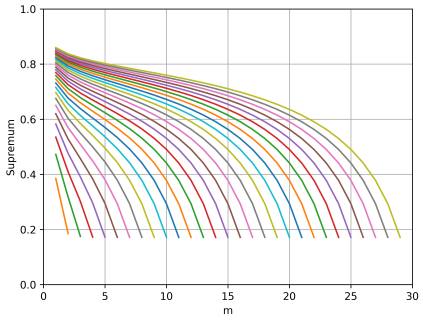


Figure 3: We plot the optima for a simplified polynomial as described in Section G.1. The individual plots correspond to n_{in} increasing from $n_{\text{in}} = 2$ to $n_{\text{in}} = 30$ (left to right) with m varying from 1 to $n_{\text{in}} - 1$ on the x-axis.

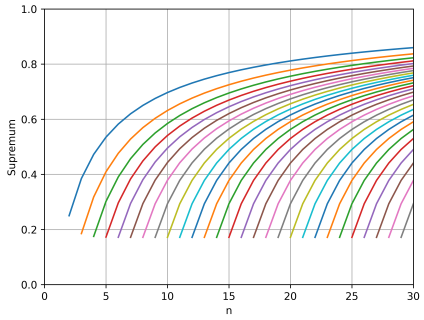


Figure 4: We plot the optima for a simplified polynomial as described in Section G.1. The individual plots correspond to m increasing from $m = 1$ to $m = 29$ (left to right) with n_{in} varying from $m + 1$ to 30 on the x-axis.

where $\zeta = \epsilon C_{\text{grad}} C_{\text{bias}} C_M \#\text{neurons}$ and $\zeta \in (0, U')$ where $U' = U C_{\text{grad}} C_{\text{bias}} C_M \#\text{neurons}$. 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 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 3. Similarly, we vary n_{in} with fixed m and report

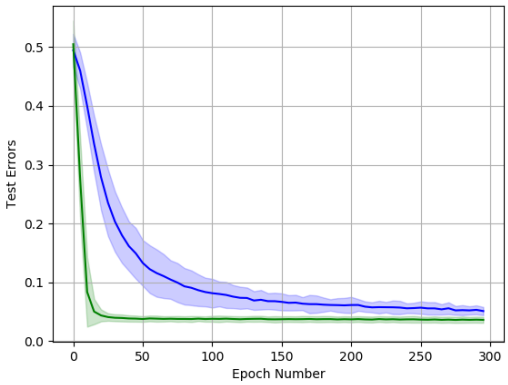


Figure 5: The test errors for the cases where data is sampled from the tractrix (blue) and the circle (green). We see that the tractrix converges slower but the magnitude of the errors remains comparable as training progresses across the two manifolds.

the supremums $C_{\text{simplified}}$ in Figure 4. 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

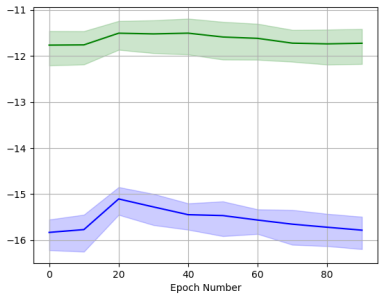
For the two supervised learning tasks with different geometries (tractrix and sphere), we uniformly sample 1000 data points from each 1D manifold to come up with samples of (x_i, y_i) pairs. We then add Gaussian noise to y. We train a DNN with 2 hidden layers, with 10 and 16 neurons in each layer and a single linear output neuron, for a total of 26 neurons with piece-wise linearity, using the 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 5). 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}

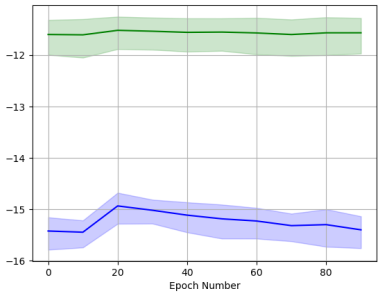
The experimental setup, hyperparameters, network architecture, target function and methods are all the same as described for the toy supervised learning problem for the case where the geometry is a sphere. The only difference is that the input dimension varies, n_{in}.

I High Dimensional Dataset

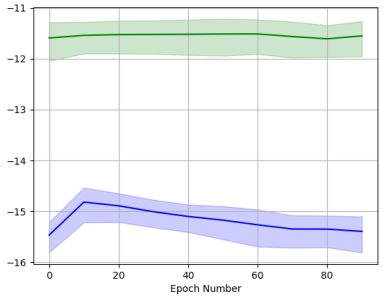
We utilise the official implementation of pretrained StyleGAN generator to generate curves of images that lie on the manifold of face images. Specifically, for each curve we sample a random pair of



(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

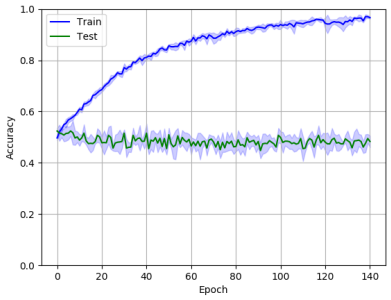
Figure 6: We report the log density of linear regions for various hyperparameters. Lr refers to the learning rate and BS is the batch size.

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 i^{th} 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`.

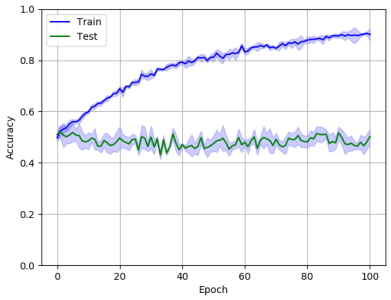
These two constructions allow us to formulate two curves in the high-dimensional setting. The straight line, with two fixed points $g(z_1)$ and $g(z_2)$, is defined as $x(t) = (1-t)g(z_1) + tg(z_2)$ with $t \in [0, 1]$. The approximated curve on the manifold is defined as $x'(t) = (1-t)g(z_i) + tg(z_{i+1})$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

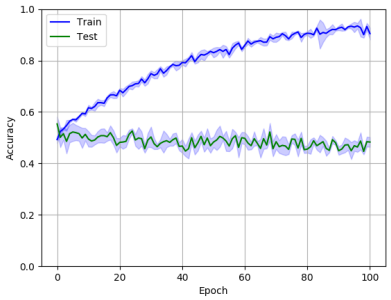
We report further results for density of linear regions with varying hyperparameters in Figure 6. We also report the training and testing accuracy for the various sets of hyperparameters in Figure 7. Note that Figure 7(a) corresponds to the test and train accuracies on MetFaces reported in the main body of the paper (Figure 9). Note all of these results are for the same architecture as described above.



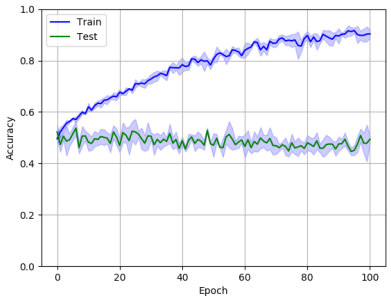
(a) LR: 0.01, momentum: 0.5, BS: 32



(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 7: We report the test and train accuracies across 5 random seeds 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`.

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_ZBXUMOWcfX-toUXa85L/view?usp=sharing`.

References

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.

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

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.

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.

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

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.

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, 2019.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.