On Explicit Curvature Regularization in Deep Generative Models

Yonghyeon Lee [1] **Frank C. Park** [2 3]

Abstract

We propose a family of curvature-based regularization terms for deep generative model learning. Explicit coordinate-invariant formulas for both intrinsic and extrinsic curvature measures are derived for the case of arbitrary data manifolds embedded in higher-dimensional Euclidean space. Because computing the curvature is a highly computation-intensive process involving the evaluation of second-order derivatives, efficient formulas are derived for approximately evaluating intrinsic and extrinsic curvatures. Comparative studies are conducted that compare the relative efficacy of intrinsic versus extrinsic curvature-based regularization measures, as well as performance comparisons against existing autoencoder training methods. Experiments involving noisy motion capture data confirm that curvature-based methods outperform existing autoencoder regularization methods, with intrinsic curvature measures slightly more effective than extrinsic curvature measures.

1. Introduction

For a large class of deep generative models, the problem of manifold learning is central to their training: given a set of data points drawn from some high-dimensional space $\mathcal{X}$ and for which the Manifold Hypothesis remains valid – that is, the data points are assumed to lie on some lower-dimensional manifold $\mathcal{M} \subset \mathcal{X}$ – the objective is to determine a mapping $f : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{Z}$ is typically a bounded region of a vector space with the same dimension as $\mathcal{M}$, such that $f(\mathcal{Z})$ closely approximates $\mathcal{M}$. The mapping f is the *generator* (or the *decoder* in the context of autoencoders), $\mathcal{Z}$ is called the *latent space*, and $\mathcal{M}$ is referred to as the *data manifold*.

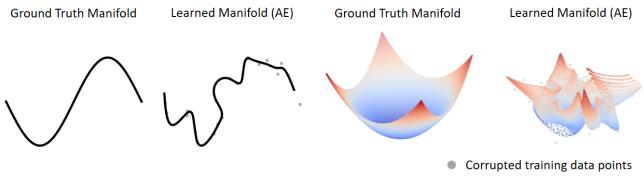

Figure 1. Autoencoder-based manifold learning examples given noise-corrupted training data.

Deep neural networks are a popular choice of parametric mapping for the generator f (LeCun et al., 2015). Methods for training such deep generative models include those based on the autoencoder (Kramer, 1991; Kingma & Welling, 2013), adversarial training (Goodfellow et al., 2014), manifold flow (Brehmer & Cranmer, 2020), and the autodecoder (Bojanowski et al., 2017). As with any function approximation problem, the presence of noise in the data can lead to overfitting, resulting in poor manifold approximations of the type shown in Figure 1. Adding a regularization term to the loss function is one means of mitigating the effects of overfitting. With few exceptions the regularization terms are formulated in terms of the gradient of f; the underlying intuition is that by minimizing, e.g., the norm of the gradient, f is forced to be close to linear, and thus less prone to overfitting to any noise-induced variations in the data.

Several recent works have pointed out the importance of formulating both the loss function and regularization terms in a *coordinate-invariant* way (Jang, 2019; Jang et al., 2020; Lee et al., 2022b). In fact, in (Lee et al., 2022b) a compelling case is made that the problem of determining f is best framed as one of finding a *minimum distortion* map between two Riemannian manifolds: the key idea is to "wrap" $\mathcal{M}$ by $\mathcal{Z}$ in such a way that the overall distortion – one can view $\mathcal{M}$ as being made of marble and $\mathcal{Z}$ of elastic, with the distortion corresponding to the elastic energy resulting from deforming $\mathcal{Z}$ – is minimized. This geometric framework allows for, among other things, classifying existing manifold and representation learning approaches based on, e.g., the choice of Riemannian metric, boundary conditions, and other a priori specified factors.

[1]Korea Institute for Advanced Study, Seoul, South Korea [2]Department of Mechanical Engineering, Seoul National University, Seoul, South Korea [3]Saige Research, Seoul, South Korea. Correspondence to: Yonghyeon Lee <ylee@kias.re.kr>, Frank C. Park <fcp@snu.ac.kr>.

Proceedings of the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023.

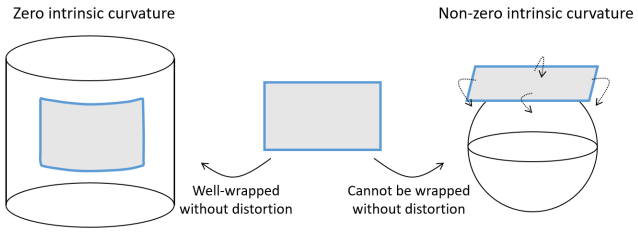

Figure 2. Left: The cylindrical surface has zero intrinsic curvature but has non-zero extrinsic curvature. *Right*: The spherical surface has non-zero intrinsic and extrinsic curvature.

In this paper we adopt a different geometric approach that focuses on the *curvature* of the data manifold. Intuitively, curvature measures how much a manifold deviates from a flat manifold, and for the purpose of learning a generator f that is robust to noise, a regularization term that attempts to minimize the curvature of the resulting data manifold would seem quite sensible. In the case of classical two-dimensional surfaces embedded in $\mathbb{R}^3$, curvature is quantified as the rate of change of the surface normal along certain directions; for this, second-order derivatives of the surface parametrization f are involved. This is a key fundamental difference with distortion-based regularization terms, which involve only first-order derivatives of f.

The theory of curvature for Riemannian manifolds is both well-developed and at the same time intimidatingly complex, but the intuitive ideas can be understood from the case of classical two-dimensional surfaces (Do Carmo, 1992; 2016). The *extrinsic curvature* of a surface captures how the surface is embedded in $\mathbb{R}^3$, whereas the *intrinsic curvature* is a property intrinsic to the surface, independent of its embedding in $\mathbb{R}^3$. For example, a cylinder and a flat piece of paper have different extrinsic curvatures but identical intrinsic curvatures – the flat paper can be rolled into a cylinder without any deformations – while a sphere and cylinder have different intrinsic curvatures (see Figure 2).

In the context of deep generative model learning, the role of extrinsic versus intrinsic curvature in the formulation of a regularization term has yet to be investigated. At first glance, it would seem obvious that extrinsic measures are more relevant, since the shape of the data manifold – specifically, how it is embedded in the higher-dimensional ambient space – is clearly important. On the other hand, generalizations of curvature to higher-dimensional manifolds are focused almost exclusively on intrinsic measures like the Ricci curvature (Ricci-Curbastro, 1904; Besse, 2007; Do Carmo, 1992); coordinate-invariant measures of extrinsic curvature for higher-dimensional manifolds embedded in Euclidean space have yet to be addressed in the literature. Moreover, formulas for both intrinsic and extrinsic curvatures are heavily computation-intensive, as they involve second-order derivatives of the generator f.

This paper reports on three contributions. First, we formulate explicit coordinate-invariant extrinsic curvature measures for multi-dimensional manifolds embedded in higher-dimensional Euclidean space. The core idea is to generalize the Gauss map for two-dimensional surfaces in $\mathbb{R}^3$ to higher-dimensional embedded manifolds using the machinery of Grassmann manifolds (Wong, 1967; Bendokat et al., 2020) and Dirichlet energy (Eells & Lemaire, 1978; 1988). Second, we derive computationally efficient algorithms for approximately computing both intrinsic and extrinsic curvature measures that are based on Hutchinson's stochastic trace estimator (Hutchinson, 1989).

The third and final contribution is a comparative study of the efficacy of extrinsic versus intrinsic curvature measures for generative deep model learning. Adopting an autoencoder framework, regularization terms that minimize intrinsic and extrinsic curvature measures are developed, and applied to motion capture data. Performance comparisons with existing autoencoder training methods (Vincent et al., 2010; Kingma & Welling, 2013; Rifai et al., 2011b; Alain & Bengio, 2014; Lee et al., 2021; 2022b) are also provided.

A key finding of our study is that both intrinsic and extrinsic curvature-based regularization terms are more effective than existing autoencoder regularization methods in mitigating the effects of noise. The same can be said when comparing our curvature-based methods to purely first-order gradient-based distortion measures. Intrinsic curvature measures, although requiring more computation than their extrinsic counterparts, appear to be slightly more effective.

2. Curvature of Riemannian Manifolds

In what follows we consider an m-dimensional Riemannian manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$. Choosing $z \in \mathbb{R}^m$ as local coordinates for $\mathcal{M}$, $\mathcal{M}$ is parametrized by the mapping $f : \mathbb{R}^m \to \mathcal{M}$, i.e., $z \mapsto x = f(z)$. Here $x \in \mathcal{M} \subset \mathbb{R}^D$. The Riemannian metric on $\mathcal{M}$ is obtained in terms of local coordinates z as $G(z) := J_f^T(z)J_f(z)$ where $J_f(z) = \frac{\partial f}{\partial z}(z)$ (this follows from the Euclidean incremental arclength formula $ds^2 = dx_1^2 + \dots dx_D^2 = dx^T dx = dz^T J_f^T J_f dz$, where $dx = \frac{\partial f}{\partial z} dz = J_f dz$). The (i, j)-th component of the Riemannian metric G is denoted by g_{ij}, and that of G^{-1} is denoted by g^{ij}.

2.1. Intrinsic Curvature

For classical two-dimensional surfaces in $\mathbb{R}^3$, the Gaussian curvature, which can be obtained as the product of the principal curvatures, is an intrinsic curvature measure. For higher-dimensional Riemannian manifolds, the study of intrinsic curvature begins with the Riemann curvature

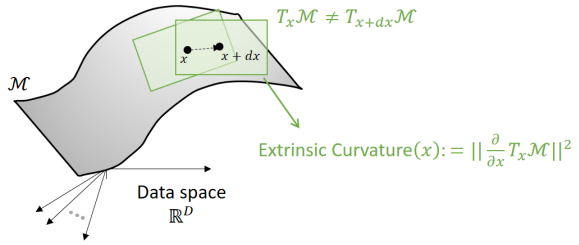

Figure 3. The tangent space to $\mathcal{M}$ at x varies as x moves to $x + dx$, i.e., $T_x\mathcal{M} \neq T_{x+dx}\mathcal{M}$; the manifold has non-zero extrinsic curvature at x.

tensor (Do Carmo, 1992), which is defined in coordinates as

$$R^a_{bcd} = \frac{\partial}{\partial z^c}\Gamma^a_{db} - \frac{\partial}{\partial z^d}\Gamma^a_{cb} + \sum_{\lambda=1}^{m}\Gamma^a_{c\lambda}\Gamma^\lambda_{db} - \sum_{\lambda=1}^{m}\Gamma^a_{d\lambda}\Gamma^\lambda_{cb}, \tag{1}$$

where Γ^a_{bc} are the Christoffel symbols of the second kind:

$$\Gamma^a_{bc} = \frac{1}{2}\sum_{\lambda=1}^{m} g^{a\lambda}\left(\frac{\partial}{\partial z^c}g_{\lambda b} + \frac{\partial}{\partial z^b}g_{\lambda c} - \frac{\partial}{\partial z^\lambda}g_{bc}\right). \tag{2}$$

The Ricci curvature is obtained by contracting the Riemann curvature tensor: $\mathrm{Ric}_{ij}(z) := \sum_{a=1}^{m} R^a_{iaj}(z)$. In the case of two-dimensional surfaces, the Ricci curvature reduces to $\mathrm{Ric}_{ij}(z) = K g_{ij}(z)$ where K is the Gaussian curvature.

The scalar curvature $R = \sum_{i,j}\mathrm{Ric}_{ij}(z)g^{ij}(z)$ is a natural scalar measure of the intrinsic curvature of a Riemannian manifold. We note that $R = 2K$ for two-dimensional surfaces. Just like Gaussian curvature, the scalar curvature can be negative or positive. In what follows we use the squared scalar curvature as a local intrinsic curvature measure:

$$\text{Intrinsic Curvature}(z) = \left(\mathrm{Tr}(G^{-1}(z)\mathrm{Ric}(z))\right)^2. \tag{3}$$

More detailed and formal treatments of the Riemann curvature tensor can be found in, e.g., (Do Carmo, 1992). A more intuitive derivation of the curvature tensor based on the notion of parallel transport can be found in (Fecko, 2006; Schutz, 2022).

2.2. Extrinsic Curvature

For a two-dimensional surface $\mathcal{S}$ in $\mathbb{R}^3$, its extrinsic curvature can be quantified as the rate of change of the surface normal. Specifically, the Gauss map $n : \mathcal{S} \to \mathbb{S}^2$ assigns to each point $x \in \mathcal{S}$ a unit vector $n(x)$ that is normal to $\mathcal{S}$ at x. The differential of the Gauss map then leads to a measure of the extrinsic curvature via the second fundamental form (Do Carmo, 2016; Kühnel, 2015).

We now generalize the above Gauss map construction to an m-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^D$ and formulate a coordinate-invariant extrinsic curvature measure as the norm of $\frac{\partial}{\partial x}T_x\mathcal{M}$, where $T_x\mathcal{M}$ denotes the tangent space to $\mathcal{M}$ at x (see Figure 3). The first step is to establish a 1-1 correspondence between the set of tangent spaces and the Grassmann manifold $\mathrm{Gr}(m, \mathbb{R}^D)$:

$$\mathrm{Gr}(m, \mathbb{R}^D) = \{P \in \mathbb{R}^{D\times D} \mid P^T = P,\ P^2 = P,\ \mathrm{rank}(P) = m\}. \tag{4}$$

$\mathrm{Gr}(m, \mathbb{R}^D)$ can be viewed as the $m(D-m)$-dimensional manifold of orthogonal projection matrices; every element $P \in \mathrm{Gr}(m, \mathbb{R}^D)$ can be uniquely identified with the linear subspace $\mathrm{Range}(P) \subset \mathbb{R}^D$ (Bendokat et al., 2020)[1]. For a generator $f : \mathbb{R}^m \to \mathcal{M}$, the tangent space of $\mathcal{M}$ at $x = f(z)$ is equal to the range of $J_f(z)$, and hence the orthogonal projection matrix $E(J_f) := J_f(J_f^T J_f)^{-1}J_f^T$, which is an element of $\mathrm{Gr}(m, \mathbb{R}^D)$, can be identified with the tangent space $T_x\mathcal{M}$ in a 1-to-1 manner. We note that the representation $E(J_f)$ is coordinate-invariant; the proof is given in Appendix A.

We next consider a generalization of the Gauss map. The mapping $\mathrm{T} : \mathcal{M} \to \mathrm{Gr}(m, \mathbb{R}^D)$ assigns to each point $x \in \mathcal{M}$ an m-dimensional linear subspace in $\mathbb{R}^D$ tangential to $\mathcal{M}$. In local coordinates z, this mapping can be expressed as

$$T(z) = E(J_f(z)) = J_f(z)(J_f(z)^T J_f(z))^{-1}J_f(z)^T. \tag{5}$$

Finally, taking the standard Riemannian metric on the Grassmann manifold given by

$$\langle V_1, V_2\rangle := \mathrm{Tr}(V_1^T V_2) \tag{6}$$

for any $V_1, V_2 \in T_P\mathrm{Gr}(m, \mathbb{R}^D)$, we use the Dirichlet energy of the mapping T (Eells & Lemaire, 1978; 1988), which is a natural coordinate-invariant smoothness measure for a mapping between two Riemannian manifolds, to define the local extrinsic curvature measure as follows:

$$\text{Extrinsic Curvature}(z) := \frac{1}{2}\mathrm{Tr}\Big(\sum_{i,j=1}^{m}(G^{-1})_{ij}\Big(\frac{\partial}{\partial z^i}T\Big)^T\frac{\partial}{\partial z^j}T\Big). \tag{7}$$

This measure is coordinate-invariant; the proof is given in Appendix A.

3. Minimum Curvature Deep Generative Models

In this section we propose a curvature minimization framework for deep generative models. We first develop efficient methods for estimating the local intrinsic and extrinsic curvature measures in Section 3.1 and Section 3.2. Finally, we explain how our local curvature measures can be used in deep generative models in Section 3.3.

Throughout we will use Hutchinson's stochastic trace estimator (Hutchinson, 1989) to simplify the above cur-

[1]This is an implicit parametrization of the Grassmann manifold viewed as being embedded in Euclidean space $\mathbb{R}^{D\times D}$.

vature measures: for an $n \times n$ matrix A, $\mathrm{Tr}(A) = \mathbb{E}_{v\sim\mathcal{N}(0,I_n)}[v^T A v]$, where I_n denotes the $n \times n$ identity matrix. For the purposes of this paper, unless otherwise specified we set the number of samples from $\mathcal{N}(0, I_n)$ in the trace estimation to be always 1 in the training phase, i.e., $\mathrm{Tr}(A) \approx v^T A v$ for $v \in \mathcal{N}(0, I_n)$. We use the Einstein summation notation (Einstein, 1922), i.e., when an index variable appears twice in a single term, it implies the summation of that term over all the values of the index, e.g., $v_i^i = \sum_i v_i^i$. Finally, we use the shorthand notation $\partial_i := \frac{\partial}{\partial z^i}$.

3.1. Intrinsic Curvature Approximation

Using Hutchinson's trace estimator, we can estimate the local intrinsic curvature in (3) as $(v^T G^{-1}(z)\mathrm{Ric}(z)v)^2$ for $v \in \mathcal{N}(0, I_m)$. Denoting $G^{-1}(z)v$ by $\tilde{v}$, the square root of the estimate is then

$$\tilde{v}^T \mathrm{Ric}(z) v = (\partial_a \Gamma^a_{ij} - \partial_j \Gamma^a_{ai} + \Gamma^a_{ab}\Gamma^b_{ij} - \Gamma^a_{ib}\Gamma^b_{aj})\tilde{v}^i v^j. \quad (8)$$

Computing the Christoffel symbols Γ^i_{jk} for all i, j, k, and using these to compute (8) is practically infeasible due to memory limitations. Instead, approximate formulas for the four terms in (8) are derived that can be computed using the Jacobian-vector and vector-Jacobian products. We assume that $\partial\tilde{v} = \partial(G^{-1}v)$ is small enough to ignore and derive a more compact expression ($\partial G^{-1} = -G^{-1}\partial G G^{-1}$, and $\|G^{-1}\|$ is often very small as the norm of J_f is typically large for high-dimensional data). Our later experiments show that the approximate intrinsic curvature with this assumption can effectively reduce the actual intrinsic curvature (Figure 4 and 10). For space limitations, we derive the approximate formula only for the first term in (8) only; formulas for the remaining terms are given in Appendix B.

Substituting (2) into the first term of (8), we get

$$\frac{1}{2}\partial_a\big(g^{a\lambda}(v^j\partial_j(g_{\lambda i}\tilde{v}^i) + \tilde{v}^i\partial_i(g_{\lambda j}v^j) - \partial_\lambda(\tilde{v}^i g_{ij} v^j))\big). \quad (9)$$

Since $\tilde{v}^i g_{ij} = v_i$, the first and third terms in (9) vanish. The first term in (8) then simplifies to

$$\partial_a \Gamma^a_{ij}\tilde{v}^i v^j = \frac{1}{2}\mathrm{Tr}\big(\nabla(G^{-1}(\tilde{v}\cdot\nabla)(Gv))\big). \quad (10)$$

Again using Hutchinson's trace estimator,

$$\frac{1}{2}\mathrm{Tr}\big(\nabla(G^{-1}(\tilde{v}\cdot\nabla)(Gv))\big) \approx \frac{1}{2}w^T\big((w\cdot\nabla)(G^{-1}(\tilde{v}\cdot\nabla)(Gv))\big) \quad (11)$$

for $w \in \mathcal{N}(0, I_m)$. The final approximate formula can be computed by repeatedly using the Jacobian-vector and vector-Jacobian products. The remaining terms in (8) can be computed similarly; see Appendix B. The full expression for the estimated local intrinsic curvature is given as follows:

$$\begin{aligned}
&\Big(\frac{1}{2}(w\cdot\nabla)(w^T G^{-2}(v\cdot\nabla)(Gv)) \\
&- \frac{1}{2}(v\cdot\nabla)(w^T G^{-2}(v\cdot\nabla)(Gw)) \\
&+ \frac{1}{4}(w^T G^{-3}(v\cdot\nabla)(G)(v\cdot\nabla)(Gw)) \\
&- \frac{1}{4}(w^T G^{-2}(v\cdot\nabla)(G)G^{-1}(v\cdot\nabla)(Gw)) \\
&- \frac{1}{4}(w^T G^{-2}(v\cdot\nabla)(G)G^{-1}(w\cdot\nabla)(Gv)) \\
&+ \frac{1}{4}(w^T G^{-1}(v\cdot\nabla)(G)G^{-2}(w\cdot\nabla)(Gv))\Big)^2, \quad (12)
\end{aligned}$$

where $v, w \in \mathcal{N}(0, I)$ and $G = J_f^T J_f$; it can be computed by using the Jacobian-vector and vector-Jacobian products multiple times.

3.2. Extrinsic Curvature Approximation

Using Hutchinson's trace estimator, we can estimate the local extrinsic curvature in (7) as

$$\begin{aligned}
\frac{1}{2}v^T g^{ij}\partial_i T\partial_j T v &= \frac{1}{2}g^{ij}\partial_i(v_k T^k_l)\partial_j(T^l_m v^m) \\
&= \frac{1}{2}\mathrm{Tr}((\nabla(Tv))^T\nabla(Tv)G^{-1}). \quad (13)
\end{aligned}$$

for $v \in \mathcal{N}(0, I_D)$. Again via Hutchinson's trace estimator, we get

$$\frac{1}{2}w^T(\nabla(Tv))^T\nabla(Tv)G^{-1}w = \frac{1}{2}\big((w\cdot\nabla)(Tv)\big)^T(\tilde{w}\cdot\nabla)(Tv), \quad (14)$$

for $w \in \mathcal{N}(0, I_m)$ and $\tilde{w} = G^{-1}w$; we can compute the final approximate formula via repeated use of the Jacobian-vector and vector-Jacobian products.

3.3. Curvature Minimization Framework

In this section, we describe our curvature minimization framework for deep generative model learning. Essentially, we integrate the local curvature measures from Section 3.1, 3.2 to define global curvature measures which are then augmented to each of the original loss functions used in existing deep manifold learning methods. It is not necessary to integrate these measures over the entire latent space $\mathbb{R}^m$. Rather, a probability density in the latent space $p(z)$ is assumed available. For GAN (Goodfellow et al., 2014) and manifold flow (Brehmer & Cranmer, 2020), the sampling distribution is defined in the latent space. For autoencoders (Kramer, 1991; Kingma & Welling, 2013), given an encoder $g : \mathbb{R}^D \to \mathbb{R}^m$, the pushforward of the data distribution by g is defined in the latent space (i.e., as the aggregated posterior). For the autodecoder (Bojanowski et al., 2017), there exists a set of latent vectors simultaneously optimized, which can then be used to construct $p(z)$.

We define the global curvature measure as an expectation of the local curvature measure over $p(z)$, which is then multiplied by the weight parameter α and added to the original loss. Further algorithmic details can be found in Appendix C.

4. Experiments

In this section, with applications of our minimum curvature framework to autoencoders, we implement Minimum Intrinsic Curvature Autoencoder (MICAE) and Minimum Extrinsic Curvature Autoencoder (MECAE). Our minimum curvature autoencoders are compared to the existing methods: vanilla Autoencoder (AE) (Kramer, 1991), Variational Autoencoder (VAE) (Kingma & Welling, 2013), Contractive Autoencoder (CAE) (Rifai et al., 2011a), De-noising Autoencoder (DAE) (Vincent et al., 2010), Reconstruction Contractive Autoencoder (RCAE) (Alain & Bengio, 2014), Isometrically Regularized Autoencoder (IRAE) (Lee et al., 2022b), and Neighborhood Reconstructing Autoencoder (NRAE) (Lee et al., 2021).

Throughout, we assume that clean data is not available during training. Among models trained with various hyperparameters, we select the one with the lowest mean validation reconstruction error. In this selection process we use clean validation data since the ability to reconstruct corrupted data well is not a desirable property of good models.

In section 4.1, with a synthetic two-dimensional manifold example, we empirically show that, as the level of noise added to the training data increases, AE tends to learn more curved manifolds in both intrinsic and extrinsic senses, and minimizing curvatures with the proposed methods helps to learn more accurate manifolds.

In section 4.2, we train ours and existing autoencoders with human skeleton pose data corrupted by Gaussian noise and compare the manifold learning performance. To evaluate quantitatively, we compare two mean test reconstruction errors. First, *clean2clean* measures the mean reconstruction error of the clean test data. Secondly, *corrupt2clean* measures the mean square error between the reconstruction of the corrupted test data and clean test data. Experimental details not visible in the main script (e.g., network architectures) and comparisons of computational costs can be found in Appendix D.

4.1. Synthetic Data

Consider a ground truth two-dimensional manifold embedded in $\mathbb{R}^3$, that is $\mathcal{M} := \{(x, y, x^2 + y^2) \mid x, y \in (-1, 1)\}$; see Figure 4 (*Upper Left*). We sample 200 random points in (x, y)-space, map them to $\mathbb{R}^3$, and add Gaussian noise to construct the training data set. We use three-layer fully connected neural networks (512 nodes per layer) for both encoder and decoder with ELU activation functions, and the latent space dimension is 2. The test data set is constructed with the 100×100 two-dimensional mesh grid in (x, y)-space (no noise added).

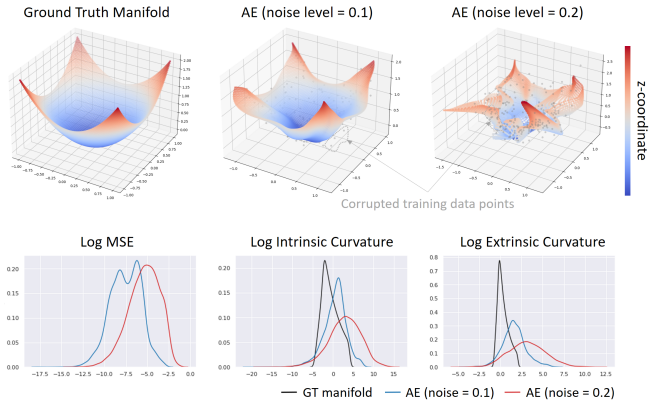

Figure 4. Upper: Ground truth manifold, and learned manifolds by AE given noisy training data sets (the noise level means the standard deviation). *Lower*: Density plots for the log MSE, Log Intrinsic Curvature, and Log Extrinsic Curvature. Log MSEs for the GT manifold are always $-\infty$.

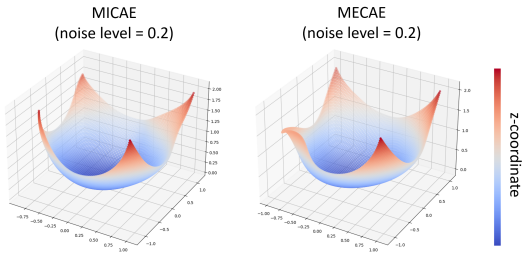

Figure 5. Learned manifolds by minimum curvature autoencoders given noisy training data sets.

Figure 4 demonstrates how AE learns manifolds as the level of noise added to the training data increases. As the noise level increases, AE learns manifolds with higher intrinsic and extrinsic curvatures, as shown in the log curvature plots, that are less accurate, as shown in the log MSE density plot (the lower, the more accurate).

Figure 5 shows the manifold learning results of the proposed minimum curvature autoencoders trained with the same training data as in Figure 4 (*Upper Right*) (the noise level is 0.2), and the density plots compared to the vanilla AE. Overall, our methods learn flatter and more accurate manifolds than AE.

Figure 6 shows how the learned manifold varies as the regularization coefficient increases. In both MICAE and MECAE, if the regularization coefficient is too small, the manifold is still overfitting to noise. If it is too large, the curvature is excessively reduced, and the manifold becomes inaccurate. We should choose an appropriate level

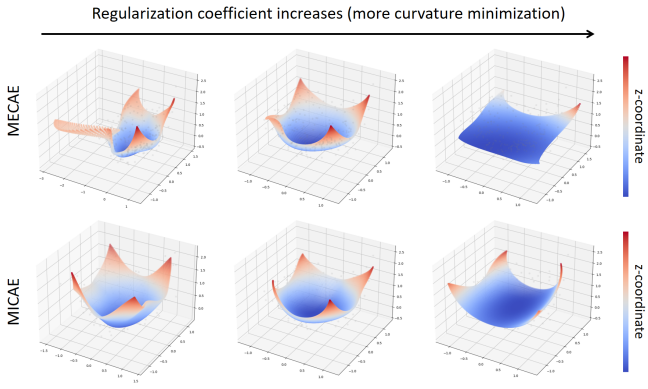

Figure 6. Illustration of how the learned manifold varies as the regularization coefficient increases. *Upper*: MECAE. *Lower*: MICAE.

Figure 7. One-dimensional manifold learning results.

of regularization coefficient. As the regularization coefficient increases, manifolds learned by MECAE and MICAE converge differently. While the MECAE learns a plane-like manifold that is extrinsically flat, the MICAE learns a cylinder-like manifold that is intrinsically flat.

Figure 7 shows one-dimensional manifold learning results trained with the noise-corrupted data, where the ground truth manifold is a sin-curve manifold. Since the intrinsic curvature of a curve is always zero, the intrinsic curvature minimization in MICAE does not affect manifold learning. On the other hand, MECAE learns a smoother and more accurate manifold.

Comparison to First-Order Distortion Minimization: Isometrically Regularized Autoencoder (IRAE) (Lee et al., 2022b), which is a first-order distortion minimization framework, attempts to make the pullback Riemannian metric $G(z) = J_f^T(z)J_f(z)$ be proportional to the identity, i.e., $G(z) = cI$ for some $c > 0$ and for all $z \in \mathcal{Z}$. In other words, the generator f is regularized to be a scaled isometry that preserves angles and scaled distances between the Euclidean latent space and the learned data manifold. Interestingly, this distortion minimization approach has a close relation to intrinsic curvature minimization.

According to Gauss's Theorema Egregium (i.e., Gauss's Remarkable Theorem), the Gaussian curvature of a surface is invariant under local isometry. In other words, if two surfaces or manifolds are mapped to each other without distortion, then their Gaussian curvatures or intrinsic curvatures should be preserved. Therefore, if f is a scaled isometry, then since the intrinsic curvature of the Euclidean latent space is everywhere 0, the resulting manifold's intrinsic curvature must be 0 everywhere as well. As a result, in IRAE, intrinsic curvatures are implicitly minimized as a byproduct of distortion minimization, whereas MICAE directly minimizes intrinsic curvatures. In the following, we compare IRAE and MICAE with the above synthetic manifold example with a noise level of 0.2.

Figure 8 shows that MICAE learns a more accurate manifold than IRAE. A potential explanation for this is that IRAE has an extra requirement of obtaining undistorted coordinates in addition to minimizing intrinsic curvature, which can make the task of the generator network more challenging compared to that of the MICAE. Meanwhile, we found both methods converge to cylinder-like manifolds as the regularization coefficient or weight parameter α increases. When the level of noise is low (≤ 0.1), the two methods did not show a significant difference.

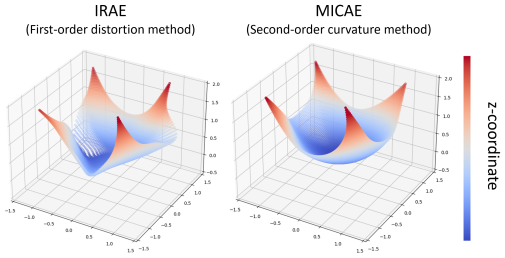

Figure 8. Manifold learning results of the first-order distortion minimization method, IRAE, and second-order (intrinsic) curvature minimization method, MICAE, for noise level 0.2.

4.2. Human Skeleton Data

In this section, we evaluate our minimum curvature autoencoders with the human skeleton pose data adopted from the NTU RGB+D dataset (Shahroudy et al., 2016). A human pose skeleton data consists of 25 three-dimensional key points and thus is considered a 75-dimensional vector. There are 60 action classes (e.g., drinking water, brushing teeth), and each action data consists of a sequence of skeleton poses. We use randomly selected 800, 200, and 9000 skeleton poses for each action class as training, validation, and test data. We add Gaussian noises of standard deviations 0.05 and 0.1 to the training data. We use two-layer fully connected neural networks (512 nodes per layer) for both encoder and decoder with ELU activation functions, and the latent space dimension is 8.

Table 1 shows the averages, and standard errors of the clean and corrupted test data set reconstruction MSEs over 60 different action classes, the lower, the better. IRAE produces the lowest reconstruction error when the noise level is 0.05, and MICAE produces the lowest reconstruction error when

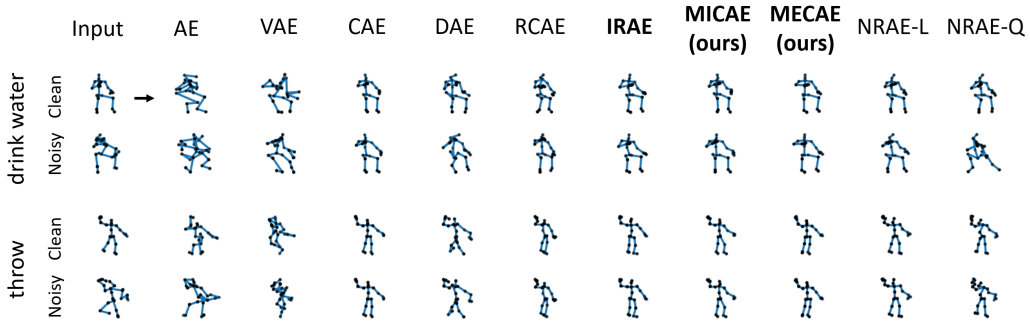

Figure 9. Clean and corrupted test data reconstruction results. Exemplar skeletons are selected from the "drink water" and "throw" action classes (noise level is 0.1). Methods that show qualitatively best reconstruction results are marked in bold (IRAE, MICAE, MECAE).

Table 1. Averages and standard errors of the skeleton test data set reconstruction MSEs with Gaussian noise of standard deviations 0.05, 0.1, the lower, the better. The best results are marked in red. Those that show comparable performance to the best results are marked in bold (if the mean errors are included in the error bars of the best results). The numbers are written in units of 10^{-3}.

Noise	0.05		0.1	
Metric	*clean2clean*	*corrupt2clean*	*clean2clean*	*corrupt2clean*
graph-free				
AE	7.52 ± 0.18	9.03 ± 0.16	15.4 ± 0.18	19.4 ± 0.18
VAE	5.56 ± 0.18	6.74 ± 0.18	13.6 ± 0.24	19.1 ± 0.26
CAE	2.32 ± 0.08	2.60 ± 0.08	2.48 ± 0.08	3.54 ± 0.08
DAE	3.09 ± 0.12	3.26 ± 0.12	5.55 ± 0.17	6.48 ± 0.18
RCAE	3.77 ± 0.14	3.99 ± 0.14	6.51 ± 0.17	7.17 ± 0.19
IRAE	**2.18 ± 0.08**	**2.46 ± 0.08**	**2.44 ± 0.09**	3.55 ± 0.09
MICAE	**2.20 ± 0.08**	**2.48 ± 0.08**	**2.37 ± 0.08**	**3.45 ± 0.08**
MECAE	2.29 ± 0.08	2.56 ± 0.08	**2.40 ± 0.08**	**3.46 ± 0.08**
graph-based				
NRAE-L	**2.19 ± 0.08**	**2.46 ± 0.08**	2.73 ± 0.08	3.79 ± 0.08
NRAE-Q	2.51 ± 0.10	2.92 ± 0.11	4.00 ± 0.11	5.93 ± 0.12

the noise level is 0.1. Overall, our minimum curvature autoencoders produce lower reconstruction errors, where MICAE slightly outperforms MECAE. Figure 9 shows some example reconstruction results, where IRAE, MICAE, and MECAE seem to be most effective at mitigating noises and producing accurate skeletons, followed by CAE and NRAEs.

Figure 10 shows the Log Intrinsic Curvature and Log Extrinsic Curvature density plots of the learned manifolds. We train autoencoders with the corrupted training data from the "throw" action class (noise level is 0.1) and compute the density plots with the test data set. As expected, MECAE produces the smallest extrinsic curvature and MICAE produces the smallest intrinsic curvature. One interesting observation is that other regularization methods, IRAE, and NRAE-L, which show some decent performance in mitigating the effects of noise, also reduce the curvature of the manifold, although their loss functions do not explicitly involve the curvature terms.

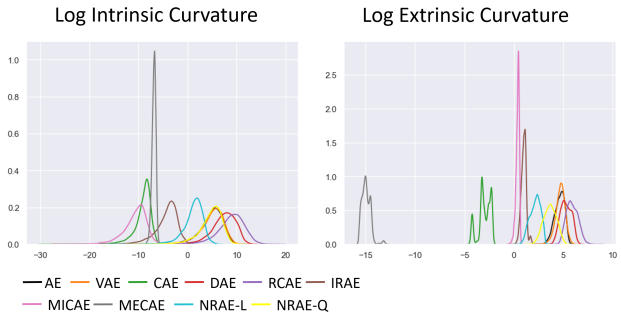

Figure 10. Density plots for the Log Intrinsic Curvature and Log Extrinsic Curvature, where the autoencoders are trained with data from the "throw" action class (noise level is 0.1).

5. Conclusions

We have proposed a class of curvature-based regularization terms for deep generative model learning. Formulas for measuring both intrinsic and extrinsic curvature, which are independent of the choice of coordinates, have been developed for data manifolds of arbitrary dimensions embedded in higher-dimensional Euclidean space. Since the formulas require a lot of computations as they require the evaluation of second-order derivatives, we have developed efficient ways to approximate these curvatures. Experiments that include noisy motion capture data have shown that curvature-based regularization methods are more effective for noise-robust manifold learning than the existing autoencoder methods, where using intrinsic curvature measures slightly outperforms using extrinsic curvature measures.

Limitations and future research directions: While approximate formulas allow for the computation of intrinsic and extrinsic curvature terms, their calculations can still be quite time-consuming. Specifically, computing the inverse of the pullback Riemannian metric $G(z)^{-1}$ requires a significant amount of time and becomes even more challenging with increasing latent space dimensionality. Therefore, it is important to develop methods for even more approximat-

ing curvature computations that can drastically reduce the computation time, thus enabling their use in data and latent spaces of much larger dimensions.

While we have assumed the ambient data space is Euclidean, a growing number of problems involve non-Euclidean data (e.g., diffusion tensor data (Fletcher & Joshi, 2007), point cloud data (Lee et al., 2022a)). To develop minimum curvature deep generative models for non-Euclidean data, we need to generalize the intrinsic and extrinsic curvature measures for manifolds embedded in higher-dimensional Riemannian manifolds.

Acknowledgements

Yonghyeon Lee is the beneficiary of an individual grant from CAINS supported by a KIAS Individual Grant (AP092701) via the Center for AI and Natural Sciences at Korea Institute for Advanced Study. Frank C. Park is supported in part by NRF-MSIT grant RS-2023-00208052, IITP-MSIT grant 2021-0-02068 (SNU AI Innovation Hub), KIAT-MOTIE grant P0020536 (HRD Program for Industrial Innovation), IITP-MSIT grant 2022-0-00480 (Training and Inference Methods for Goal-Oriented AI Agents), SNU-AIIS, SNU-IAMD, SNU BK21+ Program in Mechanical Engineering, SNU Institute for Engineering Research.

References

Alain, G. and Bengio, Y. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

Bendokat, T., Zimmermann, R., and Absil, P.-A. A grassmann manifold handbook: Basic geometry and computational aspects. *arXiv preprint arXiv:2011.13699*, 2020.

Besse, A. L. *Einstein manifolds*. Springer Science & Business Media, 2007.

Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

Brehmer, J. and Cranmer, K. Flows for simultaneous manifold learning and density estimation. In *Advances in neural information processing systems*, 2020.

Chen, N., Klushyn, A., Ferroni, F., Bayer, J., and Van Der Smagt, P. Learning flat latent manifolds with vaes. *arXiv preprint arXiv:2002.04881*, 2020.

Do Carmo, M. P. *Riemannian geometry*, volume 6. Springer, 1992.

Do Carmo, M. P. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.

Eells, J. and Lemaire, L. A report on harmonic maps. *Bulletin of the London mathematical society*, 10(1):1–68, 1978.

Eells, J. and Lemaire, L. Another report on harmonic maps. *Bulletin of the London Mathematical Society*, 20(5):385–524, 1988.

Einstein, A. The general theory of relativity. In *The Meaning of Relativity*, pp. 54–75. Springer, 1922.

Fecko, M. *Differential geometry and Lie groups for physicists*. Cambridge university press, 2006.

Fletcher, P. T. and Joshi, S. Riemannian geometry for the statistical analysis of diffusion tensor data. *Signal Processing*, 87(2):250–262, 2007.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Jang, C. *Riemannian Distortion Measures for Non-Euclidean Data*. PhD thesis, Seoul National University Graduate School, 2019.

Jang, C., Noh, Y.-K., and Park, F. C. A riemannian geometric framework for manifold learning of non-euclidean data. *Advances in Data Analysis and Classification*, pp. 1–27, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

Kühnel, W. *Differential geometry*, volume 77. American Mathematical Soc., 2015.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Lee, Y., Kwon, H., and Park, F. Neighborhood reconstructing autoencoders. *Advances in Neural Information Processing Systems*, 34:536–546, 2021.

Lee, Y., Kim, S., Choi, J., and Park, F. A statistical manifold framework for point cloud data. In *International Conference on Machine Learning*, pp. 12378–12402. PMLR, 2022a.

Lee, Y., Yoon, S., Son, M., and Park, F. C. Regularized autoencoders for isometric representation learning. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=mQxt8l7JL04.

Ricci-Curbastro, G. Direzioni et invarianti principali in una varieta qualunque. *Atti del Real Inst. Venezio*, 63: 1233–1239, 1904.

Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., and Glorot, X. Higher order contractive auto-encoder. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 645–660. Springer, 2011a.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *Icml*, 2011b.

Schutz, B. *A first course in general relativity*. Cambridge university press, 2022.

Shahroudy, A., Liu, J., Ng, T.-T., and Wang, G. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1010–1019, 2016.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Wong, Y.-C. Differential geometry of grassmann manifolds. *Proceedings of the National Academy of Sciences*, 57(3): 589–594, 1967.

Appendix

A. Coordinate-Invariance of the Extrinsic Curvature Measure

In this section, we show that (i) the representation of $T_x\mathcal{M}$, $E(J_f) = J_f(J_f^TJ_f)^{-1}J_f^T$, is coordinate-invariant and (ii) the local extrinsic curvature measure, $\frac{1}{2}\text{Tr}(\sum_{i,j=1}^m (G^{-1})_{ij}\big(\frac{\partial}{\partial z^i}T\big)^T\frac{\partial}{\partial z^j}T)$, is coordinate-invariant. Given a decoder $f : z \in \mathbb{R}^m \mapsto x \in \mathcal{M}$, consider a coordinate transformation $h : z \in \mathbb{R}^m \mapsto z' \in \mathbb{R}^m$ and the transformed decoder $f' := f \circ h^{-1} : z' \in \mathbb{R}^m \mapsto x \in \mathcal{M}$. The Jacobian of f, J_f, is transformed as follows:

$$J_f \mapsto J_{f'} = \frac{\partial f'}{\partial z'} = \frac{\partial f}{\partial z}\frac{\partial h^{-1}}{\partial z'} = J_f H, \tag{15}$$

where $H = \frac{\partial h^{-1}}{\partial z'}$ is an $m \times m$ invertible matrix at z. Therefore the representation $E(J_f)$ is coordinate-invariant:

$$E(J_f) \mapsto E(J_{f'}) = J_{f'}(J_{f'}^TJ_{f'})^{-1}J_{f'}^T = J_fH(H^TJ_f^TJ_fH)^{-1}H^TJ_f^T = E(J_f). \tag{16}$$

Next, to show the coordinate-invariance of the extrinsic curvature, firstly, investigate how the Riemannian metric G is transformed:

$$G \mapsto G' = J_{f'}^TJ_{f'} = H^TJ_f^TJ_fH = H^TGH. \tag{17}$$

Since $T' = T$ and $\frac{\partial}{\partial z'^i}T = \sum_{a=1}^m H_{ai}\frac{\partial}{\partial z^a}T$, the curvature measure is coordinate-invariant:

$$\begin{aligned}
\text{Tr}\Big(\sum_{i,j=1}^m (G^{-1})_{ij}\big(\frac{\partial}{\partial z^i}T\big)^T\frac{\partial}{\partial z^j}T\Big) &\mapsto \text{Tr}\Big(\sum_{i,j=1}^m (G'^{-1})_{ij}\big(\frac{\partial}{\partial z'^i}T'\big)^T\frac{\partial}{\partial z'^j}T'\Big) \\
&= \text{Tr}\Big(\sum_{i,j=1}^m (H^TGH)^{-1}_{ij}\big(\sum_{a=1}^m H_{ai}\frac{\partial}{\partial z^a}T\big)^T\sum_{b=1}^m H_{bj}\frac{\partial}{\partial z^j}T\Big) \\
&= \text{Tr}\Big(\sum_{i,j,k,l=1}^m (H^{-1})_{ik}(G^{-1})_{kl}(H^{-T})_{lj}\big(\sum_{a=1}^m H_{ai}\frac{\partial}{\partial z^a}T\big)^T\sum_{b=1}^m H_{bj}\frac{\partial}{\partial z^b}T\Big) \\
&= \text{Tr}\Big(\sum_{i,j,k,l,a,b=1}^m (H^{-1})_{ik}(H^{-T})_{lj}H_{ai}H_{bj}(G^{-1})_{kl}\big(\frac{\partial}{\partial z^a}T\big)^T\frac{\partial}{\partial z^b}T\Big) \\
&= \text{Tr}\Big(\sum_{k,l,a,b=1}^m \delta_{ak}\delta_{bl}(G^{-1})_{kl}\big(\frac{\partial}{\partial z^a}T\big)^T\frac{\partial}{\partial z^b}T\Big) \\
&= \text{Tr}\Big(\sum_{a,b=1}^m (G^{-1})_{ab}\big(\frac{\partial}{\partial z^a}T\big)^T\frac{\partial}{\partial z^b}T\Big),
\end{aligned} \tag{18}$$

where δ_{ij} is the Kronecker delta symbol.

B. Local Intrinsic Curvature Estimation

In this section, we provide simplified estimation formulas for the second, third, and fourth terms in (8), and summarize the strategy for computing (3).

Second Term: Plugging (2) in the second term of (8), we get

$$\partial_j \Gamma^a_{ai} \tilde{v}^i v^j = \frac{1}{2}\partial_j \big(g^{a\lambda}(\tilde{v}^i \partial_i (g_{\lambda a}) v^j + \partial_a (g_{\lambda i}\tilde{v}^i) v^j - \partial_\lambda (g_{ai}\tilde{v}^i) v^j \big). \tag{19}$$

Since $\tilde{v}^i g_{ij} = v_i$, the second and third terms vanish, and then the second term of (8) is simplified to

$$\partial_j \Gamma^a_{ai} \tilde{v}^i v^j = \frac{1}{2} v^j \partial_j \big(g^{a\lambda}(\tilde{v}^i \partial_i (g_{\lambda a})) = \frac{1}{2} v^j \partial_j \mathrm{Tr}\big(G^{-1}\tilde{v}^i \partial_i G\big). \tag{20}$$

Using Hutchinson's trace estimator again,

$$\frac{1}{2} v^j \partial_j \mathrm{Tr}\big(G^{-1}\tilde{v}^i \partial_i G\big) \approx \frac{1}{2}(v\cdot\nabla)(w^T G^{-1}(\tilde{v}\cdot\nabla)(Gw)), \tag{21}$$

for $w \in \mathcal{N}(0, I)$.

Third Term: Plugging (2) in the third term of (8), we get

$$\Gamma^a_{ab}\Gamma^b_{ij}\tilde{v}^i v^j = \frac{1}{2} g^{a\lambda}(\partial_b g_{a\lambda} + \partial_a g_{b\lambda} - \partial_\lambda g_{ab}) \times \frac{1}{2} g^{b\gamma}(v^j \partial_j (\tilde{v}^i g_{i\gamma}) + \tilde{v}^i \partial_i (v^j g_{j\gamma}) - \partial_\gamma (\tilde{v}^i g_{ij} v^j)). \tag{22}$$

Since $\tilde{v}^i g_{ij} = v_i$, the first and third terms in the second bracket vanish, and then the third term of (8) is simplified to

$$\Gamma^a_{ab}\Gamma^b_{ij}\tilde{v}^i v^j = \frac{1}{4} g^{a\lambda}(\partial_b g_{a\lambda} + \partial_a g_{b\lambda} - \partial_\lambda g_{ab}) \times g^{b\gamma}(\tilde{v}^i \partial_i (v^j g_{j\gamma})). \tag{23}$$

To further simplify, we denote the term after the multiplication sign $g^{b\gamma}(\tilde{v}^i \partial_i (v^j g_{j\gamma}))$ by

$$V^b := g^{b\gamma}(\tilde{v}^i \partial_i (v^j g_{j\gamma})) = \big(G^{-1}(\tilde{v}\cdot\nabla)(Gv)\big)^b. \tag{24}$$

Then the third term is simplified to

$$\begin{aligned}\Gamma^a_{ab}\Gamma^b_{ij}\tilde{v}^i v^j &= \frac{1}{4}\big(g^{a\lambda}V^b \partial_b g_{a\lambda} + g^{a\lambda}\partial_a V^b g_{b\lambda} - g^{a\lambda}\partial_\lambda g_{ab} V^b\big) \\ &= \frac{1}{4}\big(\mathrm{Tr}(G^{-1}(V\cdot\nabla)(G))\big),\end{aligned} \tag{25}$$

since the second and third terms cancel each other out. Using Hutchinson's trace estimator again,

$$\frac{1}{4}\big(\mathrm{Tr}(G^{-1}(V\cdot\nabla)(G))\big) \approx \frac{1}{4}\big(w^T G^{-1}(V\cdot\nabla)(Gw)\big), \tag{26}$$

for $w \in \mathcal{N}(0, I_m)$.

Fourth Term: Plugging (2) in the fourth term of (8), we get

$$\Gamma^a_{ib}\Gamma^b_{aj}\tilde{v}^i v^j = \frac{1}{2}\big(g^{a\lambda}(\tilde{v}^i \partial_i g_{\lambda b} + \partial_b (g_{\lambda i}\tilde{v}^i) - \partial_\lambda (g_{bi}\tilde{v}^i))\big) \times \frac{1}{2}\big(g^{b\gamma}(v^j \partial_j g_{\gamma a} + \partial_a (g_{\gamma j} v^j) - \partial_\gamma (g_{aj} v^j))\big). \tag{27}$$

Since $\tilde{v}^i g_{ij} = v_i$, the second and third terms in the first bracket vanish, and then the fourth term of (8) is simplified to

$$\Gamma^a_{ib}\Gamma^b_{aj}\tilde{v}^i v^j = \frac{1}{4}\big(g^{a\lambda}\tilde{v}^i \partial_i g_{\lambda b}\big) \times \big(g^{b\gamma}(v^j \partial_j g_{\gamma a} + \partial_a (g_{\gamma j} v^j) - \partial_\gamma (g_{aj} v^j))\big). \tag{28}$$

To further simplify, we denote the term before the multiplication sign $g^{a\lambda}(\tilde{v}^i \partial_i (g_{\lambda b}))$ by

$$W^a_b := g^{a\lambda}(\tilde{v}^i \partial_i (g_{\lambda b})) = \big(G^{-1}(\tilde{v}\cdot\nabla)(G)\big)^a{}_b. \tag{29}$$

We note that $V = Wv$. Then the fourth term is simplified to

$$\Gamma^a_{ib}\Gamma^b_{aj}\tilde{v}^i v^j = \frac{1}{4}\big(W^a_b g^{b\gamma}(v^j\partial_j g_{\gamma a} + \partial_a(g_{\gamma j}v^j) - \partial_\gamma(g_{aj}v^j))\big) \tag{30}$$

$$= \frac{1}{4}\big(\text{Tr}(WG^{-1}(v\cdot\nabla)(G)) + \text{Tr}(WG^{-1}\nabla(Gv) - \text{Tr}(G^{-1}W^T\nabla(Gv))\big). \tag{31}$$

Using Hutchinson's trace estimator again,

$$\Gamma^a_{ib}\Gamma^b_{aj}\tilde{v}^i v^j = \frac{1}{4}\big(w^T WG^{-1}(v\cdot\nabla)(Gw) + w^T(WG^{-1} - G^{-1}W^T)(w\cdot\nabla)(Gv)\big). \tag{32}$$

for $w \in \mathcal{N}(0, I_m)$.

Combining all four terms, finally we get the following local intrinsic curvature estimation formula:

$$\begin{aligned}
\text{EIC}(z; f) = \Big(&\frac{1}{2}(w\cdot\nabla)(w^T G^{-2}(v\cdot\nabla)(Gv)) \\
&- \frac{1}{2}(v\cdot\nabla)(w^T G^{-2}(v\cdot\nabla)(Gw)) \\
&+ \frac{1}{4}(w^T G^{-3}(v\cdot\nabla)(G)(v\cdot\nabla)(Gw)) \\
&- \frac{1}{4}(w^T G^{-2}(v\cdot\nabla)(G)G^{-1}(v\cdot\nabla)(Gw)) \\
&- \frac{1}{4}(w^T G^{-2}(v\cdot\nabla)(G)G^{-1}(w\cdot\nabla)(Gv)) \\
&+ \frac{1}{4}(w^T G^{-1}(v\cdot\nabla)(G)G^{-2}(w\cdot\nabla)(Gv))\Big)^2,
\end{aligned} \tag{33}$$

where $v, w \in \mathcal{N}(0, I)$ and $G = J_f^T J_f$; it can be computed by using the Jacobian-vector and vector-Jacobian products multiple times.

C. Algorithmic Details

In this section, we describe algorithms for MICAE and MECAE. Given a decoder $f : \mathbb{R}^m \to \mathbb{R}^D$, at a latent point $z \in \mathbb{R}^m$, denote the estimated intrinsic curvature by $\mathrm{EIC}(z; f)$ (33) and the estimated extrinsic curvature by $\mathrm{EEC}(z; f)$ (14). Let $g_\phi : \mathbb{R}^D \to \mathbb{R}^m$ be a parameterized encoder and $f_\theta : \mathbb{R}^m \to \mathbb{R}^D$ be a parameterized decoder, where θ, ϕ are learnable parameters.

The training of MICAE and MECAE is as simple as the vanilla autoencoder, where the only difference is the added regularization term in the loss function. The new loss functions have the following form:

$$\mathbb{E}_{x\sim p_{\text{data}}}[\|f_\theta(g_\phi(x)) - x\|^2] + \alpha\, \mathbb{E}_{x\sim p_{\text{data}}}[\mathrm{EC}(g_\phi(x); f_\theta)], \tag{34}$$

where α is the regularization coefficient, p_{data} is the empirical training data distribution, and $\mathrm{EC}(z; f)$ is either $\mathrm{EIC}(z; f)$ or $\mathrm{EEC}(z; f)$. In practice, $\mathbb{E}_{x\sim p_{\text{data}}}$ is replaced by $\frac{1}{N}\sum_{i=1}^N$ with the set of training data $\{x_i\}_{i=1}^N$.

D. Experimental Details

Latent Space Augmentation: Sometimes, a simple latent space data augmentation technique can improve the manifold learning performance. Following (Lee et al., 2022b; Chen et al., 2020), we use the modified mix-up data-augmentation method with a parameter $\eta > 0$. The encoded data is augmented by $z = \delta z_1 + (1 - \delta)z_2$ where $z_i, i = 1, 2$ are randomly sampled two encoded data and δ is uniformly sampled from $(-\eta, 1 + \eta)$ (we set $\eta = 0.2$). Note that we use this technique for synthetic data.

D.1. Synthetic Data

For both two-dimensional and one-dimensional manifold examples, we use three-layer fully connected neural networks (512 nodes per layer) for both encoder and decoder with ELU activation functions. We use Adam optimizer with a learning rate of 0.00001, and the number of training epochs is 10000. The regularization coefficient for the minimum curvature autoencoders is searched in (0.0001, 0.001, 0.01, 0.1, 1, 10), and the one that produces the smallest (clean2clean) test reconstruction error is reported.

For the two-dimensional manifold example, when computing the intrinsic and extrinsic curvatures for the density plots, we did not use the curvature estimation formulas, rather compute them precisely using the original formulas, which is feasible since both the data and latent space dimension are low. Table 2 shows the runtimes per epoch with the two-dimensional manifold example for a batch size of 100.

Table 2. The runtimes per epoch for a batch size of 100 (where we use the GeForce RTX 3090).

AE	MICAE	MECAE
0.003 s	0.087 s	0.029 s

D.2. Motion Capture Data

We use two-layer fully connected neural networks (512 nodes per layer) for both encoder and decoder with ELU activation functions. We use Adam optimizer with a learning rate of 0.001, and the number of training epochs is 5000. We search the hyperparameters for each method over a wide enough range and report the results of the one that produces the smallest (clean2clean) validation reconstruction error.

When computing the intrinsic and extrinsic curvatures for the density plots, we did not use the curvature estimation formulas, rather compute them precisely using the original formulas, which is feasible since both the data and latent space dimension are low. Table 3 shows the runtimes per epoch for a batch size of 100.

Table 3. The runtimes per epoch for a batch size of 100 (where we use the GeForce RTX 3090).

AE	VAE	CAE	DAE	RCAE	IRAE	MICAE	MECAE	NRAE-L	NRAE-Q
0.0130 s	0.0198 s	0.0267 s	0.0133 s	0.0299 s	0.0353 s	0.4600 s	0.1523 s	0.0280 s	0.0459 s