Curvature Estimation on Data Manifolds via Diffusion-augmented Sampling

Jason Wang JASONWANG1@COLLEGE.HARVARD.EDU
Bobak Kiani BKIANI@G.HARVARD.EDU
Melanie Weber MWEBER@SEAS.HARVARD.EDU
Harvard University, School of Engineering and Applied Sciences

Editors: List of editors' names

Abstract

Data geometry is fundamental to machine learning and data analysis, yet practical tools for characterizing the geometry of data manifolds remain limited. While intrinsic dimension estimation is well-studied, curvature, a key measure of local manifold structure, is far harder to approximate from noisy, sparsely sampled data. We introduce a diffusion-based framework for curvature estimation aiming to mitigate challenges due to low sample density. We train a diffusion model to learn a latent representation of the manifold, which we then probe to augment the raw dataset and obtain a denser sample. Compared to state-of-the-art curvature estimators applied directly to the raw data, diffusion-augmented methods achieve superior performance on heterogeneous manifolds when using high-fidelity diffusion models.
Keywords: curvature estimation, scalar curvature, normal curvature, diffusion model

1. Introduction

Understanding structure in data and leveraging it to improve learning is a central goal of machine learning. While deep neural networks excel at learning from high-dimensional data, our understanding of real-world data geometry and its role in learning remains limited. Models are known to exploit low-dimensional structure, identifiable through dimensionality reduction, but this offers only a partial view of data geometry. Curvature provides a complementary tool, capturing how a data manifold deviates from a linear subspace and characterizing its nonlinear structure. However, while manifold learning and intrinsic dimension estimation are well-studied, curvature estimation remains underexplored. Existing methods (Cao et al., 2021; Sritharan et al., 2021; Coifman and Lafon, 2006; Jones, 2024) can work well in low dimensions but degrade under noise, high intrinsic dimension, or manifold heterogeneity. This leaves a gap between current methods and real-world data settings.

The low sample density in real-world datasets reinforces these challenges, especially in high-dimensional, heterogeneous regimes. For example, many methods estimate the second fundamental form (SFF), the most general notion of curvature capturing variation across tangent directions. But the SFF scales quadratically in intrinsic and ambient dimensions, requiring an impractically large number of samples. For example, a dataset with intrinsic dimension 100 and ambient dimension 1000 would need on the order of 10 million samples from a small neighborhood to estimate the SFF around a single point.[1]

1. This is not an unrealistic scenario; for instance, Kiani et al. (2024) estimate the manifold of MNIST images to have an intrinsic dimension on the order of 100. Each of the 32×32 grayscale image has on the order of 1000 latent dimensions. For larger, state of the art data sets, these issues are even more pronounced.

© 2025 J. Wang, B. Kiani & M. Weber.

To address these limitations, we propose augmenting the raw data set with samples from a trained diffusion model. We show that on heterogeneous manifolds, diffusion-augmented regression outperforms its classical counterpart. We also introduce a novel estimator that leverages the diffusion model more directly. By sampling along geodesic paths on the learned manifold, we increase local sample density and estimate curvature via the (reciprocal) radius of curvature along those paths. Our experiments show that while the other estimator's performance deteriorates rapidly in high dimensions and under noise, the interpolation-based estimator's performance is more stable.[2]

1.1. Related Work

Learning data geometry. Geometric data analysis seeks to characterize the properties of the manifold from which a dataset is sampled with *intrinsic dimension* and *curvature* as the key quantities. In contrast to curvature estimation, intrinsic dimension estimation is comparatively well-studied. The most widely used method is Principal Component Analysis (PCA) (Pearson, 1901); others include maximum likelihood estimation (Levina and Bickel, 2004) and nearest neighbor analysis (Facco et al., 2017). Manifold learning algorithms (e.g., Isomap (Tenenbaum, 2000), MDS (Kruskal and Wish, 1978)), also implicitly estimate intrinsic dimension. More recent approaches use diffusion models (Stanczuk et al., 2022; Kamkari et al., 2024) and normalizing flows (Horvat and Pfister, 2022).

Curvature estimation. Cao et al. (2021) estimate the SFF via quadratic regression of normal dimensions from the tangent space, an approach applied in computer vision (Sritharan et al., 2021) and single-cell analysis (Ma et al., 2025). Jones (2024) uses diffusion maps (Coifman and Lafon, 2006) to estimate the SFF. Related diffusion-based approaches have been studied in biomedical data analysis (Bhaskar et al., 2022; Zhang et al., 2025). Ricci curvature (rather than the full SFF) can provably be approximated pointwise via a discrete analog of classical Ricci curvature (Ollivier, 2009; Garcia Trillos and Weber, 2023).

Geodesic interpolation via diffusion models. Prior work has shown that diffusion models can interpolate along data manifolds. Wang and Golland (2023) study interpolation schemes for image diffusion, showing that direct linear (or spherical) interpolation in latent space fails to produce semantically meaningful images. They propose to noise the endpoints, interpolate between them, and then denoise. While effective for large-scale interpolation, this approach has limitations at the small scales on which we focus here.

2. Background and Notation

Central to characterizing data geometry is the *manifold hypothesis*, which assumes high-dimensional data lies on or near a low-dimensional manifold. We assume existence of a latent representation of our data on a Riemannian manifold with homogeneous intrinsic dimension. Though unlikely to hold exactly, these assumptions are common in practice.

2.1. Notions of curvature

We provide a brief introduction of curvature notions; more details are given in Apx. A. Smooth manifolds $\mathcal{M}$ are locally Euclidean spaces equipped with a differential structure.

2. Our code and experiment scripts are open-sourced and accessible at https://github.com/Weber-GeoML/Curvature_Estimation.

At a point $x \in \mathcal{M}$, the set of tangent vectors forms the tangent space $T_x\mathcal{M}$. Here we work with Riemannian manifolds, i.e., smooth manifolds with a smoothly varying inner product defined on the tangent space at each point. Given a curve, curvature at a point is the reciprocal of the radius of its osculating circle. When the curve is not perfectly circular, the acceleration vector is not perfectly orthogonal; we can measure acceleration against the orthogonal direction to define curvature. This motivates introducing an orthonormal basis that rotates with the curve, known as a *moving frame*.

Definition 1 (Moving Frame) *Let γ be a regular curve where the 1st through mth derivatives are linearly independent. The* moving frame *is a collection of maps $\{e_i\}_{i=1}^m$ where $e_i : I \subseteq \mathbb{R} \to \mathbb{R}^m$ is such that $e_1 = \gamma'$ and $\{e_i(t)\}_{i=1}^m$ is an orthonormal basis for all $t \in I$. This can be constructed via Gram-Schmidt orthogonalization of $\{\frac{d\gamma}{dt}, \ldots, \frac{d^m\gamma}{dt^m}\}$.*

Given a moving frame, we define curvature as the change in the velocity projected in the orthogonalized direction of the acceleration:

Definition 2 (Frenet-Serret Curvature) *The* Frenet-Serret (FS) curvatures *are a vector $\kappa \in \mathbb{R}^{m-1}$ given by $\chi_i(t) = \frac{\langle e_i'(t), e_{i+1}(t)\rangle}{||\gamma\ '(t)||}$. χ_1 is the reciprocal of the radius of curvature. χ_2 is commonly referred to as torsion.*

To illustrate this notion, compare two curves drawn on the manifold with orthogonal tangents. If both curves are straight, then the surface is flat. If both are curved in the same normal direction, then the surface has *positive curvature*. If they are curved in opposing normal direction, then the surface has *negative curvature* (see Fig. 11). Characterizing the curves for every direction, yields a characterization of the curvature of the manifold:

Definition 3 (Second Fundamental Form) *The SFF is the symmetric bilinear map $\mathrm{II}_p : T_p\mathcal{M} \times T_p\mathcal{M} \to N_p\mathcal{M}$ which produces $(\nabla_u v)^\perp$, the component of the covariant derivative*[3] *orthogonal to the tangent space: $\mathrm{II}_p(X, Y) = \sum_{ij} h_{ij}^k X^i Y^j n^k$. If we let $\phi : U \subseteq \mathbb{R}^d \to \mathbb{R}^{d+n}$ be the embedding of the manifold, then h is the $d \times d \times n$ tensor $h_{ij}^k = \langle \frac{\partial^2 \phi}{\partial x^i \partial x^j}, n^k \rangle$ for the ith and jth embedding coordinates x^i, x^j and kth normal vector n^k.*

With d^2n entries, the SFF is rich in information, but difficult to estimate with limited samples, high noise, high dimensionality and high heterogeneity. Instead, one can compute subsets of the SFF, which are more tractable. The first entry of the FS curvature at a point p is the osculating circle curvature χ_1, which relates to the geodesic and normal curvatures, that correspond to projecting the acceleration vector $\frac{d^2\gamma}{d^2t}$ onto the tangent and normal space.

Definition 4 (Normal Curvature) *The Normal Curvature κ_N of a curve $\gamma \subseteq \mathcal{M}$ is the norm of the SFF applied to the velocity of the curve, $\kappa_N = \|\mathrm{II}(\gamma', \gamma')\|$. An equivalent expression is the projection of the acceleration onto the normal space, $\kappa_N = \|\operatorname{proj}_{N_p\mathcal{M}} \frac{d^2\gamma}{dt^2}\|$. The osculating circle, geodesic, and normal curvature are then related as $\chi_1^2 = \kappa_G^2 + \kappa_N^2$.*

Another perspective concerns the behavior of parallel geodesics. For flat surfaces, parallel lines stay parallel; for positively-curved surfaces they converge, and for negatively-curved surfaces they diverge. To quantify this deviation, we track how parallel transport of a vector around a loop changes its orientation (*holonomy*, Fig. 9). Measuring the noncommutativity of transport in different directions yields the *Riemannian curvature tensor*.

3. The directional derivative on a manifold, with correction for the manifold's geometry given by the metric.

Definition 5 (Riemann Curvature) *The Riemann Curvature tensor is the multi-linear map* $R : T_p\mathcal{M} \times T_p\mathcal{M} \times T_p\mathcal{M} \to T_p\mathcal{M}$ *that satisfies the relation* $R(X,Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z$. *By the Gauss-Codazzi equations,* R *can be computed as* $R_{ijkl} = \sum_\alpha h^\alpha_{ik} h^\alpha_{jl} - h^\alpha_{il} h^\alpha_{jk}$. *The notation* $[X,Y]$ *refers to the Lie Bracket or the commutator* $[X,Y] : C^\infty(\mathcal{M}) \to C^\infty(\mathcal{M})$ *such that* $[X,Y](f) = X(Y(f)) - Y(X(f))$.

With d^4 entries, the Riemannian curvature tensor is still too large to compute in practice. The trace of the Riemannian curvature tensor is the Ricci curvature tensor, which distills only the volumetric growth associated with curvature in different directions. The Ricci curvature tensor can be further contracted to the scalar curvature.

Definition 6 (Ricci and Scalar Curvatures) *The Ricci Curvature is the symmetric bilinear form* $\mathrm{Ric} : T_p\mathcal{M} \times T_p\mathcal{M} \to T_p\mathcal{M}$ *where* $\mathrm{Ric}(X,Y) = \mathrm{tr}(Z \mapsto R(X,Y)Z)$ *or* $\mathrm{Ric}_{ij} = R_{ikjk}$. *The Scalar Curvature is the trace of the Ricci Curvature* $\mathrm{Scal} = \mathrm{tr}(\mathrm{Ric})$.

2.2. Existing approaches for Curvature estimation

There are two primary existing avenues for curvature estimation: QR and DM. In this section, we give a brief overview of both methods; additional details can be found in Apx. A.6. Here, N denotes the number of points, d the intrinsic dimension.

Quadratic regression (QR). Introduced by Cao et al. (2021) and Sritharan et al. (2021), QR computes the SFF by first performing a principal component analysis (PCA) in a locally linear neighborhood to obtain a basis for the tangent and normal spaces. It then computes the SFF's entries via QR over a locally quadratic neighborhood of the normal basis against the tangent basis. Formally, we solve the regression problem $\hat{H}_{:,k} = (X^T X)^{-1} X^T Y_{:,k}$, where X is formed from the first d dimensions of the principal component scores and Y is the latter of the total n dimensions (Sritharan et al., 2021), i.e., $X = \left[1, (x_i^1)^2, \ldots, (x_i^d)^2\right]_{i\in[N]} \in \mathbb{R}^{N\times(d^2+1)}$, $Y = \left[x_i^{d+1}, \ldots, (x_i^{d+m})^2\right]_{i\in[N]} \in \mathbb{R}^{N\times n}$. This can be solved by matrix inversion or optimization; in practice, matrix inversion performs best. The method scales as $O(Nd^2n)$ (Tab. 1), where the quadratic dependence on input dimension introduces a computational bottleneck for high-dimensional data.

Diffusion maps (DM). Coifman and Lafon (2006)[4] introduced *DM* as a nonlinear manifold learning technique. It leverages the idea that geodesic distances on the manifold can be approximated by random walks between nearby points; the diffusion map is characterized by the spectral decomposition of the transition matrix of that random walk. The first eigenvector corresponds to the stationary of eigenvalue 1, but the remaining eigenvectors describe the dominant directions of variation. It turns out this spectral description describes the Laplace-Beltrami operator on the underlying manifold. This information is powerful for curvature estimation as we can use the *Carre du Champ identity* to transform estimates of the Laplace-Beltrami operator into estimates for the Riemannian Curvature Tensor (Jones, 2024). We present a brief description of the method below, but defer a detailed technical description to Apx. A.6.2. Consider the heat kernel with density-dependent bandwidth

4. Despite the similar name, DM bear no relation to diffusion models, the generative models we use to sample from the data manifold.

ρ, i.e., $K(x,y) = \exp\left(\frac{\|x-y\|^2}{\epsilon\rho(x)\rho(y)}\right)$. An $N \times N$ transition matrix can be formed from row-normalizing $P_{ij} = \frac{K(x_i,x_j)}{\sum_a K(x_a,x_j)}$ or $P = D^{-1}K$, where D denotes the weighted degree matrix. Then, P^t_{ij} denotes the time-evolved transition probabilities of the random walk starting at i and ending at j after t steps. t defines a "length scale": For small t the random walk captures only local geometry, but for larger t it traverses larger neighborhoods, capturing more global geometry. Since P is symmetric, a spectral decomposition with non-negative eigenvalues is given by $\tilde{P} = \Phi\Lambda\Phi^{-1}$, where $\tilde{P} = D^{1/2}PD^{-1/2}$. The diffusion map is then given by $\Psi_t(x) = \begin{bmatrix}\lambda_2^t\phi_2(x) & \dots & \lambda_n^t\phi_n(x)\end{bmatrix}$ and truncated for dimensionality reduction.

2.3. Diffusion Models

Diffusion models are trained to reverse Gaussian noise applied to points on a data manifold over multiple time steps. New samples are generated by denoising from white noise, with small step sizes ensuring tractability. The strengths of this approach include stable training due to a simple objective (unlike, e.g., GANs) and controllability during data generation, albeit at higher computational cost. The *Denoising Diffusion Probabilistic Model (DDPM)* (Ho et al., 2020) forms the basis of the diffusion models utilized in our framework. Its forward process is a Markov chain of T noised samples $\{x_1, \dots, x_T\}$ given by the likelihood $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$; $\beta_1, \dots, \beta_T$ is a variance schedule controlling the amount of noise added at each step. The reverse process is the procedure of recovering x_0 from x_t. Given a model θ, the reverse process estimates the likelihood $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t))$. DDPM differs from general diffusion models by using a fixed variance schedule, a fixed non-learnable variance estimator Σ_θ, and predicting added noise instead of the reverse likelihood. For more details see Apx. A.5.

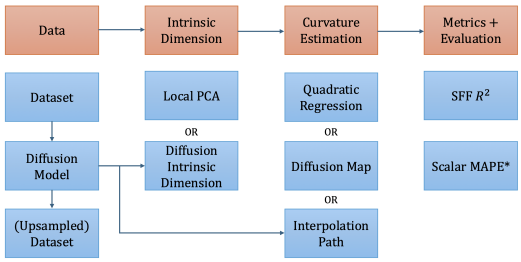

Figure 1: Overview of workflow.

3. Methods

3.1. Diffusion-based Curvature Estimation

Below we describe our diffusion-augmented curvature estimation approach (see Fig. 1):

Step 1: Diffusion Model. We train a diffusion model to learn the data manifold, which enables (i) sampling point neighborhoods at higher density, (ii) intrinsic dimension estimation via SVD of score vectors, (iii) sampling along geodesics.

Step 2: Intrinsic Dimension Estimation. We use the diffusion model to generate a dense sample in a locally linear neighborhood and apply PCA to obtain the tangent basis. The intrinsic dimension is chosen as the number of principal components carrying significant signal: For data $X = \left[x_1^{(i)}, \dots, x_N^{(i)}\right]_{i\in[m]} \in \mathbb{R}^{N\times m}$, where superscripts denote coordinates and subscripts data points, we compute eigenvectors/eigenvalues of the normalized covariance matrix $\frac{(X^T-\mu)(X-\mu)}{N}$ with mean $\mu = \left[\sum_{i=1}^N x_i^{(1)}, \dots, \sum_{i=1}^N x_i^{(m)}\right]^T$. The spectrum σ is sorted and thresholded using the scree plot's rate of decay. We systematically set the

threshold via the point of maximum curvature, using the Kneedle algorithm, which fits a spline f to sorted components σ_i and computes $\arg\max_i \frac{f''(x)}{(1+f'(x))^{3/2}}$ (Satopaa et al., 2011).

Step 3: Curvature Estimation. QR estimate the full SFF, DM the Riemannian curvature tensor, both of which we reduce to scalar curvature. DM, but not QR, requires access to the intrinsic dimension. Our proposed geodesic interpolation approach leverages a trained diffusion model to estimate normal curvature.

Step 4: Curvature Reduction and Evaluation. Since different methods estimate different curvature notions, a careful evaluation protocol is required. For QR (classical and diffusion-augmented) and DM, we compare scalar curvature, as their outputs can be contracted to this quantity. Geodesic interpolation, however, estimates normal curvature, which cannot be contracted to scalar curvature. On synthetic manifolds with known ground-truth curvature, we compute mean absolute percentage error (MAPE). We use MAPE since curvature magnitudes can vary widely, and relative error ensures equal weighting across points. For the roll dataset, where curvature is zero in one dimension, we instead use mean absolute error (MAE) to avoid division by zero.

3.2. Quadratic Regression with Adaptive Neighborhood Size

A key challenge for QR approaches is sensitivity to neighborhood size: one radius defines local linearity (for intrinsic dimension, tangent space estimation), another local quadraticity (for curvature). These radii are typically treated as hyperparameters, but without prior knowledge of the data manifold, tuning is difficult. Naively, one could brute force search several neighborhood sizes and take the best fit. Still, this requires assumptions about the scale of the neighborhood sizes. Sritharan et al. (2021) employs a log-spaced grid search based on covariance of the regression coefficients, but this remains computationally expensive. Since each neighborhood contains all smaller ones, we optimize brute-force search to match the cost of a single regression at the largest neighborhood, reducing complexity by a factor of N. This is achieved using streaming algorithms, specifically Incremental PCA (Ross et al., 2008) and Recursive Least Squares (Plackett, 1950). This procedure is particularly valuable for heterogeneous manifolds or in high-noise settings, where curvature varies significantly across the manifold, as evident from the results on the Stanford bunny below (Sec. 4). For homogeneous toy manifolds, parameter-free results are given in Apx. C.5.

3.3. Curvature Estimation via Geodesic Interpolation

Both QR and DM estimate the full SFF, which neccesitates a rapidly increasing sample size as dimension grows (see experimental results in the next section). Instead, suppose we only want to characterize curvature in one direction. We propose a novel approach, *geodesic interpolation*, for this task, which estimates curvature from data sampled along a geodesic. This requires access to a parameterized curve on the manifold in the chosen direction, which we obtain via sampling from a diffusion model trained on the raw data. In a high-fidelity model, the learned score vectors can be seen as an approximation to the exponential map, which defines a map from the tangent space back to the manifold. To construct geodesics, we (i) linearly interpolate tangent vectors at the evaluation point, and (ii) denoise (without variance). We then compute curvature via the FS formulas, using finite differences to approximate derivatives. For sufficiently small neighborhoods, the geodesic

curvature is 0, so the FS curvature equals the normal curvature. More formally, suppose we are given a diffusion model θ, evaluation point x, tangent vector v, length L, interpolation resolution N, and number of denoising timesteps T. We outline our approach:

1. Define the linear interpolation z_T consisting of N points where $z_T^i = x + (\frac{2i}{N} - 1)v$ for $i = 1, \ldots, N$, letting subscript denote the current diffusion timestep and the superscript parameterizing the interpolation curve. Since this curve may not lie on the manifold, we use the diffusion model to iteratively construct the final discrete approximation $z_0^0, \ldots, z_0^N$ of a geodesic path γ on the manifold: $z_{t-1}^i = \frac{1}{\alpha_t}(z_t^i - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(z_t^i, t))$.
2. Compute the derivatives via finite differences: $\frac{d\gamma(i)}{dt} \approx \frac{z_0^{i+1} - z_0^{i-1}}{2}$ and $\frac{d^2\gamma}{dt^2} \approx \frac{z_0^{i+1} - 2z_0^i + z_0^{i-1}}{4}$.
3. Compute the orthogonal bases $e_1(i), e_2(i)$ via Gram-Schmidt process on the derivatives.
4. The osculating circle curvature is then computed as $\chi_1 \approx \frac{\langle \frac{e_1(i+1) - e_1(i-1)}{2}, e_2(i) \rangle}{\| \frac{d\gamma(i)}{dt} \|}$.

We average over the middle half of the interpolation path to avoid edge effects. Unless stated otherwise, the tangent direction v is the first principal direction, i.e., the eigenvector of maximum curvature; without ground-truth curvature, we use the first principal component from local PCA. The method is far cheaper than others: aside from diffusion model training and inference, its cost is $O(Nm)$ (ignoring large model constants), see Tab. 1.

4. Experiments

We evaluate the classical and diffusion-augmented curvature estimators on several data sets: Three canonical 2D manifolds embedded in $\mathbb{R}^3$, high-dimensional paraboloids generated by the SFF, and a real-world data set, the Stanford bunny (Turk and Levoy, 1994), a popular computer graphics benchmark (more details in Apx. C.1). Note that in interpreting the results, it is important to consider the magnitude of error that is acceptable for the downstream task. If the goal is to simply obtain the correct sign of curvature, then even large MAPEs in the range 80-100% may still be acceptable.

Method	Complexity
Quad. Regression	$O(Nd^2n)$
Diffusion Maps	$O(N^3)$
Geod. Interpolation	$O(N(d+n))$

Table 1: Time complexities.

Experiment 1: Noise vs. sample density. We investigate the performance of all four methods with respect to number of samples ($N \in \{250, 500, 1000, 2500, 5000\}$) and noise level ($s \in \{0, 0.005, 0.01, 0.05, 0.1\}$, distributed $\epsilon \overset{iid}{\sim} \mathcal{N}(0, \frac{s}{\sqrt{m}})$). We expect performance to increase with the number of samples and to decrease with the amount of noise. In the first column of Fig. 2, we examine the roll dataset, which has ground truth of zero scalar curvature and zero normal curvature in the direction of the width. Evidently, curvature estimation on flat manifolds is much easier than on curved ones, as all four methods ace the evaluation. In the second column, we examine the results for the sphere. We observe that with a large number of samples and little noise, QR on the raw point cloud results in the best performance, achieving just 5.5% MAPE. Notably, diffusion-based sampling approaches don't suffer from the same rate of performance decay as noise increases, making regression with diffusion samples and geodesic interpolation desirable choices for high noise manifolds. In the third column, we study the curvature at the saddle point of a 3D

hyperbolic paraboloid. Here, the entire manifold is exactly quadratic, so QR does well as expected. In this case, the original samples are the highest possible quality for QR, so naturally diffusion-based augmentation can only do worse. However, when $N = 1000$, diffusion-based QR outperforms classic QR.

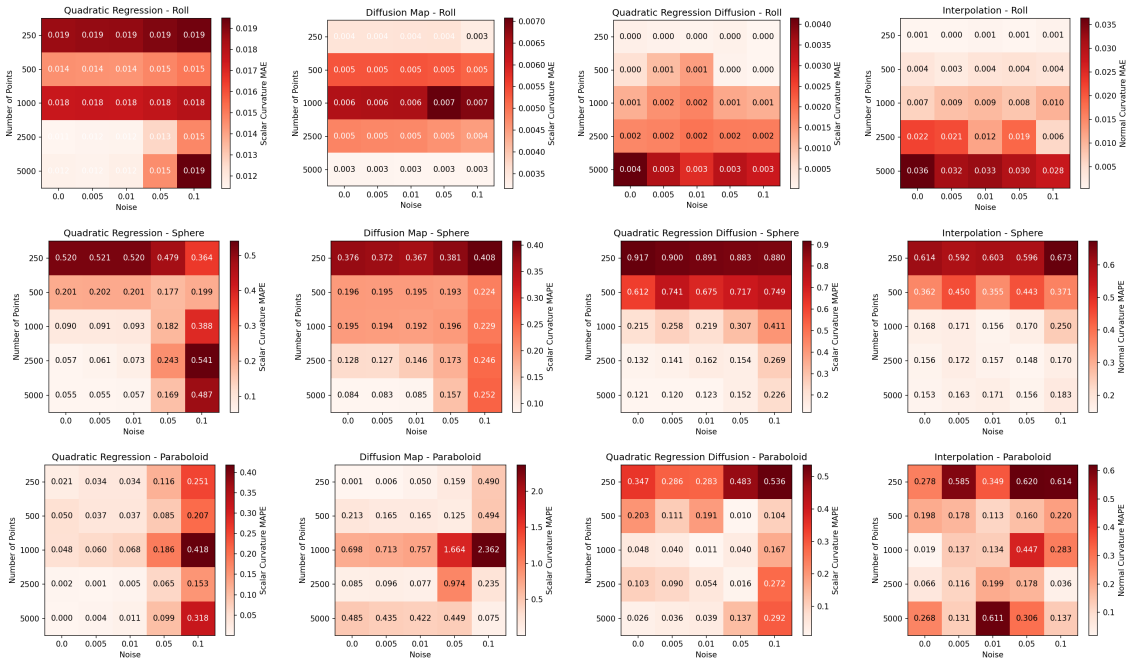

Figure 2: Results for varying number of samples and noise level across methods and datasets.

Experiment 2: Intrinsic vs. Normal Dimension. The next research question concerns the ability of these curvature estimation techniques to extend to higher dimension manifolds. We consider a high-dimensional paraboloid with eigenvalues distributed according to the Wigner semicircular distribution discussion in Apx. C.1. We vary intrinsic dimension and normal dimension in the range $\{2, 4, 8, 16, 32\}$; we sample 5000 points from the true manifold with 0 noise. This is technically a sufficient number of points for QR on a manifold with $d = 16, n = 16$ (as $d^2 n = 4096 < 5000$) but no greater. Our experimental results illustrate this threshold: Observe that only QR on the raw point cloud estimates curvature reasonably, with 38% MAPE at $d = 16, n = 16$, but accuracy deteriorates quickly beyond. DM cannot scale to higher dimensions, and diffusion-based regression performs worse on paraboloids than direct regression on samples, as in the 3D case, which we believe to be due to the higher noise in the synthetic samples. In contrast, our geodesic interpolation method, though not better than baseline in 3D, is highly effective in high dimensions, provided the normal-to-intrinsic dimension ratio is not too large. This supports the idea of leveraging diffusion to obtain a sufficiently high density in a single direction for detecting normal curvature, rather than estimating the full SFF.

Experiment 3: Real-World Data Manifold (Stanford Bunny). Our final experiment investigates performance on heterogeneous, real-world data (Fig. 4). Qualitatively, DM give the smoothest and most visually accurate curvature estimates, likely because they estimate

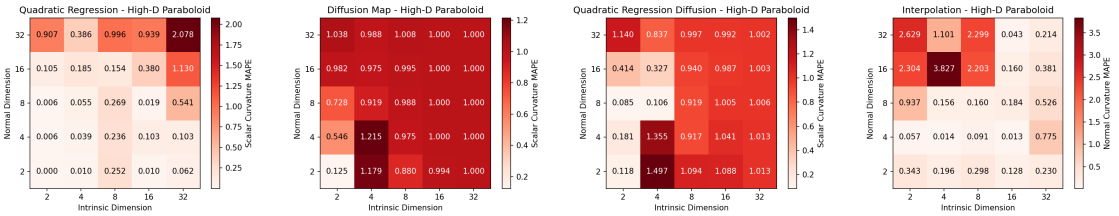

Figure 3: Results for intrinsic vs. normal dimension on the Paraboloid.

at all samples simultaneously and share information. QR, by contrast, is sample-by-sample. Both our modifications, diffusion-based sampling and parameter-free neighborhood selection, are required for QR to approach DM's performance. Although the dataset contains many samples, local density is low due to the manifold's heterogeneity. This impacts the fidelity of the diffusion model, resulting in noisy synthetic samples that impact the performance of the geodesic interpolation estimator. Further analysis of the learned manifold can be found in Apx. C.4.

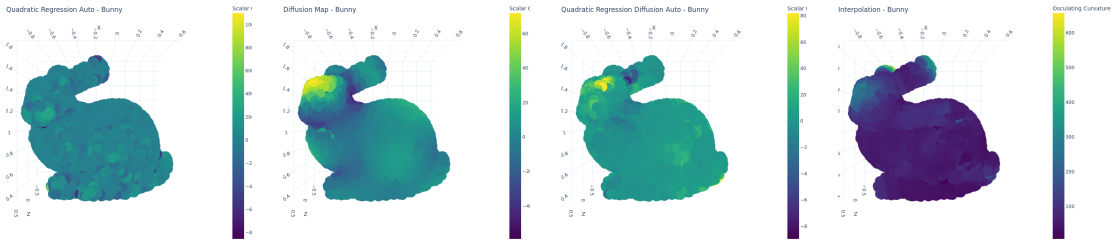

Figure 4: Comparison of classical and diffusion-augmented methods for the Stanford Bunny. From left to right: QR, DM, Diffusion-Augmented Sampling, Geodesic Interpolation.

5. Discussion

We proposed a diffusion-augmented framework for curvature estimation. Probing a trained diffusion model yields denser samples, albeit at the cost of added noise; our experimental results show that this trade-off is beneficial for regression-based curvature estimators on heterogeneous manifolds. The diffusion-based framework further enables curvature estimation via geodesic interpolation, which exhibits more stable performance as dimension increases.

We observe that performance depends on model fidelity: Augmentation with low-fidelity diffusion models can hurt performance, especially on simple manifolds (e.g., the sphere), where classical methods outperform diffusion-based ones on sparsely sampled raw data. On heterogeneous manifolds (e.g., the bunny), the additional synthetic samples improve accuracy, with diffusion-augmented QR outperforming the classical variant. Remaining challenges include sensitivity to hyperparameters and high computational cost, with DM being especially difficult to scale to high dimensions. Future directions include optimization and parallelization, as well as extensions of the current framework to newer diffusion models (possibly with higher fidelity) and to other data modalities.

Although high-dimensional curvature estimation remain challenging, the proposed framework already yields good results in low-dimensions, a setting which is important in areas such as shape analysis and geometry processing.

Acknowledgments

BTK and MW were supported by the Harvard Data Science Initiative Competitive Research Fund and NSF award DMS-2406905. MW was supported by an Alfred P. Sloan Fellowship in Mathematics.

References

Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. In *ACM SIGGRAPH 2003 Papers*, pages 485–493. 2003.

Tyrus Berry and John Harlim. Variable bandwidth diffusion kernels. *Applied and Computational Harmonic Analysis*, 40(1):68–96, 2016.

Dhananjay Bhaskar, Kincaid MacDonald, Oluwadamilola Fasina, Dawson Thomas, Bastian Rieck, Ian Adelstein, and Smita Krishnaswamy. Diffusion curvature for estimating local curvature in high dimensional data. *Advances in Neural Information Processing Systems*, 35:21738–21749, 2022.

Nithya Bhasker, Hattie Chung, Louis Boucherie, Vladislav Kim, Stefanie Speidel, and Melanie Weber. Contrastive poincaré maps for single-cell data analysis. In *ICLR Workshop on Machine Learning for Genomics Explorations*, 2024.

Yueqi Cao, Didong Li, Huafei Sun, Amir H Assadi, and Shiqiang Zhang. Efficient weingarten map and curvature estimation on manifolds. *Machine Learning*, 110(6):1319–1344, 2021.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. In *ACL*, 2020.

Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. In *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*, pages 63–70, 2007.

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1): 12140, 2017.

Jean Feydy. *Analyse de données géométriques, au delà des convolutions.* PhD thesis, Université Paris-Saclay, 2020.

Nicolas Garcia Trillos and Melanie Weber. Continuum limits of ollivier's ricci curvature on data clouds: pointwise consistency and global lower bounds. *arXiv preprint arXiv:2307.02378*, 2023.

Martin Hanik, Benjamin Ducke, Hans-Christian Hege, Friederike Fless, and Christoph Von Tycowicz. Intrinsic shape analysis in archaeology: a case study on ancient sundials. *ACM Journal on Computing and Cultural Heritage*, 16(4):1–26, 2023.

Wenchong He, Zhe Jiang, Chengming Zhang, and Arpan Man Sainju. Curvanet: Geometric deep learning based on directional curvature for 3d shape analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2214–2224, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Susung Hong. Smoothed energy guidance: Guiding diffusion models with reduced energy curvature of attention. *Advances in Neural Information Processing Systems*, 37:66743–66772, 2025.

Christian Horvat and Jean-Pascal Pfister. Estimating the intrinsic dimensionality using normalizing flows. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 12225–12236, 2022.

Iolo Jones. Manifold diffusion geometry: Curvature, tangent spaces, and dimension. *arXiv preprint arXiv:2411.04100*, 2024.

Hamidreza Kamkari, Brendan Leigh Ross, Rasa Hosseinzadeh, Jesse C Cresswell, and Gabriel Loaiza-Ganem. A geometric view of data complexity: Efficient local intrinsic dimension estimation with diffusion models. *arXiv preprint arXiv:2406.03537*, 2024.

Brian Keng. Tensors, tensors, tensors. https://bjlkeng.io/posts/tensors-tensors-tensors/, 2018.

Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6418–6428, 2020.

Bobak Kiani, Jason Wang, and Melanie Weber. Hardness of learning neural networks under the manifold hypothesis. *Advances in Neural Information Processing Systems*, 37:5661–5696, 2024.

Anna Klimovskaia, David Lopez-Paz, Léon Bottou, and Maximilian Nickel. Poincaré maps for analyzing complex hierarchies in single-cell data. *Nature communications*, 11(1):2966, 2020.

Joseph B. Kruskal and Myron Wish. *Multidimensional Scaling*. SAGE, 1978. ISBN 978-0-8039-0940-3. Google-Books-ID: ZzmIPcEXPf0C.

John Lee. *Introduction to Riemannian Manifolds*. Springer International Publishing AG, 2018.

Elizaveta Levina and Peter Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17, 2004.

Marvin Li, Jason Wang, Jeffrey Wang, and Seth Neel. Mope: Model perturbation-based privacy attacks on language models. *arXiv preprint arXiv:2310.14369*, 2023.

Tianlin Liu. A detailed derivation of the diffusion map, 2021. URL https://tianlinliu.com/2021/05/29/difussion-maps/.

Yanbiao Ma, Licheng Jiao, Fang Liu, Maoji Wen, Lingling Li, Wenping Ma, Shuyuan Yang, Xu Liu, and Puhua Chen. Predicting and enhancing the fairness of dnns with the curvature of perceptual manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Puskar Mondal. Math 136: Introduction to differential geometry, 2023.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9078–9086, 2019.

Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572, 1901.

Ronald L Plackett. Some theorems in least squares. *Biometrika*, 37(1/2):149–157, 1950.

David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1):125–141, 2008.

Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

Justin Solomon. Topic 5: Curvature, lecture notes in mit 6.838: Shape analysis, Spring 2021.

Duluxan Sritharan, Shu Wang, and Sahand Hormoz. Computing the riemannian curvature of image patch and single-cell rna sequencing data manifolds using extrinsic differential geometry. *Proceedings of the National Academy of Sciences*, 118(29):e2100473118, 2021.

Jan Stanczuk, Georgios Batzolis, Teo Deveney, and Carola-Bibiane Schönlieb. Your diffusion model secretly knows the dimension of the data manifold. *arXiv preprint arXiv:2212.12611*, 2022.

Yinhang Tang, Huibin Li, Xiang Sun, Jean-Marie Morvan, and Liming Chen. Principal curvature measures estimation and application to 3d face recognition. *Journal of Mathematical Imaging and Vision*, 59:211–233, 2017.

J. B. Tenenbaum. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, December 2000. ISSN 00368075, 10959203. doi: 10.1126/science.290.5500.2319. URL http://www.sciencemag.org/cgi/doi/10.1126/science.290.5500.2319.

Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, 1994.

Clinton Wang and Polina Golland. Interpolating between images with diffusion models. *arXiv preprint arXiv:2307.12560*, 2023.

Yuanzhen Wang, Beibei Liu, and Yiying Tong. Linear surface reconstruction from discrete fundamental forms on triangle meshes. In *Computer Graphics Forum*, volume 31, pages 2277–2287. Wiley Online Library, 2012.

Yanlei Zhang, Lydia Mezrag, Xingzhi Sun, Charles Xu, Kincaid Macdonald, Dhananjay Bhaskar, Smita Krishnaswamy, Guy Wolf, and Bastian Rieck. Principal curvatures estimation with applications to single cell data. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

Appendix A. Extended Background

We provide a more comprehensive overview of the differential geometry background of this work. We begin with preliminary notations, mostly following from Puskar Mondal's *Introduction to Differential Geometry* (Mondal, 2023) and John Lee's *Introduction to Riemannian Manifolds* (Lee, 2018).

A.1. Notation

In this section, we introduce commonly used notations and conventions that will reoccur throughout. We proceed to define these notions in the following sections.

Notation	Definition
γ	Regular curve
e_i	Basis vector
χ	Frenet-Serret Curvature
$\mathcal{M}$	Riemannian manifold
d	Intrinsic dimension
n	Normal dimension
m	Ambient dimension
$T_x\mathcal{M}$	Tangent space of $\mathcal{M}$ at x
$N_x\mathcal{M}$	Normal space of $\mathcal{M}$ at x
$\frac{\partial}{\partial x^i}$	Tangent basis vector
∂x^i	Cotangent basis vector
${X_i}^j$	i is a covariant index, j is a contravariant index
g	Riemannian metric
∇	Covariant derivative
$[X, Y]$	Lie bracket
II	Second fundamental form
S	Shape operator
κ	Principal curvatures
κ_N	Normal curvature
κ_G	Geodesic curvature
R	Riemannian curvature tensor
Ric	Ricci curvature tensor
Ricci	Ricci curvature
Sec	Sectional curvature
Scal	Scalar curvature

Table 2: Mathematical Notation

An overview of the curvature notions we introduce:

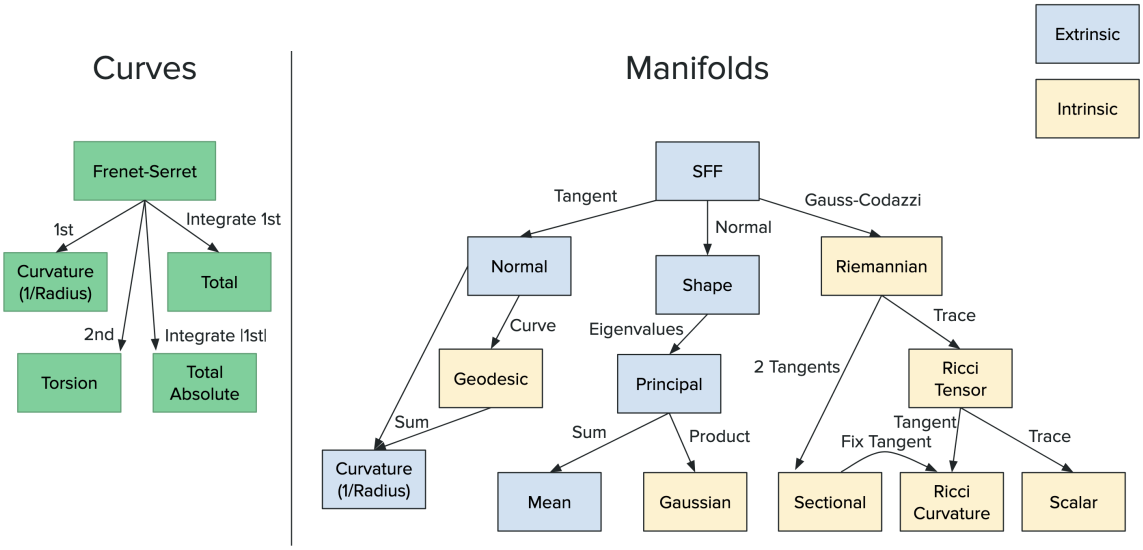

Figure 5: Summary chart of relationships between curvature notions.

A.2. Curves

What is curvature? Informally, we understand curvature as how much a curve deviates from being a straight line. If you keep bending the curve slightly, it will come around and form a circle of large radius. If you bend the curve more significantly, it will form a circle of smaller radius. Thus, curvature could be understood as the reciprocal of the radius of the osculating circle—the circle that best approximates the curve locally.

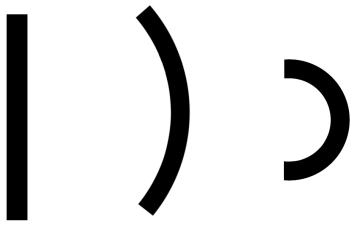

Figure 6: Increasing Curvature from Left to Right

Another interpretation, borrowing from physics, is the idea of a curvature as the force that pulls the curve in away from a straight line (centripetal acceleration), which corresponds proportionally to the inverse radius of curvature. For these quantities to be defined, the curves need to be "well-behaving"; in particular, characterized as having continuous velocity, acceleration, etc.

Definition 7 (Regular Curve) *A smooth parametrized curve is an infinitely differentiable map $\gamma : I \subseteq \mathbb{R} \to \mathbb{R}^m$. The velocity of the curve is notated as $\frac{d\gamma}{dt} = \gamma'$. A curve γ is regular if it is smooth parameterized and $\gamma'(t) \neq 0 \ \forall t \in I$. We will use the arc-length parameterization of γ, which defines $\gamma_{new}(t) = \int_0^t |\gamma'(t)| dt$*[5].

A curve may not be perfectly circular, in which case the acceleration will not point directly orthogonally to the curve, which is the direction we want to measure the acceleration against to obtain a radius of curvature. This motivates an orthonormal basis to quantify the curvature in each dimension. For non-straight curves, this basis will need to rotate with the curve. We call this a moving frame.

We begin with an illustration of the geometric characterization that curvature provides, before stating formal definitions. Given a curve, the curvature at a point is defined as the reciprocal of the radius of the osculating circle at that point. A curve may not be perfectly circular; in such cases, the acceleration vector does not point exactly orthogonally to the curve. However, it is this orthogonal direction against which we wish to measure the acceleration in order to determine the radius of curvature. This consideration motivates the introduction of an orthonormal basis to quantify curvature along each dimension. For non-linear curves, such a basis must rotate in accordance with the curve. This rotating basis is referred to as a moving frame. To be more formal, we first define the notion of a *moving frame*:

Definition 8 (Moving Frame (Frenet-Serret)) *Let γ be a regular curve where the $1st$ through mth derivatives are linearly independent. The Frenet-Serret Frame is a collection of maps $\{e_i\}_{i=1}^m$ where $e_i : I \subseteq \mathbb{R} \to \mathbb{R}^m$ is such that $e_1 = \gamma\,'$ and $\{e_i(t)\}_{i=1}^m$ is an orthonormal basis for all $t \in I$. This can be constructed by performing Gram-Schmidt orthogonalization of $\{\frac{d\gamma}{dt}, \ldots, \frac{d^m\gamma}{dt^m}\}$.*

Given this moving frame, we can now define curvature in the vein that we discussed above as the change in the velocity projected in the orthogonalized direction of the acceleration, and so forth.

Definition 9 (Frenet-Serret Curvature) *The Frenet-Serret curvatures are a vector $\kappa \in \mathbb{R}^{m-1}$ given by $\chi_i(t) = \frac{\langle e_i'(t), e_{i+1}(t)\rangle}{||\gamma\,'(t)||}$. χ_1 is the reciprocal of the radius of curvature. χ_2 is commonly referred to as torsion.*

As a concrete example, consider the 3D helix: $\gamma(t) = (r\cos t, r\sin t, c\,t)$. Then, $\gamma\,'(t) = (-r\sin t, r\cos t, c)$ and the Frenet-Serret Frame is given by $e_1(t) \propto (-r\sin t, r\cos t, c)$, $e_2(t) \propto (-r\cos t, -r\sin t, 0)$, and $e_3(t) \propto (c\sin t, -c\cos t, r)$. The Frenet-Serret curvatures are thus $\chi_1(t) = \frac{r}{r^2+c^2}, \chi_2(t) = \frac{c}{r^2+c^2}$.

5. This assumes a metric of the ambient space which for now is the typical Euclidean norm; we discuss the details of such a metric in the next section.

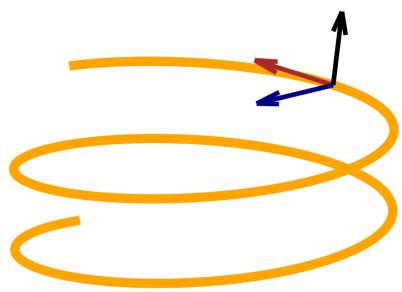

Figure 7: Frenet-Serret Frame on a Helix

A.3. Manifolds

Curvature of the space of the manifold represents an amalgamation of curvatures of different curves lying on the manifold. Since there are many ways to draw curves on the manifold and many ways to combine the resulting curvatures, the number of curvature notions for manifolds is much greater.

In the previous section, we discuss curves mapping to $\mathbb{R}^m$. This can be generalized to curves mapping onto any manifold. A manifold is a very general concept that can be thought of as any space that looks locally Euclidean. One example of a manifold is the 2-sphere in $\mathbb{R}^3$ which e.g., may represent locations on the surface of the Earth.

Definition 10 (Manifold) *A manifold is a topological space $\mathcal{M}$ such that $\forall x \in \mathcal{M}$, there exists a homeomorphism from a neighborhood of x to d-dimensional Euclidean space.*

If we want to compute curvature on a manifold, there may be many possible curves that you could draw on the manifold, each going in a different direction and having potentially different curvatures. We need a coordinate system for an organized way to refer to these curves. The tangent space provides this.

Definition 11 (Tangent Space) *The tangent space to a manifold $\mathcal{M}$ at point x, denoted $T_x\mathcal{M}$, is the set of all tangent vectors at x, where a tangent vector v is an equivalence class of curves γ with the same tangent $v = \frac{d}{dt}(\varphi \circ \gamma)|_{t=0}$.*

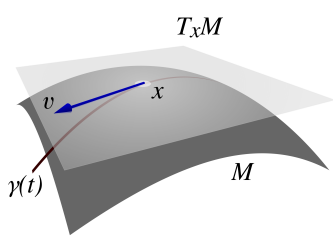

Figure 8: Tangent Space

The number of orthogonal directions that a curve on a manifold could possibly take tells us the degrees of freedom available in a representation of the local space. We refer

to this as the intrinsic dimension. If a manifold exists in a higher dimensional space, the extra dimensions available that lead away from the manifold are the normal dimensions, characterized by the set of vectors orthogonal to the tangent space.

Definition 12 (Intrinsic Dimension, Normal Dimension) *The intrinsic dimension d of a manifold $\mathcal{M}$ is the dimension of $T_x\mathcal{M}$. The normal dimension n of a manifold $\mathcal{M}$ is the dimension of the normal space $N_x\mathcal{M}$, the set of vectors orthogonal to all of $T_x\mathcal{M}$.*

Finally, to concretely measure curvature, we need a metric. A metric is a function that gives us an inner product $\langle \cdot, \cdot \rangle$, which is essential for specifying geometric notions. Specifically, an inner product induces a norm $|v| = \sqrt{\langle v, v \rangle}$ and an angle $\theta = \arccos \frac{\langle u,v \rangle}{|u||v|}$. Manifolds equipped with a metric that induces non-degenerate, symmetric norms and angles are called Riemannian manifolds.

Definition 13 (Riemannian Metric) *A metric g is a $(0,2)$-tensor*[6] *field that assigns for each point x an inner product function $\langle \cdot, \cdot \rangle : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}_{\geq 0}$, verifying symmetry $g(x,y) = g(y,x)$ and non-degeneracy $g(x,\cdot) = 0 \implies x = 0$. A Riemannian manifold is a smooth manifold equipped with a Riemannian metric.*

The general form of an inner product is $\langle u, v \rangle_g = \sum_{i,j} g_{ij} u^i v^j$. The Riemannian metric that we use, the Euclidean inner product, is $\langle u, v \rangle = \sum_i u_i v_i$, which correspond to $g_{ij} = \mathbf{1}(i = j)$ or the identity matrix. This important observation allows us to handily elide the metric in most of our cases.

A.4. Curvature on Manifolds

Before delving into the mathematical formalism of curvature on manifolds, we first lay the intuition for what curvature entails on easily-visualizable 2-dimensional surfaces before generalizing to higher dimensions.

We start by comparing just two curves drawn on the manifold with orthogonal tangents. If both curves are straight, then the surface is flat. If one is flat and one is curved, then the surface is cylindrical, showing curvature in one direction. If both are curved in the same normal direction, then the surface is spherical and has *positive curvature*. If they are curved in opposing normal direction, then the surface is hyperbolic and has *negative curvature*. Informally, if we know how curves on the manifold look like for every direction on this surface (i.e., the Hessian), then we know the curvature of the overall manifold. This is called the *second fundamental form*. This kind of curvature analysis is known as the *extrinsic curvature*, as it uses the curvature of curves on the manifold relative to the ambient space.

Another perspective on curvature of manifolds concerns the behavior of parallel geodesics. For flat and cylindrical surfaces, parallel lines stay parallel. For spherical surfaces, parallel lines converge, and for hyperbolic surfaces, parallel lines diverge. This means that triangles will appear normal for flat and cylindrical surfaces, bulged out for spherical, and shrunken in for hyperbolic. To "measure" the degree of this deviation, a first instinct might be to try computing how many extra degrees there are in the triangle. More generally, we can

6. For a good introduction to tensors, see Brian Keng's *Tensors, Tensors, Tensors* (Keng, 2018).

observe holonomy, which is how moving a vector around a loop while staying parallel to the geodesic movement actually changes the orientation of the vector (see Figure 9). Informally, if we measure this noncommutativity of transporting in one direction vs. another, we get the *Riemannian curvature tensor*. This kind of curvature analysis is known as the *intrinsic curvature*, as it does not rely on the embedding of the manifold in its ambient space.

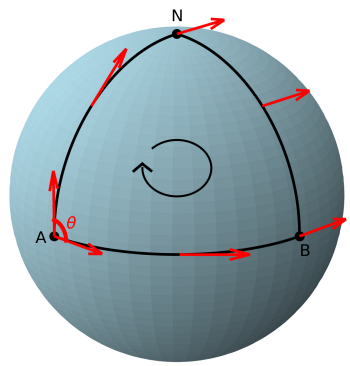

Figure 9: Holonomy on a Sphere

These two perspectives of extrinsic vs. intrinsic curvature, though measuring different quantities, can be reconciled (intrinsic can be computed from extrinsic). Now, we present both extrinsic and intrinsic curvature defintitions formally, and in full generality with respect to dimension.

First, the second fundamental form (SFF) is the most complete notion of curvature. It captures the curvature from curves in all possible tangent directions, for all normal directions. Quantitatively, it looks like the Hessian matrix of the best quadratic approximation to the manifold.

Definition 14 (Second Fundamental Form) *The Second Fundamental Form is the symmetric bilinear map* $\mathrm{II}_p : T_p\mathcal{M} \times T_p\mathcal{M} \to N_p\mathcal{M}$ *which produces* $(\nabla_u v)^\perp$*, the component of the covariant derivative*[7] *orthogonal to the tangent space. We write* $\mathrm{II}_p(X, Y) = \sum_{ij} h^k_{ij} X^i Y^j n^k$*. In particular, if we let* $\phi : U \subseteq \mathbb{R}^d \to \mathbb{R}^{d+n}$ *be the embedding of the manifold, then* h *is the* $d \times d \times n$ *tensor* $h^k_{ij} = \langle \frac{\partial^2 \phi}{\partial x^i \partial x^j}, n^k \rangle$ *for the* i *and* j*th embedding coordinates* x^i, x^j *and* k*th normal vector* n^k*.*

The SFF contains an enormous amount of information, having $d^2 n$ entries. Thus, estimating this quantity from data can be difficult due to insufficient sample density, noise, high dimensionality, and computational constraints. Instead, it can be more feasible to look at different contractions and subsets of the SFF, which are also widely studied.

For instance, focusing on a single normal dimension gives us the shape operator.

Definition 15 (Shape Operator) *The Shape Operator* $S^n_p : T_p\mathcal{M} \to T_p\mathcal{M}$ *is defined as by* $S^n_p(X) = \sum_k h^k_{ij} n^k$*. As a consequence, it satisfies the relation* $\langle S^n_p(X), Y \rangle = \langle \mathrm{II}_p(X, Y), n \rangle$*, which means that* $\mathrm{II}_p(X, Y) = \langle S^k_p(X), Y \rangle n$*.*

7. The directional derivative on a manifold, with correction for the manifold's geometry given by the metric.

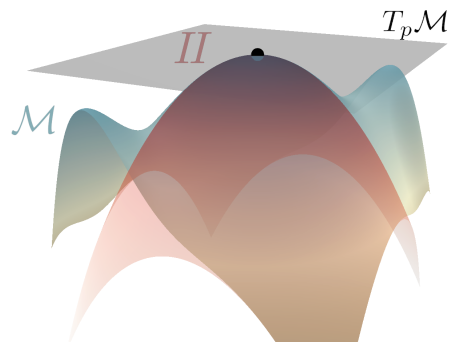

Figure 10: Second Fundamental Form

Still, the shape operator has d^2 coefficients. Going further, we can look at just the eigenvalues of this matrix, which tell us the curvatures along the main directions inferred from the Hessian (which will be orthogonal by spectral theorem on symmetric matrices). These eigenvalues are called principal curvatures. Two widely used contractions of the principal curvatures are the Gaussian curvature (the product of eigenvalues) and the Mean Curvature (the average of eigenvalues). The Gaussian curvature, although derived extrinsically, is actually an intrinsic notion. The mean curvature is useful in identifying minimal surfaces and in mean curvature flow.

Definition 16 (Principal, Gaussian, and Mean Curvatures) *The Principal Curvatures $\kappa_1, \ldots, \kappa_d$ for a given normal vector n are the eigenvalues of the shape operator S_p^n. The Gaussian Curvature is the determinant of S_p^n (i.e., $\prod_i \kappa_i$) and the Mean Curvature is the trace of S_p^n (i.e., $\sum_i \kappa_i$).*

On the other hand, from the intrinsic curvature perspective, the Riemannian curvature tensor is the fundamental notion of curvature, implicitly denoting the holonomy for all possible transports. Though, via the Gauss-Codazzi equations, it turns out that the Riemannian curvature tensor can be computed from the SFF (and not the other way around).

Definition 17 (Riemann Curvature) *The Riemann Curvature tensor is the trilinear map $R : T_p\mathcal{M} \times T_p\mathcal{M} \times T_p\mathcal{M} \to T_p\mathcal{M}$ that satisfies the relation $R(X,Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z$. By the Gauss-Codazzi equations, R can be computed as $R_{ijkl} = \sum_\alpha h_{ik}^\alpha h_{jl}^\alpha - h_{il}^\alpha h_{jk}^\alpha$. The notation $[X,Y]$ refers to the Lie Bracket or the commutator $[X,Y] : C^\infty(\mathcal{M}) \to C^\infty(\mathcal{M})$ such that $[X,Y](f) = X(Y(f)) - Y(X(f))$.*

Much like the SFF, the Riemannian curvature tensor is quite large to deal with in practice (d^4 coefficients). The trace of the Riemannian curvature tensor is the Ricci curvature tensor, which distills only the volumetric growth associated with curvature in different directions. The Ricci curvature tensor can be further contracted to the scalar curvature, which is a single number average.

Definition 18 (Ricci and Scalar Curvatures) *The Ricci Curvature is the symmetric bilinear form $\mathrm{Ric} : T_p\mathcal{M} \times T_p\mathcal{M} \to T_p\mathcal{M}$ where $\mathrm{Ric}(X,Y) = \mathrm{tr}(Z \mapsto R(X,Y)Z)$ or $\mathrm{Ric}_{ij} = R_{ikjk}$. The Scalar Curvature is the trace of the Ricci Curvature $\mathrm{Scal} = \mathrm{tr}(\mathrm{Ric})$.*

Alternatively, a different way of contracting the Riemannian curvature tensor is to specify a two-dimensional plane of the tangent space to compute curvature on. This is called sectional curvature, and can be seen as a generalization of the Gaussian curvature in higher dimensions.

Definition 19 (Sectional Curvatures) *The Sectional Curvature is the map* $\mathrm{Sec} : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ *where* $\mathrm{Sec}(u, v) = \frac{\langle R(u,v)v,u\rangle}{\langle u,u\rangle\langle v,v\rangle - \langle u,v\rangle^2}$.

The average of the sectional curvatures is the Ricci curvature.

Finally, the last branch of the tree of curvatures relates how curves that lie on the manifold can be decomposed. Given a curve $\gamma \subseteq \mathcal{M}$, we are already familiar with the Frenet-Serret curvature at a point p given in Definition 9, whose first entry is the osculating circle curvature χ_1. We reference an alternative interpretation relating χ_1 to the geodesic and normal curvatures, which correspond to projecting the acceleration vector $\frac{d^2\gamma}{d^2t}$ onto the tangent space $T_p\mathcal{M}$ (intrinsic curvature) and the normal space $N_p\mathcal{M}$ (extrinsic curvature) respectively.

Definition 20 (Geodesic Curvature) *The Geodesic Curvature* κ_G *of a curve* $\gamma \subseteq \mathcal{M}$ *is the norm of the covariant derivative* $\kappa_G = ||\nabla_{\gamma'}\gamma'||$. *An equivalent expression is the projection of the acceleration onto the tangent space:* $\kappa_G = ||\operatorname{proj}_{T_p\mathcal{M}} \frac{d^2\gamma}{dt^2}||$. *Geodesics on a manifold have zero geodesic curvature, or* $\nabla_{\gamma'}\gamma' = 0$.

Definition 21 (Normal Curvature) *The Normal Curvature* κ_N *of a curve* $\gamma \subseteq \mathcal{M}$ *is the norm of the second fundamental form applied to the velocity of the curve:* $\kappa_N = ||\mathrm{II}(\gamma', \gamma')||$. *An equivalent expression is the projection of the acceleration onto the normal space:* $\kappa_N = ||\operatorname{proj}_{N_p\mathcal{M}} \frac{d^2\gamma}{dt^2}||$. *The osculating circle curvature, geodesic curvature, and normal curvature are then related by the Pythagorean theorem:* $\chi_1^2 = \kappa_G^2 + \kappa_N^2$.

In summary, extrinsic and intrinsic notions of curvature measure different aspects of curvature. In high dimensions, these can be prohibitively large tensors, but there exists useful and interpretable contractions of each. The web of connections between these notions of curvature is deep; most notably, it is possible to derive intrinsic curvature from extrinsic curvature. Now, we have all the differential geometry tools to work with and convert between all notions of curvatures.

A.5. Diffusion Models

A.5.1. Denoising Diffusion Probabilistic Model

Diffusion models work by training a model to undo the application of Gaussian noise applied to a point on the data manifold, broken down over a number of timesteps. The reason we would like this is that we can then generate new samples by using our model to denoise from white noise while keeping each denoising step small enough to be a tractable problem. The strengths of this approach are that training is stable because of the simple objective (as opposed to adversarial training dynamics found in GANs), and that we can exert control/guidance at any point during the generation process. This is at the cost of rather sequential computing power.

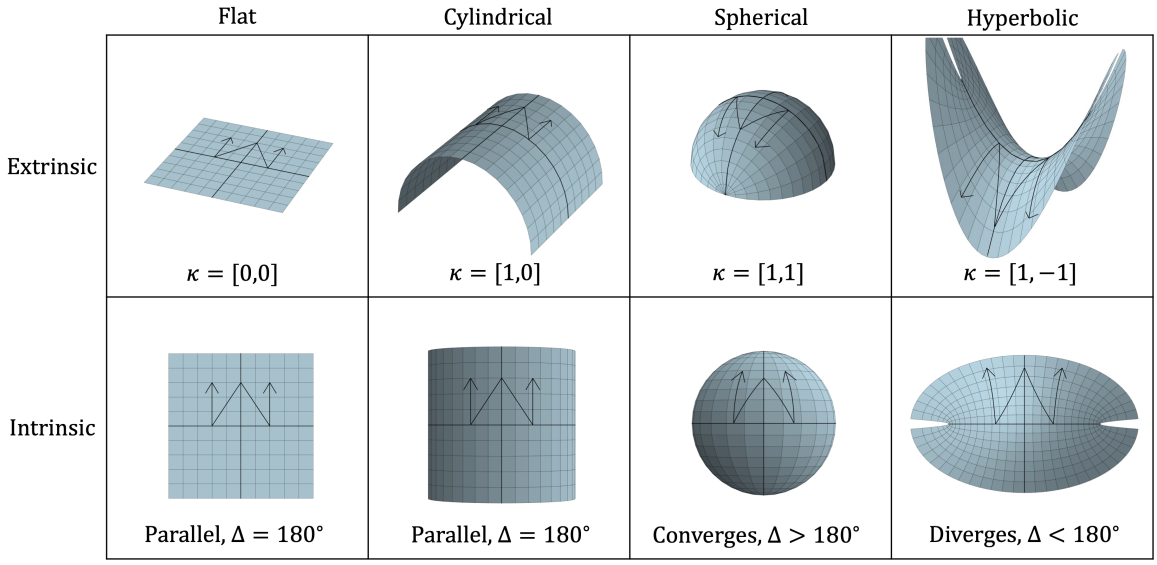

Figure 11: Extrinsic and intrinsic curvature on examples of two-dimensional surfaces.

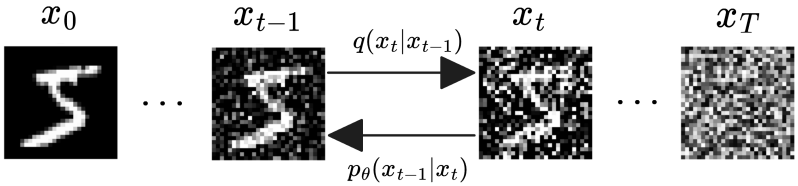

Figure 12: The Diffusion Process

The diffusion models leveraged in this work build on the *Denoising Diffusion Probabilistic Model* (Ho et al., 2020), which we define below.

More formally, we define a forward diffusion process that specifies the amount of noise to add at each timestep, and a reverse diffusion process that leverages a trainable model to reverse the noise.

Definition 22 (Forward Diffusion Process) *Let $x_0 \sim q(x)$ be a data point from the data manifold. The forward process is the Markov chain of T noised samples $\{x_1, \ldots, x_T\}$ given by the likelihood $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$. $\beta_1, \ldots, \beta_T$ is a variance schedule controlling the amount of noise added at each step. The notation $\mathcal{N}(x; \mu, \Sigma)$ is the normal pdf of $\mathcal{N}(\mu, \sigma)$ at x. For efficient sampling, there is the closed form $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$ where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.*

Definition 23 (Reverse Diffusion Process) *The reverse process is the procedure of recovering x_0 from x_t. Given a model θ, the reverse process estimates likelihood $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. The training loss is typically the ELBO loss $L = \mathbb{E}[-\log p_\theta(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}] \geq \mathbb{E}[-\log p_\theta(x_0)]$.*

Specifically, the DDPM diffusion model makes a few choices that differentiates it from the general diffusion model: it uses a fixed nonlearnable variance schedule, uses a fixed

nonlearnable variance estimator Σ_θ, and predicts the added noise rather than the reverse likelihood. One iteration of the training process is given by first sampling a training data point $x_0 \sim q(x)$, a timestep $t \sim \text{Unif}(\{1, \ldots, T\})$, a noise $\epsilon \sim \mathcal{N}(0, I)$ which generates $x_t = q(x_t|x_{t-1}, \epsilon)$ and then applying gradient descent with the gradient $\nabla_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2$. Sampling from the diffusion model involves a noisy sample x_T, then for each t from T to 1, computing a noise $z \sim \mathcal{N}(0, I)$ and $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sigma_t z$ for fixed variance estimates σ_t.

A.5.2. Learning data manifolds with diffusion models

While the DDPM architecture seems designed primarily for generating samples, the authors of DDPM note that ϵ_θ corresponds to score matching $\nabla_x \log q(x)$ (the direction that maximizes likelihood of data). Intuitively, this makes sense, as the denoising should push towards the original data. We can use this behavior to interact with the data manifold. Specifically, diffusion models allow us to (i) generate data points locally, (ii) obtain the tangent space of the data manifold at a point, and (iii) draw geodesic curves that lie on the manifold.

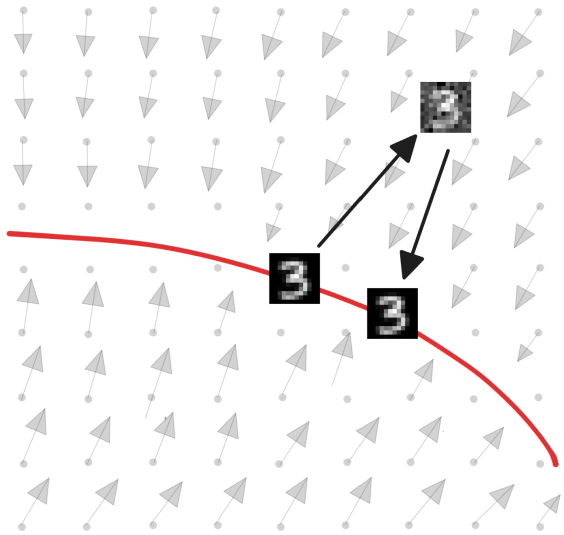

Figure 13: Score Points to the Manifold

Access to a trained diffusion model on the dataset allows us to obtain more realistic data samples around a point, as many as is needed for the curvature estimation algorithm. The method to do this is to apply a small amount of noise (a few timesteps) as to come off the data manifold, and to run the last few steps of the reverse process to get back onto the data manifold. The amount of noise will determine the size of the neighborhood on the manifold that you want to sample from. There are two challenges to this approach, but these are challenges not easily solvable in general. The first is that this will not necessarily result in a uniform sample on the manifold, and the second is that the right amount of noise can be difficult to gauge a priori.

Another useful application of diffusion models is the ability to estimate the tangent and normal space of the learned data manifold. Stanczuk et al. (2022) developed an algorithm using diffusion models for intrinsic dimension estimation (which is also important to know for curvature estimation). They showed that by running the forward process many times and obtaining the score vectors predicted by the diffusion model at these noised positions, one can run a singular value decomposition (SVD) on the matrix formed by these vectors and pick out the intrinsic dimension from the number of significant singular values in the resulting spectrum. Indirectly, the singular vectors also provides a basis for the normal space, from which one can construct the tangent space. Equivalently, we can also use our generated samples strategy to obtain a large number of samples on the manifold and apply principal component analysis (PCA) to obtain a basis for the tangent space, from which one can construct the normal space.

Finally, we can use diffusion models to obtain curves on a manifold. Estimating the full second fundamental form can be prohibitive. Instead, one can get a small picture of the overall curvature by looking at the Frenet-Serret curvature of a few sample curves on the manifold. There are many methods for interpolating between two points. Linear interpolation in the ambient space would likely drift off the data manifold for curved manifolds. Wang and Golland (2023) review methods of interpolation, including a linear interpolation followed by denoising by the diffusion model to get the curve to return to the data manifold, among others. This could be seen as being related to the connection operator from differential geometry.

A.5.3. Implementation Details

In our experiments, we use a fully connected SiLU residual neural network of 6 hidden layers each of width 1024, concatenating 128-dimensional sinusoidal time embeddings to the input. We train for 300 epochs using the AdamW optimizer with a learning rate of 0.001 and a batch size of 256, using a cosine annealing learning schedule with warm restarts ever 20 epochs. For the bunny manifold specifically, we use 50 epochs, a width of 512, and a learning rate of 0.0005. We use the DDPM noise scheduler with 1000 total timesteps and a cosine beta schedule. Notably, when generating samples for our experiments, we do not denoise with variance, which we find empirically to provide more stable curvature estimations. We run training and inference on our diffusion models using a single NVIDIA A100 GPU.

A.6. Curvature Estimation Methods

A.6.1. Quadratic Regression

The most complete notion of curvature is the Second Fundamental Form (recall Definition 14), which captures the extrinsic curvature of the manifold in ambient space. Each coordinate h_{ij}^k has the interpretation of being the quadratic coefficient describing the best quadratic approximation to the hypersurface specified by the kth normal dimension in terms of basis the ith and jth intrinsic dimensions. Note that the other notions of curvature can be reduced from the Second Fundamental Form; namely, the intrinsic curvature Riemannian curvature tensor which can be reduced from the Second Fundamental Form via the Gauss-Codazzi equations.

First introduced by Cao et al. (2021); Sritharan et al. (2021), the quadratic regression approach involves first performing a principal component analysis (PCA) in a locally linear neighborhood to obtain a basis for the tangent and normal spaces, and then performing a quadratic regression in a locally quadratic neighborhood of the normal basis against the tangent basis, which directly gives the SFF coefficients.

More formally, we consider the following regression where X is formed from the first d dimensions of the principal component scores and Y is the latter n dimensions of the principal component scores: (Sritharan et al., 2021)

$$
\begin{aligned}
X &= \begin{bmatrix} 1 & (x_1^1)^2 & (x_1^2)^2 & \dots (x_1^d)^2 \\ & & \vdots & \\ 1 & (x_N^1)^2 & (x_N^2)^2 & \dots (x_N^d)^2 \end{bmatrix} \in \mathbb{R}^{N\times(d^2+1)} \\
Y &= \begin{bmatrix} x_1^{d+1} & x_1^{d+2} & \dots x_1^m \\ & & \vdots \\ x_N^{d+1} & x_N^{d+2} & \dots x_N^m \end{bmatrix} \in \mathbb{R}^{N\times n} \\
H &= \begin{bmatrix} c_1 & c_2 & \dots & c_n \\ h_{1,1}^1 & h_{1,1}^2 & \dots & h_{1,1}^n \\ h_{1,2}^1 & h_{1,2}^2 & \dots & h_{1,2}^n \\ & & \vdots & \\ h_{d,d}^1 & h_{d,d}^2 & \dots & h_{d,d}^n \end{bmatrix} \in \mathbb{R}^{(d^2+1)\times n} \\
\hat{H}_{:,k} &= (X^TX)^{-1}X^TY_{:,k}
\end{aligned}
$$

Cao et al. (2021) show that the mean squared error of this estimator under the appropriate assumptions accompanying linear regression (normality and independence of residuals, homoskedasticity) is $O(N^{-\frac{4}{d+4}})$ where N is the number of points and d is the intrinsic dimension.

Note that the second fundamental form (H excluding the first row of constants) is necessarily symmetric, so estimating H can be done in a number of ways:

- **Closed-form matrix inversion:** Directly computing $\beta = (X^TX)^{-1}X^TY$ requires a non-parallelizable matrix inversion, but a faster approach is to use the Moore-Penrose pseudo-inverse $\beta = X^\dagger Y$, computable via SVD. Assuming $N > d^2$, computing $X = U\Sigma V^T$ takes $O(N(d^2)^2)$ time, and computing the matrix multiplications is $d^2d^2N + d^2Nn$ which gives final time complexity of $O(Nd^2(d^2+n))$.

- **Gradient-based approach**: instead of computing the matrix inversion, perform stochastic gradient descent. That is, compute $\hat{H}^{(t+1)} = \hat{H}^{(t)} - \eta \cdot \frac{1}{B}\sum_{i=1}^{B} ||X_iH - Y_i||_2^2$ for some learning rate η and batch size B until convergence. This has the advantage of being parallelizable over the batch and can better take advantage of GPUs to compute the matrix multiplications. the symmetry constraint is enforced at the end by computing $H = \frac{1}{2}(\hat{H} + \hat{H}^T)$. The time complexity of this is dominated by computing XH which is $O(Nd^2n)$.

- **Optimization over the manifold of symmetric matrices:** Since H is symmetric, we can use Riemannian manifold optimization to explore over the more limited space of symmetric matrices. Furthermore, if we add in an assumption of the rank of H, we can use Riemannian optimization over the space of fixed rank matrices. However, this method suffers due to a vastly larger overhead from computing Riemannian gradients and other geometric tools, which are more expensive than their Euclidean counterparts.

We found that, in the practise, that the matrix inversion is the best choice whenever computationally feasible, and gradient descent is the preferred method for larger matrices (see Section 4 for more details). Thus, we use matrix inversion for toy datasets and gradient descent for the larger image datasets.

We consider two applications of quadratic regression, one on the raw data manifold, and one on the samples generated from a diffusion model. For performing quadratic regression on samples from a diffusion model, we run a two-step process where we first uniformly sample a ball of radius r_{diff} and denoise for T timesteps, then run local PCA to obtain a tangent basis. Second, we uniformly sample a ball of radius r_{diff} restricted to the tangent basis, and denoise for T samples. This last set of samples is the set of samples that we use for quadratic regression. The reason to run this two-step process is that we don't want to waste samples reiterating the normal dimensions.

A.6.2. Diffusion Maps

Diffusion maps (Coifman and Lafon, 2006) is a nonlinear manifold learning technique (remark: despite the similar name, it bears no relation to diffusion models, the generative models we use to sample from the data manifold). It leverages the idea that geodesic manifold distance can be approximated by random walks between nearby points. A diffusion map is just a description of the spectral decomposition of the transition matrix of that random walk. The first eigenvector corresponds to the stationary of eigenvalue 1, but the remaining eigenvectors describe the dominant directions of variation. It turns out this spectral description describes the Laplace-Beltrami operator, or the extension of a Laplacian defined on a curved manifold. This information is powerful for curvature estimation as we can use the Carre du Champ identity to transform estimates of the Laplace-Beltrami operator into estimates for the Riemannian Curvature Tensor (recall Definition 17).

More formally, Coifman and Lafon (2006) and Liu (2021) define a kernel function K to assign weights to random walk transition probabilities between two data points satisfying symmetry ($K(x,y) = K(y,x)$) and non-negativity ($K(x,y) \geq 0$). We employ the heat kernel (radial basis function, or RBF) with variable bandwidth: $K(x,y) = \exp(\frac{||x-y||^2}{\epsilon\rho(x)\rho(y)})$ where ρ is a density-dependent bandwidth function (Berry and Harlim, 2016), though this is not the sole feasible choice. An $N \times N$ transition matrix can be formed from row-normalizing $P_{ij} = \frac{K(x_i,x_j)}{\sum_a K(x_a,x_j)}$ or $P = D^{-1}K$ for weighted degree matrix D. Then, $(P^{\,t})_{ij}$ denotes the time-evolved transition probabilities of starting at i and ending at j after t steps, according to the random walk. t has the additional interpretation of a length scale, as random walk probability distributions will only capture local geometry with low t, but with more time, they can traverse larger and larger neighborhoods and capture more global geometry. To understand the powers $P^{\,t}$, a spectral analysis is conducted.

Consider a density-normalized $\tilde{P} = D^{1/2}PD^{-1/2} = D^{1/2}(D^{-1}K)D^{-1/2} = D^{-1/2}KD^{-1/2}$, observing that $I - D^{-1/2}KD^{-1/2} = I - \tilde{P}$ is the normalized graph Laplacian of K. $\tilde{P}$ is similar to P as it is related by a change of basis $D^{1/2}$. Therefore, it shares the same eigenvalues and eigenvectors up to a change of basis. $\tilde{P}$ is also symmetric because K is symmetric: $\tilde{P}^T = (D^{-1/2}KD^{-1/2})^T = D^{-1/2}K^TD^{-1/2} = D^{-1/2}KD^{-1/2} = \tilde{P}$. Thus, by the spectral theorem, symmetric matrices can be decomposed as $\tilde{P} = \Phi\Lambda\Phi^{-1}$ with all real eigenvalues—in fact, they are nonnegative as K is defined as PSD so $\tilde{P}$ is as well.

By the ergodic theorem (the Markov chain is irreducible, aperiodic, and finite), there exists a limiting distribution which is the stationary distribution $\pi = \pi P$ that is given by the first eigenvector which for a random walk is the diagonal $\frac{D}{\mathrm{tr}(D)}$; in fact, the spectral radius (max eigenvalue) of a stochastic matrix is 1, so we know that the eigenvalues are diminishing $1 = \lambda_1 \geq \cdots \geq \lambda_n \geq 0$.

The diffusion map is then defined as the map $\Psi_t(x) = \begin{bmatrix}\lambda_2^t\phi_2(x) & \dots & \lambda_n^t\phi_n(x)\end{bmatrix}$, which can be truncated for dimensionality reduction. Observe that the first eigenvalue/eigenvector is dropped as it is the constant 1 vector. The interpretation is that as t grows (larger length scale), the eigenvectors with the smaller eigenvalues decay. In other words, the first few eigenvectors capture more global geometry while the last few eigenvectors capture more local geometry. Another interpretation of this diffusion map arises from the diffusion distance, or the difference in random walk destination probability based on starting point, defined as $d_t(i,j)^2 = \sum_k((P^{\ t})_{ik} - (P^{\ t})_{jk})^2/\pi_k$. This is equal to the Euclidean distance between two points after the applying the diffusion map; $d_t(x,y)^2 = \sum_{i\geq 2}\lambda_j^{2t}(\phi_j(x) - \phi_j(y))^2$. The time complexity for an eigendecomposition of a $N \times N$ matrix is $N^{\ 3}$.

This is useful for curvature as the discrete graph Laplacian $I - \tilde{P}$ is an estimate for the continuous Laplace-Beltrami operator $\Delta = \lim_{N\to\infty,\epsilon\to 0}\frac{I-\tilde{P}}{\epsilon}$, which captures intrinsic geometry as the divergence $\Delta_p : T_p\mathcal{M} \to \mathbb{R}$. Then, the carre du champ operator gives us the Riemannian metric as $g_p(x,y) = \frac{1}{2}(x\Delta_p(y) + y\Delta_p(x) - \Delta_p(xy))$. Forming the Gram matrix $G_{ij} = g(x^i, x^j)$, the eigendecomposition of G produces a basis for the manifold. After separating out the tangent dimensions from the normal dimensions, iterating the carre du champ again gives us the second fundamental form $II_z(x,y) = \frac{1}{2}(g(x, g(y,z)) + g(y, g(x,z)) - g(z, g(x,y)))$, from which we can derive the Riemannian Curvature Tensor via the Gauss-Codazzi equations referenced in Definition 17) (Jones, 2024).

Appendix B. Extended Related Work

In this section we provide an overview of applications of curvature as a geometric characteristic in data science and machine learning applications. In many of these works, curvature plays a central role, often without being explicitly estimated from data.

Shape Analysis Curvature estimation restricted to 3-dimensional spaces connects with the field of shape analysis (Solomon, 2021). Shape analysis seeks to detect similarly shaped objects (in part, using curvature), which has applications in archaeology (Hanik et al., 2023), medical imaging (Feydy, 2020), and facial recognition (Tang et al., 2017). More generally, curvature is a tool in visualization, reconstruction (Wang et al., 2012), smoothing (Desbrun et al., 1999), contouring (DeCarlo and Rusinkiewicz, 2007), and remeshing (Alliez et al., 2003). Recently, He et al. 2020 (He et al., 2020) invented curvature filters which look at

principal curvature and pair with graph convolutional neural networks to perform better at shape matching.

Learning Complexity Curvature may also have implications for learning theory. Kiani et al. (2024) developed theory on the hardness of learning neural networks assuming the manifold hypothesis. They characterize three regimes; a learnable regime with lower-bounded Ricci curvature, a hard-to-learn regime with upper-bounded reach (which itself lower bounds the radius of curvature), and a middle regime. Thus, understanding the curvature of the data manifold may be able to indicate which learnability regime the data is for learning neural networks.

Regularization Curvature may also be useful for regularization in neural networks. Ma et al. (2025) show that curvature imbalance on the perceptual manifold (the embedding space of the neural network before the classification head) leads to model bias. Hence, they suggest training neural networks with curvature regularization on that embedding space. Curvature regularization of the loss surface (Moosavi-Dezfooli et al., 2019) has also been shown to help with adversarial robustness. Conversely, curvature of the loss surface can be exploited in data privacy attacks (Li et al., 2023). Another application of curvature is the ability to steer diffusion models (Hong, 2025) to better quality outputs by regularizing the curvature of the energy landscape of attention.

Non-Euclidean Embeddings Researchers and practitioners alike will often make educated guesses on the right curvature embedding space to use for their data. For example, hyperbolic embeddings (in negatively curved space), are often used for hierarchical data such as phylogenetic trees, knowledge graphs, or image classifications (Khrulkov et al., 2020; Chami et al., 2020; Klimovskaia et al., 2020; Bhasker et al., 2024). Curvature estimation may be able to provide another perspective on the true curvature of data, to better inform the choice of embedding space.

Appendix C. Additional Experimental Results

C.1. Overview of data sets

We choose four canonical 2D manifolds embedded in 3D space, and high-dimensional paraboloids generated by the second fundamental form.

Sphere The simplest non-flat manifold is the sphere $S^{n-1} = \{x : ||x||_2^2 = r\}$, which has constant positive curvature. The SFF is given by the identity matrix.

Roll The roll is an example of a manifold that has extrinsic curvature but no intrinsic curvature. It is given by the Archimedean spiral extruded to give depth, parameterized by $(x, y, z) = (at\cos t, at\sin t, s)$ where $t \in [0, 3\pi]$ parameterizes the spiral and $s \in [-5, 5]$ parameterizes the flat dimension. It has increasing positive Gaussian curvature on the dimension rolled up and zero Gaussian curvature on the other. The SFF for a given (t, s) is the matrix $\begin{bmatrix} (t^2+2)/(a(t^2+1)^{3/2}) & 0 \\ 0 & 0 \end{bmatrix}$.

Paraboloid The paraboloid is a manifold fully specified by the SFF at the vertex, formed by the quadratic form $y = x^T(2H)x$. In theory, quadratic regression at the vertex should be able to perfectly reconstruct the entire manifold. We test on a paraboloid with eigenvalues (principal curvatures) $1, -1$ which results in a saddle point at the vertex.

High-Dimensional Paraboloid The paraboloid is easily extendable to higher dimensions, as it is fully specified by the SFF. This makes it easy to compute the R^2 with the ground truth SFF in quadratic regression. We generate paraboloids with eigenvalues randomly generated along the semicircular distribution between -2 and 2, which is the resultant eigenvalue distribution according to Wigner's semicircular law for symmetric matrices with random standard normal entries.

C.2. Visualizations of Curvature Estimation on the Sphere

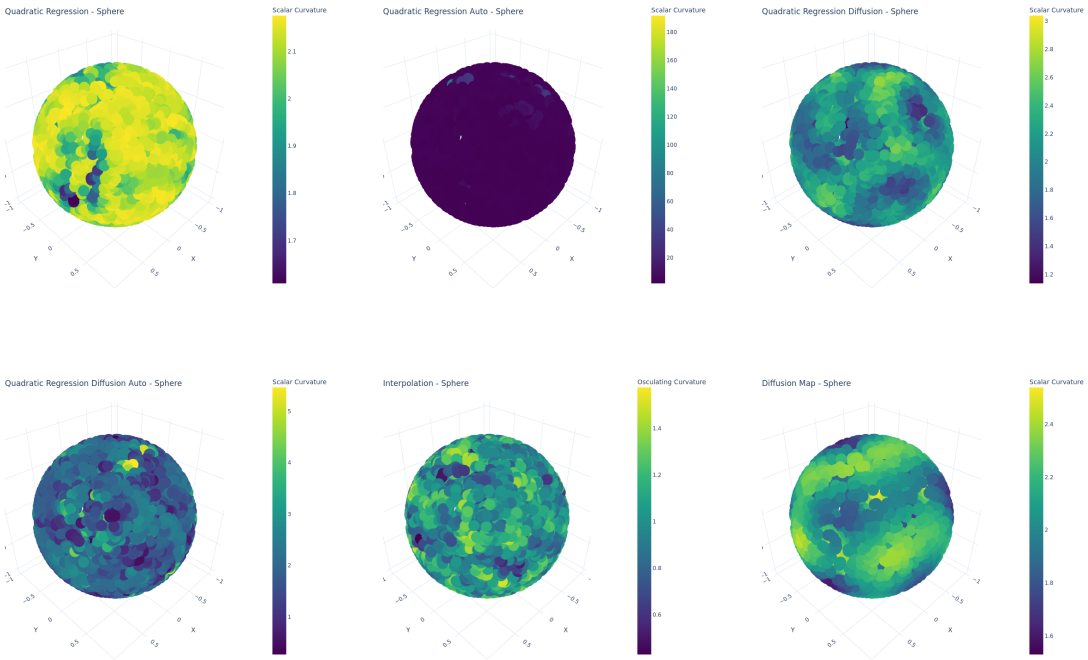

Figure 14: Results of curvature estimation on the sphere for all methods.

C.3. Diffusion Model Training Curves

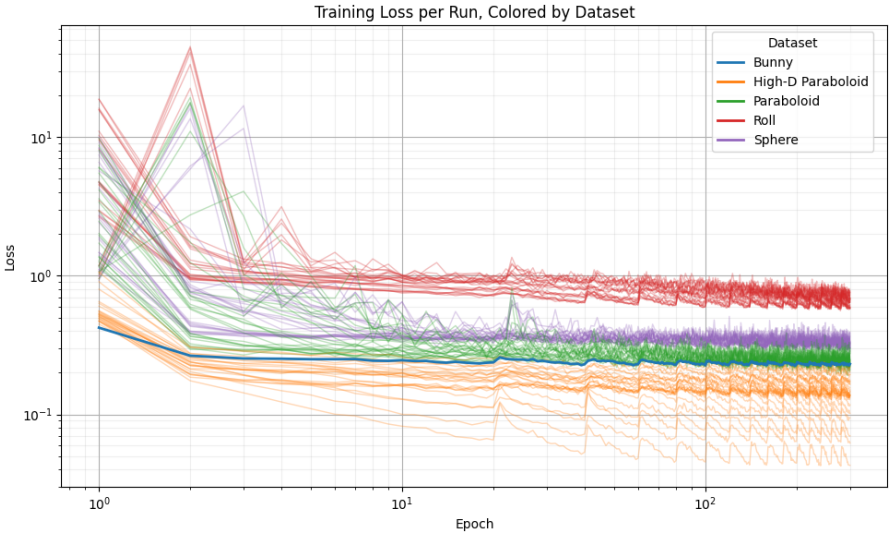

Figure 15: Diffusion Model Training ($N = 5000, s = 0$)

C.4. Visualization of the Diffusion Model's learned manifolds

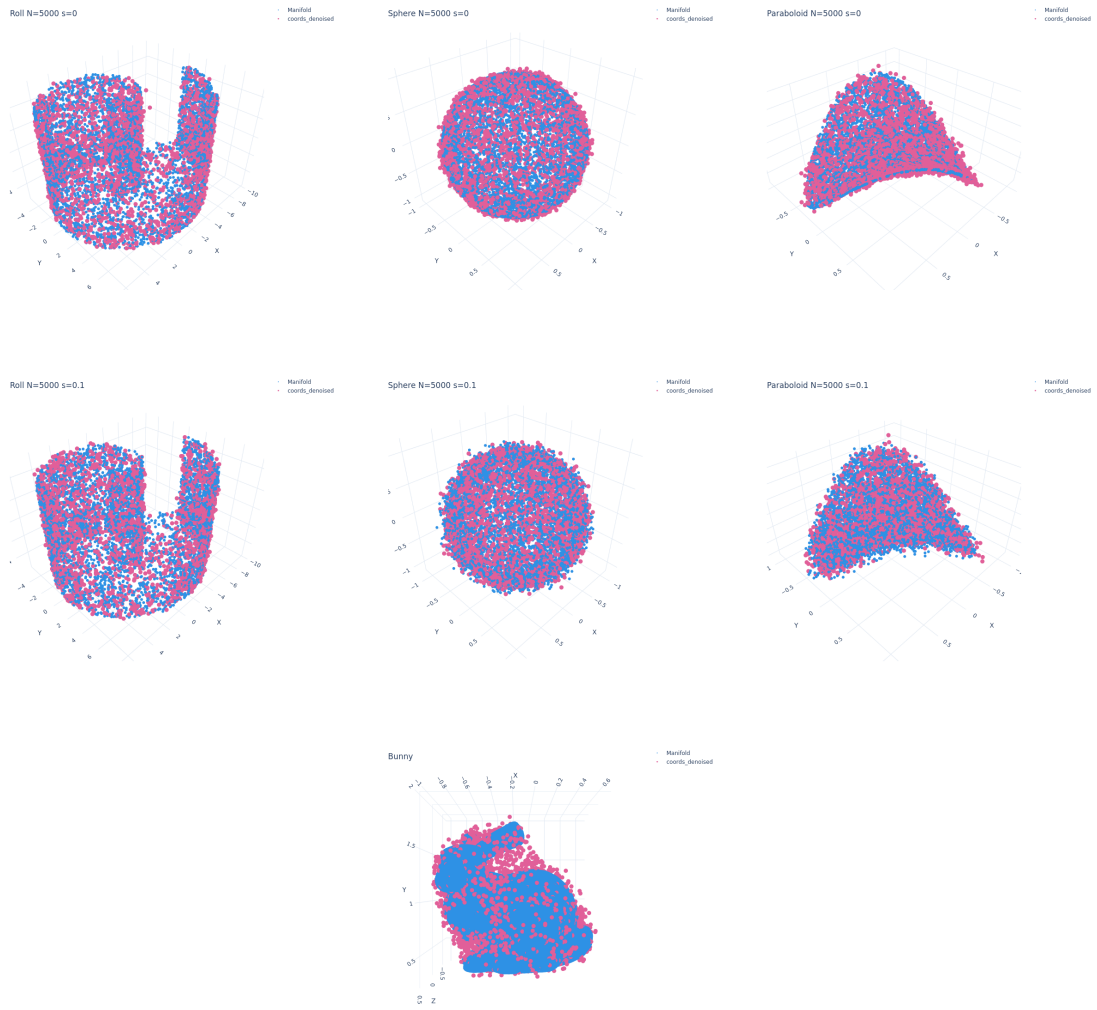

Figure 16: The diffusion model's learned manifolds for synthetic and real-world data at two noise levels.

C.5. Quadratic regression with adaptive neighborhood sizes on synthetic data manifolds

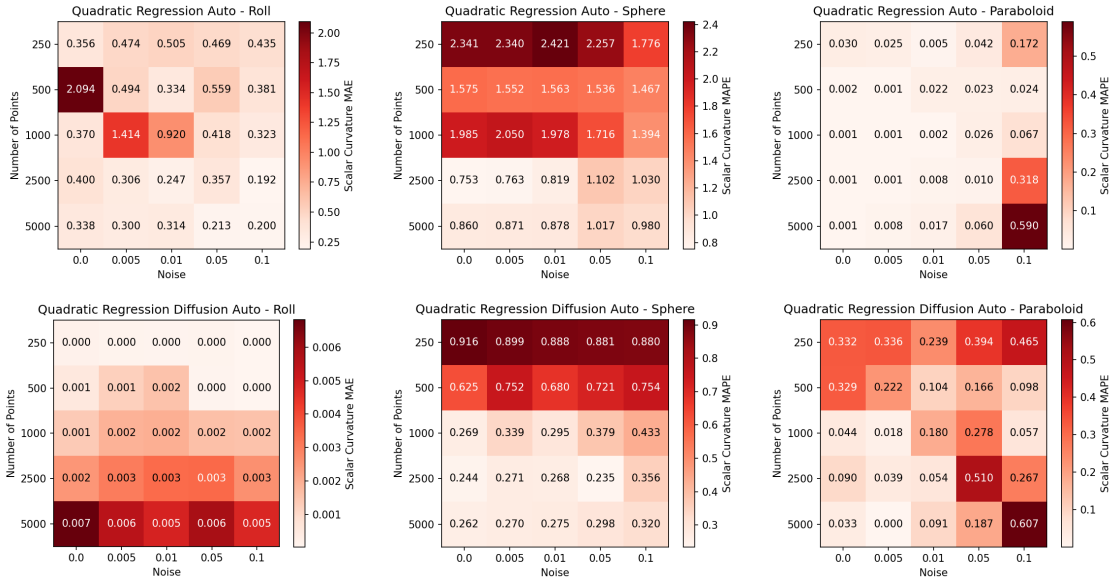

Figure 17: Results for quadratic regression with adaptively chosen neighborhood sizes across datasets.

C.6. Additional results for quadratic regression

We provide R^2 values of the quadratic fit for the different approaches, rather than the actual MAPE presented in the main text.

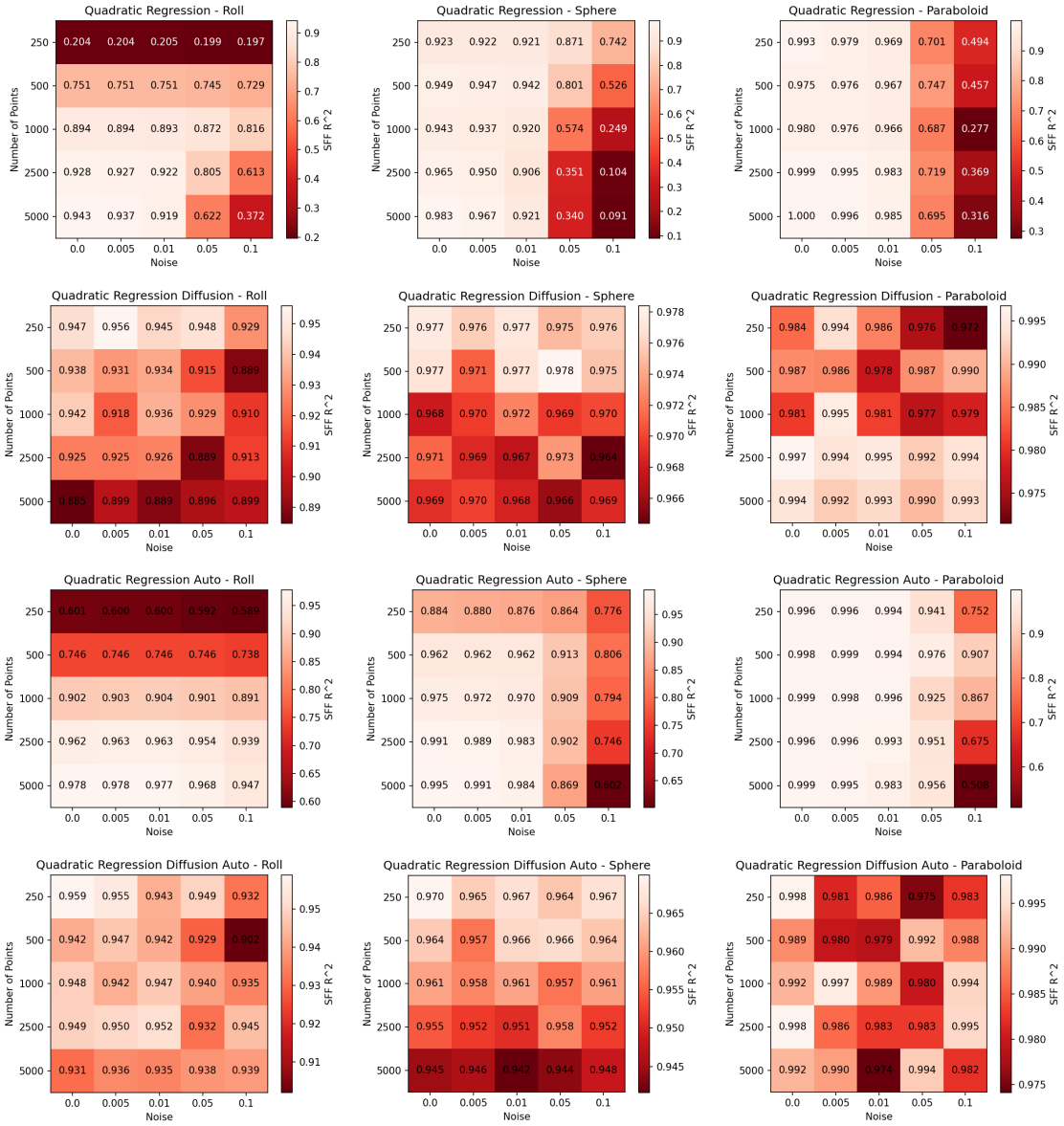

Figure 18: Quadratic Regression Curvature Estimations for Each Toy Manifold (Number of Samples vs. Noise)

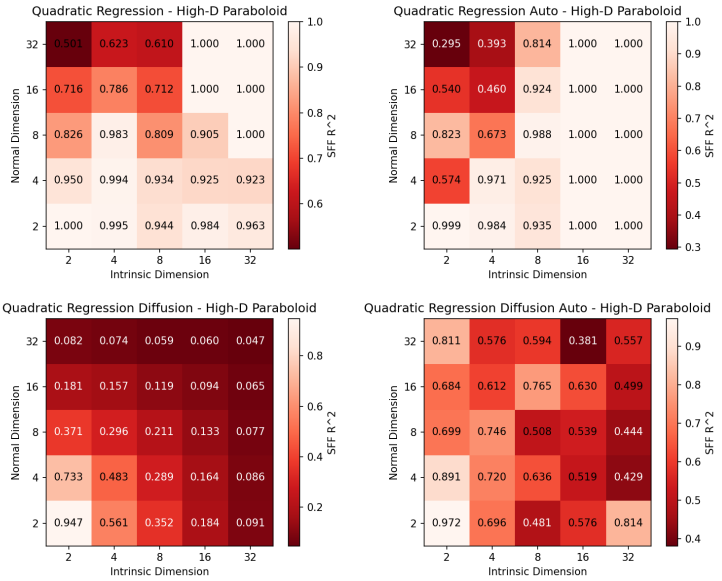

Figure 19: Intrinsic Dimension vs. Normal Dimension on the Paraboloid

C.7. Manual vs. automatic neighborhood selection on the Stanford Bunny

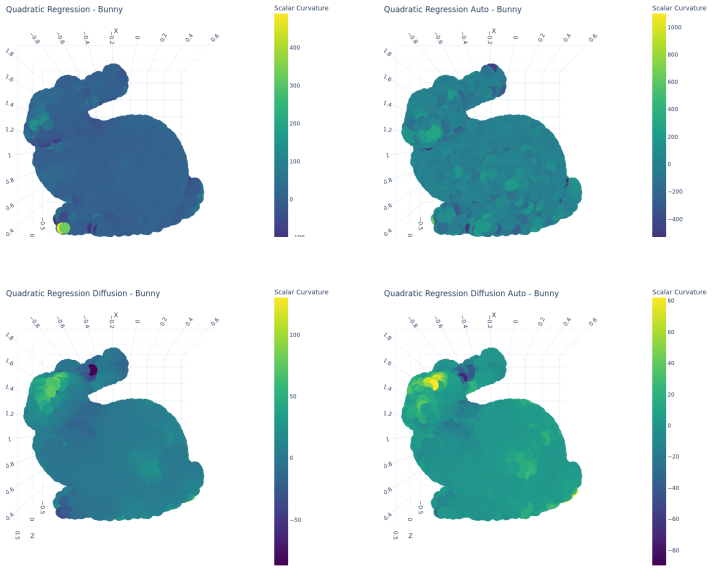

Figure 20: Comparison of Manual vs. Automatic Neighborhood Selection on the Stanford Bunny