A Statistical Manifold Framework for Point Cloud Data

Yonghyeon Lee [*1] **Seungyeon Kim** [*1] **Jinwon Choi** [2] **Frank C Park** [1 3]

Abstract

Many problems in machine learning involve data sets in which each data point is a point cloud in $\mathbb{R}^D$. A growing number of applications require a means of measuring not only distances between point clouds, but also angles, volumes, derivatives, and other more advanced concepts. To formulate and quantify these concepts in a coordinate-invariant way, we develop a Riemannian geometric framework for point cloud data. By interpreting each point in a point cloud as a sample drawn from some given underlying probability density, the space of point cloud data can be given the structure of a statistical manifold – each point on this manifold represents a point cloud – with the Fisher information metric acting as a natural Riemannian metric. Two autoencoder applications of our framework are presented: (i) smoothly deforming one 3D object into another via interpolation between the two corresponding point clouds; (ii) learning an optimal set of latent space coordinates for point cloud data that best preserves angles and distances, and thus produces a more discriminative representation space. Experiments with large-scale standard benchmark point cloud data show greatly improved classification accuracy vis-á-vis existing methods. Code is available at https://github.com/seungyeon-k/SMF-public.

1. Introduction

Many machine learning problems involve data sets in which each data point is a point cloud in $\mathbb{R}^D$. For example, to measure the similarity between two shapes, point cloud representations of the two shapes can be obtaind with a depth camera, and a distance metric on the space of point clouds, e.g., the Hausdorff distance, the chamfer distance (Hausdorff, 1914), or earth mover distance (Rubner et al., 2000) used to measure their similarity.

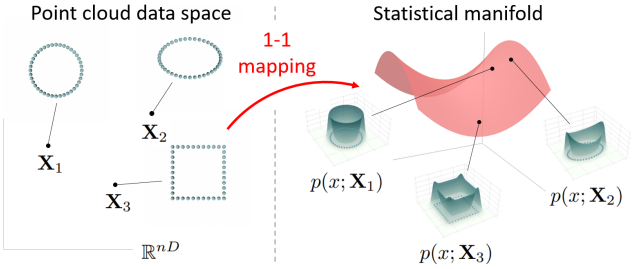

Figure 1: Illustration of statistical manifold obtained from the 1-1 mapping between the space of point cloud data and the space of probability density functions.

The distance metric measures just one aspect of point cloud data; other applications may require more advanced concepts and tools. For example, in the case of a moving point cloud, one may seek any number of quantities such as the velocity, acceleration, relative heading direction, or the swept volume. For multiple point cloud samples, one may seek a measure of their dispersion, e.g., the covariance or higher moments.

In fact, a growing number of applications involving point cloud data require a means of measuring not only distances, but also angles, volumes, derivatives, and other advanced geometric and analytical concepts. In principle one could choose some arbitrary coordinates to parametrize the space of point clouds, and extend the usual Euclidean notions of angles, volumes, and derivatives, but such an approach would not only be ad hoc, but likely not be invariant with respect to the choice of coordinates.

To formulate and quantify these concepts in a coordinate-invariant, geometrically meaningful way, as a first contribution we develop a *Riemannian geometric* framework for point cloud data. The key idea behind our approach draws upon information geometry (Amari & Nagaoka, 2000; Amari, 2016): by interpreting each point in a point cloud as a sample drawn from some known probability density, the space of point cloud data can be given the structure of a *statistical manifold* – each point on this manifold represents a point cloud – with the Fisher information metric acting

[*]Equal contribution [1]Department of Mechanical Engineering, Seoul National University, Seoul, South Korea [2]Kakao Enterprise, Seongnam, Kyonggi-do, South Korea [3]Saige Research, Seoul, South Korea. Correspondence to: Frank C Park <fcp@snu.ac.kr>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

as a natural Riemannian metric. Under some mild assumptions, i.e., the number of points in a point cloud is fixed, and all points are distinct, a one-to-one mapping between the space of point clouds and probability densities can be constructed. That is, a point cloud $\mathbf{X} = \{x_1, ..., x_N \,|\, x_i \in \mathbb{R}^D\}$ is mapped to a density function $p(x; \mathbf{X})$ on $\mathbb{R}^D$ in a 1-1 fashion as illustrated in Figure 1.

We remark that the idea of interpreting each point in a point cloud as a sample drawn from some given probability density is well-known, and has been applied to problems ranging from point set registration (Jian & Vemuri, 2005; Wang et al., 2006; Myronenko & Song, 2010; Hasanbelliu et al., 2014; Zhou et al., 2014; Min et al., 2018; Li et al., 2021) to point cloud de-noising (Zaman et al., 2017; Luo & Hu, 2021). These applications, however, only require a similarity measure between point clouds, typically formulated in terms of some divergence measure.

Two autoencoder applications of our framework are presented: (i) smoothly deforming one 3D object (a cylinder) into another (a cone), and (ii) learning an optimal set of latent space coordinates for point cloud data that best preserves distances and angles. In the former case, a pre-trained autoencoder is used to encode two 3D point clouds – one representing the cylinder, the other the cone – and the minimal geodesic with respect to the natural Riemannian metric is then constructed between these two objects. The shape evolution obtained for this iemannian metric is seen to be far more natural and intuitive than that obtained for the straight line interpolant in latent space.

In the second application, we use the statistical manifold framework to find a set of distortion minimizing latent space coordinates, in the sense that Euclidean straight lines in the latent space closely approximate minimal geodesics on the statistical manifold. Such a set of coordinates offers a more discriminative representation for the data manifold (Chen et al., 2020; Lee et al., 2022) that results in, e.g., higher linear SVM classification accuracy vis-á-vis existing state-of-the art methods. Experiments are carried out with both synthetic and standard benchmark datasets (*ShapeNet* (Chang et al., 2015), *ModelNet* (Wu et al., 2015)).

2. Statistical Manifold Framework for Point Cloud Data

We begin this section with some information geometric preliminaries. A *statistical manifold* is an infinite-dimensional Riemannian manifold each of whose points is a probability density, with the Fisher information metric acting as a natural Riemannian metric. Finite-dimensional statistical manifolds can be obtained by considering a family of parametric probability density functions:

Definition 2.1. Given an m-dimensional differentiable manifold Θ and a smooth 1-1 map from Θ to the space of probability density functions $\boldsymbol{\theta} \mapsto p(x; \boldsymbol{\theta})$, the image of this mapping, denoted $\mathcal{S} := \{p(x; \boldsymbol{\theta}) | \boldsymbol{\theta} \in \Theta\}$, is an m-dimensional statistical manifold.

Let $\theta = (\theta^1, ..., \theta^m) \in \mathbb{R}^m$ be local coordinates for $\boldsymbol{\theta} \in \Theta$, which by trivial extension also act as local coordinates for $\mathcal{S}$. Throughout we use italics to represent local coordinates, e.g., $\boldsymbol{\theta} \in \Theta$ has local coordinates $\theta \in \mathbb{R}^m$. In this coordinate system, elements g_{ij} of the Fisher information metric $G(\theta) \in \mathbb{R}^{m\times m}$ are given by

$$g_{ij}(\theta) := \int p(x;\theta) \frac{\partial \log p(x;\theta)}{\partial \theta^i} \frac{\partial \log p(x;\theta)}{\partial \theta^j} \, dx, \quad (1)$$

where $i, j = 1, \ldots, m$. The length of a curve on $\mathcal{S}$ parametrized by $\theta(t)$, $t \in [0, T]$, can then be computed as the integral $\int_0^T ds$, where the infinitesimal length ds on $\mathcal{S}$ is given by $ds^2 = d\theta^T G(\theta) d\theta$. Further details on statistical manifolds and the Fisher information metric can be found in, e.g., (Amari & Nagaoka, 2000; Efron & Hinkley, 1978; Rissanen, 1996; Han & Park, 2014).

With the above statistical manifold preliminaries, we now construct a Riemannian geometric structure for the space of point cloud data. Section 2.1 defines a statistical manifold from the point cloud data, while Section 2.2 uses the Fisher information metric to construct a Riemannian metric for point cloud data. To keep the definitions and ensuing results simple, we assume throughout that all point clouds consist of exactly n distinct points in $\mathbb{R}^D$, i.e., each point cloud is of the form

$$\mathbf{X} = \{x_1, ..., x_n \,|\, x_i \in \mathbb{R}^D, x_i \neq x_j \text{ if } i \neq j\}. \quad (2)$$

The set of all point clouds is denoted $\mathcal{X}$. Later we discuss methods for dealing with point clouds that do not satisfy these assumptions. Proofs of all propositions in this section can be found in Appendix B.

2.1. Statistical Manifold of Point Cloud Data

Given a point cloud $\mathbf{X}$, a parametric probability density function $p(x; \mathbf{X})$ can be defined in terms of a positive kernel function $\mathbf{X}$ itself as the parameter (Parzen, 1962; Davis et al., 2011):

Definition 2.2. Given a positive kernel function $K : \mathbb{R}^D \to \mathbb{R}$ such that $\int_{\mathbb{R}^D} K(u)\, du = 1$, and a $D \times D$ symmetric positive-definite matrix Σ (the bandwidth matrix), the kernel density estimate

$$p(x; \mathbf{X}) := \frac{1}{n\sqrt{|\Sigma|}} \sum_{i=1}^{n} K(\Sigma^{-\frac{1}{2}}(x - x_i)) \quad (3)$$

is said to be a statistical representation of the point cloud $\mathbf{X} \in \mathcal{X}$. The set of statistical representations is denoted $\mathcal{S} := \{p(x; \mathbf{X}) \,|\, \mathbf{X} \in \mathcal{X}\}$.

To ensure that $\mathcal{S}$ is a statistical manifold, recall from Definition 2.1 that the following two conditions need to be satisfied: (i) $\mathcal{X}$ is a differentiable manifold; (ii) A 1-1 mapping $h : \mathcal{X} \to \mathcal{S}$, $h(\mathbf{X}) = p(x; \mathbf{X})$ must be defined. The first condition can be satisfied with the "distinct points" assumption:

Proposition 2.3 (Corollary 2.2.11. in (Knudsen, 2018))**.** *The set of point clouds in which each point cloud is a set of n distinct points of dimension D, is an nD-dimensional differentiable manifold.* [1]

To satisfy the second condition, additional assumptions are needed. The following proposition provides a sufficient condition for h to be 1-1:

Proposition 2.4. *If the kernel function*

$$\Psi(x, y) = K(\Sigma^{-\frac{1}{2}}(x - y)) \tag{4}$$

is strictly positive-definite,[2] *then the mapping $h : \mathcal{X} \to \mathcal{S}$ given by*

$$h(\mathbf{X})(x) = \frac{1}{n\sqrt{|\Sigma|}} \sum_{i=1}^{n} K(\Sigma^{-\frac{1}{2}}(x - x_i)) \tag{5}$$

is 1-1.

The above proposition offers guidance on the choice of kernel in our statistical manifold framework. Throughout the remainder of the paper, we use the standard and widely used strictly positive-definite kernel function

$$K(u) = \frac{1}{\sqrt{(2\pi)^D}} \exp(-\frac{u^T u}{2}), \tag{6}$$

with the scaled identity bandwidth matrix, i.e., $\Sigma = \sigma^2 \mathbf{I}$. We note that other choices of kernel function are possible, e.g., the Laplacian kernel, or inverse multiquadratic kernel (Sriperumbudur et al., 2010).

From Propositions 2.3 and 2.4 we have established that, under the distinct points assumption and using the normal kernel function, the mapping $h : \mathcal{X} \to \mathcal{S}$ is 1-1; $\mathcal{S}$ can therefore be given the structure of statistical manifold. Figure 2 illustrates statistical manifold representations of some example point clouds.

2.2. Information Riemannian Metric for Point Cloud Data Space

We now equip the point cloud statistical manifold $\mathcal{S}$ with the Fisher information metric, which we refer to as the info-Riemannian metric and denote by $\mathbf{H}$. The first task is to define a local coordinate system on the space of point clouds $\mathcal{X}$. Toward this end, we use the matrix representation $X \in \mathbb{R}^{n \times D}$ of a point cloud $\mathbf{X}$. Observe that the matrix representation is not unique: given an $n \times n$ permutation matrix $P \in \mathbb{R}^{n \times n}$, then X and PX represent the same point cloud $\mathbf{X}$. Fortunately, this does not cause problems since $p(x; X)$ is defined in a permutation-invariant way, i.e., $p(x; X) = p(x; PX)$ for any $n \times n$ permutation matrix P. We again note that we use italics to denote local coordinate representations, e.g., $\mathbf{X}$ has local coordinates $X \in \mathbb{R}^{n \times D}$, the tangent vector $\mathbf{V} \in T_{\mathbf{X}}\mathcal{X}$ has local coordinates $V \in \mathbb{R}^{n \times D}$.

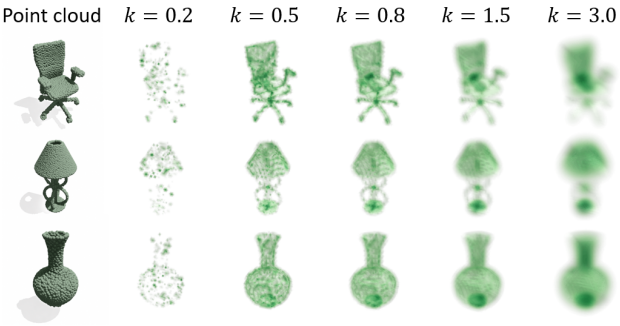

Figure 2: Probability heat maps for various k (the greener, the higher) for some examples from the ShapeNet dataset (Chang et al., 2015), where we set $\sigma = k \times \text{MED}$ for $k \in (0, \infty)$. MED denotes the median of the distances between the points in the point cloud and their nearest points.

The info-Riemannian metric $\mathbf{H}$ can be expressed in local coordinates coordinates X as follows:

$$H_{ijkl}(X) := \int p(x; X) \frac{\partial \log p(x; X)}{\partial X^{ij}} \frac{\partial \log p(x; X)}{\partial X^{kl}} \, dx, \tag{7}$$

for $i, k = 1, ..., n$ and $j, l = 1, ..., D$. Given two tangent vectors $\mathbf{V}, \mathbf{W} \in T_{\mathbf{X}}\mathcal{X}$ with respective matrix representations $V, W \in \mathbb{R}^{n \times D}$, their inner product is then computed as follows:

$$\langle \mathbf{V}, \mathbf{W} \rangle_{\mathbf{X}} := \sum_{i,k=1}^{n} \sum_{j,l=1}^{D} H_{ijkl}(X) V^{ij} W^{kl}. \tag{8}$$

The coordinate expression of the info-Riemannian metric $H_{ijkl}(X)$ results in a permutation-invariant inner product, i.e., $\sum H_{ijkl}(X) V^{ij} W^{kl} = \sum H_{ijkl}(PX)(PV)^{ij}(PW)^{kl}$ for any $n \times n$ permutation matrix P, showing that the metric is geometrically well-defined.

Using the standard normal kernel function, the coordinate expression of the info-Riemannian metric H_{ijkl} has a simple analytic expression as follows:

[1] Without the distinct points assumption, the set of point clouds is no longer a manifold, but only an orbifold that is locally a finite group quotient of a Euclidean space.

[2] A kernel function $\Psi(\cdot, \cdot)$ is said to be *strictly positive-definite* if the matrix $(\Psi(x_i, x_j))_{1 \le i,j \le m}$ is positive-definite for all positive integers m and all mutually distinct $x_1, ..., x_m$.

Proposition 2.5. *With the standard (multivariate) normal kernel function and the bandwidth parameter σ, the information Riemannian metric $H_{ijkl}(X)$ is given by*

$$\int p(x;X)\frac{K(\frac{x-x_i}{\sigma})K(\frac{x-x_k}{\sigma})}{(\sum_{m=1}^{n}K(\frac{x-x_m}{\sigma}))^2}\Big[\frac{(x-x_i)(x-x_k)^T}{\sigma^4}\Big]_{jl}dx, \tag{9}$$

Figure 3 shows that, given two moving point cloud data whose velocity matrices have equal Euclidean norm (i.e., $||\mathbf{V}||^2 = \sum_{i=1}^{n}\sum_{j=1}^{D} V^{ij}V^{ij}$), the velocity norms under the info-Riemannian metric are significantly different: the velocity A has a value of 0.2626, while the velocity B has a value of 2.2×10^{-8}. In particular, observe that the tangential velocity in the case B, which does not change the overall distribution of the point cloud, has a very small velocity norm under the info-Riemannian metric as it should.

3. Applications to Point Cloud Autoencoders

Riemannian geometric formulations of autoencoders for representation learning have recently been introduced and extensively studied in (Shao et al., 2018; Arvanitidis et al., 2018; Yang et al., 2018a; Chen et al., 2018; Kalatzis et al., 2020; Arvanitidis et al., 2020; Chen et al., 2020; Lee et al., 2022). In these works, the image of the decoder function is viewed as a low-dimensional manifold embedded in the high-dimensional data space – we refer to this manifold as the *decoded manifold* – and a Riemannian metric for the decoded manifold is obtained by projecting the data space Riemannian metric to this manifold. In contrast, this perspective cannot be reasonably extended to existing point cloud autoencoders (e.g., FoldingNet (Yang et al., 2018b), AtlasNet (Groueix et al., 2018), AtlasNetV2 (Deprelle et al., 2019), and TearingNet (Pang et al., 2021)), due to the absence of a geometrically well-formulated Riemannian manifold structure.

In this section, using the info-Riemannian metric, we extend this perspective by defining a Riemannian metric for the decoded manifold of the point cloud autoencoder. With this info-Riemannian metric, we examine two case studies: (i) interpolation between two points of latent space via the minimal geodesic; (ii) learning an optimal set of latent space coordinates that best preserves distances and angles (or intuitively, minimizes distortion).

Consider a point cloud decoder function with the m-dimensional latent space $f : \mathbb{R}^m \to \mathbb{R}^{n\times D}$, where the output is expressed in terms of the matrix representation. The projection of the info-Riemannian metric on the point cloud statistical manifold to the decoded manifold is then expressed in latent space coordinates $z \in \mathbb{R}^m$ as follows:

$$G_{ab}(z) := \sum_{i,k=1}^{n}\sum_{j,l=1}^{D} H_{ijkl}(f(z))(J_f)_a^{ij}(z)(J_f)_b^{kl}(z), \tag{10}$$

where J_f denotes the Jacobian of f, and the indices a, b both range from 1 to m. The latent space is then assigned a Riemannian metric $G_{ab}(z)$; the following two applications rely on $G(z) \in \mathbb{R}^{m\times m}$.

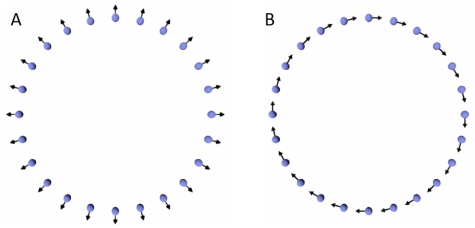

Figure 3: Two moving point clouds with different velocity matrices.

Geodesic interpolation: The latent space metric $G(z)$ can be used to find the minimal geodesic curve connecting two point clouds (i.e., the shortest length curve in the decoded manifold). Let z_1, z_2 be the encoded values of the two point clouds in the latent space $\mathbb{R}^m$. In terms of the metric $G(z)$, the geodesic curve connecting these two points can be determined as a solution to the following optimization problem (Do Carmo, 2016):

$$\min_{z(t)} \int_0^1 \dot{z}(t)^T G(z(t))\dot{z}(t)\, dt, \tag{11}$$

subject to $z(0) = z_1$ and $z(1) = z_2$. Parametrizing $z(t)$ by a cubic spline with fixed boundary points z_1, z_2 then leads to an unconstrained optimization problem. To avoid excessive memory consumption when computing the objective function and its gradient, instead of the usual Riemann sum approximation of the integral, we interpret the integral as an expectation over the uniform distribution $t \sim U(0, 1)$ and accordingly use the mini-batch sampling technique.

Learning optimal latent space coordinates: The latent space metric $G(z)$ can be used to formulate a regularization term when training an autoencoder to learn an optimal set of latent space coordinates; by "optimal" we mean $G(z) = cI$ for some positive scalar c, so that the decoder preserves distances and angles as much as possible. Recently, a regularization technique for this purpose has been introduced in (Chen et al., 2020). Specifically, the following regularization term is added to the reconstruction loss function:

$$\mathbb{E}_{z\sim P}[\|G(z) - cI\|_F^2], \tag{12}$$

where $\|\cdot\|_F$ is the Frobenius norm, $c = \mathbb{E}_{z\sim P}[\frac{1}{m}\mathrm{Tr}(G(z))]$, and P is defined via the modified mix-up augmentation, i.e.,

$z \sim P \iff z = \alpha z_1 + (1-\alpha) z_2$ where z_1, z_2 are sampled from the set of encoded training data and $\alpha \sim U(-\eta, 1+\eta)$ for $\eta > 0$. For latent spaces whose dimension m is large, in order to avoid the expensive and memory-consuming computation of $G(z) \in \mathbb{R}^{m \times m}$, we use the following regularization term in the subsequent experiments:

$$\mathbb{E}_{z \sim P}[\mathbb{E}_{v \sim \mathcal{N}(0,I)}[\|v^T G(z) v - c v^T v\|^2]], \qquad (13)$$

where we use mini-batch sampling to estimate the expectations. This can be done much more efficiently since we need only compute the Jacobian-vector product, i.e., $\sum_a (J_f)^{ij}_a v^a$.

4. Experimental Results

We now verify the effectiveness of the info-Riemannian metric for the two point cloud autoencoder applications described above using both a synthetic 3D basic shape dataset and standard benchmark point cloud datasets.

The synthetic 3D basic shape dataset consists of cylinders, cubes, cones, and ellipsoids with various aspect ratios of the shape parameters (e.g., radius versus height for the cylinder). We sample 512 points from the surface mesh of the shapes using a greedy sample elimination algorithm, so that each sampled point is approximately the same distance from its neighborhood points (Yuksel, 2015). Each point cloud is then normalized so that the two farthest points are a unit distance apart. Further details about the synthetic 3D basic shape dataset generation are provided in Appendix C.

The standard benchmark point cloud dataset consists of ModelNet (Wu et al., 2015) and ShapeNet (Chang et al., 2015), where ModelNet consists of ModelNet10 and ModelNet40, each of which consist of 10 and 40 shape classes, respectively. Each point cloud data has 2048 points; we normalize these into a unit sphere as done in (Yang et al., 2018b).

4.1. Synthetic 3D Basic Shape Dataset

In Section 4.1.1, we use a dataset consisting of cones, cylinders, and ellipsoids, which are split into training/validation/test sets of size 3196/800/804. We then confirm the validity of the proposed metric by comparing the results of several shape interpolation methods in the latent space.

In Section 4.1.2, to study the effects of the regularization term when learning the optimal latent coordinates (with respect to the info-Riemannian metric), we use a dataset consisting of boxes, cones, and ellipsoids divided into training/validation/test sets of size 720/240/240.

We use DGCNN as the encoder (Wang et al., 2019) and a fully-connected neural network as the decoder. The latent space is assumed to be two-dimensional. For the reconstruction loss term, we use the Chamfer distance. The regularization term in Equation (12) is multiplied by a coefficient $\lambda > 0$ and added to the reconstruction loss term. Further details about the network architectures and training are provided in Appendix C.

4.1.1. Example 1: Cone, Cylinder, and Ellipsoid

Figure 4 shows the test data encoded in the latent space together with the interpolation trajectories and generated point clouds from those interpolants. In the case of intra-class interpolations (i.e., interpolants between cylinders), the linear interpolant clearly passes through the red ellipsoid region in the latent space, with some of the generated point clouds clearly ellipsoids. The geodesic interpolations under the Euclidean and info-Riemannian metrics both avoid ellipsoid regions in the latent space. In particular, the geodesic interpolants between two cylinders are also cylinders; this is well-aligned with human intuition. However, if we look at the generated point clouds in detail, while the info-Riemannian metric produces clearly blue cylinders, some of the generated point clouds with the Euclidean metric are non-blue cylinders (i.e., relatively closer to the ellipsoid region) with noisy side surfaces. For the inter-class interpolations (i.e., interpolants between a cylinder and a cone), the linear interpolant also clearly passes through the red ellipsoid region. The geodesic interpolation under the Euclidean metric produces many non-blue and non-green color shapes during the transition from cylinders to cones, while geodesic interpolation under the info-Riemannian metric produces such cases far less. Overall, it can be observed that the geodesic interpolants under the info-Riemannian metric have minimal shape class changes.

Figure 5 shows the latent spaces produced by the regularized autoencoders using the Euclidean and Info-Riemannian metrics. Compared to the latent space in Figure 4 (trained without regularization), the following observations can be made: (i) the encoded latent spaces of cones and cylinders are flattened, and (ii) the ellipsoids and cones become more discriminative. Furthermore, we note that the encoded latent space curves of cones and cylinders in the right figure (where the info-Riemannian metric is used) are clearly flatter than those in the left figure (where the Euclidean metric is used). In other words, if we linearly interpolate between two cylinders or two cones, the interpolants in the right case will most likely remain in the same class, unlike the left case (the generated point clouds from the linear interpolants are provided in Appendix E).

4.1.2. Example 2: Box, Cone, and Ellipsoid

Figure 6 shows the test data encoded in the latent space together with the visualization of the Riemannian metric, the fitted Gaussian Mixture Model and its samples, and pair-

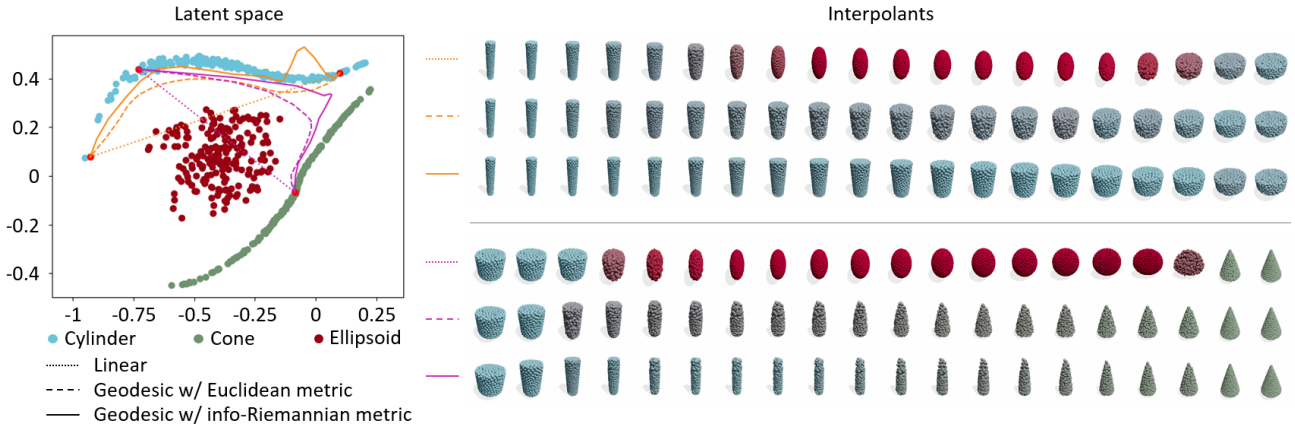

Figure 4: *Left*: Latent space with linear and geodesic interpolants. The orange interpolants connect a wide cylinder to a tall cylinder, while the magenta interpolants connect a cylinder to a cone. Linear interpolants and geodesic interpolants under the Euclidean and info-Riemannian metrics are drawn as dotted, dashed, and solid lines, respectively. *Right*: Generated point clouds from those interpolants. To visually indicate which class generated point cloud belong to, we color these according to the ratio of the Chamfer distances to the nearest point cloud for each class (see Appendix C). For example, when it is uncertain which class a generated data belongs to (i.e., the nearest distances to each class are similar), it is assigned some color other than blue, red, or green.

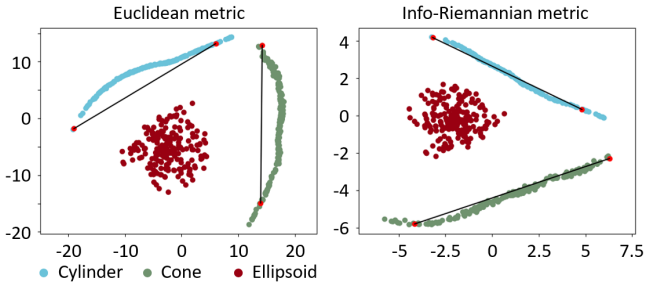

Figure 5: Latent spaces produced by regularized autoencoders, each of which is trained with the Euclidean (*Left*) and info-Riemannian metric (*Right*). Representative intra-class linear interpolants between two cylinders and two cones are drawn as black solid lines.

wise Euclidean distances. From the first column of Figure 6, by comparing the results with and without regularization, the following two key observations can be made: (i) in the vanilla autoencoder case (upper case), the major axes of the gray ellipses are aligned with the decision boundary (i.e., a hypersurface that partitions the different class regions), which implies that shapes of different classes are actually more distant in the learned manifold (under the info-Riemmanian metric) than shown in the latent space, and (ii) in the regularized autoencoder case (lower case), by encouraging the metric to be isotropic (i.e., turning ellipses into circles), the gaps between different class regions are widened. The second column confirms that the components of the GMM are better separated after regularization; each component of the GMM on the regularized autoencoder generates high-quality, even samples from the same class shape as shown in the third column. The heat maps of the pairwise distances in the last column also indicate that shapes in different classes are more distant, and therefore more easily separable in the latent space of the regularized autoencoder.

To quantitatively verify, we repeat this experiment with multiple different synthetic datasets (details are in Appendix C), and report the averaged GMM clustering scores, Normalized Mutual Information (NMI) and Adjusted Rand Index (ADI), in Table 1. Ours shows higher clustering accuracy.

Table 1: Averaged clustering scores over 27 different datasets, each of which consists of diverse shapes of boxes, cones, and ellipsoids; the higher the better.

METHOD	NMI	ADI
Vanilla AE	0.7624 ± 0.2132	0.7209 ± 0.2598
Regularized AE	$\mathbf{0.9484 \pm 0.1391}$	$\mathbf{0.9368 \pm 0.1737}$

4.2. Standard Benchmark Data

To show that the regularization technique with the info-Riemannian metric can benefit unsupervised representation learning from the perspective of discriminative representation learning, we compare the transfer classification accuracy of ShapeNetCore.v2 to ModelNet following the same experimental procedure outlined in (Yang et al., 2018b). When training autoencoders with ShapeNet, random rotations about an axis parallel to the direction of gravity are applied to each point cloud. We use four different point cloud autoencoders: *FcNet* and *FoldingNet* adopted from (Yang et al., 2018b), *PointCapsNet* adopted from (Zhao et al., 2019), and *DGCNN-FcNet* using DGCNN (Wang

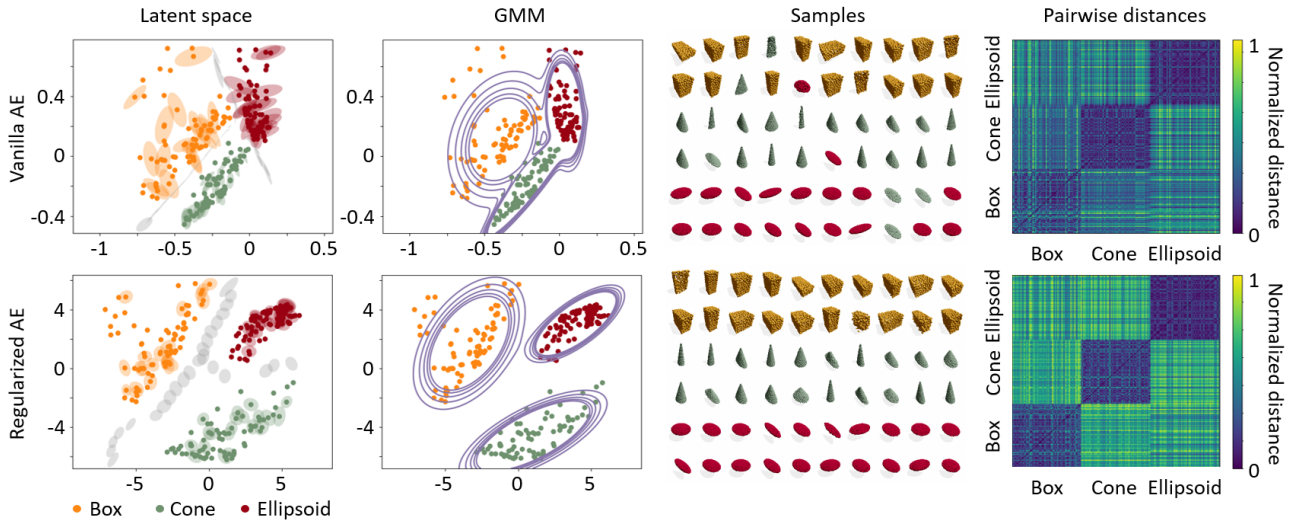

Figure 6: From *left* to *right*: latent spaces with equidistant ellipse ($\{z|(z-z^*)^T G(z^*)(z-z^*)=1\}$ for center z^*) centered on some selected points and sampled points from interspaces, Gaussian Mixture Model (GMM) fitting results, generated samples from the GMM, and the heat map of the pairwise Euclidean distances in the latent space of all test data. The *upper* figure is a vanilla autoencoder trained without regularization, while the *lower* figure is trained with regularization (using the info-Riemannian metric). For the samples in the third column, we assign colors using the same method of Section 4.1.1 to visually express which classes the samples are likely to belong to.

et al., 2019); the latent space is 512-dimensional for all. The four autoencoders are trained with and without regularization. In the former case, the regularization terms of Equation (13) under both the Euclidean and info-Riemannian metrics are used while varying the regularization coefficient λ. We distinguish between regularized autoencoders using the Euclidean and info-Riemannian metrics by an "+E" or "+I" after the network name. After network training is finished, we train linear SVM classifiers with the encoded data for ModelNet10 and ModelNet40. These are split into training/test sets of sizes 3991/909 and 9843/2468, respectively. Further experimental detail are provided in Appendix C.

Table 2 shows a comparison of transfer classification accuracy from ShapeNet to ModelNet10 (MN10) and ModelNet40 (MN40) for various recent state-of-the-art methods. In the upper table (*Adopted from References*), the numbers are adopted from previous papers (the experimental procedures may differ slightly from ours). In the lower table (*Implemented by Authors*), we report the best numbers obtained (adopted from Appendix E). First, although their performance is not directly comparable due to differences in the experimental procedures, it can be seen that our regularized autoencoders are comparable to the state-of-the-art methods. Second, at least for our implementation, regularization using the info-Riemannian metric improves classification accuracy over vanilla autoencoders, with higher accuracy compared to the Euclidean metric case.

Next, based on the intuition that kernel-based statistical representations are robust to noise, we also conduct additional experiments to determine how much more robust the representation obtained with our regularization approach is for noisy point cloud data. We add noise with different levels of standard deviation (1%, 5%, 10%, and 20% of the diagonal length of the point cloud bounding box) to point cloud data (see Appendix C for details). Then *FcNet* is trained with and without regularization in the same way as above.

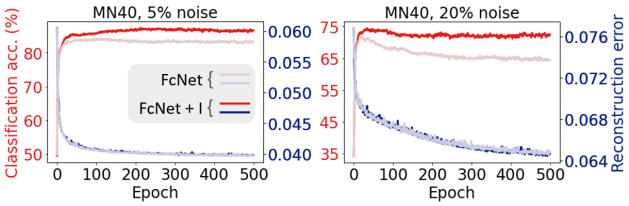

Figure 7: Learning curves in the presence of noise (*left*: 5% noise; *right*: 20% noise), ModelNet40 transfer classification accuracy and reconstruction error as functions of the training epoch.

Table 3 shows a comparison of transfer classification accuracy in the presence of noise. As the noise level increases, the classification accuracy obviously decreases, but the reduction is the least dramatic for regularized autoencoders under Info-Riemannian metric. Figure 7 shows learning curves, ModelNet40 transfer classification accuracy and reconstruction error as functions of the training epoch, in the presence of noise. Throughout the learning process, compared to vanilla autoencoders (light colored lines), reg-

Table 2: Classification accuracy by transfer learning for ModelNet10 (MN10) and ModelNet40 (MN40) from ShapeNet.

METHOD	MN40	MN10
Adopted from References		
SPH (Kazhdan et al., 2003)	68.2%	79.8%
LFD (Chen et al., 2003)	75.5%	79.9%
VConv-DAE (Sharma et al., 2016)	75.5%	80.5%
3D-GAN (Wu et al., 2016)	83.3%	91.0%
Latent-GAN (Achlioptas et al., 2018)	84.5%	**95.4%**
FoldingNet (Yang et al., 2018b)	88.4%	94.4%
PointFlow (Yang et al., 2019)	86.8%	93.7%
Multi-Task (Hassani & Haley, 2019)	89.1%	-
PointCapsNet (Zhao et al., 2019)	**89.3%**	-
Implemented by Authors		
FcNet	88.3%	93.5%
FcNet + E (ours)	89.3%	93.7%
FcNet + I (ours)	90.4%	94.3%
FoldingNet	89.3%	93.7%
FoldingNet + E (ours)	88.9%	94.4%
FoldingNet + I (ours)	90.1%	94.5%
PointCapsNet	87.2%	93.6%
PointCapsNet + E (ours)	88.1%	93.7%
PointCapsNet + I (ours)	88.5%	93.9%
DGCNN-FcNet	90.3%	94.5%
DGCNN-FcNet + E (ours)	89.9%	94.4%
DGCNN-FcNet + I (ours)	**91.0%**	**95.2%**

ularized autoencoders under Info-Riemannian metric (dark colored lines) show higher classification accuracy and similar levels of reconstruction errors. In particular, as shown in the right figure, as the learning progresses, the classification accuracy of the vanilla autoencoder largely decreases in expense of minimizing the reconstruction error, while that of our regularized autoencoder is maintained. This shows our regularization approach helps autoencoders to learn robust representations to noise as expected.

Lastly, we compare the semi-supervised transfer classification accuracy. In the semi-supervised settings, not all the training data have labels, which are actually more frequent situations in reality. When we train linear SVM classifier, we use different numbers of training data (1%, 5%, 10%, and 50% of the overall training data, see Appendix C for details). Table 4 shows a comparison of transfer classification accuracy according to the percentage of training label used. As the label rate decreases, the classification accuracy decreases, but the reduction is more dramatic for vanilla autoencoders. Together with the results in Table 2, this clearly shows that our regularization approach helps autoencoders to learn more discriminative representation spaces, and the effect is greater with a small number of labeled data.

Overall, it is indeed somewhat surprising that unsupervised classification accuracy can be improved (i.e., more discriminative representation space can be obtained) with a simple regularization technique in lieu of a complex neural network architecture or loss function. We include additional experimental results and analysis in Appendix D.

Table 3: Classification accuracy by transfer learning for ModelNet10 (MN10) and ModelNet40 (MN40) from ShapeNet under the noise levels of 1%, 5%, 10%, and 20%.

	MN40			
METHOD	1%	5%	10%	20%
FcNet	87.8%	83.2%	75.6%	64.5%
FcNet + E (ours)	86.6%	85.1%	79.1%	70.4%
FcNet + I (ours)	**89.0%**	**86.6%**	**81.4%**	**72.4%**
	MN10			
	1%	5%	10%	20%
FcNet	92.4%	91.9%	88.4%	79.8%
FcNet + E (ours)	92.2%	91.1%	88.2%	82.6%
FcNet + I (ours)	**93.3%**	**92.6%**	**91.6%**	**84.8%**

Table 4: Classification accuracy by transfer learning for ModelNet10 (MN10) and ModelNet40 (MN40) from ShapeNet under the different percentages of labeled training data for linear SVM classifier (50%, 10%, 5%, and 1%).

METHOD	MN40			
	50%	10%	5%	1%
FcNet	85.7%	78.0%	70.6%	50.3%
FcNet + I (ours)	**87.9%**	**81.6%**	**76.8%**	**57.4%**
	MN10			
	50%	10%	5%	1%
FcNet	91.7%	90.1%	87.2%	74.1%
FcNet + I (ours)	**93.2%**	**91.2%**	**88.3%**	**78.1%**

5. Discussion and Conclusion

This paper has proposed a new Riemannian geometric structure for the space of point cloud data. We have defined a statistical representation of point cloud data and constructed a statistical manifold in a mathematically rigorous way. Then a natural Riemannian metric – Fisher information metric – is assigned to the point cloud statistical manifold, which provides geometrically well-defined measures needed for applications. We demonstrate its advantages through two applications involving point cloud autoencoders: (i) minimal geodesic interpolants under info-Riemannian metric have minimal shape changes compared to the standard linear interpolants, and (ii) the optimal latent coordinates learned using our method produce more discriminative representation spaces than existing methods. In particular, transfer classification accuracy has been greatly improved in noisy data and semi-supervised settings.

As a potential issue, the "fixed number of points" assumption used in our construction of the statistical manifold may be violated in real world problems. In such cases, we can easily mitigate this issue by matching the number of points in each point cloud through a simple upsampling/downsampling algorithm. Further, the kernel function used in our

current implementations, the standard normal kernel function, may not be an optimal choice. Other choices can be explored to enhance our algorithms as long as the conditions of Proposition 2.4 are satisfied.

Compared to research on distance metric, there are relatively few studies on Riemannian metric despite its importance and utility. This paper is the starting point of research on point cloud space Riemannian metric, and we believe the study on diverse Riemannian metrics, just as various distance metrics with different properties have been developed, should be continued.

In certain real-life scenarios containing multiple 3d objects that (i) are only partially observed (e.g., only one side of the underlying surface is observed), and (ii) local densities of the measured points for each object are different, using a single probability density function with a single kernel-type and fixed bandwidth parameter to represent the measured point cloud can be problematic. For example, consider a point cloud data obtained through LiDAR; the point cloud can include diverse objects such as cars, pedestrians, trees, buildings, and lanes. They are obviously partially observed, and local point cloud densities are different since their distances from the sensors are different. One way to approach this problem is to (i) first decompose the measured point cloud into several multiple point clouds, each of which represents a single object (for example, by using existing object detection techniques); (ii) use point cloud completion and super resolution algorithms as needed to make each point cloud rich enough to represent the corresponding object, and (iii) apply our methods by using different kernels and bandwidth parameters suitable to represent each point cloud. Other approaches are also possible, and we believe it is an worthwhile future research topic to examine which approaches are best suited to different application domains.

Acknowledgements

This work was supported in part by SRRC NRF grant 2016R1A5A1938472, 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, and the SNU Institute for Engineering Research.

References

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Amari, S.-i. *Information geometry and its applications*, volume 194. Springer, 2016.

Amari, S.-i. and Nagaoka, H. *Methods of information geometry*, volume 191. American Mathematical Soc., 2000.

Arvanitidis, G., Hansen, L. K., and Hauberg, S. Latent space oddity: on the curvature of deep generative models. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Arvanitidis, G., Hauberg, S., and Schölkopf, B. Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*, 2020.

Besl, P. J. and McKay, N. D. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pp. 586–606. International Society for Optics and Photonics, 1992.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chen, D.-Y., Tian, X.-P., Shen, Y.-T., and Ouhyoung, M. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pp. 223–232. Wiley Online Library, 2003.

Chen, N., Klushyn, A., Kurle, R., Jiang, X., Bayer, J., and Smagt, P. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1550. PMLR, 2018.

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.

Cosmo, L., Norelli, A., Halimi, O., Kimmel, R., and Rodolà, E. Limp: Learning latent shape representations with metric preservation priors. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 19–35. Springer, 2020.

Davis, R. A., Lii, K.-S., and Politis, D. N. Remarks on some nonparametric estimates of a density function. In *Selected Works of Murray Rosenblatt*, pp. 95–100. Springer, 2011.

Deprelle, T., Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. Learning elementary structures for 3d shape generation and matching. *arXiv preprint arXiv:1908.04725*, 2019.

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

Efron, B. and Hinkley, D. V. Assessing the accuracy of the maximum likelihood estimator: Observed versus expected fisher information. *Biometrika*, 65(3):457–483, 1978.

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224, 2018.

Han, M. and Park, F. C. Dti segmentation and fiber tracking using metrics on multivariate normal distributions. *Journal of mathematical imaging and vision*, 49(2):317–334, 2014.

Hasanbelliu, E., Giraldo, L. S., and Príncipe, J. C. Information theoretic shape matching. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2436–2451, 2014.

Hassani, K. and Haley, M. Unsupervised multi-task feature learning on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8160–8171, 2019.

Hausdorff, F. Grundzüge der mengenlehre. 1914.

Hausdorff, F. *Felix Hausdorff-Gesammelte Werke Band III: Mengenlehre (1927, 1935) Deskripte Mengenlehre und Topologie*, volume 3. Springer-Verlag, 2008.

Jian, B. and Vemuri, B. C. A robust algorithm for point set registration using mixture of gaussians. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pp. 1246–1251. IEEE, 2005.

Kalatzis, D., Eklund, D., Arvanitidis, G., and Hauberg, S. Variational autoencoders with riemannian brownian motion priors. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pp. 156–164, 2003.

Knudsen, B. Configuration spaces in algebraic topology. *arXiv preprint arXiv:1803.11165*, 2018.

Lee, Y., Baek, J., Kim, Y. M., and Park, F. C. Imat: The iterative medial axis transform. In *Computer Graphics Forum*. Wiley Online Library, 2021.

Lee, Y., Yoon, S., Son, M., and Park, F. Regularized autoencoders for isometric representation learning. *International Conference on Learning Representation*, 2022.

Li, F., Fujiwara, K., and Matsushita, Y. Toward a unified framework for point set registration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12981–12987. IEEE, 2021.

Luo, S. and Hu, W. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4583–4592, 2021.

Min, Z., Wang, J., and Meng, M. Q.-H. Robust generalized point cloud registration using hybrid mixture model. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4812–4818. IEEE, 2018.

Myronenko, A. and Song, X. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

Nguyen, T., Pham, Q.-H., Le, T., Pham, T., Ho, N., and Hua, B.-S. Point-set distances for learning representations of 3d point clouds. *arXiv preprint arXiv:2102.04014*, 2021.

Pang, J., Li, D., and Tian, D. Tearingnet: Point cloud autoencoder to learn topology-friendly representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7453–7462, 2021.

Parzen, E. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3): 1065–1076, 1962.

Paulsen, V. I. and Raghupathi, M. *An introduction to the theory of reproducing kernel Hilbert spaces*, volume 152. Cambridge university press, 2016.

Rissanen, J. J. Fisher information and stochastic complexity. *IEEE transactions on information theory*, 42(1):40–47, 1996.

Rubner, Y., Tomasi, C., and Guibas, L. J. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Shao, H., Kumar, A., and Thomas Fletcher, P. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 315–323, 2018.

Sharma, A., Grau, O., and Fritz, M. Vconv-dae: Deep volumetric shape learning without object labels. In *European Conference on Computer Vision*, pp. 236–250. Springer, 2016.

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11:1517–1561, 2010.

Wang, F., Vemuri, B. C., and Rangarajan, A. Groupwise point pattern registration using a novel cdf-based jensen-shannon divergence. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pp. 1283–1288. IEEE, 2006.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38 (5):1–12, 2019.

Wu, J., Zhang, C., Xue, T., Freeman, W. T., and Tenenbaum, J. B. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 82–90, 2016.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4541–4550, 2019.

Yang, T., Arvanitidis, G., Fu, D., Li, X., and Hauberg, S. Geodesic clustering in deep generative models. *arXiv preprint arXiv:1809.04747*, 2018a.

Yang, Y., Feng, C., Shen, Y., and Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 206–215, 2018b.

Yuksel, C. Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum*, volume 34, pp. 25–32. Wiley Online Library, 2015.

Zaman, F., Wong, Y. P., and Ng, B. Y. Density-based denoising of point cloud. In *9th International Conference on Robotic, Vision, Signal Processing and Power Applications*, pp. 287–295. Springer, 2017.

Zhao, Y., Birdal, T., Deng, H., and Tombari, F. 3d point capsule networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1009–1018, 2019.

Zhou, Z., Zheng, J., Dai, Y., Zhou, Z., and Chen, S. Robust non-rigid point set registration using student's-t mixture model. *PloS one*, 9(3):e91381, 2014.

Appendix

A. Existing Geometric/Statistical Methods for Point Cloud Data

A.1. Geometric Methods

The Hausdorff distance measures the distance between two non-empty subsets of a metric space (Hausdorff, 1914; 2008). Given two point clouds $\mathbf{X} = \{x_1, ..., x_n \mid x_i \in \mathbb{R}^D\}$ and $\mathbf{Y} = \{y_1, ..., y_n \mid y_i \in \mathbb{R}^D\}$ and metric $\|x - y\|^2$ in $\mathbb{R}^D$, the Hausdorff distance can be computed as follows:

$$\max(\max_{x\in\mathbf{X}}(\min_{y\in\mathbf{Y}}\|x-y\|^2), \max_{y\in\mathbf{Y}}(\min_{x\in\mathbf{X}}\|x-y\|^2)).$$

The Hausdorff distance is susceptible to outliers; hence, in practice, the average Hausdorff distance is used more often:

$$\frac{1}{|\mathbf{X}|}\sum_{x\in\mathbf{X}}\min_{y\in\mathbf{Y}}\|x-y\|^2 + \frac{1}{|\mathbf{Y}|}\sum_{y\in\mathbf{Y}}\min_{x\in\mathbf{X}}\|x-y\|^2,$$

where $|\mathbf{X}|$ denotes the number of elements in the set $\mathbf{X}$. A slightly modified version of this is often referred to as the Chamfer distance (Yang et al., 2018b). The popular point cloud registration algorithm ICP relies on these classes of metrics (Besl & McKay, 1992).

Another popular similarity measure between two point cloud data is the Earth Mover's Distance (EMD) (Rubner et al., 2000):

$$\sum_{x\in\mathbf{X}}\min_{\phi:\mathbf{X}\to\mathbf{Y}}\|x-\phi(x)\|^2,$$

where ϕ is a bijective mapping. Although the EMD is computationally more expensive than the above Hausdorff distances, comparing point clouds with optimal matching in EMD provides a more robust and well-behaved similarity measure.

The Chamfer distance and EMD are often used to measure the distances between two point clouds, but they are typically computationally expensive. Recently, the sliced Wasserstein distance and its variants have been proposed to more efficiently measure distances (Nguyen et al., 2021). In another study, each point cloud data is represented as a matrix of pairwise Euclidean distances between all points, and the Frobenius norm of the difference between the two matrices is used as the distance between two point clouds (Cosmo et al., 2020).

A.2. Statistical Methods

Interpreting the point cloud data as a set of samples from some underlying probability distribution is very intuitive and natural, and has been adopted in many previous works (Jian & Vemuri, 2005; Wang et al., 2006; Myronenko & Song, 2010; Hasanbelliu et al., 2014; Zhou et al., 2014; Min et al., 2018; Li et al., 2021). Commonly, a mixture model is used to describe the point cloud, written as follows:

$$p(x; w, \theta) := \sum_{i=1}^{k} w_i \phi(x|\theta_i),$$

where the w_i are weights and $\phi(x|\theta_i)$ are primitive density functions with parameter θ_i (e.g., $\phi(x|\theta_i)$ can be a standard Gaussian with mean θ_i). Given a point cloud $\mathbf{X} := \{x_1, \ldots, x_n | x_i \in \mathbb{R}^D\}$, the parameters w, θ are either fit with data or specified by the user, then the mixture model is used as a statistical representation of the point cloud data. Besides the most popular choice for ϕ, the Gaussian (Jian & Vemuri, 2005), other choices such as the t-distribution (Zhou et al., 2014) or hybrid model (Min et al., 2018) have been explored. The main purpose behind using statistical representations in existing works are to use the well-known information-theoretic divergence measures such as the KL-divergence to compute the similarity between point clouds.

However, all these methods focus on distance metrics that measure just one aspect of point cloud data, yet effective mathematical concepts and tools for defining and measuring other important geometric aspects of point cloud are still lacking.

B. Proof of the Propositions

B.1. Proof of Proposition 2.4

Proof. Start with the proof of Proposition B.1 which is an easier version of the original proposition.

Proposition B.1. *Assume that all point cloud data in $\mathcal{X}$ consists of exactly n distinct points in $\mathbb{R}^D$. If the set of functions $\{K(\Sigma^{-\frac{1}{2}}(x - x_i))\}_{x_i \in \mathcal{F}}$ are linearly independent[3] for any arbitrary finite subset $\mathcal{F} \subset \mathbb{R}^D$ with $|\mathcal{F}| \leq 2n$, the mapping $h : \mathcal{X} \to \mathcal{S}$ is 1-1.*

Proof. Let's consider two point clouds $\{y_i\}_{i=1}^n$ and $\{z_i\}_{i=1}^n$. To show that the mapping $h : \mathcal{X} \to \mathcal{S}$ is 1-1 (especially injective), we have to prove the following statement:

$$p(x; \{y_i\}_{i=1}^n) = p(x; \{z_i\}_{i=1}^n) \implies \{y_i\}_{i=1}^n = \{z_i\}_{i=1}^n. \tag{14}$$

The conditional statement can be rewritten as follows:

$$\sum_{i=1}^{n} K(\Sigma^{-\frac{1}{2}}(x - y_i)) = \sum_{i=1}^{n} K(\Sigma^{-\frac{1}{2}}(x - z_i)). \tag{15}$$

Let denote $B = \{y_i\}_{i=1}^n \cap \{z_i\}_{i=1}^n$ and $|B| = m$, and assume that $m < n$. Then the above equation is reduced to

$$\sum_{y \in \{y_i\}_{i=1}^n - B} K(\Sigma^{-\frac{1}{2}}(x - y)) - \sum_{z \in \{z_i\}_{i=1}^n - B} K(\Sigma^{-\frac{1}{2}}(x - z)) = 0. \tag{16}$$

Since the sets $\{y_i\}_{i=1}^n - B$ and $\{z_i\}_{i=1}^n - B$ are disjoint and each set has $n - m$ elements, the LHS has $2(n - m) \leq 2n$ terms and the terms are different to each other. Then, by the assumption, the above $2(n - m)$ terms are linearly independent, and so the above equation cannot hold. There is a contradiction and the assumption $m < n$ must be wrong. Therefore, $m = n$, so $\{y_i\}_{i=1}^n = \{z_i\}_{i=1}^n$. □

With the above proposition, any kernel function that satisfies the linear independence condition is sufficient to ensure the existence of a 1-1 mapping h. Indeed, the strict positive definiteness of the kernel function, satisfied by various kernel functions such as normal (Gaussian) or Laplacian function, implies the linear independence condition as stated below:

Proposition B.2 (Corollary of Proposition 4.3. in (Paulsen & Raghupathi, 2016))**.** *Let $\Psi : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a positive function and $\mathcal{F} = \{x_1, ..., x_n\}$ be a finite set of mutually distinct points. Then the set $\{\Psi(\cdot, x_i)\}_{x_i \in \mathcal{F}}$ is linearly independent if and only if the matrix $(\Psi(x_i, x_j))_{i,j=1,...,n}$ is positive definite.*

From Propositions B.1 and B.2, Proposition 2.4 can be easily proved. From the assumption that a function $\Psi : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ defined by

$$\Psi(x, y) = K(\Sigma^{-\frac{1}{2}}(x - y)) \tag{17}$$

is strictly positive definite, the matrix $(\Psi(x_i, x_j))_{i,j=1,...,n}$ is always positive definite for any finite set of mutually distinct points $\mathcal{F} = \{x_1, ..., x_n\}$, so the set of functions $\{K(\Sigma^{-\frac{1}{2}}(x - x_i)\}_{x_i \in \mathcal{F}} = \{\Psi(x, x_i)\}_{x_i \in \mathcal{F}}$ is always linearly independent by Proposition B.2. Then, directly from Proposition B.1, the mapping $h : \mathcal{X} \to \mathcal{S}$ is 1-1.

□

[3] The linear independence of a set of functions implies that only a trivial linear combination of the functions equals the zero function.

B.2. Proof of Proposition 2.5

Proof. Since $\frac{\partial \log p(x;X)}{\partial X^{ij}} = \frac{1}{p(x;X)} \frac{\partial p(x;X)}{\partial X^{ij}}$, the Riemannian metric H_{ijkl} in equation (7) is

$$\int p(x;X) \frac{1}{p^2(x;X)} \frac{\partial p(x;X)}{\partial X^{ij}} \frac{\partial p(x;X)}{\partial X^{kl}} \, dx.$$

By plugging $p(x;X)$ in equation (6) in $\frac{\partial p(x;X))}{\partial X^{ij}}$, we get the following expression:

$$\begin{aligned}
n\sqrt{|\Sigma|} \frac{\partial p(x;X)}{\partial X^{ij}} &= \frac{\partial}{\partial X^{ij}} \sum_{a=1}^{n} K(\Sigma^{-1/2}(x - x_a)) \\
&= \frac{\partial}{\partial X^{ij}} K(\Sigma^{-1/2}(x - x_i)) \\
&= J_K(\Sigma^{-1/2}(x - x_i))\Sigma^{-1/2} \frac{\partial}{\partial X^{ij}}(x - x_i) \\
&= J_K(\Sigma^{-1/2}(x - x_i))\Sigma^{-1/2} \frac{\partial}{\partial X^{ij}}(x - x_i) \\
&= -[J_K(\Sigma^{-1/2}(x - x_i))\Sigma^{-1/2}]_j,
\end{aligned}$$

$J_K : \mathbb{R}^D \to \mathbb{R}^D$ is the Jacobian of the kernel function K. This consequently leads to

$$H_{ijkl}(X) := \int p(x;X) \frac{(J_K\big|_{h(x,x_i)}\Sigma^{-\frac{1}{2}})_j (J_K\big|_{h(x,x_k)}\Sigma^{-\frac{1}{2}})_l}{(\sum_{m=1}^{n} K(h(x,x_m)))^2} \, dx,$$

where $h(x, x_i) = \Sigma^{-\frac{1}{2}}(x - x_i)$.

If K is the standard normal kernel function, then we get $J_K(x) = -K(x)x^T$. By plugging this in the above equation with $\Sigma = \sigma^2 I$, we get the following:

$$\begin{aligned}
H_{ijkl}(X) &= \int p(x;X) \frac{(J_K\big|_{h(x,x_i)}\Sigma^{-\frac{1}{2}})_j (J_K\big|_{h(x,x_k)}\Sigma^{-\frac{1}{2}})_l}{(\sum_{m=1}^{n} K(h(x,x_m)))^2} \, dx \\
&= \int p(x;X) \frac{K(h(x,x_i))K(h(x,x_k))}{(\sum_{m=1}^{n} K(h(x,x_m)))^2} \Big[\frac{(x - x_i)(x - x_k)^T}{\sigma^4}\Big]_{jl} \, dx.
\end{aligned}$$

□

C. Implementation Details for the Experiments

C.1. Synthetic 3D Basic Shape Dataset Generation

For synthetic 3D basic shape dataset, we define 5 shape classes that consist of cylinder, cone, elliptic cone, ellipsoid, and box. Figure 8 shows the representative shape of each class and the shape parameters used to define the shape class. We sample 512 points from the surface mesh of the shapes using a greedy sample elimination algorithm, and each point cloud is then normalized so that the two farthest points are a unit distance apart.

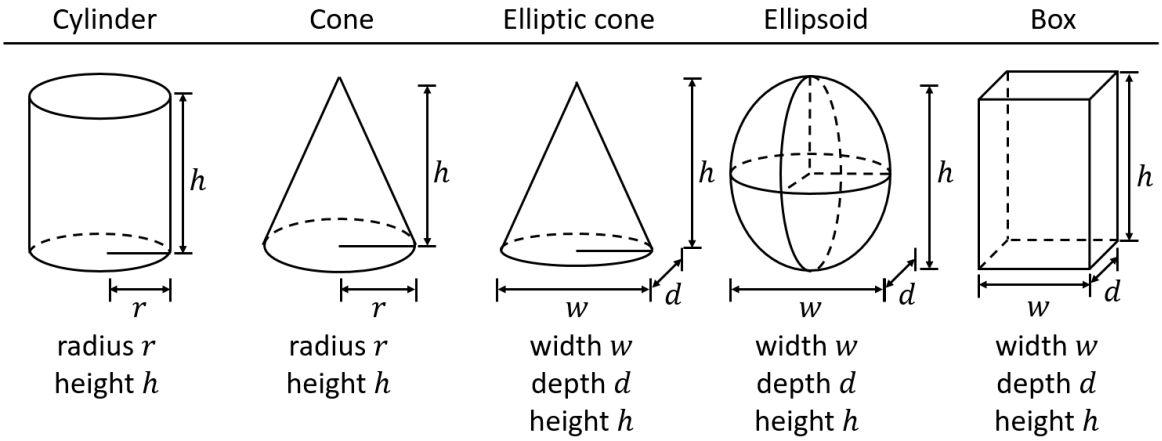

Figure 8: Representative shape to each class and the shape parameters required to define it. There are 5 shape classes including cylinder, cone, elliptic cone, ellipsoid, and box.

In Section 4.1.1, we use a dataset consisting of cones, cylinders, and ellipsoids, which are split into training/validation/test sets of size 3196/800/804. The detail ranges of the exact value of the shape parameters are shown in Table 5.

Table 5: The ranges of the shape parameters of the dataset used in Section 5.1.1

SHAPE	param	min	max	param	min	max	param	min	max
Cylinder	r	0.01	0.12	h	0.05	0.45			
Cone	r	0.02	0.15	h	0.02	0.45			
Ellipsoid	w	0.03	0.12	d	0.03	0.12	h	0.03	0.12

In Section 4.1.2, we use a dataset consisting of boxes, cones, and ellipsoids divided into training/validation/test sets of size 720/240/240. The detail ranges of the aspect ratios of the shape parameters are shown in Table 6.

Table 6: The ranges of the shape parameters of the dataset used in Section 5.1.2

SHAPE	param	min	max	param	min	max
Elliptic cone	d/w	0.33	3	h/w	0.33	3
Ellipsoid	d/w	0.33	3	h/w	0.125	0.33
Box	d/w	0.33	3	h/w	0.33	3

C.2. Details for Experiments on Synthetic 3D Basic Shape Dataset

We used an encoder with a structure similar to the classification network used in DGCNN (Wang et al., 2019). The input point cloud with dimension 3×512 passes through five EdgeConv layers with point-wise latent space dimensions (64, 64, 128, 256) and a max pooling layer (we do not use a batch normalization layer unlike the original DGCNN classification network, but other settings are the same, e.g., $k = 20$, leaky relu activation), then we can obtain a 1024-dimensional feature

vector. Then this feature vector again passes through three fully-connected neural networks with dimensions (512, 256, 2) with leaky relu activation functions and linear output activation function; the latent space is two-dimensional. For the decoder model, we simply use a fully-connected neural network as the decoder. The two-dimensional vector on the latent space passes through three fully-connected neural networks with dimensions (256, 512, 3×512) with relu activation functions and linear output activation function; the output is a 3D point cloud with the number of points 512.

Section 5.1.1: To train the networks, we use ADAM with a learning rate of 0.001 and batch size of 16; the total number of the epochs is 500. The mean value of MEDs of the dataset is 0.0339, and we use the bandwidth value k to 0.5. We use Chamfer distance as the reconstruction loss; for regularization figures, the regularization term with the version of Equation (12) is multiplied by a coefficient $\lambda = 10^7$ with the info-Riemannian metric and by a coefficient $\lambda = 1$ with the Euclidean metric and added to the reconstruction loss term for each metric case. The value of η is set to be 0.0. For geodesic computation, we parametrize the curve $z(t)$ by a cubic spline with fixed boundary points z_1, z_2 and 10 control points. The control points are first initialized with equally spaced linear interpolants between z_1 and z_2. Then, for each iteration of optimization, we randomly sample 40 points on $t_i \sim U(0,1), i = 1, ..., 40$ and calculate an expectation $\frac{1}{40}\sum_{i=1}^{40} \dot{z}(t_i)^T G(z(t_i))\dot{z}(t_i)$ over the sampled points as the approximation of the objective function. We use ADAM with a learning rate of 0.001 and the total number of the iterations is 5000.

Section 5.1.2: To train the networks, we use ADAM with a learning rate of 0.001 and batch size of 16; the total number of the epochs is 3000. The mean value of MEDs of the dataset is 0.0341, and we use the bandwidth value k to 0.5. We use Chamfer distance as the reconstruction loss, and the regularization term with the version of Equation (12) is multiplied by a coefficient $\lambda = 10^7$ and added to the reconstruction loss term. The value of η is set to be 0.0.

To quantitatively evaluate how much the regularization approach improves class separability, more diverse synthetic datasets are made and experiments are conducted. We use datasets consisting of boxes, elliptic cones, and ellipsoids. In details, we generate short, normal, and tall shapes for each shape class, and the aspect ratios of the shape parameters are shown in Table 7. We conduct a total of 27 experiments with 3^3 combinations. Each dataset is divided into training/validation/test sets of size 720/240/240. Training configurations are the same with the above experiment, except that the mean values of MEDs are different to each other (but we consistently use the bandwidth value k to 0.5) and the total number of epochs is 500.

Table 7: The ranges of the shape parameters of the dataset used in quantitative analysis on synthetic dataset

SHAPE	param	min	max	param	min	max
Elliptic cone short	d/w	0.33	3	h/w	0.125	0.33
Elliptic cone normal	d/w	0.33	3	h/w	0.33	3
Elliptic cone tall	d/w	0.33	3	h/w	3	8
Ellipsoid short	d/w	0.33	3	h/w	0.125	0.33
Ellipsoid normal	d/w	0.33	3	h/w	0.33	3
Ellipsoid tall	d/w	0.33	3	h/w	3	8
Box short	d/w	0.33	3	h/w	0.125	0.33
Box normal	d/w	0.33	3	h/w	0.33	3
Box tall	d/w	0.33	3	h/w	3	8

In the obtained representation spaces, after fitting the Gaussian mixture model by using the training and validation data, the clustering scores are measured with the test data.

Color assigning method: To visually indicate which class generated point clouds belong to, we color these according to the ratio of the Chamfer distances to the nearest point cloud for each class. In detail, the smallest value (distance to nearest point cloud) is found by comparing the distance between the given point cloud and all point clouds of each class in the dataset. Since we are using 3 shape classes in both examples, we call the nearest distance to each class d_1, d_2, and d_3. After that, the vector $d = (d_1, d_2, d_3)$ is normalized with 2-norm so that the 2-norm of the vector to be 1. Finally, the value $0.2 \times \mathrm{Softmax}(1/d_1, 1/d_2, 1/d_3)$ is regarded as the ratio of the distances and a color is assigned to a given point cloud according to this ratio (i.e., linear weighted sum in the RGB coordinate).

C.3. Details for Experiments on Standard Benchmark Dataset

We use four different point cloud autoencoders: *FcNet*, *FoldingNet*, *PointCapsNet*, and *DGCNN-FcNet*; the latent space is 512-dimensional. For *FcNet* and *FoldingNet*, we use the exactly same point cloud autoencoder structures both adopted from (Yang et al., 2018b). For *PointCapsNet*, we also use the exactly same point cloud autoencoder structure adopted from (Zhao et al., 2019); we use 16×32 capsules to restrict the latent space to a reasonable size of 512. For *DGCNN-FcNet*, we use DGCNN classification network as encoder (i.e., the same encoder architecture used in experiments on synthetic 3D basic shape dataset, see Appendix C.2), and the same decoder structure from *FcNet* as decoder (i.e., three fully-connected neural networks with dimension (1024, 2048, 3×2048) with relu activation function and linear output activation function). To train the networks, we use ADAM with a learning rate of 0.0001, betas of $[0.9, 0.999]$, and weight decay of 0.000001 and batch size of 16; the total number of the epochs is 500. The mean value of MEDs of the dataset is 0.0356, and we use the bandwidth value k to 0.8. We use Chamfer distance as the reconstruction loss and regularization term with the version of Equation (13) with the value of η to be 0.2. The regularization term is multiplied by various coefficients, where the values of the regularization coefficients are summarized in Appendix C.

C.4. Details for Experiments on Standard Benchmark Dataset with Noise

We use the exactly same point cloud autoencoder structures adopted from (Yang et al., 2018b), *FcNet*, where the latent space is 512-dimensional. We add noise to each point $\mathbf{x}$ in point cloud of the dataset (ShapeNet, ModelNet10, and ModelNet40) according to $\mathbf{x} \mapsto \mathbf{x} + m\mathbf{v}$, where $\mathbf{v}$ is uniformly sampled on the unit sphere and m is sampled from the Gaussian distribution with zero mean and different levels of standard deviation (1%, 5%, 10%, and 20% of the diagonal length of the point cloud bounding box) as done in (Lee et al., 2021). The training configuration is the same with the case of Appendix C.3 except the followings. The regularization term is multiplied by $\lambda = 8000$. The mean values of MEDs of the dataset are 0.0320, 0.0364, 0.0442, and 0.579 for the cases of the noise levels 1%, 5%, 10%, and 20%, respectively, and we use the bandwidth value k to 0.8.

C.5. Details for Experiments on Standard Benchmark Dataset (Semi-Supervised Classification)

We train *FcNet* whose latent space is 512-dimensional with and without regularization. The training configuration is the same with the case of Appendix C.3 except the following: the regularization term of the regularized autoencoder (i.e., FcNet + I) is multiplied by $\lambda = 8000$. In this case, when we training linear SVM classifier, we use the different numbers of training data (1%, 5%, 10%, and 50% of the overall training data).

D. Additional Experimental Results

D.1. Synthetic Dataset

D.1.1. Qualitative Results of Table 1 in 4.1.2.

More examples related to the experiment in 4.1.2 are shown in Figure 9. The trend of the experimental results is similar to the experimental results in Section 4.1.2. For all five results, the gray ellipses are aligned well with the decision boundary in the vanilla autoencoder (we show the decision boundary in Figure 9 while it is not included in the main manuscript due to lack of space). In the regularized autoencoder, these gray ellipses (or Riemannian metrics) try to become isotropic, so the gaps on the decision boundaries get widened. As a result, different class clusters become farther away from each other.

D.1.2. Linear Interpolations using Regularized Autoencoders (related to Figure 5 in 4.1.1)

The generated point clouds from the representative intra-class linear interpolants between two cylinders and two cones with the regularized autoencoders under the Euclidean metric and info-Riemannian metric are drawn in Figure 10.

D.2. Standard Benchmark Dataset

D.2.1. Performance Analysis with Varying Regularization Coefficients

Figure 11 shows graphs of the classification accuracy versus reconstruction error for the trained AEs measured on ModelNet datasets, for a range of regularization coefficients. The reconstruction error is measured by the modified Chamfer distance as in (Yang et al., 2018b). Compared to vanilla autoencoders (red), regularized autoencoders under the info-Riemannian metric (blue) show overall higher classification accuracy regardless of the regularization coefficients. At the same time, they do not significantly increase the reconstruction error. On the other hand, when comparing the performance of regularized autoencoders under the Euclidean metric (green), most of these are clearly inferior to the vanilla autoencoder; the others perform even worse. Overall, regularization under the info-Riemannian metric is much more robust to the choice of regularization coefficients compared to using the Euclidean metric.

The linear SVM classification accuracy and reconstruction error (modified Chamfer distance) according to regularization coefficient are shown in Table 8 and Table 9. The tables are also arranged according to autoencoder models (FcNet vs. Foldingnet vs. PointCapsNet vs. DGCNN-FcNet) and regularization types (Vanilla vs. Euclidean vs. info-Riemannian).

D.2.2. Learning Curves for Noisy Point Cloud Data

Figure 12 shows how the linear SVM classification accuracy and reconstruction error (modified Chamfer distance) evolve as the training proceeds, where datasets are ModelNet10 and ModelNet40 and noise levels are 1%, 5%, 10%, and 20% (details about noise are in Appendix C.4). Compared to vanilla autoencoders (light colored lines), regularized autoencoders under Info-Riemannian metrc (dark colored lines) show overall higher classification accuracy, while they show similar levels of reconstruction errors. The increase in classification accuracy becomes more pronounced as the noise level increases. Especially, when the noise level is 10% and 20%, the classification accuracy of the vanilla autoencoder and regularized autoencoder under Euclidean metric gradually decreases as the learning progresses (as the epoch increases). However, such phenomenons do not appear in the regularized autoencoders under the Info-Rimennian metric. This result implies that our method is very advantageous in situations where there is noise in the data.

D.2.3. Learning Curves for Semi-Supervised Classification

Regularized autoencoders (i.e., FcNet + I) show overall higher classification accuracy compared to vanilla autoencoders (i.e., FcNet), while their reconstruction errors are not significantly different to vanilla autoencoders'. Moreover, the increase in classification accuracy becomes more pronounced as the label rate decreases. In other words, our performance is more effective as the number of labels decreases. Figure 13 shows how the linear SVM classification accuracy and reconstruction error evolve as the training proceeds, where label rate levels are 1%, 5%, 10%, and 50%. Also, similarly, when the label rate is 1%, the classification accuracy of the vanilla autoencoder gradually decreases as the learning progresses (as the epoch increases), but such phenomenons do not appear in the regularized autoencoders. This result implies that our method is also very advantageous in semi-supervised settings.

Table 8: Classification accuracy and reconstruction error according to regularization coefficient. The table is also arranged according to model (FcNet vs. FoldingNet) and regularization type (Vanilla vs. Euclidean vs. info-Riemannian). For Riemannian metric cases, the regularization coefficients used in the actual experiments are σ^2 times λ shown in the table.

MODEL	**METRIC**	λ	MD40 acc	MD40 recon	MD10 acc	MD10 recon
FcNet	Euclidean	0.0001	89.343598	0.029069	92.951542	0.029527
FcNet	Euclidean	0.0010	88.330632	0.030464	93.612335	0.030684
FcNet	Euclidean	0.0100	88.249595	0.029877	93.722467	0.030054
FcNet	Euclidean	0.1000	86.993517	0.029878	92.841410	0.030900
FcNet	Euclidean	1.0000	87.115073	0.031966	92.951542	0.034272
FcNet	Euclidean	10.0000	86.709887	0.031523	92.400881	0.032318
FcNet	Euclidean	100.0000	85.696921	0.031841	92.511013	0.035172
FcNet	Euclidean	1000.0000	85.899514	0.035089	92.400881	0.036145
FcNet	Euclidean	10000.0000	86.345219	0.034149	92.400881	0.034811
FcNet	Riemannian	100.0000	89.586710	0.029032	94.052863	0.029435
FcNet	Riemannian	500.0000	89.829822	0.028944	94.273128	0.029928
FcNet	Riemannian	1000.0000	89.951378	0.028864	93.722467	0.029430
FcNet	Riemannian	2000.0000	90.194489	0.029165	94.052863	0.029606
FcNet	Riemannian	8000.0000	90.397083	0.028869	93.722467	0.028944
FcNet	Riemannian	10000.0000	89.991896	0.029603	94.162996	0.030278
FcNet	Riemannian	20000.0000	89.748784	0.028980	93.832599	0.029412
FcNet	Riemannian	50000.0000	89.667747	0.029161	93.392070	0.028894
FcNet	Riemannian	100000.0000	89.748784	0.028961	93.942731	0.030085
FcNet	Vanilla	0.0000	88.330632	0.028938	93.502203	0.029895
FoldingNet	Euclidean	0.0001	88.897893	0.030540	94.162996	0.029130
FoldingNet	Euclidean	0.0010	87.844408	0.031623	94.273128	0.031726
FoldingNet	Euclidean	0.0100	87.520259	0.029126	93.612335	0.032219
FoldingNet	Euclidean	0.1000	87.641815	0.029391	93.722467	0.030318
FoldingNet	Euclidean	1.0000	88.128039	0.030564	93.171806	0.031435
FoldingNet	Euclidean	10.0000	88.290113	0.032129	93.171806	0.032975
FoldingNet	Euclidean	100.0000	88.087520	0.032728	94.383260	0.032885
FoldingNet	Euclidean	1000.0000	87.722853	0.033305	92.951542	0.036244
FoldingNet	Euclidean	10000.0000	87.844408	0.033546	93.502203	0.036820
FoldingNet	Riemannian	100.0000	89.667747	0.029467	94.052863	0.030789
FoldingNet	Riemannian	500.0000	90.113452	0.029916	94.052863	0.029346
FoldingNet	Riemannian	1000.0000	89.870340	0.030090	94.493392	0.029996
FoldingNet	Riemannian	2000.0000	89.546191	0.030471	94.273128	0.031142
FoldingNet	Riemannian	8000.0000	89.627229	0.028949	94.162996	0.029745
FoldingNet	Riemannian	10000.0000	89.505673	0.029926	94.052863	0.033548
FoldingNet	Riemannian	20000.0000	89.384117	0.032798	94.162996	0.029463
FoldingNet	Riemannian	50000.0000	89.708266	0.029262	94.273128	0.030002
FoldingNet	Riemannian	100000.0000	89.262561	0.029511	94.162996	0.030355
FoldingNet	Vanilla	0.0000	89.343598	0.029599	93.722467	0.030528

Table 9: Classification accuracy and reconstruction error according to regularization coefficient. The table is also arranged according to model (PointCapsNet vs. DGCNN-FcNet) and regularization type (Vanilla vs. Euclidean vs. info-Riemannian). For Riemannian metric cases, the regularization coefficients used in the actual experiments are σ^2 times λ shown in the table.

MODEL	**METRIC**	λ	MD40 acc	MD40 recon	MD10 acc	MD10 recon
PointCapsNet	Euclidean	0.0001	88.087520	0.039522	93.722467	0.043112
PointCapsNet	Euclidean	0.0010	87.601297	0.046227	93.171806	0.047749
PointCapsNet	Euclidean	0.0100	87.155592	0.058326	92.621145	0.060827
PointCapsNet	Euclidean	0.1000	86.628849	0.070735	91.519824	0.072615
PointCapsNet	Euclidean	1.0000	85.858995	0.087722	91.299559	0.087930
PointCapsNet	Euclidean	10.0000	83.954619	0.110708	90.638767	0.107402
PointCapsNet	Euclidean	100.0000	79.659643	0.124266	88.215859	0.115549
PointCapsNet	Euclidean	1000.0000	76.823339	0.128694	88.546256	0.121284
PointCapsNet	Euclidean	10000.0000	75.567261	0.130566	88.325991	0.121398
PointCapsNet	Riemannian	100.0000	88.492707	0.035034	93.942731	0.042224
PointCapsNet	Riemannian	1000.0000	88.168558	0.035816	93.942731	0.039131
PointCapsNet	Riemannian	2000.0000	87.884927	0.036824	93.392070	0.040169
PointCapsNet	Riemannian	5000.0000	87.884927	0.035990	93.832599	0.039729
PointCapsNet	Riemannian	8000.0000	87.641815	0.037986	93.392070	0.040947
PointCapsNet	Riemannian	10000.0000	87.763371	0.037507	93.722467	0.040428
PointCapsNet	Riemannian	20000.0000	87.763371	0.038454	93.281938	0.048909
PointCapsNet	Riemannian	50000.0000	87.641815	0.042233	93.281938	0.044333
PointCapsNet	Riemannian	100000.0000	87.317666	0.046607	93.171806	0.051187
PointCapsNet	Vanilla	0.0000	87.155592	0.033936	93.612335	0.037172
DGCNN-FcNet	Euclidean	0.0001	89.910859	0.028852	94.052863	0.029049
DGCNN-FcNet	Euclidean	0.0010	89.546191	0.029489	94.383260	0.029480
DGCNN-FcNet	Euclidean	0.0100	88.492707	0.030448	93.061674	0.030531
DGCNN-FcNet	Euclidean	0.1000	87.884927	0.030046	93.392070	0.030966
DGCNN-FcNet	Euclidean	1.0000	87.520259	0.030605	93.502203	0.034671
DGCNN-FcNet	Euclidean	10.0000	87.196110	0.032174	92.841410	0.032151
DGCNN-FcNet	Euclidean	100.0000	87.277147	0.032385	93.171806	0.033750
DGCNN-FcNet	Euclidean	1000.0000	87.682334	0.033359	93.171806	0.033906
DGCNN-FcNet	Euclidean	10000.0000	86.385737	0.034471	92.621145	0.036056
DGCNN-FcNet	Riemannian	100.0000	90.397083	0.028761	94.493392	0.028591
DGCNN-FcNet	Riemannian	1000.0000	90.964344	0.028278	94.493392	0.028806
DGCNN-FcNet	Riemannian	2000.0000	90.680713	0.028299	94.493392	0.028832
DGCNN-FcNet	Riemannian	5000.0000	90.883306	0.028520	94.493392	0.029128
DGCNN-FcNet	Riemannian	8000.0000	90.802269	0.028727	95.154185	0.028971
DGCNN-FcNet	Riemannian	10000.0000	90.680713	0.028583	94.383260	0.028820
DGCNN-FcNet	Riemannian	20000.0000	90.559157	0.028677	94.713656	0.029067
DGCNN-FcNet	Riemannian	50000.0000	90.761750	0.029794	94.713656	0.029695
DGCNN-FcNet	Riemannian	100000.0000	90.235008	0.028934	94.052863	0.031275
DGCNN-FcNet	Vanilla	0.0000	90.275527	0.028867	94.493392	0.029454

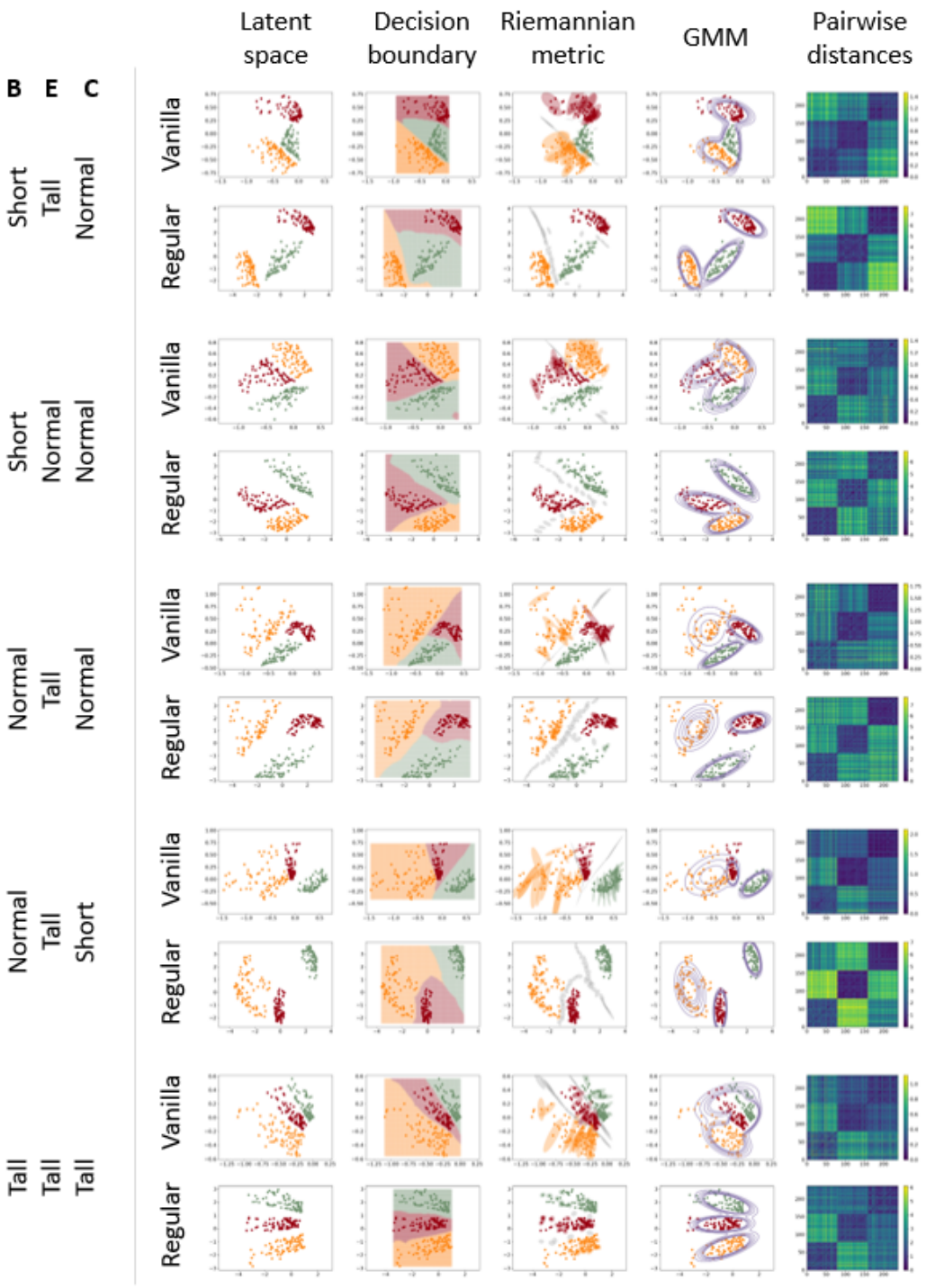

Figure 9: The representative five examples of the regularization experiments on the synthetic dataset. From left to right: latent spaces, decision boundary according to the color assigning method introduced in Appendix C.2, latent spaces with equidistant ellipse ($\{z|(z-z^*)^T G(z^*)(z-z^*)=1\}$ for center z^*) centered on some selected points and sampled points from interspaces, Gaussian Mixture Model (GMM) fitting results, and the heat map of the pairwise Euclidean distances in the latent space of all test data. For each experiment, the upper figure is a vanilla autoencoder trained without regularization, while the lower figure is trained with regularization.

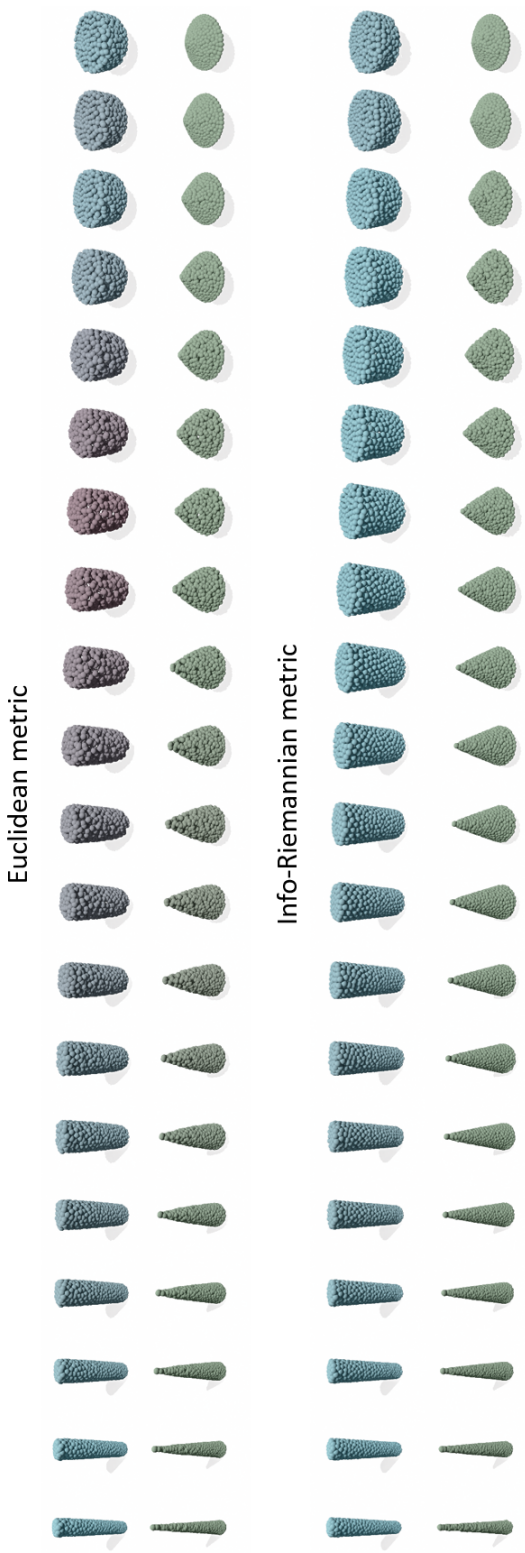

Figure 10: The generated point clouds from the linear interpolants of the regularized autoencoders with the Euclidean metric (*Upper*) and info-Riemannian metric (*Lower*).

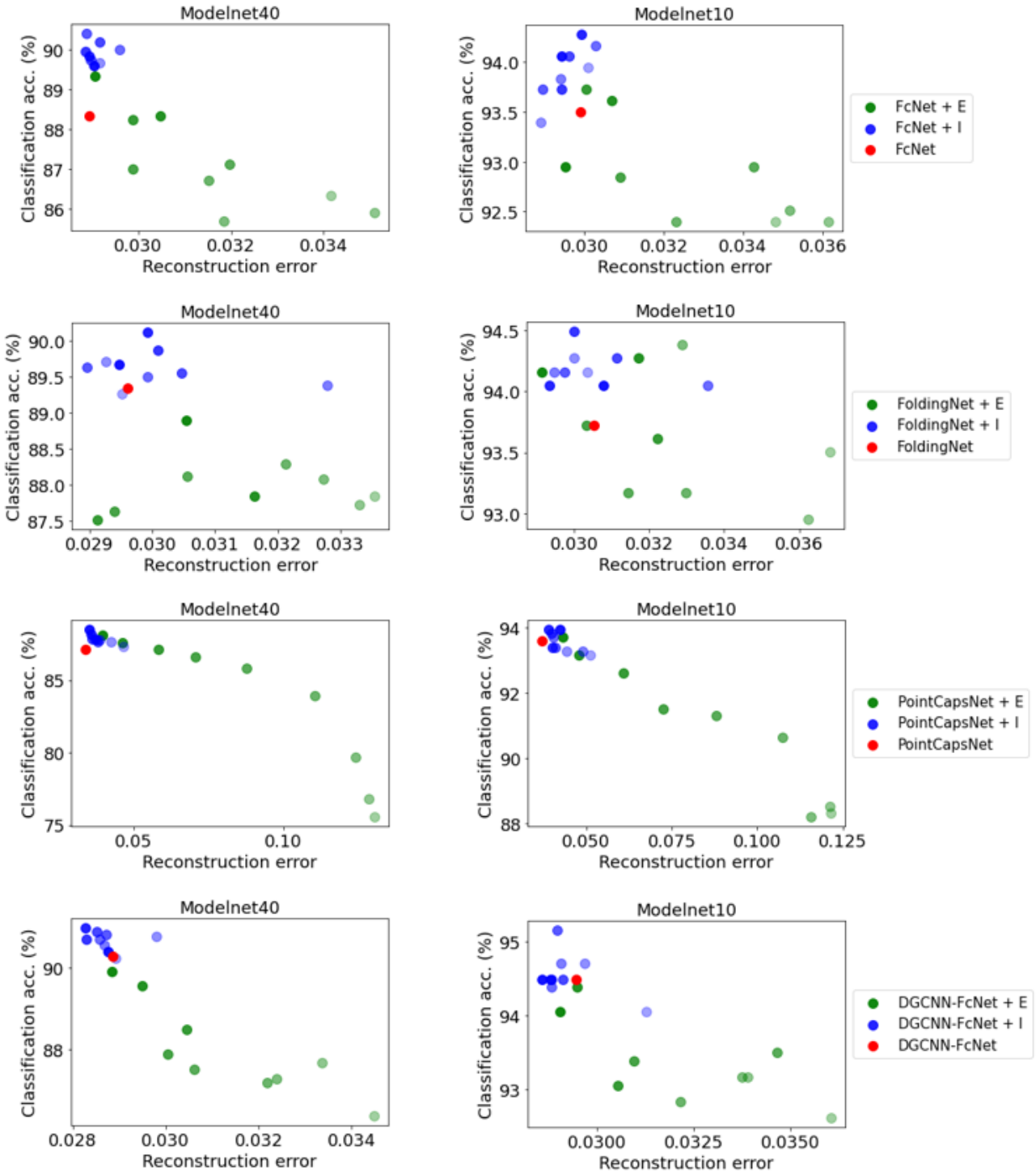

Figure 11: Graphs of classification accuracy versus reconstruction error measured on ModelNet datasets. More transparent markers have larger coefficients λ; detailed values are in Table 8 and Table 9.

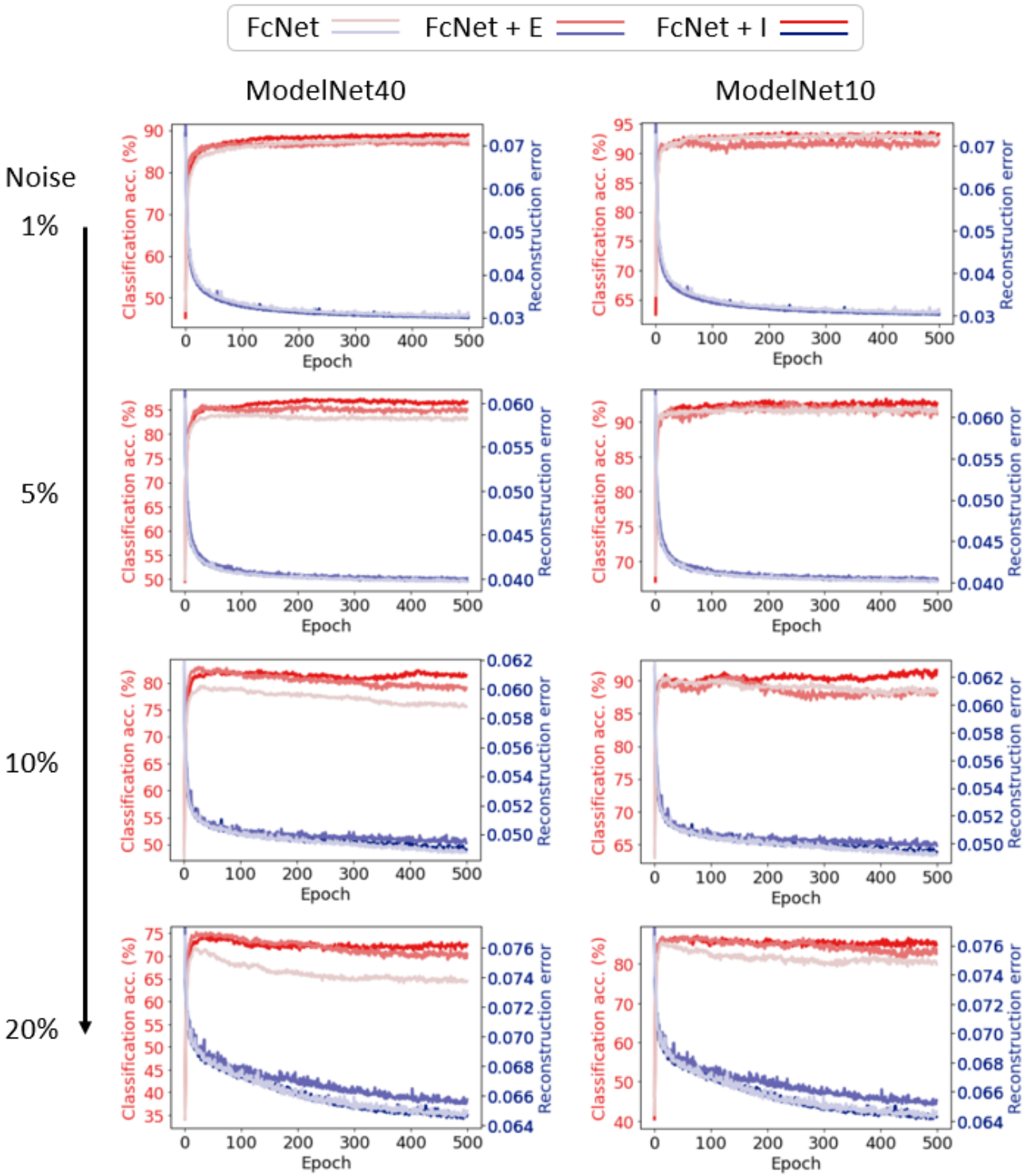

Figure 12: Learning curves of classification accuracy and reconstruction error measured on ModelNet datasets (ModelNet40 and ModelNet10) according to the noise levels (1%, 5%, 10%, and 20%). In each plot, the light colored lines are the result of the non-regularized autoencoders (i.e., FcNet), and the dark colored lines are the result of the regularized autoencoders (i.e., FcNet + E and FcNet + I).

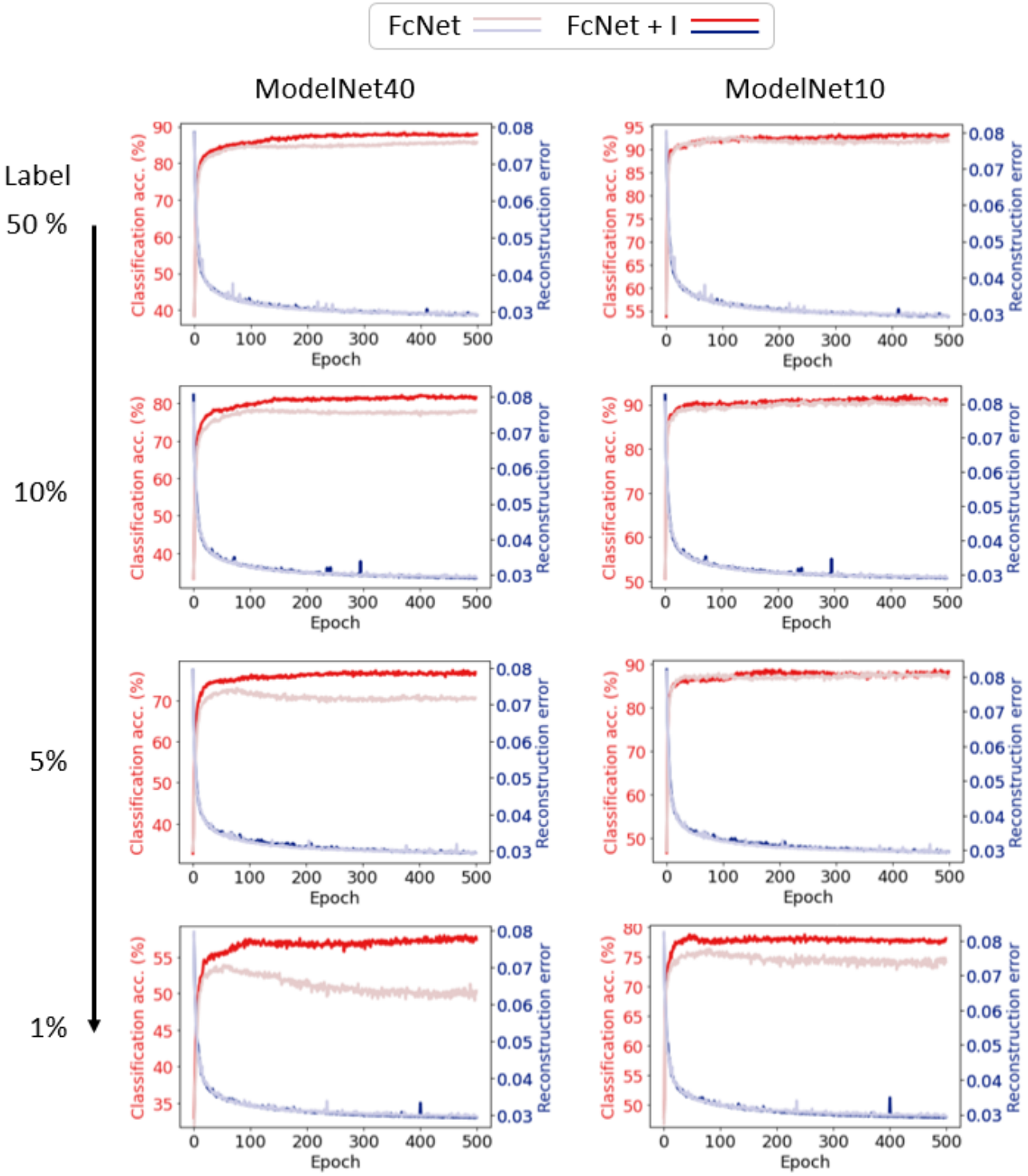

Figure 13: Learning curves of classification accuracy and reconstruction error measured on ModelNet datasets (ModelNet40 and ModelNet10) according to the label rates (50%, 10%, 5%, and 1%). In each plot, the light colored lines are the result of the non-regularized autoencoders (i.e., FcNet), and the dark colored lines are the result of the regularized autoencoders (i.e., FcNet + I).