Distance Learner: Incorporating Manifold Prior to Model Training

Aditya Chetan* ACHETAN@CS.CORNELL.EDU
Cornell University, Ithaca, NY, USA

Nipun Kwatra NKWATRA@MICROSOFT.COM
Microsoft Research India, Bengaluru, Karnataka, India

Editors: Sophia Sanborn, Christian Shewmake, Simone Azeglio, Nina Miolane

Abstract

The manifold hypothesis (real-world data concentrates near low-dimensional manifolds) is suggested as the principle behind the effectiveness of machine learning algorithms in very high-dimensional problems that are common in domains such as vision and speech. Multiple methods have been proposed to explicitly incorporate the manifold hypothesis as a prior in modern Deep Neural Networks (DNNs), with varying success. In this paper, we propose a new method, *Distance Learner*, to incorporate this prior for DNN-based classifiers. *Distance Learner* is trained to predict the *distance* of a point from the underlying manifold of each class, rather than the class label. For classification, *Distance Learner* then chooses the class corresponding to the closest predicted class manifold. *Distance Learner* can also identify points as being out of distribution (belonging to neither class), if the distance to the closest manifold is higher than a threshold. We evaluate our method on multiple synthetic datasets and show that *Distance Learner* learns much more meaningful classification boundaries compared to a standard classifier. We also evaluate our method on the task of adversarial robustness and find that it not only outperforms standard classifiers by a large margin but also performs at par with classifiers trained via well-accepted standard adversarial training.

Keywords: Manifold Learning, Manifold Hypothesis, Adversarial Robustness

1. Introduction

Most real world classification problems involve very high dimensional data inputs. However, learning in high dimensions suffers from what is popularly known as the *curse of dimensionality*. Consider, for example, a simple nearest neighbor based-classifier. For a new point to be "reasonably" close to a point in the training dataset, the number of points in the dataset need to be exponential in the data dimension. This can become prohibitive very quickly. Fortunately, as per the manifold hypothesis, real world data is believed to concentrate near low-dimensional manifolds even though the data is represented in much higher dimensions. Domingos (2012) calls this the "blessing of non-uniformity" as learners can implicitly or explicitly take advantage of the much lower effective dimension.

Deep Neural Networks (DNNs) have been extremely successful in achieving state-of-the-art performance on high dimensional classification tasks across multiple domains — e.g, image (Yu et al., 2022), text (Devlin et al., 2019; Raffel et al., 2020) and speech (Shen et al., 2018). A remarkable property of DNNs is their ability to generalize well on unseen test

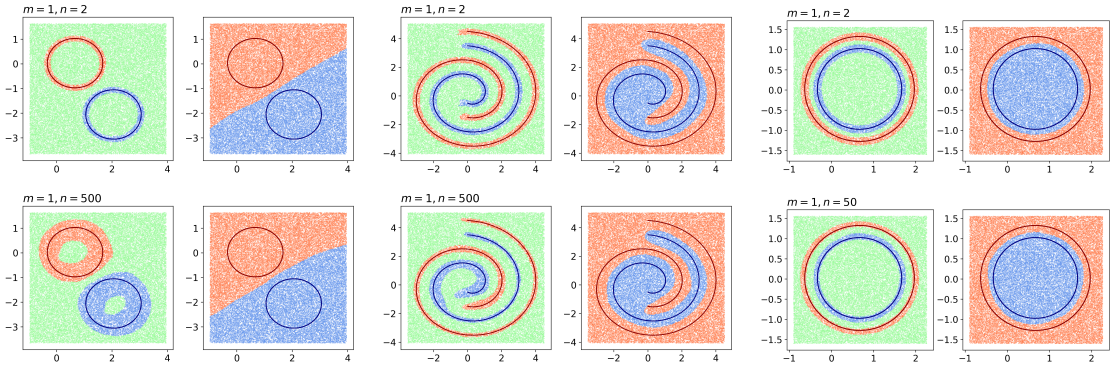
* Work done while Aditya was working at Microsoft Research India

data. Understanding the generalization properties of DNNs is an open area. The famous "no free lunch" theorems (Wolpert and Macready, 1997; Wolpert, 1996, 2002) state that on an average no learner can beat random guessing over all the possible learnt functions. Then, what makes DNNs so successful? Fortunately, the underlying functions for these real world tasks are not sampled randomly from the set of all possible functions. What makes learning work is incorporating some prior knowledge about the underlying function into the learning. Even simple priors such as smoothness can do very well. For example, neural networks encode the smoothness prior via construction of the network where all building blocks are smooth. Convolutional Neural Networks (CNN) encode the prior knowledge that the underlying classification function should be translation invariant. Similarly, attention based architectures (Vaswani et al., 2017) encode prior knowledge about semantic understanding in natural languages. A whole new field — *geometric deep learning* — looks at encoding symmetry constraints of the underlying domain into the network architecture (Cohen and Welling, 2016; Bronstein et al., 2021; Bietti et al., 2021). Incorporating these priors has shown significant improvements in accuracy as well as sample complexity (number of data points needed to achieve a given accuracy).

Given the importance of incorporating prior knowledge into learning and the strong prior provided by the manifold hypothesis, multiple methods have been proposed to incorporate this geometric knowledge into learners, either explicitly or implicitly. Traditional machine learning saw a collection of very successful methods, termed as *manifold learning* (Tenenbaum et al., 2000; Roweis and Saul, 2000; Zhang and Zha, 2004b), which aimed to directly learn the manifold. For incorporating manifold prior into modern DNNs, the most popular and commonly used method is domain-aware augmentation, where new samples are generated via manifold preserving transformations known from domain knowledge (e.g. translation, scaling, rotation, etc. for image classification). Other methods such as Manifold Tangent Classifier (Rifai et al., 2011) which encourage network invariance along learnt tangent spaces have also been proposed. However, other than augmentations, such methods have seen limited adoption.

In this paper, we propose a new method, *Distance Learner*, to incorporate the manifold prior for DNN-based classifiers. While standard classifiers are trained to learn the class label for each input point, *Distance Learner* is trained to predict the distance of an input point from the underlying manifold of each class. This is enabled by generating points (similar to augmentation) at a *known* distance to the underlying manifold. We do this by approximating the local manifold near each training point and adding controlled perturbations to the point in both the tangent space and normal space (orthogonal complement of the tangent space). The *Distance Learner* is then trained via an MSE loss to minimize the error between predicted and actual distance of these generated points, as well as the original training points (which have a distance of zero to the manifold). During inference, *Distance Learner*, chooses the class corresponding to the closest predicted class manifold.

Distance Learner has many advantages. First, it can learn much more meaningful decision regions (Figure 1), as classification is now based on distances to the class manifolds. By providing a distance along with the class label, our method is able to take advantage of the richer geometric information per sample. Second, *Distance Learner* can identify points that are out of distribution (belonging to neither class), as their predicted distance to all class manifolds will be very high. Standard classifiers on the other hand, are forced to give



(a) Separated Spheres (b) Swiss Roll (c) Concentric Spheres

Figure 1: **Decision Regions**. Side-by-side comparison of decision regions for the *Distance Learner* (left) and *Standard Classifier* (right). For easy visualization, only manifold dimensions of $m = 1$ are used. The two rows correspond to experiments with different values of embedding dimension n. Dark solid lines correspond to the actual class-manifolds, while dots correspond to samples in the test set. The blue and red dots are color-coded with the color of the predicted class, while green dots in the *Distance Learner* plots correspond to points identified as out-of-domain.

an output for *any* input. For example, a standard classifier trained to classify images of cats and dogs, will classify images of unrelated objects, say a car or even random noise, as a cat or a dog. DNNs often make such predictions with a high confidence (Hein et al., 2019) making identification of these cases hard. Third, learning meaningful boundaries can help make *Distance Learner* more robust to adversarial attacks. Vulnerability to adversarial attacks (Goodfellow et al., 2014; Huang et al., 2017; Akhtar and Mian, 2018; Chakraborty et al., 2018; Madry et al., 2018) has been attributed by many researchers to the high dimensionality of the representation space as well as that of the underlying data manifold (Gilmer et al., 2018; Stutz et al., 2019). Providing more structured geometric information about the manifold, can help alleviate these issues.

We perform experiments with multiple synthetic datasets to evaluate *Distance Learner*. We first visualize the distance boundaries learnt by *Distance Learner* and standard classifier on low-dimensional datasets, for a qualitative evaluation. Our experiments show that *Distance Learner* is able to learn much more meaningful boundaries, corresponding to the actual distance to underlying class manifolds. For experiments with high dimensional data, we build on the concentric spheres dataset proposed in Gilmer et al. (2018). We evaluate on the task of adversarial robustness, and show that *Distance Learner* not only outperforms the standard classifier significantly, but also performs comparable to classifiers trained via the standard adversarial training (Madry et al., 2018).

The contributions of our work can be summarized as follows:

1. A new learning paradigm, *Distance Learner*, which incorporates geometric information about the underlying data manifold by learning distances from class manifolds.
2. An efficient method to generate points at a known distance to the class manifolds, enabled by learning the local manifold near each training point.

3. Evaluation of the proposed approach on multiple synthetic datasets. In particular, we demonstrate that *Distance Learner* has adversarial robustness comparable to adversarial training (Madry et al., 2018). Our code is open-sourced at https://github.com/microsoft/distance-learner.

2. Distance Learner

The manifold hypothesis states that real world high-dimensional data lies on or near low-dimensional manifolds. Consider a classification dataset consisting of $C \in \mathbb{Z}^+$ classes: $\mathcal{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathbb{Z}^+, 1 \leq y_i \leq C\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ is an input point and y_i is the corresponding class label. Let $\mathcal{D}_c$ be the set of all points in class c, i.e. $\mathcal{D}_c = \{\mathbf{x}_i | (\mathbf{x}_i, y_i) \in \mathcal{D}, y_i = c\}$. The manifold hypothesis then states that the set $\mathcal{D}_c$ lies on a topological manifold, say $\mathcal{M}_c$, of dimension m much lower than n. Our aim is to incorporate geometric information about these manifolds, $\mathcal{M}_c$, into learning. For ease of exposition, we will use $\mathcal{M}$ to denote $\mathcal{M}_c$.

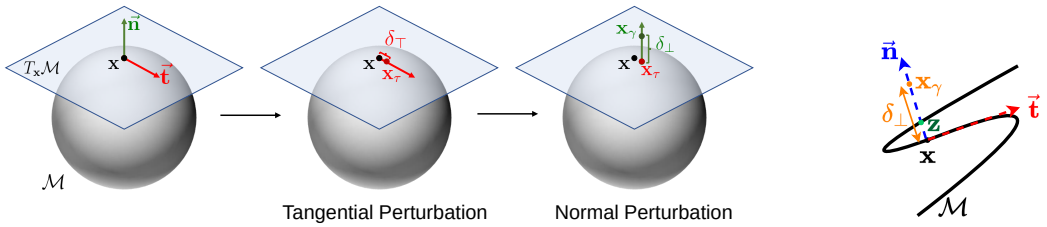
We propose to incorporate this geometric information via learning a model which can predict the distance of a given point to the manifold[1]. That is, we want the model to learn a function $f(\mathbf{x}) = \mathbf{d}$, where $\mathbf{x} \in \mathbb{R}^n$ and the output $\mathbf{d} \in \mathbb{R}^C$ contains the predicted distance to each class manifold. We call such a learner, which captures the distance field of a manifold, the *Distance Learner*. We note that the distance field of a manifold captures the entire manifold information since one can fully recover the actual manifold as its zero level-set. Thus distance learning can be a powerful way of incorporating manifold information.

For training *Distance Learner*, we propose a simple augmentation-based strategy. For each point in the dataset $\mathcal{D}$, we generate augmented points that are at a *known* distance to the underlying manifold. We discuss our augmentation method in section 2.1. Generating these augmented points, however, requires knowledge of the local manifold around each data point, which is not always available. We discuss a strategy to infer the local manifold for a given dataset $\mathcal{D}$ in section 2.2. Finally, *Distance Learner* is trained via an MSE loss to minimize the error between predicted and actual distances of the augmented points, as well as the original training points, which are at a distance of zero to the manifold. We discuss this further in section 2.3.

2.1. Off-manifold Point Sampling

We need to generate augmented points that are at a known distance to the underlying class manifold. Naïvely generating points via random sampling will be very expensive as computing distance to class manifolds will require brute-force comparison to all points in $\mathcal{D}$. Moreover, such a method will be very sample inefficient — we want to sample points close to class manifolds to learn fine-resolution distance fields near the manifolds, but randomly sampled points will be uniformly distributed in the entire domain of interest. Here, we propose an efficient sampling method which generates sample points close to the manifold, and at a known distance. For this, we take a point $\mathbf{x}$ from the training set $\mathcal{D}$ (points in $\mathcal{D}$ are assumed to be on the class manifold), and add a small perturbation to it. To perturb the point to a given distance from the manifold, we will need information about the manifold

1. Distance of a point $\mathbf{p}$ to a manifold $\mathcal{M}$ is defined as the minimum distance of $\mathbf{p}$ to any point on $\mathcal{M}$.



(a) (b)

Figure 2: **(a) Off-manifold sampling.** $\mathbf{x} \in \mathcal{M}$ is first perturbed in the tangent space $T_{\mathbf{x}}\mathcal{M}$ by a magnitude of $\delta_\top$ to obtain $\mathbf{x}_\tau$. Next, a perturbation of magnitude $\delta_\perp$ is added in the normal space, $N_{\mathbf{x}}\mathcal{M}$, to obtain an off-manifold point $\mathbf{x}_\gamma$. The distance of $\mathbf{x}_\gamma$ from $\mathcal{M}$ is then $\sim \delta_\perp$ **(b) Failure case at high curvature.** The closest manifold point to $\mathbf{x}_\gamma$ is now $\mathbf{z}$, which is at a distance of less than $\delta_\perp$.

locally near $\mathbf{x}$. For ease of exposition, we will assume in this section that this information is available. This assumption will be relaxed in section 2.2.

Let the underlying data manifold corresponding to $\mathbf{x}$ be $\mathcal{M}$ of dimensionality m. The manifold can be thought of as embedded in $\mathbb{R}^n$. The tangent space $T_{\mathbf{x}}\mathcal{M}$ at the point $\mathbf{x} \in \mathcal{M}$ is then informally defined as the m-hyperplane passing through $\mathbf{x}$ and tangent to $\mathcal{M}$. We assume that $T_{\mathbf{x}}\mathcal{M}$ is known, along with an orthonormal spanning basis $BT_{\mathbf{x}}$. Let $N_{\mathbf{x}}\mathcal{M}$ be the $(n-m)$-dimensional *normal*-space at $\mathbf{x}$, defined as the orthogonal complement of the tangent space w.r.t the embedded space. Let $BN_{\mathbf{x}}$ be a corresponding orthonormal spanning basis.

Once the tangent and normal spaces of the manifold near $\mathbf{x}$ are known, we generate an augmented point as follows. We first sample two random vectors $\mathbf{t}$ and $\mathbf{n}$ from the tangent and normal spaces, respectively. For sampling $\mathbf{t}$, we simply sample m scalars uniformly at random from $[0, 1]$ and use them as coefficients of the basis $BT_{\mathbf{x}}$. Sampling $\mathbf{n}$ proceeds similarly. The tangent and normal vectors are then combined to generate an off-manifold point via:

$$\mathbf{x}_\gamma = \mathbf{x} + \delta_\top \frac{\mathbf{t}}{\|\mathbf{t}\|} + \delta_\perp \frac{\mathbf{n}}{\|\mathbf{n}\|}. \tag{1}$$

That is, we perturb the on-manifold point $\mathbf{x}$ in the tangent space by an amount $\delta_\top$ and in the normal space by $\delta_\perp$. $\delta_\top$ and $\delta_\perp$ are sampled uniformly at random from $[0, max\_tangent]$ and $[0, max\_norm]$, where $max\_tangent$ and max_norm are hyperparameters.

If we assume $\delta_\top$, $\delta_\perp$ to be reasonably small, then the distance of $\mathbf{x}_\gamma$ from manifold $\mathcal{M}$ can be approximated to be $\delta_\perp$ (see Figure 2(a)). Since the tangential perturbation $\delta_\top$ is small, the tangentially perturbed point $\mathbf{x}_\tau = \mathbf{x} + \delta_\top \frac{\mathbf{t}}{\|\mathbf{t}\|}$ can be assumed to stay on the manifold. Also, we can assume that the tangent space and thus the normal space at $\mathbf{x}_\tau$ is the same as that at $\mathbf{x}$. Then, since the augmented point $\mathbf{x}_\gamma$ is generated by adding a small perturbation perpendicular to the tangent space at $\mathbf{x}_\tau$, the closest on-manifold point to $\mathbf{x}_\gamma$ will be $\mathbf{x}_\tau$, and thus the distance of the augmented point from the manifold will be $\|\mathbf{x}_\gamma - \mathbf{x}_\tau\| = \delta_\perp$. Note, however, that if the manifold has a very high curvature relative to $\delta_\top$ and $\delta_\perp$, this assumption may break (see Figure 2(b)). Thus, $max\_tangent$ and max_norm need to be chosen reasonably. Figure 6 shows a few examples of the generated points.

To summarize, given a point $\mathbf{x} \in \mathcal{D}$ and the tangent, normal spaces of the class manifold $\mathcal{M}$ at $\mathbf{x}$, we can generate an augmented point at a distance of $\delta_\perp$ from $\mathcal{M}$ using equation 1. Multiple augmentations can be generated for each $\mathbf{x}$ by random sampling of $\mathbf{t}, \mathbf{n}, \delta_\top$ and $\delta_\perp$.

2.2. Inferring the Local Manifold

In section 2.1, we assumed availability of the local manifold information near each point $\mathbf{x} \in \mathcal{D}$. In this section we relax this assumption, and discuss how we can infer the local manifold from the dataset $\mathcal{D}$. We use a simple method based on finding k nearest neighbors and principal component analysis (Jolliffe, 1986), similar to the method in Zhang and Zha (2004a). Note that *Distance Learner* is agnostic to the specific choice of method used to infer the local manifold, and one can use more complex methods, such as the ones discussed in Lin and Zha (2008); Rifai et al. (2011).

To infer the local manifold near a point $\mathbf{x} \in \mathcal{D}$, we first find the k nearest neighbors (kNNs) of $\mathbf{x}$ in the dataset $\mathcal{D}$. We denote this set of kNNs as $\text{NN}_{\mathcal{M}}(\mathbf{x}, k)$, where $\mathcal{M}$ is the underlying class manifold of $\mathbf{x}$. Note that for nearest neighbor search, we use the Euclidean distance in the embedded space $\mathbb{R}^n$, and not the geodesic distance on $\mathcal{M}$. As a result, even though the neighbors are close in $\mathbb{R}^n$, they may not be close in $\mathcal{M}$. However, if $\mathcal{M}$ satisfies certain properties (e.g. bounded curvature), we can assume closeness w.r.t geodesic distance as well. See Lin and Zha (2008) for a discussion on these properties and failure cases.

Locally, a manifold is isomorphic to $\mathbb{R}^m$, where m is the manifold dimensionality. We assume the close neighbors, $\text{NN}_{\mathcal{M}}(\mathbf{x}, s)$, to lie on this local $\mathbb{R}^m$ subspace, or equivalently in the span of the tangent space. The tangent space $T_{\mathbf{x}}\mathcal{M}$ can then be constructed as follows. Let $L_{\mathcal{M}}(\mathbf{x}) = \{\mathbf{x}_j - \mathbf{x} | \mathbf{x}_j \in \text{NN}_{\mathcal{M}}(\mathbf{x}, k)\}$. The tangent space, $T_{\mathbf{x}}\mathcal{M}$, is then the span of vectors in $L_{\mathcal{M}}$ (as long as the number of independent vectors in $L_{\mathcal{M}}$ is $\geq m$). However, note that because of the discrete nature of $\mathcal{D}$, the nearest neighbors of $\mathbf{x}$ may not lie exactly in the span of the tangent space $T_{\mathbf{x}}\mathcal{M}$. Thus, the span of $L_{\mathcal{M}}$ may include components outside the tangent space. The severity of this problem will depend on the sampling density of $\mathcal{D}$. To remove these noisy components, we do a Principal Component Analysis on $L_{\mathcal{M}}$, and extract the top-m most important principal components (based on their explained variance). These components then form the orthonormal basis, $BT_{\mathbf{x}}$, of the tangent space $T_{\mathbf{x}}\mathcal{M}$.

Finally, we need to compute an orthonormal basis $BN_{\mathbf{x}}$ of the $(n-m)$-dimensional *normal*-space, $N_{\mathbf{x}}\mathcal{M}$, at $\mathbf{x}$. Since all vectors in $N_{\mathbf{x}}\mathcal{M}$ are orthogonal to the vectors in $T_{\mathbf{x}}\mathcal{M}$ (i.e., $\mathbf{n} \cdot \mathbf{t} = 0$, $\forall \mathbf{n} \in N_{\mathbf{x}}\mathcal{M}, \mathbf{t} \in T_{\mathbf{x}}\mathcal{M}$), the normal-space $N_{\mathbf{x}}\mathcal{M}$ is the null-space of $T_{\mathbf{x}}\mathcal{M}$. The basis, $BN_{\mathbf{x}}$, of this null-space can thus be computed by SVD of a matrix whose rows are elements of $BT_{\mathbf{x}}$.

Note that here we have assumed knowledge of the intrinsic dimensionality of the manifold $\mathcal{M}$, to choose the number of principal components, as well as to choose the minimum number of neighbors, k. There are multiple methods to compute the intrinsic dimensionality of a manifold (e.g., (Lin and Zha, 2008)) which can be used for the purpose.

2.3. Learning Distance

The augmented points generated above can now used to train our *Distance Learner*. However, note that our generated point $\mathbf{x}_\gamma$ has its distance known to only *one* class manifold — the manifold on which the point $\mathbf{x}$ lies. But we want our *Distance Learner* to predict

distances to *all* class manifolds. To enable this, we simply set the distance of the generated point to all other-class manifolds at a fixed high value, *high_distance*. This is reasonable, since the different class manifolds can be assumed to be well separated and points on (or near) one manifold will be far from all other manifolds. Setting a high value for distances to other-class manifolds captures the coarse distance information. Similarly, points in the dataset $\mathcal{D}$ itself have their distance set to zero for the class manifold they belong to, and to *high_distance* for all other classes.

Also note that since the normal perturbation $\delta_\perp \in [0, max\_norm]$, our augmented points are only generated in *max_norm* thick bands around each class manifold (see Figure 6). However, since these bands contain fine distance information, it is enough to encode the manifold structure (the manifold can still be recovered as the zero level-set). Moreover, the other-class manifolds provide samples at a high distance, which encourages the model to predict high values for points far outside these bands. In our experiments, we observe that even outside these bands, *Distance Learner* is able to learn a smoothly increasing function (Figure 7). We note that it is a common practice in level-sets-based surface representation methods (e.g. liquid surface representation for simulating fluids on a grid), where only small bands around the surface store the actual signed distance function (Osher and Sethian, 1988; Osher and Fedkiw, 2001; Enright et al., 2002).

Our method is agnostic to the specific architecture of the model. The model should have C outputs, corresponding to the predicted distance to each of the C classes. While training we minimize the mean squared error between the predicted and ground-truth distances.

2.4. Classification using Distance Learner

For classification using *Distance Learner*, we use two variants. In the first variant, we classify input points as belonging to one of the C classes. This is done by outputting the class with the closest predicted class manifold. In the second variant, we also allow classifying points as being out-of-domain (i.e., belonging to neither class), if the distance to the closest manifold is higher than a tolerance threshold, say *tol*. The tolerance corresponds to the distance from the class manifold which is considered close enough to still belong to that class. It can be determined from domain knowledge or empirically from a validation set. Since *Distance Learner* learns fine-grained distances only in *max_norm* bands around the class manifolds, we should choose $max\_norm > tol$ during training, so that *Distance Learner* has high fidelity predictions in the tolerance region. Note that standard classifiers have to classify all points (even unrelated points far from any class manifold) to one of the C classes the classifier was trained on. *Distance Learner* does not have this limitation.

3. Experiments

We evaluate *Distance Learner* on multiple synthetic datasets and tasks. Each dataset consists of m-dimensional manifolds embedded in $\mathbb{R}^n$ ($m < n$). For robust evaluation we experiment with m ranging from 1 to 50 and n ranging from 1 to 500. First, we do a set of qualitative evaluations to visualize the distance function and decision boundaries learnt by *Distance Learner*. To allow visualization, we only consider manifolds of 1 and 2 dimensions during the qualitative analysis. The manifolds can be embedded in much higher dimensions,

and we visualize using appropriate 2D slices. This is followed by quantitative experiments to evaluate 1) distance prediction accuracy, and 2) adversarial robustness of *Distance Learner*.

Model Architecture Our *Distance Learner* is based on a simple MLP-based architecture, inspired by works on learning neural implicit fields for 3D shapes, e.g. Park et al. (2019). Figure 5 gives a detailed description of the architecture.

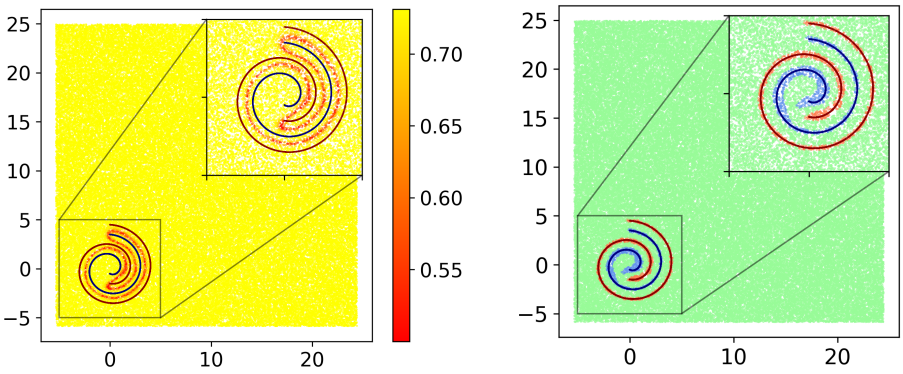
Datasets We evaluate on three sets of synthetic data — separated spheres, intertwined swiss rolls and concentric spheres (see appendix A.2). In each, m-dimensional manifolds are embedded in $\mathbb{R}^n$ ($m < n$). We experiment with multiple values of m and n for each dataset. Details of dataset generation and our embedding method are discussed in appendix C.

A note on accuracy In a test set of 20 million samples, Gilmer et al. (2018) observed no errors on the concentric spheres dataset. As a result, the true error rate of the model was unknown, and only a statistical upper bound was available. We observed similar behavior for all our datasets with both *Distance Learner* and *Standard Classifier*. However, in spite of this, we find adversarial examples, similar to Gilmer et al. (2018). Thus, we focus on adversarial robustness for quantitative evaluation.

Distance Prediction Accuracy *Distance Learner* is able to predict distances of points to the class manifolds with high accuracy. Table 1 (in appendix B.1) shows that even for high dimensions of manifold (m) and embedding space (n), the *Distance Learner* is able to achieve very low test losses. For example, in the case of $m = 1$, $n = 50$ for the concentric spheres dataset, we obtain test and train losses of $5.114\ (\pm 0.523) \times 10^{-9}$ and $4.757\ (\pm 0.478) \times 10^{-9}$, respectively. Similarly, even for higher manifold dimensions, e.g., $m = 50$ and $n = 500$, we obtain test and train losses of $1.380\ (\pm 0.547) \times 10^{-5}$ and $1.132\ (\pm 0.047) \times 10^{-6}$, respectively. Although there is an increase in losses as the dimensions increase (reflecting the hardness of learning at higher dimensions), the errors stay reasonably small. Note that we only use points sampled on the manifold itself and in *max_norm* bands around the manifold to compute these losses (full details in appendix D). Figure 7 (in appendix B.1) shows a visualization of the learnt distances as a heatmap. Here, we also include points outside the *max_norm* bands. Note that even though there are no augmented points outside the *max_norm* bands during training, the network is able to learn a smoothly increasing function for these points (e.g. see the plot boundaries). The observations hold even when the embedded dimension is very high ($n = 500$ in the 2^{nd} row of Figure 7).

Out-of-domain Classification & Decision Boundaries Since *Distance Learner* is able to predict distances to the class manifolds with good accuracy, we can use it to identify points that are out-of-domain (i.e., belonging to neither class). As discussed in section 2.4, points whose predicted distance to the closest manifold is higher than a tolerance threshold, *tol*, are classified as out-of-domain. Figure 1 shows a visualization (over 2D slices) of the classification done by *Standard Classifier* and *Distance Learner*. While the decision regions learned by *Standard Classifier* are very coarse and include even points very far from the manifold, *Distance Learner* is able to learn fine regions containing only points close the class manifolds. The observation holds true for both low and high dimensions.

Unlike *Standard Classifier*, where all points in the domain are assigned to one of the classes, *Distance Learner* only classifies points truly close to a class. This allows *Distance*



(*a*) *Standard Classifier* (*b*) *Distance Learner*

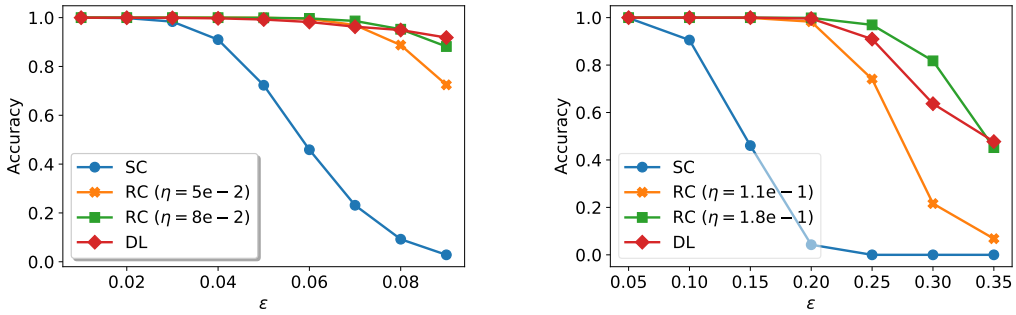
Figure 3: **Far region predictions for swiss roll dataset with** $m = 1, n = 500$**.** (a) Heatmap of prediction confidence for *Standard Classifier* in a large domain. Even points very far from the manifolds are assigned a class with high confidence of $\sim$1. Only for points close to the class manifolds (see inset), meaningful confidence scores are predicted. (b) *Distance Learner* decision region in a large domain. Far points are correctly predicted as out-of-domain, while accurate decision boundaries are obtained in the closer region (inset).

Learner to learn much more meaningful decision boundaries as can be seen in Figure 1. Another problem with standard classifiers is that not only do they assign a class to out-of-domain points, but they can do so with high confidence (Hein et al., 2019). We demonstrate this problem in Figure 3, which shows the confidence (softmax probability) heatmap for points in a 2D slice. As shown, even for points far from any class, *Standard Classifier* classifies them as one of the classes with high confidence.

Adversarial Robustness We now evaluate the robustness of *Distance Learner* to adversarial attacks. Here, we use an additional baseline, *Robust Classifier*, which is the *Standard Classifier* but trained via the adversarial training method (or robust-training) proposed by Madry et al. (2018). In robust-training, the model is trained by optimizing the min-max problem: $\min_\theta \mathbb{E}_{(x,y)\in\mathcal{D}} \left[\max_{\|\delta\|_2<\eta} \mathcal{L}(f(\mathbf{x}_i+\delta;\theta), y_i)\right]$, where θ denotes the network parameters, f denotes the network function, and η is the search radius for the adversarial perturbation in the inner loop. In our experiments, the inner loop is run for 40 iterations.

For a sample $(\mathbf{x}, y) \in \mathcal{D}$, we find an adversarial point by maximizing the loss $\mathcal{L}(f(\mathbf{x}+\delta;\theta), y)$ in an ϵ neighborhood around the point. For *Standard Classifier* $\mathcal{L}$ corresponds to the cross-entropy loss. For *Distance Learner* we maximize the predicted distance to class y minus the predicted distance to the other class, to drive towards misclassification. We solve this constrained optimization problem via the white-box projected gradient descent (PGD) method (Madry et al., 2018). We run 100 PGD iterations with a step size of 5e$-$3.

We focus on the Concentric Spheres and Intertwined Swiss Rolls datasets for analysis of adversarial robustness. Our analysis is inspired by Gilmer et al. (2018), which used



(a) Concentric Spheres (b) Intertwined Swiss Rolls

Figure 4: **Adversarial Robustness.** Comparisons of *Distance Learner* (**DL**), *Standard Classifier* (**SC**), and *Robust Classifier* (**RC**) over different values of ϵ (maximum adversarial perturbation). *Distance Learner* outperforms *Standard Classifier* and is comparable to *Robust Classifier*.

the concentric spheres dataset to extensively analyze the connection between adversarial robustness and high dimensional geometry. We focus on the case of $m = 50$ to mimic real image datasets. This is motivated by the findings of Pope et al. (2021), where for all popular image datasets (e.g. ImageNet), their highest estimate of intrinsic dimensionality was < 50.

Our results are shown in Figure 4. We evaluate adversarial robustness for multiple values of the attack neighborhood radius, ϵ. For *Robust Classifier*, we show results with $\eta \in \{5\text{e}{-2}, 8\text{e}{-2}\}$ for Concentric Spheres and $\eta \in \{1.1\text{e}{-1}, 1.8\text{e}{-1}\}$ for Intertwined Swiss Rolls. We use $tol = 0.14$ for Concentric Spheres, while for Intertwined Swiss Rolls, we classify using minimum predicted distance. For higher values of η, we found the performance of *Robust Classifier* to significantly degrade. As shown in the figure, *Distance Learner* not only outperforms the standard classifier significantly but also performs at par with adversarially trained classifiers. Since *Distance Learner* incorporates manifold geometry information into training, we suspect that it learns smoother and more meaningful decision boundaries, making it more robust to adversarial attacks.

4. Conclusion

The well-known no-free-lunch theorems drive the point that incorporating prior knowledge about the underlying learning problem can significantly improve outcomes. In this paper, we propose a novel method, *Distance Learner*, to incorporate the manifold prior while training DNNs. *Distance Learner* is trained to predict the distance of a point from the underlying class manifolds. We do this by efficiently generating augmented points at a known distance to the class manifolds. Our evaluation reveals that *Distance Learner* is robust to adversarial attacks, and is able to predict distances accurately. Moreover, *Distance Learner* is able to identify out-of-domain points and learn much more meaningful decision regions.

References

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

Alberto Bietti, Luca Venturi, and Joan Bruna. On the sample complexity of learning under geometric stability. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18673–18684. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/9ac5a6d86e8924182271bd820acbce0e-Paper.pdf.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL https://arxiv.org/abs/2104.13478.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/cohenc16.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, oct 2012. ISSN 0001-0782. doi: 10.1145/2347736.2347755. URL https://doi.org/10.1145/2347736.2347755.

Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics*, 183(1): 83–116, 2002.

Justin Gilmer, Luke Metz, Fartash Faghri, Sam Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres, 2018. URL https://openreview.net/forum?id=SyUkxxZ0b.

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

Sachin Goyal, Aditi Raghunathan, Moksh Jain, Harsha Vardhan Simhadri, and Prateek Jain. Drocc: Deep robust one-class classification. In *ICML 2020*,

July 2020. URL https://www.microsoft.com/en-us/research/publication/drocc-deep-robust-one-class-classification/.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

Nikhil Iyer, V Thejas, Nipun Kwatra, Ramachandran Ramjee, and Muthian Sivathanu. Wide-minima density hypothesis and the explore-exploit learning rate schedule. *arXiv preprint arXiv:2003.03977*, 2020.

I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

Tong Lin and Hongbin Zha. Riemannian manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):796–809, 2008.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018.

Stephen Marsland. *Machine Learning - An Algorithmic Perspective*. Chapman and Hall / CRC machine learning and pattern recognition series. CRC Press, 2009. ISBN 978-1-4200-6718-7.

Stanley Osher and Ronald P Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational physics*, 169(2):463–502, 2001.

Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79 (1):12–49, 1988.

Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 165–174. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00025. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Park_DeepSDF_Learning_Continuous_Signed_Distance_Functions_for_Shape_Representation_CVPR_2019_paper.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XJk19XzGq2J.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140): 1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Salah Rifai, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. *Advances in neural information processing systems*, 24, 2011.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098.

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Z. Chen, Yu Zhang, Yuxuan Wang, R. J. Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783, 2018.

David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. 2019.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

David H Wolpert. The supervised learning no-free-lunch theorems. *Soft computing and industry*, pages 25–42, 2002.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022. URL https://arxiv.org/abs/2205.01917.

Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing*, 26(1): 313–338, 2004a. doi: 10.1137/S1064827502419154. URL https://doi.org/10.1137/S1064827502419154.

Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004b.

Appendix A. Experimental Setup

A.1. Model Architecture

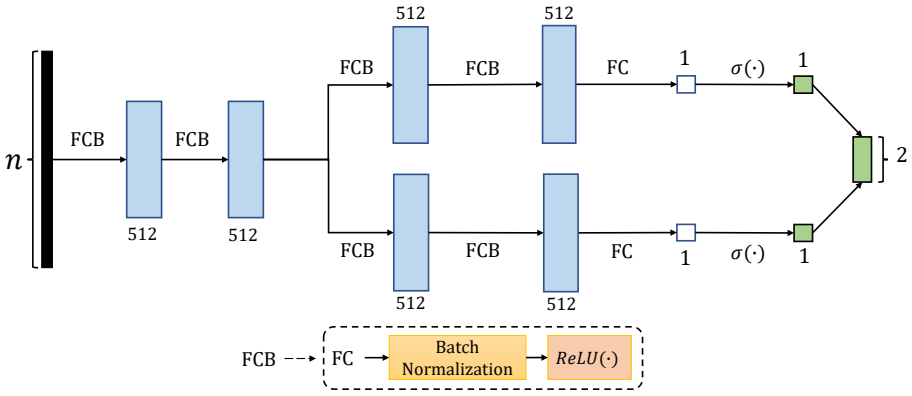


Figure 5: *Distance Learner* model architecture. The boxes denote tensors and the arrows denote operations. **FC** refers to a fully-connected layer, and **FCB** refers to a fully-connected layer followed by batch normalization and ReLU activation.

Figure 5 shows the architecture of our model. The first two layers of our model consist of **FCB** blocks (fully connected layers of 512 neurons, followed by batch normalization (Ioffe and Szegedy, 2015) and a ReLU activation). After this, the network branches into multiple pathways, one for each manifold. Each branch consists of two more **FCB** blocks, followed by a fully connected layer (**FC**) and finally a sigmoid activation (σ). The output of each branch gives the distance of the point from the branch's corresponding manifold. We collect the outputs of each branch into a single tensor which is then used for computing MSE loss. The branched network architecture was inspired by hard parameter-sharing techniques used in multi-task learning (Ruder, 2017). In our experiments, we found the branched architecture to improve the quality of the learnt distance function.

Our model was implemented using PyTorch (Paszke et al., 2019).

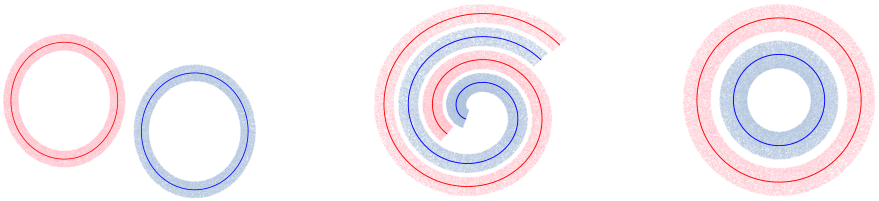
All experiments were performed on a V100 GPU with 16GB memory. We have open-sourced our code at https://github.com/microsoft/distance-learner.

A.2. Datasets

In this section, we provide details about the synthetic datasets used in our experiments.

Separated Spheres This is a simple dataset consisting of two hyperspheres of the same radius $r = 1$, where each sphere belongs to a different class (Figure 6(a)). The sphere centers are separated by a distance of 2.5 to prevent overlap.

Intertwined Swiss Rolls Swiss roll (Marsland, 2009) is a commonly used benchmark dataset for manifold learning techniques. A 2D Swiss roll embedded in $\mathbb{R}^3$ can be parameterized as: $\mathbf{x}(\phi, \psi) = (\phi \sin \phi,\ \phi \cos \phi,\ \psi)$. This can be generalized for an m dimensional Swiss roll embedded in $\mathbb{R}^{m+1}$ as, $\mathbf{x}(\phi, \psi_1, \psi_2, \ldots \psi_{m-1}) = (\phi \sin \phi,\ \phi \cos \phi,\ \psi_1,\ \psi_2,\ \ldots,\ \psi_{m-1})$.



(*a*) Separated Spheres (*b*) Intertwined Swiss Rolls (*c*) Concentric Spheres

Figure 6: **Synthetic datasets.** Representative plots with manifold dimension $m = 1$, and embedded dimension $n = 2$. Dark red and blue points lie on the manifold, while lightly colored points are the augmentations generated to train *Distance Learner*.

Embedding in dimensions higher than $m+1$ and other details are discussed in appendix C. Our complete dataset consists of two intertwined Swiss rolls (Figure 6(*b*)), where each Swiss roll corresponds to a different class.

Concentric Spheres This dataset consists of two concentric hyperspheres, with each sphere belonging to a different class (Figure 6(*c*)). The dataset is based on Gilmer et al. (2018), where they used this dataset extensively to analyze the connection between adversarial examples and high dimensional geometry. As in Gilmer et al. (2018), the inner and outer spheres' radii are set to 1.0 and 1.3, respectively. However, unlike Gilmer et al. (2018), where they used $m = 499$, $n = 500$, we used settings with m much lower than n. We do this to more closely model real-world datasets where the intrinsic dimensionality of the underlying data is much smaller than the embedded space dimension. In section 3, we focus on the case $m = 50$. This is motivated by the findings of Pope et al. (2021), where for all popular image datasets (e.g. ImageNet), their highest estimate of intrinsic dimensionality was less than 50. Experiments with other values of m are discussed in section 3 and appendix B.2. We use $n = 500$ in all experiments.

Appendix B. Results

In this section, we provide the complete results shared in the main paper in section 3, along with some additional results.

B.1. Distance Prediction Accuracy

As discussed in section 3, *Distance Learner* is able to predict accurate distances of points from the manifold. In Table 1, we report the test/train losses with mean and standard deviation computed over three runs started with different random seeds.

Heatmap of Learnt Distances In Figure 7, we show the heatmap of learnt distances talked about in section 3. For visualization, we only consider relevant 2D slices. Note that even though there are no points outside the *max_norm* bands during training, the network

Table 1: Train and test loss of *Distance Learner*

Dataset	m	n	Train Loss (av)	Test Loss (av)	# Test Examples
Separated Spheres	1	2	$2.450\ (\pm 0.157) \times 10^{-8}$	$2.719\ (\pm 0.185) \times 10^{-8}$	20,000
	1	50	$1.268\ (\pm 0.049) \times 10^{-6}$	$3.151\ (\pm 0.265) \times 10^{-7}$	20,000
	1	500	$2.287\ (\pm 0.143) \times 10^{-6}$	$3.151\ (\pm 0.231) \times 10^{-7}$	20,000
	2	500	$2.729\ (\pm 0.093) \times 10^{-7}$	$2.973\ (\pm 0.058) \times 10^{-6}$	20,000
Intertwined Swiss Rolls	1	2	$3.700\ (\pm 0.289) \times 10^{-7}$	$3.884\ (\pm 0.317) \times 10^{-7}$	20,000
	1	50	$9.792\ (\pm 1.657) \times 10^{-7}$	$2.256\ (\pm 0.550) \times 10^{-6}$	20,000
	1	500	$9.470\ (\pm 0.984) \times 10^{-6}$	$1.843\ (\pm 0.248) \times 10^{-5}$	20,000
Concentric Spheres	1	1	$4.757\ (\pm 0.478) \times 10^{-9}$	$5.114\ (\pm 0.523) \times 10^{-9}$	200,000
	1	50	$2.754\ (\pm 0.053) \times 10^{-7}$	$3.506\ (\pm 0.208) \times 10^{-7}$	200,000
	2	50	$5.118\ (\pm 0.230) \times 10^{-8}$	$1.283\ (\pm 0.215) \times 10^{-7}$	200,000
	25	500	$1.163\ (\pm 0.024) \times 10^{-6}$	$2.641\ (\pm 2.198) \times 10^{-6}$	200,000
	50	500	$1.132\ (\pm 0.047) \times 10^{-6}$	$1.380\ (\pm 0.547) \times 10^{-5}$	200,000

is able to learn a smoothly increasing function for these points (e.g. see the plot boundaries). The observations hold even when the embedded dimension is very high ($n = 500$ in the 2^{nd} row). Since the other-class manifold points are trained to have a *high_distance* value (1.0 in these experiments), the predicted values near the blue class manifolds are close to 1.0. To enable enough color resolution in the heatmaps, in Figure 7(b), we remove points with a predicted distance of $> .5$ in the 1^{st} and $> .25$ in the 2^{nd} and 3^{rd} rows. These points correspond to the blank white region. In Figure 7(a) we plot the full region (log scale), and in Figure 7(c), the complement of the point set plotted in Figure 7(b). For points closer to the other (blue) manifold, we can see that the *Distance Learner* predicts a very high value, as intended.

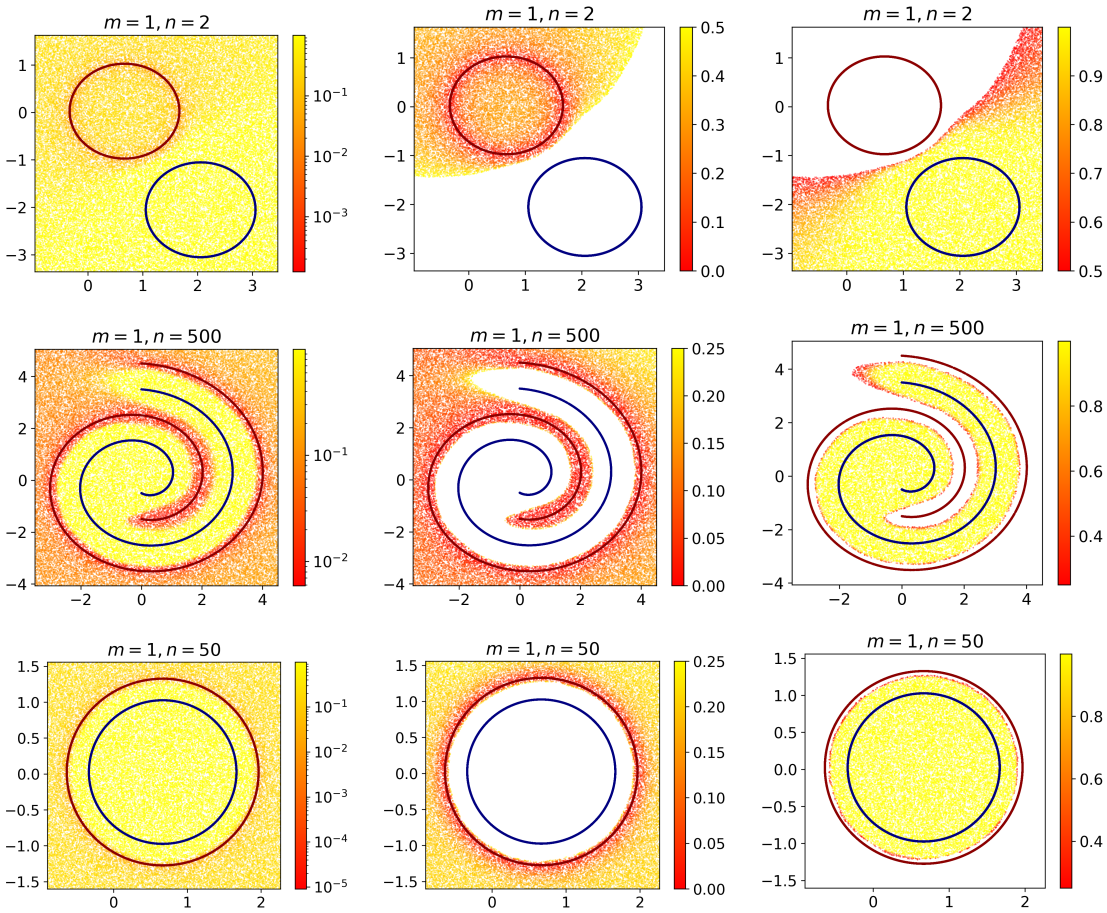
B.2. Adversarial Robustness

In section 3, we discussed the robustness of *Distance Learner* to adversarial attacks. In this section, we expand upon our experimental setup and present the results from our experiments.

Specifically, we evaluate the robustness of *Distance Learner* against white-box projected gradient descent (PGD) attacks. For this task, we compare *Distance Learner* against *Standard Classifier*, and a new baseline, the *Robust Classifier*. *Robust Classifier* is trained using robust-training proposed by Madry et al. (2018). Robust-training is a well-accepted standard method for making *Standard Classifier* robust to PGD attacks.

In a standard PGD attack, given a sample $(\mathbf{x}, y) \in \mathcal{D}$, we find an adversarial example by maximizing the loss $\mathcal{L}(f(\mathbf{x} + \delta; \theta), y)$ in an ϵ neighborhood around the point. For the *Standard Classifier* and *Robust Classifier*, we maximize the cross-entropy loss used in classification. For *Distance Learner* we maximize the predicted distance to the class manifold for the correct class y minus the predicted distance to the incorrect class. This would drive the *Distance Learner* towards misclassification.

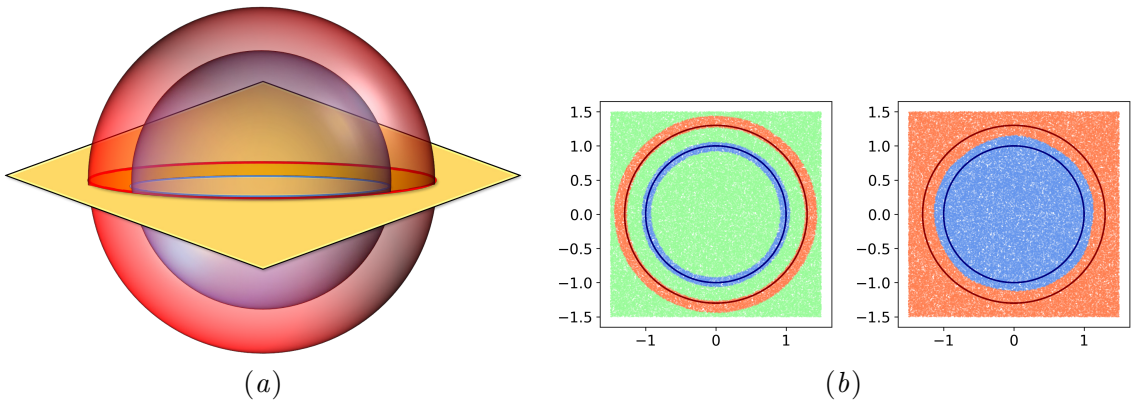
In our experiments, we focus on the Concentric Spheres and the Intertwined Swiss Rolls datasets. For each of these datasets, we generate points and perform training using the



(a) All points (b) Lower Distance Points (c) Higher Distance Points

Figure 7: Learnt Distances: Heatmap of predicted distance from the red class manifold. **(a)** Heatmap with all points, where the colors are decided by mapping the predicted distances to colors on a log scale.
For higher fidelity visualization we also show heatmaps on subsets of points: **(b)** points where the predicted value of distances are $\leq .5$ (1^{st} row) and $\leq .25$ (2^{nd} & 3^{rd} rows). **(c)** Heatmap of points complementary to those shown in (b), i.e. points with predicted distances $> .5$ (1^{st} row) and $> .25$ (2^{nd} & 3^{rd} rows).

method described in section 2 and appendix C. For the Intertwined Swiss Rolls dataset, we used the analytic form of the normals at each point instead of inferred normal directions for computing off-manifold points, as the inferred manifold was not a good approximation of the actual manifold. We set the intrinsic dimensionality of the datasets, $m = 50$. This is to mimic real-world datasets, based on the findings of Pope et al. (2021), where for all popular image datasets like ImageNet, their highest estimate of m was ≤ 50. We evaluate adversarial robustness for multiple values of the attack neighborhood radius, ϵ. For the classification of



(a) (b)

Figure 8: **(a)** Illustration of a plane used for plotting decision regions for concentric spheres dataset with $m = 2$, $n = 500$. **(b)** Decision regions for *Distance Learner* (left) and *Standard Classifier* (right).

adversarial examples with the *Distance Learner*, we use $tol = 0.14$ for Concentric Spheres, while for Intertwined Swiss Rolls, we classify using the minimum predicted distance.

Figure 4 shows our results. We can observe that *Distance Learner* not only outperforms *Standard Classifier* significantly but also performs at par with *Robust Classifier*. We suspect that due to priors about manifold geometry incorporated by the *Distance Learner*, it is able to learn a smoother and more meaningful decision boundary, which makes it more robust to adversarial attacks.

B.3. Out-of-domain Classification & Decision Boundaries

Figure 3 shows the confidence heatmap of far-off regions for the *Standard Classifier* as well as the decision boundaries learned by the *Distance Learner* as discussed in section 3. As discussed earlier, the *Standard Classifier* provides high-confidence predictions even for far-off regions, while the *Distance Learner* is able to use its learnt distance predictions to learn more accurate decision boundaries, labeling far-off regions as out-of-domain.

In Figure 1, we visualized decision regions only for $m = 1$ dimensional manifolds. Here we show decision regions for Concentric Spheres with $m = 2$ (Figure 8). Since a sphere of dimension, 2 has a canonical embedding in 3D space, in order to visualize the decision regions, we consider a 2D plane passing through the center of the two concentric spheres. Figure 8(a) shows an illustration of the place used for our visualization. Then, we visualize the predicted classes of points sampled on this plane using the *Distance Learner* and *Standard Classifier*. These decision regions are visualized in Figure 8(b). As we can observe, even for a higher intrinsic dimension $m = 2$, *Distance Learner* learns accurate decision regions, not only correctly identifying the two classes but also identifying out-of-domain points.

Appendix C. Synthetic Data Generation

In this section, we discuss the steps we followed to generate our synthetic datasets. To generate our synthetic datasets, we need to first sample points on the m-dimensional manifold, and then embed these points in the n-dimensional space $\mathbb{R}^n$. We perform this task in three successive steps – 1) sampling the m-dimensional manifold in an $m+1$-dimensional embedding (we call this the canonical embedding), 2) embedding these points in $\mathbb{R}^n$ via what we call *trivial* embedding and 3) applying a random transformation.

C.1. Sampling Canonical Embeddings

We first sample points on the m-dimensional manifold embedded in $\mathbb{R}^{m+1}$ (we call this the canonical embedding). This process depends on the dataset that we are using.

Separated Spheres Sampling the canonical embeddings amounts to sampling points uniformly from an m-sphere of required radius r and center $\mathbf{c}_m$ in $\mathbb{R}^{m+1}$. As a first step, we sample points uniformly from a unit-sphere centered at the origin using the steps described in Gilmer et al. (2018), i.e., sampling a point $\mathbf{z} \in \mathbb{R}^{m+1}$ from the standard normal distribution, $\mathcal{N}(\mathbf{0}, I)$, and the scaling $\mathbf{z}$, as follows: $\mathbf{y} = r\frac{\mathbf{z}}{\|\mathbf{z}\|}$. Finally, to centre the sphere at $\mathbf{c}_m$ apply a translation, $\mathbf{x} = \mathbf{y} + \mathbf{c}_m$. We generate centers by first sampling a random center for one of the spheres, and then perturbing it using a random perturbation of size d_c for the other sphere. We choose d_c so that there is no overlap between the two spheres.

Intertwined Swiss Rolls We discussed how the Swiss Roll is parameterized in the main text (see section 3). In order to obtain intertwined Swiss Rolls, we generate the inner Swiss Roll by changing the parameterization to $\mathbf{x}(\phi, \psi_1, \psi_2, \dots \psi_{m-1}) = ((\phi - \mu)\cos\phi, (\phi - \mu)\sin\phi, \psi_1, \psi_2, \dots \psi_3)$, where μ controls the gap between the Swiss Rolls, and is chosen appropriately to prevent overlap. For our experiments, we use $\phi \in [1.5, 4.5]$, $\psi \in [0, 21]$, and $\mu = 1$.

Concentric Spheres We follow the same process as Separated Spheres, except that only one common centre is sampled for both the spheres, and the difference between the radii of the two spheres is chosen suitably to prevent overlap.

C.2. Generating Trivial Embeddings

Let the canonical embeddings of the samples be stored as rows of a matrix $\mathbf{P}_c$ with $(m+1)$ columns. Once $\mathbf{P}_c$ is obtained, we embed the points trivially in $\mathbb{R}^n$ by concatenating the points with $(n-m-1)$ 0's. That is, the trivial embedding $\mathbf{P}_{\text{tr}}$ can be obtained as $\mathbf{P}_{\text{tr}} = \begin{bmatrix}\mathbf{P}_c & \mathbf{0}\end{bmatrix}$, where $\mathbf{0}$ is a zero matrix.

C.3. Random Transformations

Note that the embedding of the manifold in $\mathbb{R}^n$ is a trivial one, obtained by concatenating the canonical embeddings with 0's. To make this embedding more generalized, we apply random translation and rotation transforms to the data. In order to generate a random rotation transform, $\mathbf{Q} \in \mathbb{R}^{n\times n}$, we sample a random matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$, and decompose it using QR factorization to obtain an orthogonal matrix $\mathbf{Q}$ and a residual matrix $\mathbf{R}$. The

orthogonal matrix $\mathbf{Q}$ can be used as the rotation transform. We sample a random vector, $\mathbf{T} \in \mathbb{R}^n$ as a translation transform. We obtain the final transformed samples matrix $\mathbf{P}$ by applying these transforms row-wise to $\mathbf{P}_{\text{tr}}$. For training, we normalize these points so that they lie in a unit-hypercube. This helped significantly with training stability.

Appendix D. Hyperparameters

In this section we describe the various hyperparameter settings used in our experiments. Table 2 shows the hyperparameter values used for training. For all our experiments, we used the *Knee* learning rate schedule (Iyer et al., 2020), and increase the learning rate linearly from zero to its maximum value till epoch 10, and then decrease linearly to zero from epoch 700 to 1000. All models were trained for 1000 epochs.

Table 2: Values of hyperparameters for *Distance Learner*. $\star$ denotes that a hyperparameter sweep was conducted for this dataset. $N_{\text{on/off}}$ denote to the number of on-/off-manifold points respectively.

Dataset	m	n	*max_norm*	N_{on} $(\times 10^6)$	N_{off} $(\times 10^6)$	Learning Rate	Batch Size	
Separated Spheres	1	2	0.10	0.50	1.00	1.0×10^{-5}	512	
	1	50	0.10	0.50	1.00	1.0×10^{-5}	512	
	1	500	0.10	0.50	1.00	1.5×10^{-5}	4096	$\star$
	2	500	0.10	0.05	1.00	1.0×10^{-5}	512	
Intertwined Swiss Rolls	1	2	0.40	0.05	0.05	1.0×10^{-5}	512	$\star$
	1	50	0.40	0.05	0.05	1.0×10^{-5}	512	
	1	500	0.40	0.05	1.00	1.0×10^{-6}	4096	$\star$
Concentric Spheres	1	2	0.10	0.50	1.00	1.5×10^{-5}	4096	
	1	50	0.10	0.50	1.00	1.5×10^{-5}	4096	
	2	50	0.14	0.50	2.00	1.5×10^{-5}	4096	
	25	500	0.14	0.50	6.00	1.5×10^{-5}	4096	
	50	500	0.14	0.50	6.00	1.5×10^{-5}	4096	$\star$

Typically, if the dataset has a high value of n, we sampled a higher number of off-manifold augmentations (N_{off}) for the same number of on-manifold points (N_{on}). This is because the volume of an off-manifold band of width *max_norm* in n dimensions would roughly be of the order of $max\_norm^{(n-m)}$. Although the volume increases exponentially in $(n-m)$, because of small *max_norm*, we required only modest increases in the number of off-manifold augmentations, N_{off}. For instance, for Intertwined Swiss Roll with $m = 1$, as we go from $n = 2$ to $n = 500$, we are able to obtain extremely low classification error rates with just a $20\times$ increase in N_{off}.

In almost all cases, we obtained low error rates from the first set of parameters that we chose. However, we have performed hyperparameter tuning for a few settings (marked by $\star$ in Table 2) and found that it is possible to improve loss statistics further. We searched

for the optimal values of N_{off} ($\in [0.05, 6.00] \times 10^6$), learning rate ($\in [0.01, 8.00] \times 10^{-5}$), and batch size ($\in \{512, 2048, 4096\}$). Due to resource and time constraints, we could not tune hyperparameters for the other settings. Further hyperparameter tuning may improve performance in the other settings as well.

Appendix E. Limitations and Future Work

We acknowledge a few limitations of our current work. Firstly, we have evaluated *Distance Learner* only on synthetic datasets. Although it reveals important insights and promise of the current work, we would like to extend our evaluation to real-world datasets. This may require further work on accurately estimating local manifolds from training data. Also, in our current method, augmented points are sampled uniformly in small bands around the manifold. However, this may not be the most sample-efficient. We want to explore more efficient sampling techniques by incorporating ideas similar to Goyal et al. (2020). Finally, we would like to extend our method to also incorporate unsupervised data. Since inferring local manifold and sampling off-manifold points, only relies on nearest neighbors, unsupervised data can be easily incorporated for these steps. Given the abundant availability of unsupervised data for many tasks, this can provide significant gains.