Geometric Autoencoders – What You See is What You Decode

Philipp Nazari [1] **Sebastian Damrich** [1 2] **Fred A. Hamprecht** [1]

Abstract

Visualization is a crucial step in exploratory data analysis. One possible approach is to train an autoencoder with low-dimensional latent space. Large network depth and width can help unfolding the data. However, such expressive networks can achieve low reconstruction error even when the latent representation is distorted. To avoid such misleading visualizations, we propose first a differential geometric perspective on the decoder, leading to insightful diagnostics for an embedding's distortion, and second a new regularizer mitigating such distortion. Our "Geometric Autoencoder" avoids stretching the embedding spuriously, so that the visualization captures the data structure more faithfully. It also flags areas where little distortion could not be achieved, thus guarding against misinterpretation.

1. Introduction

The acquisition of larger and more complex datasets – with a dimensionality of a few thousand, for example in bioinformatics (Zheng et al., 2017; Zilionis et al., 2019; Packer et al., 2019) – has boosted the development of recent machine learning algorithms. While such high dimensionality allows for encoding an increasing amount of information, it also makes human interpretation more difficult. A common method for exploring high-dimensional datasets is two- or three-dimensional visualization.

Today's state-of-the-art algorithms for dimensionality reduction are UMAP (McInnes et al., 2018; Damrich & Hamprecht, 2021) and t-SNE (van der Maaten & Hinton, 2008; van der Maaten, 2014; Kobak & Berens, 2019). Both UMAP and t-SNE tend to preserve local structure, which gives them great unfolding power, at the expense of rendering global structure faithfully (see Figure 1a).

In contrast to standard UMAP and t-SNE, autoencoders (Hinton & Salakhutdinov, 2006) find representations that afford approximate reconstruction of the original, high-dimensional dataset. They also allow the embedding of additional measurements after training. If the autoencoder is linear, it reduces to PCA (Pearson, 1901; Kramer, 1991; Plaut, 2018), see also Appendix B.1. Non-linear autoencoders have much greater representational power. In Figure 1a we show that autoencoders can produce meaningful maps of the globe where PCA (projecting New Zealand onto Italy) and t-SNE (distorting land mass beyond recognition) both fail.

While non-linearity enables autoencoders to unfold the data, it can also hinder the interpretability of an autoencoder's latent representation. Powerful encoders can introduce distortions that equally powerful decoders can resolve, leaving the reconstruction loss unaffected. The result is a misleading embedding with near perfect reconstruction, subverting the idea that a 2D latent space forces only the most salient features to be visualized. In deep networks this defect can be amplified to the point that even a simple dataset, such as the *Earth* dataset (see Appendix D.1) in Figure 1a, can become hardly recognizable. In the vanilla autoencoder's embedding, South and North America each seem to be bigger than Eurasia and Africa combined. Nevertheless, the autoencoder achieves low reconstruction loss, because the decoder contracts the Americas while expanding the rest of the world, so that the reconstruction accurately reflects the continents' actual sizes.

We take an intuitive geometric approach to different methods of measuring how encoder and decoder introduce distortion in the embedding, which assist the practitioner in understanding a given embedding. To make sure that what we see in the embedding is closer to what the decoder reconstructs, and thus to the structure of the dataset, we propose taming the decoder's geometric properties by constructing a regularizer that pushes the decoder towards being area-preserving. This approach is similar to the one in Lee et al. (2022) which encourages the decoder to be a scaled isometry – a more restrictive property than area-preservation.

The decoder maps the low-dimensional latent space to the

[1]HCI/IWR at University of Heidelberg, 69120 Heidelberg, Germany [2]University of Tübingen, 72074 Tübingen, Germany. Correspondence to: Philipp Nazari <philipp.nazari@gmail.com>, Sebastian Damrich <sebastian.damrich@uni-tuebingen.de>, Fred Hamprecht <fred.hamprecht@iwr.uni-heidelberg.de>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

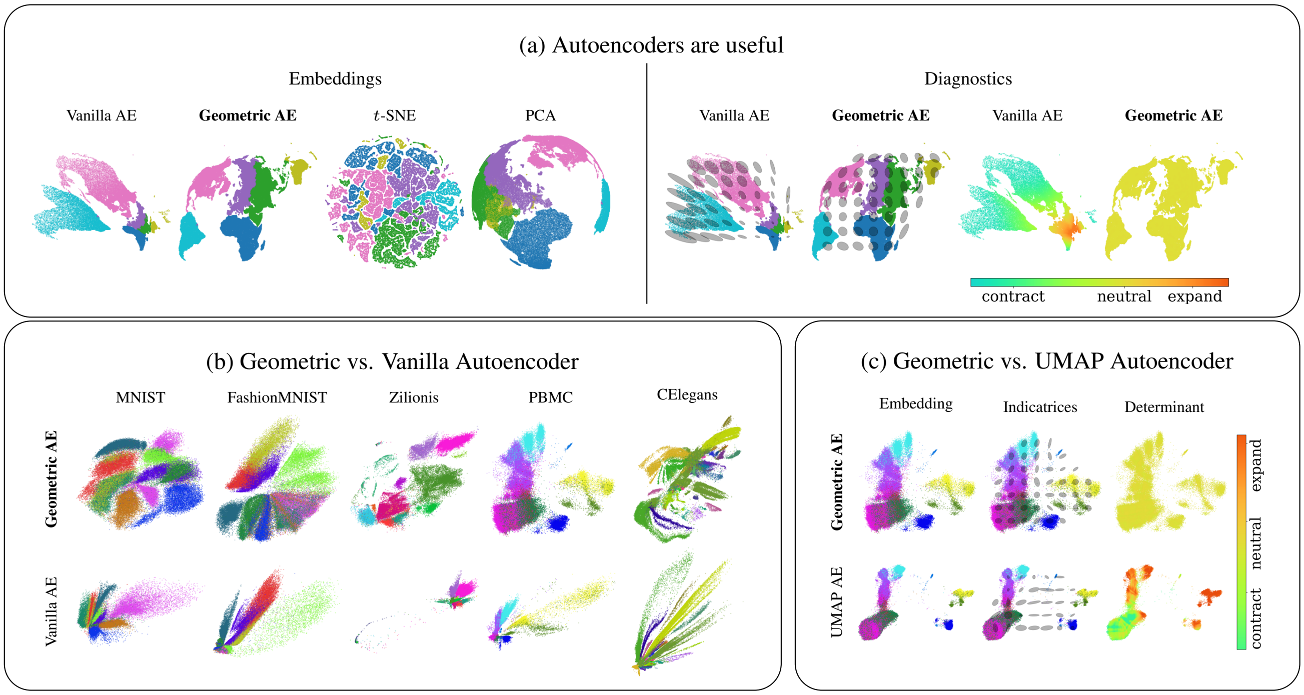


Figure 1. Panel (a) provides an example for the usefulness of autoencoders for visualization. While autoencoders are able to unravel the Earth dataset, t-SNE disregards global structure, and PCA projects Eurasia onto Australia. The geometric autoencoder ensures that the relative sizes of the continents are much better preserved than by the vanilla autoencoder. We use diagnostics, indicatrices and a determinant plot, to demonstrate that our geometric autoencoder's embedding is more faithful, contracting more homogeneously. Panel (b) shows how our geometric regularizer improves upon a vanilla autoencoder on a number of datasets. Panel (c) compares our geometric autoencoder with a UMAP autoencoder on the PBMC dataset. The diagnostics show that the UMAP decoder expands much more inhomogeneously than our proposed method, distorting the relative cluster sizes.

high-dimensional output space. If, for example, we consider a two-dimensional latent space and a three-dimensional ambient space, the autoencoder learns to place a curved surface into $\mathbb{R}^3$ which, in a suitable measure, best approximates the higher dimensional dataset. Intuitively, one can visualize the decoder's task as fitting a surface into output-space, stretching it arbitrarily. While some stretching might be necessary to approximate the dataset well, excessive stretching introduces unnecessary distortions in latent space. Loosely speaking, the geometric regularizer makes the surface resist stretching intrinsically.

We propose visualizing the decoder's expansion, which we can think of as the surface's stretching, by a heatmap of the *generalized Jacobian determinant* (closely related to the "Riemannian Volumeform" (Lee, 2000) and to the work of Chen et al. (2018)) and by *indicatrices* (Laskowski, 1989; Brun & Knutsson, 2009). While the generalized Jacobian determinant measures the decoder's undirected contraction, indicatrices additionally show its anisotropy. Their size and elongation enable the practitioner to more faithfully interpret any autoencoder, see Figures 1a, c. We further endow latent-space with a *pullback metric* which allows us to measure and mitigate the decoder's variance in contraction.

To sum up, we propose diagnostic tools for visualizing local distortion of two-dimensional autoencoders and construct a geometric regularizer reducing those distortions, leading to a more faithful embedding. We provide the code as an open-source package for PyTorch. It can be found at https://github.com/hci-unihd/GeometricAutoencoder.

2. Preliminaries

2.1. Problem Setting

Throughout this work, we assume that there is a dataset X living in some high-dimensional Euclidean space $\mathbb{R}^n$. We view an autoencoder as a concatenation of two functions $\mathbb{R}^n \xrightarrow{E} \mathbb{R}^l \xrightarrow{D} \mathbb{R}^n$, where E is the encoder, D the decoder and $l < n$ is the dimensionality of latent space. Both decoder and encoder are realized as (deep) neural networks, which are jointly trained to minimize the ℓ_2 loss between the dataset and its reconstruction. Under some mild assumptions (Section 7), the decoder's image defines an l-dimensional manifold M (the "reconstruction manifold") living in $\mathbb{R}^n$ with an atlas consisting of only a single chart, the decoder's inverse $D^{-1}\colon M \to \mathbb{R}^l$. The encoder $E\colon \mathbb{R}^n \to \mathbb{R}^l$ can be seen as placing an input point onto the codomain of the chart. The decoder thus defines the manifold on which the autoencoder can place reconstructions. The encoder specifies the position on the manifold by outputting the position on the global chart. During training, updating the decoder changes the reconstruction manifold, while updating the encoder changes the position on the chart and thus on the manifold.

If an autoencoder was trained to optimal reconstruction loss, it would essentially project data points orthogonally to the reconstruction manifold, see Appendix E.1. Even in this case, the encoder could still locally stretch and contract the embedding as long as the decoder undoes these distortions. A priori, this is not visible in the embedding space. For example, the vanilla autoencoder's embedding of the Earth dataset in Figure 1a disproportionally expands the Americas. Despite this visual distortion, the reconstruction loss is half that of the geometric autoencoder. This is only possible if the decoder contracts the enlarged embedding of the Americas again. We will present a way of measuring such avoidable contraction and ultimately mitigating it as much as possible.

It is known from multivariate calculus that the Jacobian determinant of a continuously differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}^n$ at a point $p \in \mathbb{R}^n$ measures how f transforms an infinitesimal volume centered at p. In order to develop the concept of a Jacobian determinant for the decoder, which in general acts between spaces of different dimensionality, we first generalize the ordinary case to smooth immersions $F\colon M \to N$, smooth maps with injective differential everywhere, between manifolds. This requires some machinery which we introduce in the following paragraph.

2.2. A Note on Differential Geometry

In this section we introduce basic concepts from differential geometry. For a detailed treatment, see Lee (2018; 2000).

One of the core concepts from differential geometry is that of a *(smooth) manifold*, a space that locally looks like Euclidean Space; it can be covered by open sets U, each of which is homeomorphic to an open subset of $\mathbb{R}^n$. Such a homeomorphism is called a *chart*. Furthermore, we require the transition maps between charts to be diffeomorphisms. The directions of a manifold M at a point $p \in M$ are captured by the *tangent space* T_pM at p, which can be thought of as the best linear approximation of the manifold.
A smooth map $f\colon M \to N$ between two manifolds can be linearly approximated around each point $p \in M$. This approximation is called the *differential* of f at p, denoted by d_pf, and maps from the tangent space corresponding to p to that of its image, $d_pf\colon T_pM \to T_{f(p)}N$. In coordinates, it is given by the Jacobian matrix J_pF of F at p.
Distances and angles at a point p of a Riemannian manifold are determined by the *metric tensor*, a bilinear, positive definite map acting on the tangent space at p. The Euclidean metric tensor g_e is given by the Euclidean inner product.

2.3. The Generalized Jacobian Determinant

In this section we generalize the concept of a Jacobian determinant to smooth immersions $F\colon M \to N$ between manifolds of dimension m and n, from which the ordinary case emerges as a special case.

Assume (N, g) to be a Riemannian manifold, then F induces a volume form on $F(M)$ in the following way:

Proposition 2.1. *Assume F is a diffeomorphism onto its image and M is oriented. Then there exists a volume form ω_g on $F(M)$, the Riemannian volume form, which in the smooth oriented coordinates $x_1, ..., x_l$ induced by F is given by* $\omega_g = \sqrt{\det\left[(J_pF)^t\, J_pF\right]}dx_1 \wedge ... \wedge dx_l$.

Proof. Use Proposition 15.6 and 15.31 in Lee (2000). □

The square-root factor of the Riemannian volume form shows how volumes are changed locally by F, and can thus be seen as a generalization of the Jacobian determinant. We call its square the *generalized Jacobian determinant*, which captures information about the distortion of angles and directed stretching. This gives rise to the "pullback metric", which we introduce in the next paragraph.

2.4. The Pullback Metric

In order to faithfully interpret the latent space of an autoencoder, it is crucial to know how angles and distances would appear after decoding. This can be achieved by equipping latent space with a metric tensor that measures angles and distances as they would be mapped to the output manifold. The resulting metric tensor on latent space is the *pullback metric* (Lee, 2000). See Figure 2a for an illustration.

To construct the pullback metric tensor, we endow ambient space with the Euclidean metric g_e, making it a Riemannian Manifold $(\mathbb{R}^n, g_e)$. An immersion $F\colon \mathbb{R}^l \to \mathbb{R}^n$, which we will later choose to be the decoder D, induces a pullback metric F^*g_e on its domain in the following way: Given a point $p \in \mathbb{R}^l$ and two tangent vectors $v, w \in T_p\mathbb{R}^l$, their inner product in the pullback metric is defined as the inner product of their images under the decoder's differential,

$$F^*g_{e_p}(v, w) := g_{e_{F(p)}}(d_pFv, d_pFw), \tag{1}$$

where $d_pF\colon T_p\mathbb{R}^l \to T_{F(p)}\mathbb{R}^n$ is the differential of F at p.

In coordinates, this pullback metric takes a very simple form, just depending on the Jacobian of F:

Proposition 2.2 (Pullback Metric in Coordinates)**.** *The pullback of g_e under F at $p \in \mathbb{R}^l$ is in coordinates given by*

$$\langle \cdot, \cdot \rangle_p := F^* g_{e_p} = (J_p F)^t \, J_p F \in \mathbb{R}^{l,l}, \tag{2}$$

where $J_p F \in \mathbb{R}^{n \times l}$ is the Jacobian matrix of F at p.

Proof. See Appendix B.2. □

Equation (2) indicates the connection between the pullback metric and the generalized Jacobian determinant introduced in Section 2.3. Indeed, the pullback metric measures lengths in latent space as lengths along the immersed manifold.

2.5. Indicatrices

While the generalized Jacobian determinant provides information about the decoder's undirected contraction, it does not tell us anything about its isotropy or directed contraction. Therefore, we propose visualizing the pullback metric tensor fields using "indicatrices" (Laskowski, 1989; Brun & Knutsson, 2009). Consider a smooth immersion F between two manifolds M and N, as in the setting of Section 2.3.

Definition 2.3 (Indicatrix)**.** An *indicatrix* at a point $p \in M$ is the unit sphere in the pullback metric induced by F at p.

Since the differential linearly approximates F at p, we may think of an indicatrix as consisting of those points around p which F approximately maps to a unit sphere around $F(p)$. An indicatrix centered at p thus tells us which directions are squeezed and which are expanded. It makes distortions originating from the function F visible. A set of indicatrices, distributed over the dataset, allows to identify regions which are contracted or expanded as well as the direction of the stretching. See Figure 2b for a visual explanation.

2.6. Application to Autoencoders

For the geometric autoencoder, we equip the decoder's image with the restriction of the Euclidean metric g_e, which we then pull back using the decoder D. In particular, all the above applies to the special case where $F = D$ is the decoder. For limitations of our method, see Section 7.

3. Geometric Autoencoders

Unregularized autoencoders tend to contract the embedding inhomogeneously. In this section, we discuss diagnostics for this distortion, as well as a regularizer mitigating the variance in contraction. See Figure 1 for an overview.

3.1. Diagnostics

3.1.1. Generalized Jacobian Determinant

To prevent misinterpreting an embedding due to inhomogeneous contraction of the decoder, we propose highlighting areas in latent space based on the generalized Jacobian determinant, which we plot as a heat map on the embedding as opposed to the background shading in Chen et al. (2018). This helps interpreting embeddings more faithfully: In Figure 1a, the determinant plot reveals that the heavily clustered data lies in an area which the decoder expands. Thus, one can infer that Europe, Russia and Africa combined are not actually smaller than each of the two Americas.

3.1.2. Indicatrices

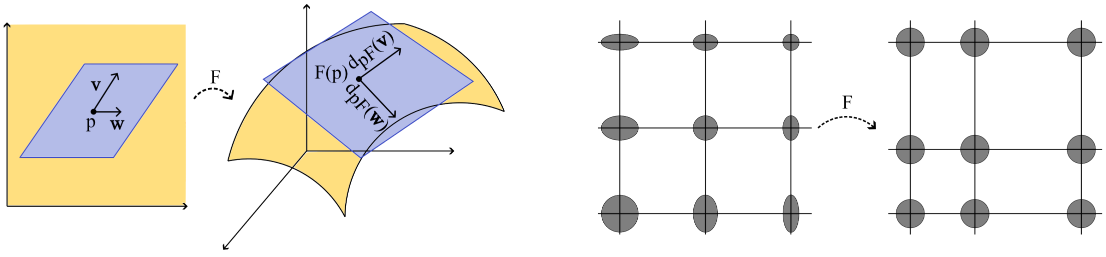
Given a decoder $D\colon \mathbb{R}^l \to \mathbb{R}^n$, we approximate the indicatrix at $p \in \mathbb{R}^l$ by the convex hull of the family of vectors $v_i / \sqrt{\langle v_i, v_i \rangle_p}$, where the v_i are sampled uniformly from the Euclidean unit circle at p. As a visualization technique, we plot for a given point p in latent space the convex hull of the vectors v_i as a patch around p. See Figure 2b for an example. The points p are chosen as a regular grid in the convex hull of the embedding. The inhomogeneity of the embedding is reflected in the indicatrices. The variance of the decoder's undirected contraction is indicated by the variance of the indicatrices' volumes. For example, in Figure 1a the vanilla decoder's indicatrices on Europe are smaller than those on North America (Figure 1a). The decoder's directed contraction is encoded by the shape of a single indicatrix; for the vanilla decoder's in Figure 1a most of them are elongated towards Europe, indicating that the decoder has to contract in that direction in order to reconstruct the dataset. A locally isotropic decoder, for example the one of PCA, has round indicatrices (Figure S3, column 4).

3.2. Regularization

We discussed above how the generalized Jacobian determinant measures the local contraction and expansion of a decoder. A faithful embedding avoids any stretching unnecessary for reconstruction. Therefore, it is natural to regularize the decoder to have uniform generalized Jacobian determinant. To achieve that, we calculate the generalized Jacobian determinant at every embedding point in the embedding of a minibatch B and calculate the variance of their logarithm. This defines our regularizer $\mathcal{L}_{\det}$,

$$\mathcal{L}_{\det} = \mathrm{Var}_{x \sim \mathcal{U}(B)} \left[\log \left(\det \left(J_{E(x)} D \right)^t J_{E(x)} D \right) \right]. \tag{3}$$

The total loss amounts to $\mathcal{L} \to \mathcal{L}_{\text{rec}} + \alpha \mathcal{L}_{\det}$, where α is a hyperparameter controlling the importance of the regularizer compared to the usual reconstruction loss $\mathcal{L}_{\text{rec}}$, typically the mean squared error. In Appendix D.5 we explain why gradients resulting from the regularizer are propagated through



(a) Pullback Metric (Concept)

(b) Indicatrices (Concept)

Figure 2. Panel (a) illustrates the pullback metric along a smooth map $F\colon \mathbb{R}^2 \to \mathbb{R}^3$. Given a point $p \in \mathbb{R}^2$ and two vectors u and v from the tangent space at p (purple), their product in the pullback metric is defined as the product of their images under F's differential d_pF (on the right). While v and w are not orthonormal in the euclidean metric, they are orthonormal in the pullback metric, since their images are. Panel (b) illustrates indicatrices. Consider a map $F\colon \mathbb{R}^2 \to \mathbb{R}^2$, $(x, y) \mapsto (x^2, y^2)$ which distorts a regular grid as displayed. The shape of the indicatrices makes this distortion visible in the input space.

both the encoder and the decoder.

Computing the variance of logarithms ensures that the autoencoder cannot minimize the secondary objective by globally expanding the embedding:

Lemma 3.1 (Scale Invariance of the Regularizer)**.** *If the decoder scales with a factor $\beta \in \mathbb{R}\setminus\{0\}$, the objective $\mathcal{L}_{\mathrm{det}}$ stays invariant.*

Proof. See Appendix B.3. □

4. Related Work

Since the invention of autoencoders (Rumelhart et al., 1986) and their application to visualization, see e.g. Hinton & Salakhutdinov (2006), numerous regularizations have been proposed to avoid over-fitting.

Two popular strategies are contractive autoencoders (Rifai et al., 2011) and denoising autoencoders (Vincent et al., 2010). Both are geared towards classification, rather than visualization, as they aim to produce locally constant embeddings. Sparse autoencoders (Ng et al., 2011; Makhzani & Frey, 2014) are regularized to have sparse hidden activations instead of compressing to a bottleneck dimension, making them unsuitable for visualization as well. Therefore, we omitted these three classical regularized autoencoders from our quantitative evaluation.

Variational autoencoders (Kingma & Welling, 2014; 2019) are tailored towards generating samples from a prespecified prior distribution in latent space, typically a Gaussian. As a result, the embeddings are usually densely packed together to allow smooth interpolation. This is not ideal for visualization, especially of clustered datasets. Ghosh et al. (2020) suggest to replace the variational framework with various deterministic regularizers. However, none of them directly address the geometric properties of the embedding. Furthermore, Ghosh et al. (2020) do not consider visualization.

More similar to our method are topological autoencoders (Moor et al., 2020). Effectively, they encourage the encoder to preserve the minimum spanning tree of the dataset. Recently, Trofimov et al. (2023) proposed to regularize autoencoders based on a more refined method for comparing the topology between point clouds. Instead of regularizing the topology of the embedding, our proposed method addresses the geometry. Other works in this area have tried to turn the decoder into an isometry, a map that preserves pairwise distances on a local scale. The Markov-Lipschitz autoencoder (Li et al., 2020) directly regularizes local distances and across several layers of the network. Isometric autoencoders (Gropp et al., 2020) try to achieve isometry of the decoder instead by preserving the norm of Monte-Carlo sampled unit vectors in latent space under multiplication with the decoder's Jacobian. The work of Chen et al. (2020) regularizes the pullback metric tensor directly via a Frobenius norm, but their coordinate-dependent measure has a bias for decoders with Jacobian of small norm (Lee et al., 2022). Instead, Lee et al. (2022) propose a different regularization of the pullback metric that induces the decoder to become a scaled isometry. Our approach aligns closely with that of Lee et al. (2022), but is less restrictive and only encourages the decoder to become area-preserving, see Appendix C.

Other regularized autoencoders include neighborhood reconstructing autoencoders (Lee et al., 2021) that try to reconstruct neighborhoods of data points by local approximation of the decoder. The geometry regularized autoencoders of Duque et al. (2022) regularize the latent layer to stay close to a previously computed embedding, e.g., a neighbor embedding with UMAP. Our method does not employ neighborhood relations, but only uses the reconstruction of individual point and our geometric regularizer. The recently proposed geometrically regularized autoencoder of Jang et al. (2023) extends denoising and contractive autoencoders to the setting where the data and possibly the latent space are known, non-Euclidean Riemannian manifolds. Crucial for visualization, we only consider 2D Euclidean latent spaces. While not explored here, our work readily applies to a general Riemannian data space by pulling back its metric tensor to latent space.

Improving the structure of latent space activations in the supervised setting is an active area of research, too (Zhao et al., 2018; Scott et al., 2021).

The most popular non-parametric dimensionality reduction algorithms are the neighbor embedding methods t-SNE (van der Maaten & Hinton, 2008; Kobak & Berens, 2019; van der Maaten, 2014) and UMAP (McInnes et al., 2018). Their relation is discussed in Damrich et al. (2023). Both t-SNE and UMAP usually do not include a decoder. There is, however, a parametric implementation of UMAP, which can be implemented as an autoencoder with UMAP loss on the embedding (Sainburg et al., 2021). In this setup, our diagnostics revealed that UMAP embeddings have significant variance in local contraction and expansion, see Figures S2 and S3.

We use indicatrices for visualizing the pullback metric. These are related to Tissot indicatrices (Laskowski, 1989), commonly used to visualize distortions in world maps, and Tensor Glyphs (Brun & Knutsson, 2009). Both of those methods are more complicated than ours which is more in line with the equidistance-lines and -plots in Chen et al. (2018); Lee et al. (2022). We recommend plotting indicatrices across the entire embedding instead of only at isolated points. Magnification factor plots have been put forward in Chen et al. (2018) though we suggest restricting them to embedding points. This is meaningful, since judging the area-distortion is most relevant in regions that contain data.

5. Experiments

5.1. Experimental Setup

Datasets Besides the classical image datasets MNIST (LeCun et al., 1998) and FashionMNIST (Xiao et al., 2017), we use the three single-cell datasets Zilionis (Zilionis et al., 2019), PBMC (Zheng et al., 2017) and CElegans (Packer et al., 2019). For illustration only, we generate an *Earth* dataset consisting of points randomly sampled from the unit sphere $S^2 \subset \mathbb{R}^3$, wherever there would be landmass on earth. More information can be found in Appendix D.1.

Baselines We use UMAP (McInnes et al., 2018), t-SNE (van der Maaten & Hinton, 2008) and PCA (Pearson, 1901) as baselines, as well as a vanilla autoencoder, an autoencoder with UMAP side-loss (Sainburg et al., 2021) and the topological autoencoder (Moor et al., 2020). For the former non-parametric models as well as for the UMAP autoencoder's side loss, we use the default parameters. For the topological autoencoder, we weigh the topological loss for all datasets by $\lambda = 0.5$, recommended by Moor et al. (2020) for the MNIST dataset.

Architecture and Training Encoder and decoder of all the autoencoder models have four layers of width 100, with ELU (Clevert et al., 2015) activations. This architecture is very similar to the standard architecture of Sainburg et al. (2021), and differs from it only by an additional layer as well as in the activation function. See Appendix D.2 for more information about our training procedure. For the proposed geometric autoencoder, we found $\alpha = 0.1$ to be a good weight for the geometric loss term.

5.2. Evaluation

Qualitative Evaluation We evaluate the geometric autoencoder as well as suitable baselines using indicatrices and the generalized Jacobian determinant. For the latter we create a heatmap plot of the logarithm of the generalized Jacobian determinant in units of their mean, and subtract 1 from the result in order to center the scale. All values outside of the 5% quantiles are collapsed to the extreme values inside the quantiles. Our results are shown in Figures 3, S2, S3.

Quantitative Evaluation We evaluate all models' embeddings with metrics from Moor et al. (2020) and Kobak & Berens (2019). Our local metrics are *kNN*, *Trust* and $KL_{0.1}$. The *kNN* metric calculates the kNN recall from embedding to input. *Trust* is a metric based on the k nearest neighbor rankings, and $KL_{0.1}$ measures the Kullback-Leibler divergence based on density estimates in input- and latent-space with length-scale 0.1. Our global metrics are KL_{100}, *Stress*, and *Spear*. The KL_{100} metric considers a density estimate on a more global scale. The *Stress* metric is the loss of multidimensional scaling, and *Spear* calculates the Spearman coefficient between distances in input and embedding. For a more detailed discussion of the metrics, see Appendix D.4.

We train on and visualize all data, and evaluate the metrics on a 10% random subset for speed reasons. To aggregate results of different metrics and datasets, we rank the models for each metric and average over all datasets. Averaging these aggregated ranks over all metrics gives our final metric $\langle$Rank$\rangle$. The results can be found in Tables 1 and S5.

6. Results

Qualitative Results In the following paragraphs we evaluate the performance of the autoencoders based on our proposed diagnostics, as well as the embeddings themselves. We computed them for all datasets considered, see Figures S1–S3. In the main paper, we illustrate our findings with the MNIST dataset for which we depict the embeddings, determinant heat maps and indicatrices in Figure 3.

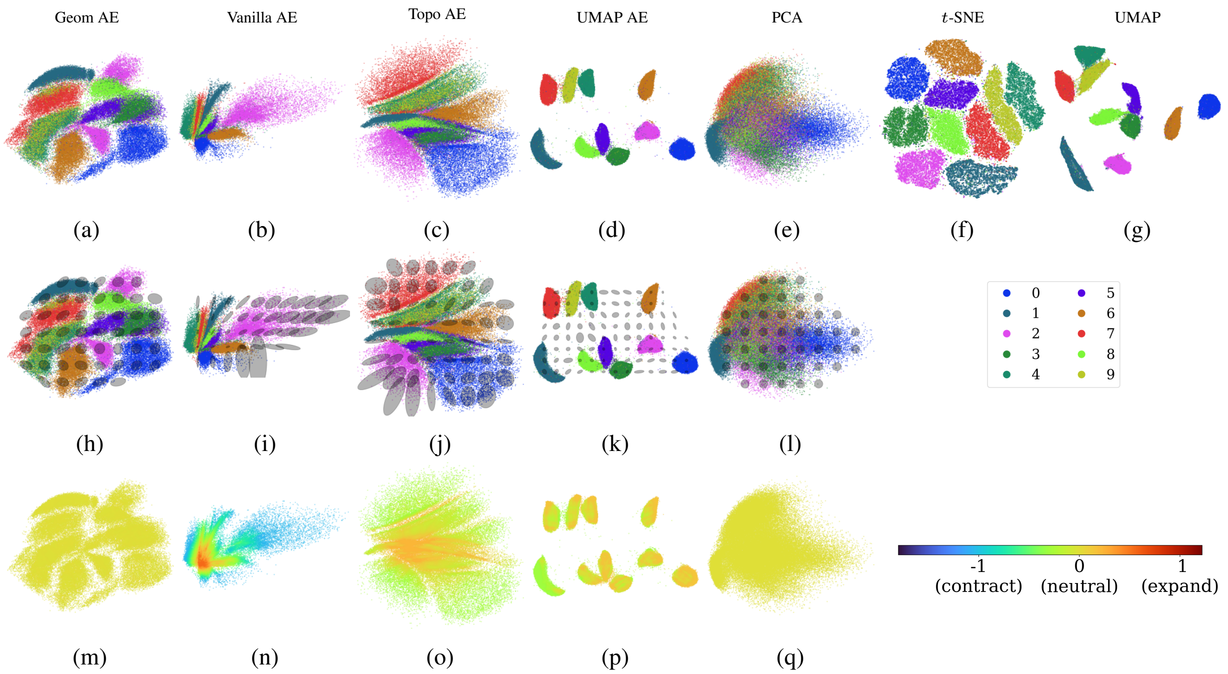


Figure 3. Comparing different embedding methods on MNIST. From left to right we consider the geometric, the vanilla, the topological autoencoder, and the UMAP autoencoder followed by PCA, t-SNE, and UMAP. From top to bottom we show the embedding, the indicatrices and the generalized Jacobian determinant. We show our diagnostics only for those methods with a decoder.

The geometric autoencoder produces balanced embeddings for all the datasets considered, see first column of Figure S1. Its class separation is better than that of the vanilla autoencoder and PCA, slightly better than that of the topological autoencoder, but worse than t-SNE's and UMAP's.

While the geometric autoencoder produces visually pleasing embeddings, its true strength becomes apparent when combining it with the information obtained from our diagnostics, the determinant heatmap and the indicatrices. As we regularize the autoencoder by the generalized Jacobian determinant, it is reassuring to see that this determinant indeed varies very little for the final embedding, see Figure 3m and the first column of Figure S2. Visually, the determinant heatmap looks as uniform as that of PCA, which has constant generalized Jacobian determinant. As a result, we can trust the relative sizes in the geometric autoencoders' plots.

The determinant heatmaps on the vanilla autoeocoders' plots (Figure S2, column 2) are striking and allow us to understand the corresponding embeddings much better. Those embeddings have an extremely crowded area and a few data points or classes which take up most of the embedding space in common. For instance, it appears that the embedding of the digit 2 (pink) in Figure 3b takes up roughly as much space as the rest of the embedding. This observation is similar to the distorted continent sizes in Figure 1a. Consulting the determinant heatmap prevents from false conclusions: The decoder massively expands the cluttered region and contracts those clusters that take up most of the embedding area. This means that the data embedded into the cluttered area takes up much more space than appears in the embedding and conversely, the embedding of the digit 2 in Figure 3b do not actually take up much more space than the other classes. Indeed, in the embedding of the geometric autoencoder, which does not distort relative sizes, the different classes of MNIST are depicted roughly equisized.

Both the topological autoencoder and the UMAP autoencoder show more variation in local contraction and repulsion than the geometric autoencoder, see Figure S2 columns 3 and 4. For the topological autoencoder, we see that the typically densely packed center of the embedding gets expanded by the decoder. A more spread-out layout would improve the faithfulness of the embedding. For the UMAP autoencoder, it is generally the boundary of clusters that appears too contracted, indicating that the actual cluster separation is exaggerated in the UMAP plot. Similarly, the indicatrices show that the whitespace gets contracted by the decoder.

The information encoded in the indicatrices refines the interpretation of the various embeddings further. For the geometric autoencoder, indicatrices have mostly the same area, reflecting the uniform determinant heatmaps (column 1 of Figure S3). Even though not regularized for this explicitly, they are also more circular than for other methods. This

Table 1. Quantitative evaluation of our method. We rank each method for a given metric and calculate the mean over all datasets. The ⟨RANK⟩ is the average over metrics. Bold and underlined indicates first, bold second place.

	Local			Global			
	$\text{KL}_{0.1}$	kNN	Trust	Stress	KL_{100}	Spear	⟨rank⟩
Geom AE (ours)	**2.6**	3.4	**2.2**	3.4	**2.2**	3.4	**2.9**
Vanilla AE	5.4	5.4	4.4	6.2	4.8	5.0	5.2
Topo AE	**2.8**	4.8	4.2	4.8	**2.2**	**1.8**	**3.4**
UMAP AE	4.4	**1.6**	**1.8**	**2.6**	6.0	5.0	3.6
UMAP	5.2	3.4	4.0	**1.6**	5.6	4.2	4.0
t-SNE	4.0	**2.4**	4.4	6.8	3.8	7.0	4.7
PCA	3.6	7.0	7.0	2.6	3.4	**1.6**	4.2

demonstrates that the geometric autoencoder does little directed stretching, leading to a recognizable world map of the Earth dataset (Figure 1a). We measured the mean 2-norm condition number on the MNIST dataset for the pullback metric and found that our method, after PCA, has the most isotropic indicatrices (see Table S1). Analyzing the positions where an indicatrix is elongated helps us to correctly understand our embeddings. On MNIST, the indicatrix in the long protrusion of the embedding of the digit 0 (blue) in Figure 3a is elongated in the same direction as the protrusion. This implies that the protrusion should not be as long in the real data as depicted in the embedding.

The indicatrices for the vanilla autoencoder help us understand its embedding better: Not only is the pink class in Figure 3b depicted deceitfully large, but, in particular, stretched too much horizontally. Similarly, the embedding of the digit 7 (red) is stretched vertically. Jointly, the indicatrices seem to point towards the most densely packed region of the vanilla autoencoder embeddings for all datasets (see Figure S3 column 2). Investigating further, we noticed that this dense region is typically close to the origin in embedding space. We found that at the beginning of training all embedding points are clustered tightly around the origin due the initialization of the network. During training, some classes separate by "expanding away" from the origin, while others stay near the origin. This explains the typical "star-shape" of vanilla autoencoder embeddings. We include a video of the training of a vanilla autoencoder on MNIST illustrating this process in the GitHub repository. Overall, our diagnostics enabled us to unravel the peculiar appearance of vanilla autoencoder plots.

The indicatrices for PCA are perfect circles, since the decoder of PCA consists of a pair of orthonormal vectors. Thus, PCA scores perfectly in our diagnostics, but produces embeddings with the least structure. This shows that a certain level non-linearity is necessary for salient feature extraction.

Spotting artefacts in the data is a major use-case of data visualization. Indeed, the embedding of the initial version of the PBMC dataset revealed suspiciously regular structures in the data, which turned out to be a preprocessing artefact, compare Figures S6c, d. The geometric autoencoder highlights this artefact particularly well (Figure S6e). UMAP, for example, disguises it completely.

Our geometric autoencoder visualizes semantic information even on a finer level than digit class in MNIST. For instance, it separates digits 2 with straight lower stroke from those with curved lower strokes. The vanilla autoencoder fails to depict such subtle structure successfully. See Appendix E.4 for more details.

Quantitative Results Our quantitative results are reported in Tables 1 and S5. We find that the geometric regularizer influences the reconstruction loss slightly less than the topological autoencoder (see Table S5). It furthermore has competitive reconstruction loss compared to the vanilla autoencoder, which shows that our regularizer does not lead to a major impairment of the reconstruction. The geometric autoencoder beats the vanilla baseline in all metrics except for the reconstruction loss. It furthermore achieves top rank in the KL_σ metrics in both the local and more global setting, striking a good compromise between the preservation of local and global structure. Overall, the geometric autoencoder balances the demands of the different metrics best as it achieves top aggregated rank. Its closest competitors are the topological autoencoder and the UMAP autoencoder, underlining the power of autoencoders for visualization when properly regularized.

7. Limitations

For the image of the decoder D to be a manifold with a single chart, we require D to be a smooth embedding. Choosing ELU activations ensures that the decoder is continuously differentiable. Since our regularizer penalizes the Jacobian for having zero determinant, the inverse function theorem gives us locally continuously differentiable invertibility. Not fulfilling these assumptions rigorously impedes neither our regularization nor our visualization. We could just not call the decoder's image a manifold.

When defining the pullback metric, we additionally need to assume that the decoder is an immersion. If this was not the case and the differential failed to be injective at an embedding point p, then the metric tensor at p would not be positive definite. Our diagnostics would detect this in the form of an infinitely flat indicatrix at p. However, our regularization loss would become infinite, hence mitigating the problem in practice.

8. Discussion and Conclusion

Low-dimensional visualization is key for understanding high-dimensional datasets. An embedding should capture the most salient features, while representing the dataset faithfully. Thus, our contribution consists of two components. We provide insightful diagnostics that allow identifying distortions in an embedding, as well as a novel regularizer mitigating them. The resulting embedding is more faithful when it comes to relative sizes and shapes.

Our geometric regularizer is fairly simple: It minimizes how much local expansion varies. On a range of datasets, including image and single-cell data, we used our diagnostics to show that the geometric autoencoder produces visualizations with homogeneous expansion leading to a good resolution in all parts of the embedding.

We furthermore showed that the parametric version of UMAP, when combined with an autoencoder, creates clustered and separated embeddings by contracting the dataset rather inhomogeneously, especially at the border of clusters.

Future Work Equipping latent space with a metric allows for a variety of geometric diagnostics of the decoder different from our proposed indicatrices and the generalized Jacobian determinant. For example, it allows us to measure the decoder's curvature or to perform parallel transport, which is a way of moving coordinate systems "parallel" along a curved manifold. We imagine sampling an orthogonal coordinate system at an arbitrary point in latent space and then parallel transporting it along geodesics. The result would be a "curved" grid on the latent space that captures the intrinsic geometry of the high-dimensional dataset. We believe that such a latitude-longitude-like grid would be highly informative, and hope to overcome the numerical challenges encountered with existing parallel transport implementations (Guigui & Pennec, 2022; Miolane et al., 2020).

Acknowledgements This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2181/1 - 390900948 (the Heidelberg STRUCTURES Excellence Cluster) as well as by "Informatics for Life", funded by the Klaus Tschira Foundation.

References

Böhm, J. N., Berens, P., and Kobak, D. Unsupervised Visualization of Image Datasets using Contrastive Learning. In *International Conference on Learning Representations*, 2023.

Brun, A. and Knutsson, H. Tensor Glyph Warping: Visualizing Metric Tensor Fields using Riemannian Exponential Maps. In *Visualization and Processing of Tensor Fields*, pp. 139–160. Springer, 2009.

Chazal, F., Cohen-Steiner, D., and Mérigot, Q. Geometric Inference for Probability Measures. *Foundations of Computational Mathematics*, 11(6):733–751, 2011.

Chen, N., Klushyn, A., Kurle, R., et al. Metrics for Deep Generative Models. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1550. PMLR, 2018.

Chen, N., Klushyn, A., Ferroni, F., et al. Learning Flat Latent Manifolds with VAEs. *arXiv preprint arXiv:2002.04881*, 2020.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Damrich, S. and Hamprecht, F. A. On UMAP's True Loss Function. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5798–5809, 2021.

Damrich, S., Böhm, J. N., Hamprecht, F. A., et al. Contrastive learning unifies t-SNE and UMAP. In *International Conference on Learning Representations*, 2023.

Duque, A. F., Morin, S., Wolf, G., et al. Geometry Regularized Autoencoders. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Ghosh, P., Sajjadi, M. S. M., Vergari, A., et al. From variational to deterministic autoencoders. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1g7tpEYDS.

Gropp, A., Atzmon, M., and Lipman, Y. Isometric Autoencoders. *arXiv preprint arXiv:2006.09289*, 2020.

Guigui, N. and Pennec, X. Numerical Accuracy of Ladder Schemes for Parallel Transport on Manifolds. *Foundations of Computational Mathematics*, 22(3):757–790, 2022.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313 (5786):504–507, 2006.

Jang, C., Lee, Y., Noh, Y.-K., et al. Geometrically regularized autoencoders for non-Euclidean data. In *International Conference on Learning Representations*, 2023.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, pp. 1–15, 2015.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

Kingma, D. P. and Welling, M. An Introduction to Variational Autoencoders. *Foundations and Trends in Machine Learning*, 2019.

Kobak, D. and Berens, P. The art of using t-SNE for single-cell transcriptomics. *Nature Communications*, 10(1):1–14, 2019.

Kobak, D., Linderman, G., Steinerberger, S., et al. Heavy-tailed kernels reveal a finer cluster structure in t-SNE visualisations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 124–139. Springer, 2020.

Kramer, M. A. Nonlinear Principal Component Analysis using Autoassociative Neural Networks. *AIChE journal*, 37(2):233–243, 1991.

Laskowski, P. H. The Traditional and Modern Look at Tissot's Indicatrix. *The American Cartographer*, 16(2): 123–133, 1989.

LeCun, Y., Bottou, L., Bengio, Y., et al. Gradient-based learning applied to document recognition. *IEEE*, 86(11): 2278–2324, 1998.

Lee, J. M. *Introduction to Smooth Manifolds*. Springer, 2nd edition, 2000.

Lee, J. M. *Introduction to Riemannian Manifolds*, volume 2. Springer, 2018.

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

Lee, Y., Yoon, S., Son, M., et al. Regularized Autoencoders for Isometric Representation Learning. In *International Conference on Learning Representations*, 2022.

Li, S. Z., Zang, Z., and Wu, L. Markov-Lipschitz Deep Learning. *arXiv preprint arXiv:2006.08256*, 2020.

Makhzani, A. and Frey, B. k-Sparse Autoencoders. In *International Conference on Learning Representations*, 2014.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Miolane, N., Guigui, N., Le Brigant, A., et al. Geomstats: A Python Package for Riemannian Geometry in Machine Learning. *Journal of Machine Learning Research*, 21 (223):1–9, 2020.

Moor, M., Horn, M., Rieck, B., et al. Topological Autoencoders. In *International Conference on Machine Learning*, pp. 7045–7054. PMLR, 2020.

Narayan, A., Berger, B., and Cho, H. Assessing single-cell transcriptomic variability through density-preserving data visualization. *Nature Biotechnology*, 39(6):765–774, 2021.

Ng, A. et al. Sparse Autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

Packer, J. S., Zhu, Q., Huynh, C., et al. A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution. *Science*, 365(6459):eaax1971, 2019.

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

Plaut, E. From Principal Subspaces to Principal Components with Linear Autoencoders. *arXiv preprint arXiv:1804.10253*, 2018.

Rifai, S., Vincent, P., Muller, X., et al. Contractive Auto-Encoders: Explicit Invariance during Feature Extraction. In *28th International Conference on Machine Learning*, pp. 833–840, 2011.

Rumelhart, D., Hinton, G., and Williams, R. Learning internal representations by error propagation. *Parallel distributed processing*, 1:318–363, 1986.

Sainburg, T., McInnes, L., and Gentner, T. Q. Parametric UMAP embeddings for representation and semisupervised learning. *Neural Computation*, 33(11):2881–2907, 2021.

Scott, T. R., Gallagher, A. C., and Mozer, M. C. Von Mises-Fisher loss: An Exploration of Embedding Geometries for Supervised Learning. In *IEEE/CVF International Conference on Computer Vision*, pp. 10612–10622, 2021.

Trofimov, I., Cherniavskii, D., Tulchinskii, E., et al. Learning Topology-Preserving Data Representations. In *International Conference on Learning Representations*, 2023.

van der Maaten, L. Accelerating t-SNE using Tree-Based Algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

van der Maaten, L. and Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

Venna, J. and Kaski, S. Visualizing Gene Interaction Graphs with Local Multidimensional Scaling. In *The European Symposium on Artificial Neural Networks*, 2006.

Vincent, P., Larochelle, H., Lajoie, I., et al. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11(12), 2010.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhao, Y., Zhao, D., Wan, S., et al. Softmax Supervision with Isotropic Normalization, 2018. preprint.

Zheng, G. X., Terry, J. M., Belgrader, P., et al. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8(1):1–12, 2017.

Zilionis, R., Engblom, C., Christina, P., et al. Single-Cell Transcriptomics of Human and Mouse Lung Cancers Reveals Conserved Myeloid Populations across Individuals and Species. *Immunity*, 50(5):1317–1334, 2019.

A. Extended Figures

In Figures S1, S2 and S3 we show the embeddings, determinant heatmap plots and indicatrices for all the datasets and models considered. In Figure S5 we show the labels of all datasets.

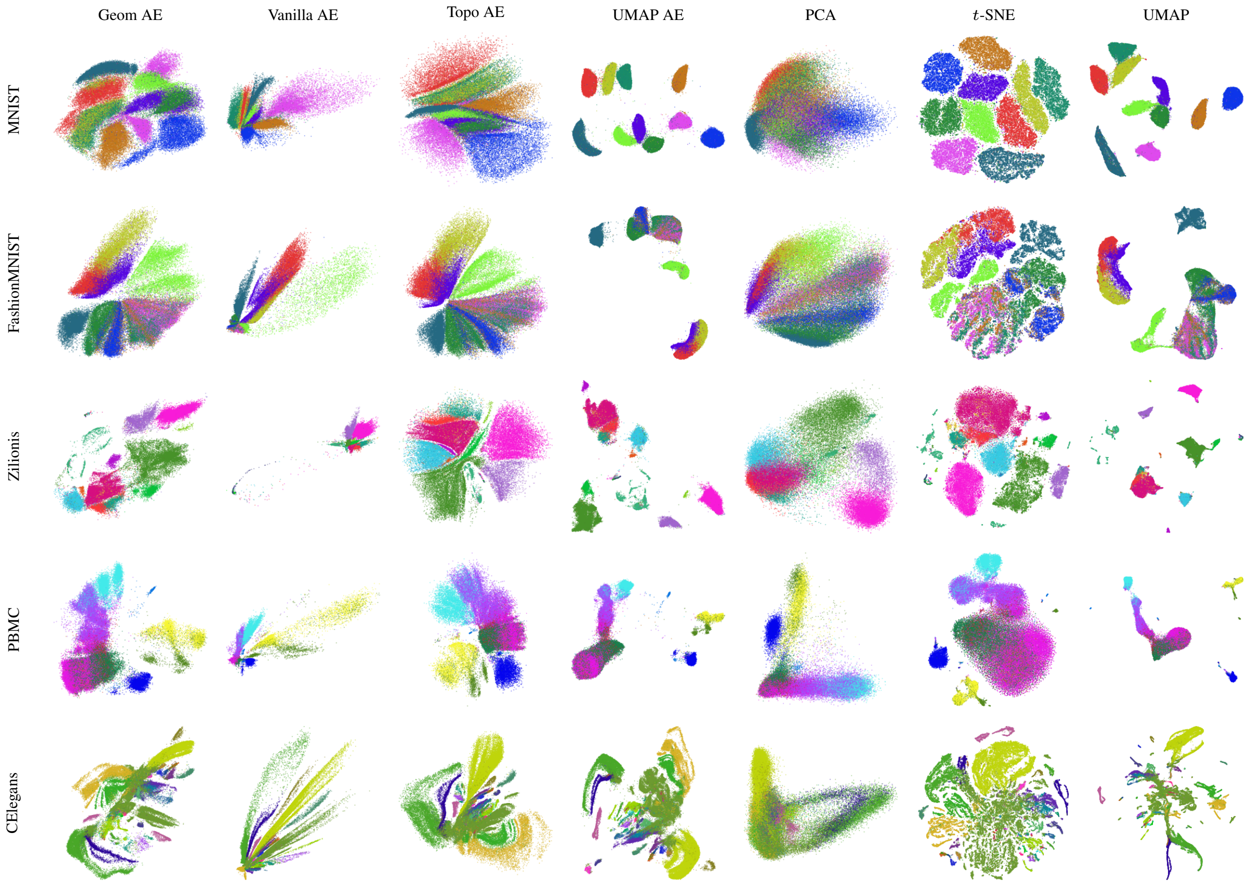


Figure S1. Embeddings of all datasets created with all models.

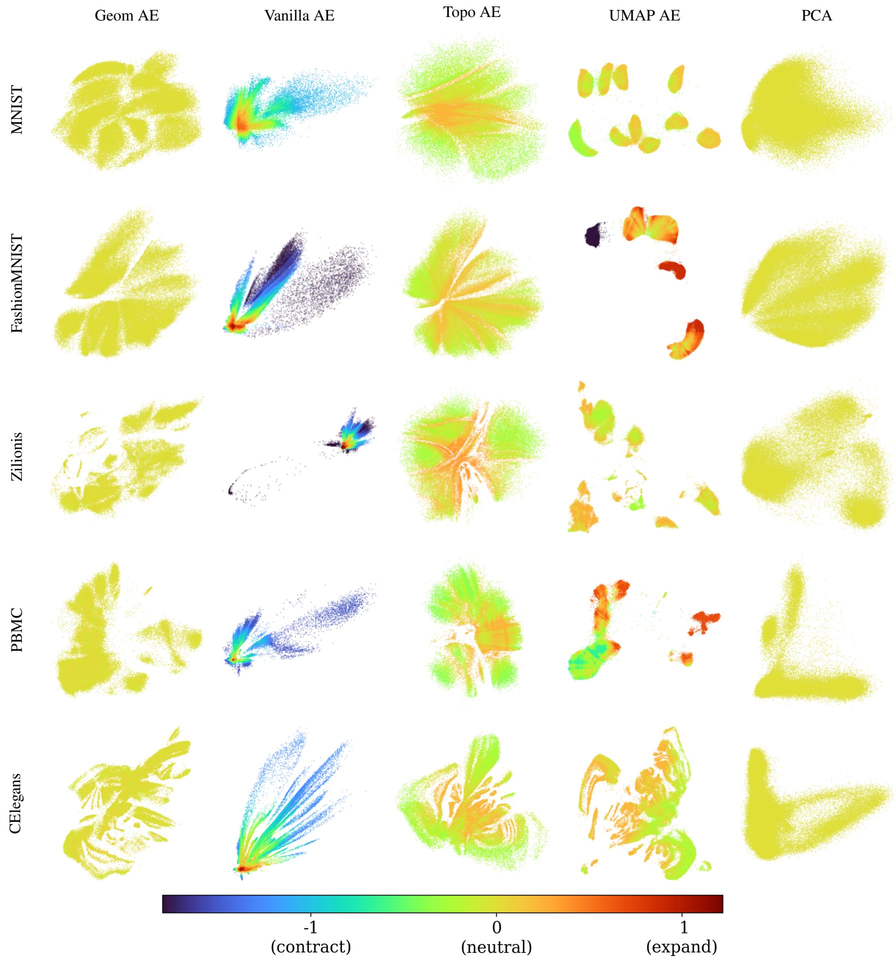


Figure S2. Determinant diagnostic for all suitable models on all datasets.

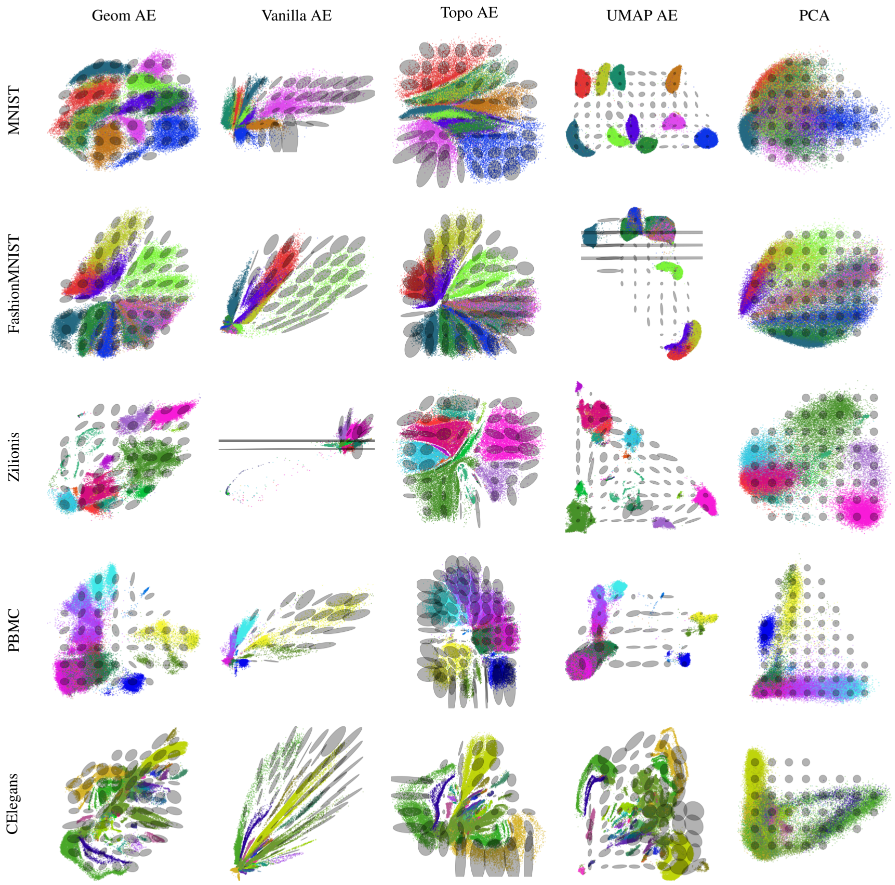


Figure S3. Indicatrix diagnostics for all suitable models on all datasets.

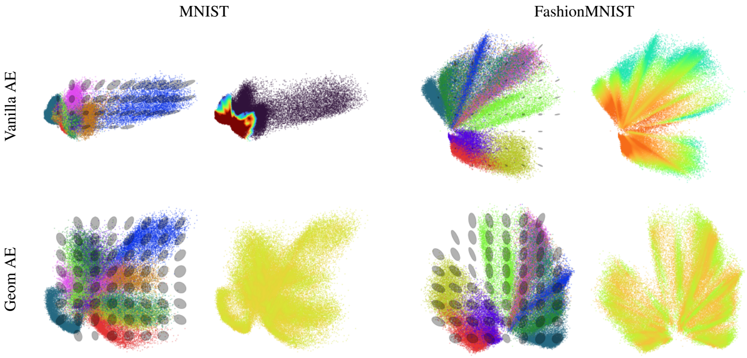
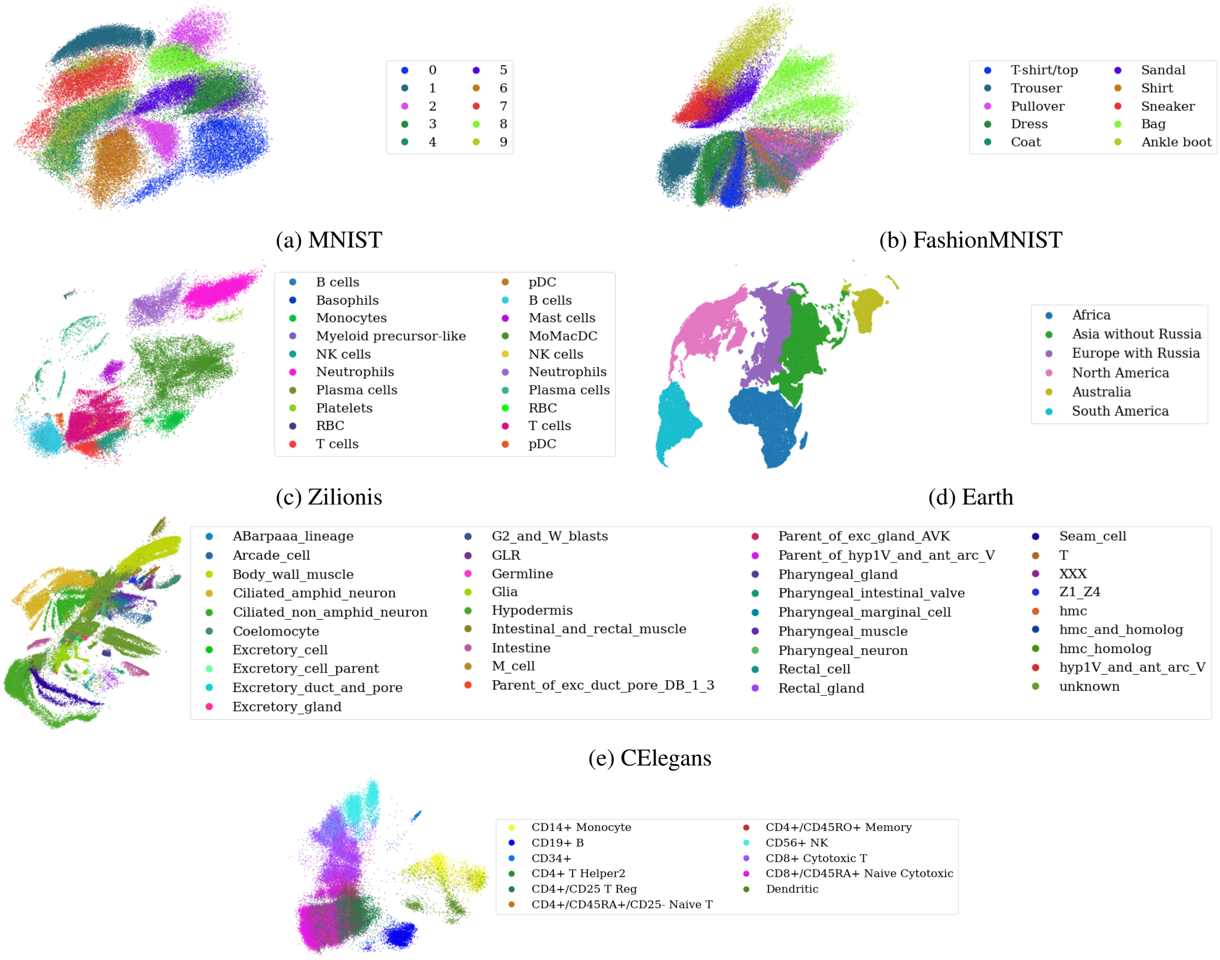


Figure S4. Convolutional autoencoders trained and evaluated on MNIST and FashionMNIST.



(a) MNIST

(b) FashionMNIST

(c) Zilionis

(d) Earth

(e) CElegans

(f) PBMC

Figure S5. Datasets with labels. Embeddings created with geometric autoencoder.

B. Theorems and Proofs

B.1. PCA and Linear Autoencoders

We show that PCA can be understood as emerging as an edge case from autoencoders, placing it into the same family of dimensionality reduction techniques. The proof up to the consideration of weight decay can in this or a similar form also be found in Kramer (1991); Plaut (2018). We repeat the complete argument here for the reader.

Theorem B.1. *Let X be a zero-centered dataset whose first l singular values are strictly larger than the $l+1$-st. Further, denote by $\mathcal{S}$ the set of autoencoders that have linear encoder and decoder without biases, bottleneck dimension l and achieve optimal reconstruction loss on X. Then an autoencoder $(E, D) \in \mathcal{S}$ learns PCA (up to a rotation or reflection) if and only if it also achieves minimal weight decay loss among the autoencoders in $\mathcal{S}$.*

Proof. Let $E \in \mathbb{R}^{l\times n}$ be the linear encoder, $D \in \mathbb{R}^{n\times l}$ the linear decoder. Furthermore, let $W \in \mathbb{R}^{l\times n}$ be the PCA solution, i.e., the matrix whose l rows are the first l principal components. By assumption about X's singular values, the set of the first l principal components and thus the subspace $V' \subset \mathbb{R}^n$ that they span are unique.

Let $X \in \mathbb{R}^{n,m}$ be the data matrix and set $\mathcal{W} = \{W' \in \mathbb{R}^{l,n} \mid W' \text{ has orthonormal rows}\}$. Then the PCA objective (up to rotation and reflection) can be written as

$$W \in \underset{W'\in\mathcal{W}}{\arg\min} \|X - W'^T W' X\|_2^2. \tag{4}$$

The autoencoder objective is given by

$$E, D = \underset{E'\in\mathbb{R}^{l,n}, D'\in\mathbb{R}^{n,l}}{\arg\min} \|X - D'E'X\|_2^2. \tag{5}$$

We want to argue that modulo a multiplication by an invertible matrix in latent space, the two objectives defined in Equations (4) and (5) agree.

<u>Claim:</u> The possible autoencoder solutions are precisely those matrices of the form $(E, D) = (AW, W^T A^{-1})$ for an $A \in \mathrm{GL}(l, \mathbb{R})$.

<u>Proof:</u> First, we want to show that the PCA solution minimizes the autoencoder objective. If we can show this, one implication follows trivially. Note that the image of DE is an l-dimensional subspace V of the vector space $\mathbb{R}^n$, so that the autoencoder objective in Equation (5) comes down to mapping the dataset into $V \subset \mathbb{R}^n$ while minimizing the ℓ_2 distance between a data point and its image. In the Hilbert space $\mathbb{R}^n$, the minimality condition implies that such a map must be given by an orthogonal projection, independent of the subspace we project onto. In other words, let $B \in \mathbb{R}^{l,n}$ be a matrix whose rows are an orthonormal basis of V. Then the orthogonal projection $\mathbb{R}^n \to V$ is given by $v \mapsto B^T Bv$. By the minimality criterion, we have $DE = B^T B$. As B is feasible for the PCA objective, the PCA solution also solves the autoencoder objective.

Second, we have to show that every autoencoder solution (E, D) is of the form $(E, D) = (AW, W^T A^{-1})$ for an $A \in \mathrm{GL}(l, \mathbb{R})$. The orthogonal projection to V', the space given by the first l principal components, is given by $W^T W$. By our argument above and the uniqueness of V', the matrix DE must equal $W^T W$. Since W has orthogonal rows, it is surjective and thus D and W^T must have the same row space. In other words, there is some $A \in \mathrm{GL}(l, \mathbb{R})$ such that $D = W^T A^{-1}$. Orthogonality of the rows of W implies $WW^T = I$. Multiplying $DE = W^T W$ by W from the left yields

$$A^{-1}E = WW^T A^{-1}E = W(DE) = W(W^T W) \tag{6}$$
$$= W. \tag{7}$$

#

In the following, we find further restrictions on the matrix A resulting from weight decay on E and D.

<u>Claim:</u> If E and D have minimal Frobenius norm among all $(E, D) \in \{(AW, W^T A^{-1}) | A \in \mathrm{GL}(l, \mathbb{R})\}$, then the additional matrix A is a rotation or reflection.

<u>Proof:</u> First recall that for a real matrix $M \in \mathbb{R}^{m,n}$, the Frobenius norm is $\|M\|_F^2 = \mathrm{tr}(MM^t)$. Consequently the Frobenius-Norm is invariant under multiplication by an orthogonal matrix. Second, recall that the only freedom E and D have lies in the additional invertible matrix A. Performing an SVD of A, we obtain $A = U^t \Sigma V$ where U and V are $l \times l$ orthogonal and $\Sigma = (\sigma_1, \sigma_2, ..., \sigma_{l-1}, \sigma_l)$ is $l \times l$ diagonal with $\sigma_i \neq 0$. This allows to evaluate the Frobenius norm of the decoder D as

$$\begin{aligned} \|D\|_F &= \|W^T A^{-1}\|_F \\ &= \|W^T V^T \Sigma^{-1} U\|_F \\ &= \|\Sigma^{-1}\|_F, \end{aligned} \tag{8}$$

where we used that U, V and W all have orthonormal rows. Analogously, we obtain

$$\|E\|_F = \|\Sigma\|_F. \tag{9}$$

This shows that

$$\begin{aligned} \text{loss}_{\text{weight decay}} &= \|D\|_F^2 + \|E\|_F^2 \\ &= \|\Sigma^{-1}\|_F^2 + \|\Sigma\|_F^2 \\ &= \sum_i \sigma_i^2 + \sigma_i^{-2}, \end{aligned} \tag{10}$$

which is minimal if and only if

$$\sigma_i = \pm 1. \tag{11}$$

Consequently, weight decay restricts the autoencoders degree of freedom to a matrix of the form

$$A = U^t \operatorname{diag}(\pm 1) V \in O(l), \tag{12}$$

which is orthogonal since U, $\operatorname{diag}(\pm 1)$ and V are so. #

This shows that the autoencoder differs from PCA only by a rotation and/or reflection, which completes the proof. □

We believe that Theorem B.1 closely applies in practice, where autoencoders are typically trained to minimize a weighted sum of reconstruction and weight decay loss, as long as the weight of the regularizer is reasonably small.

B.2. Pullback Metric in Coordinates

The pullback of the Euclidean metric under the decoder D takes a very simple form in coordinates:

Theorem B.2 (Pullback Metric in Coordinates)**.** *Given a point $p \in \mathbb{R}^l$, the pullback metric at p in coordinates is*

$$\langle \cdot, \cdot \rangle_p \coloneqq D^* g_{e_p} = (J_p D)^t \, J_p D \in \mathbb{R}^{2,2}, \tag{13}$$

where $J_p D$ is the Jacobian of the decoder at p.

Proof. After choosing coordinates on $\mathbb{R}^l$ and $\mathbb{R}^n$, the differential of D at p becomes the Jacobian matrix which we denote by $J_p D$, and the inner product in latent space $(\mathbb{R}^l, D^* g)$ is given by

$$\begin{aligned} \langle v, w \rangle_p &\coloneqq D^* g_p(v, w) = d_p D(v)^t d_p D(w) && (14)\\ &= (J_p D v)^t \, J_p D w = v^t \left(J_p D^t J_p D \right) w. && (15) \end{aligned}$$

□

B.3. Properties of the Determinant Regularization Objective

In this subsection, we prove properties of our secondary objective defined in Equation (3), in particular its minimum and some of its invariances.

Theorem B.3.

1. $\mathcal{L}_{\det}(D) \geq 0$
2. $\mathcal{L}_{\det}(D) = 0$ *if and only if for all $x, x' \in X$ we have* $\det((J_{E(x)}D)^t J_{E(x)}D) = \det((J_{E(x')}D)^t J_{E(x')}D)$.
3. *If D and $\tilde{D}$ are two decoders and there is some $c > 0$ such that for all $x \in X$ we have* $\det((J_{E(x)}D)^t J_{E(x)}D) = c \det((J_{E(x)}\tilde{D})^t J_{E(x)}\tilde{D})$, *then $\mathcal{L}_{\det}(D) = \mathcal{L}_{\det}(\tilde{D})$.*
4. *Let $F : \mathbb{R}^l \to \mathbb{R}^l$ be a scaled area–preserving diffeomorphism, so that $\det(J_z F)$ is a constant in z. Then the autoencoder (E, D) and the map given by $(F^{-1} \circ E, D \circ F)$ have the same output, the same reconstruction loss, and the same geometric regularizer value $\mathcal{L}_{\det}(D) = \mathcal{L}_{\det}(D \circ F)$.*

Proof.

1. Variances are non-negative.

2. The "if" part is clear since the variance of a constant is zero. The "only if" part follows since all data samples have equal probability mass in each batch and batches are also collected uniformly from the whole dataset.

3. Analogous to 4.

4. Since the applications of F and F^{-1} cancel, the outputs and reconstruction losses agree.

Let $d = \det(J_z F)$ be the determinant of F's Jacobian. For $x \in X$, we have

$$J_{F^{-1}(E(x))}(D \circ F) = J_{E(x)}(D) J_{F^{-1}(E(x))}(F) \tag{16}$$

and so

$$\begin{aligned} &\det\left(J_{F^{-1}(E(x))}(D \circ F)^t J_{F^{-1}(E(x))}(D \circ F) \right) && (17)\\ &\quad = d^2 \det\left(J_{E(x)}(D)^t J_{E(x)}(D) \right). && (18) \end{aligned}$$

Thus, the value of the regularizer remains unchanged

$$\begin{aligned} &\mathcal{L}_{\det}(D \circ F) && (19)\\ &= \operatorname*{Var}_{x \sim \mathcal{U}(B)} \Big(\log \Big(\det \Big(J_{F^{-1}(E(x))}(D \circ F)^t && (20)\\ &\qquad\qquad J_{F^{-1}(E(x))}(D \circ F) \Big) \Big) \Big) && (21)\\ &= \operatorname*{Var}_{x \sim \mathcal{U}(B)} \left(\log \left(d^2 \det(J_{E(x)}(D)^t J_{E(x)}(D)) \right) \right) && (22)\\ &= \operatorname*{Var}_{x \sim \mathcal{U}(B)} \Big(\log(\det(J_{E(x)}(D)^t J_{E(x)}(D))) && (23)\\ &\qquad\qquad + 2 \log(d) \Big) && (24)\\ &= \operatorname*{Var}_{x \sim \mathcal{U}(B)} \left(\log(\det(J_{E(x)}(D)^t J_{E(x)}(D))) \right) && (25)\\ &= \mathcal{L}_{\det}(D). && (26) \end{aligned}$$

□

The first two parts of Theorem B.3 show that our regularizer becomes minimal exactly when the decoder is area-preserving at the embedding points. The second two parts describe invariances of our regularizer. Our regularizer is insensitive to area-preserving changes of the decoder, or equivalently, of the embedding. This implies scale invariance, a useful property for visualization:

Corollary B.4 (Scale Invariance)**.** *Let the first layer of the decoder scale by a factor of $\beta \in \mathbb{R} \setminus \{0\}$, and the embedding by β^{-1}. Not only does this fix the primary objective (the reconstruction loss), but also our secondary geometric objective.*

Proof. This is a special case of Theorem B.3 in which F is the multiplication with β. □

In particular, our regularizer does not favor decoders with Jacobian of small norm, in contrast to the regularizer of Chen et al. (2020), see (Lee et al., 2022).

C. Relation to Lee et al. (2022)

Lee et al. (2022) describe a hierarchy of geometry-preserving mappings consisting, from strongest geometry-preservation to weakest, of isometries, scaled isometries, conformal maps and area-preserving maps. Their proposed regularizers tackle the case of scaled isometries and they explicitly refrain from exploring area-preserving maps.

In turn, our regularizer promotes area-preservation. We will first explain how the functional form of our regularizer differs from that of Lee et al. (2022) and then discuss how our regularizer achieves a similar goal as that of Lee et al. (2022) in practice.

Denote the determinant of the pullback metric at point z by $d(z) := \det((J_z D)^t J_z D)$ and by $\lambda_1(z), \dots, \lambda_l(z)$ the eigenvalues of the pullback metric at z. In slight abbuse of notation, we will write $z \sim \mathcal{U}(B)$ when we mean $z = E(x)$ and x being sampled uniformly from the batch $x \sim \mathcal{U}(B)$.

Rewriting our regularizer from Equation (3), we get

$$\mathcal{L}_{\det} \tag{27}$$

$$= \operatorname*{Var}_{z\sim\mathcal{U}(B)} \left[\log\left(d(z)\right)\right] \tag{28}$$

$$= \mathop{\mathbb{E}}_{z\sim\mathcal{U}(B)} \left[\left(\log(d(z)) - \mathop{\mathbb{E}}_{z'\sim\mathcal{U}(B)} \log(d(z')) \right)^2 \right] \tag{29}$$

$$= \mathop{\mathbb{E}}_{z\sim\mathcal{U}(B)} \left[\left(\sum_{i=1}^{l} \log(\lambda_i(z)) \right. \right. \tag{30}$$

$$\left. \left. - \mathop{\mathbb{E}}_{z'\sim\mathcal{U}(B)} \sum_{j=1}^{l} \log(\lambda_j(z')) \right)^2 \right] \tag{31}$$

Lee et al. (2022)'s regularizer requires the choice of a probability distribution $\mathbb{P}$ on latent space, a map $h : \mathbb{R} \to [0, \infty)$ with $h(1) = 0$ and $h'(\lambda) = 0$ if and only if $\lambda = 1$ and finally a symmetric map $S : \mathbb{R}^l \to \mathbb{R}$ with

$$S(\alpha\lambda_1, \dots, \alpha\lambda_l) = \alpha S(\lambda_1, \dots, \lambda_l) \tag{32}$$

$$S(1, \dots, 1) = 1. \tag{33}$$

Given $\mathbb{P}$, h and S, the regularizer is

$$\mathcal{L}_{\text{Lee}}(\mathbb{P}, h, S) = \tag{34}$$

$$\mathop{\mathbb{E}}_{z\sim\mathbb{P}} \left(\sum_{i=1}^{l} h\left(\frac{\lambda_i(z)}{\mathbb{E}_{z'\sim\mathbb{P}} S(\lambda_1(z'), \dots, \lambda_l(z'))} \right) \right) \tag{35}$$

Table S1. The pullback metric's 2-norm condition number acts as a measure for the decoder's isotropy. We evaluate it on the intersection between a regular grid of 100 × 100 points and the embedding's convex hull. The results are for the MNIST dataset and we report average and standard deviation over five different initializations. The last two rows refer to the convolutional autoencoders described in Section E.2.

MODEL	MEAN CONDITION NUMBER
GEOM AE (OURS)	2.5 ± 1.1
VANILLA AE	100.0 ± 900.0
TOPO AE	10.1 ± 22.1
UMAP AE	3.6 ± 3.1
PCA	1.0 ± 0.0
CONV VANILLA AE	14.6 ± 13.8
CONV GEOMAE (OURS)	2.5 ± 1.2

The following admissible choices yield the closest match to our regularizer:

$$\mathbb{P} \text{ to be given by } z = E(x), x \sim \mathcal{U}(B) \tag{36}$$

$$h(\lambda) = \log(\lambda)^2 \tag{37}$$

$$S(\lambda_1, \dots, \lambda_l) = \frac{1}{l} \sum_{i=1}^{l} \lambda_i \tag{38}$$

While they are not the main choices employed by Lee et al. (2022)), they yield the regularizer

$$\mathcal{L}_{\text{Lee}} = \mathbb{E}_{z\sim\mathcal{U}(B)} \left[\sum_{i=1}^{l} \left(\log(\lambda_i(z)) \right. \right. \tag{39}$$

$$\left. \left. - \log\left(\frac{1}{l} \sum_{j=1}^{l} \mathbb{E}_{z'\sim\mathcal{U}(B)} \lambda_i(z') \right) \right)^2 \right] \tag{40}$$

Comparing Equations (31) and (40), the difference between our regularizer and Lee et al. (2022)'s amounts to a different ordering of taking expectation, logarithm, sum and square.

Practically, this means that our method promotes only scaled area-preservation instead of a scaled isometry, see Theorem B.3. While we only regularize towards scaled area-preservation, we observe qualitatively in Figure S3 that it also seems to favor isotropic decoders. Lee et al. (2022) propose the pullback metric determinant's 2-norm condition number as a measure for the decoder's isotropy. It is equivalent to the ratio of the length of the indicatrices' main axes; an isotropic decoder would have a condition number of one. We calculate the condition number on a regular, sufficiently dense grid in latent space intersected with the datasets convex hull for MNIST. The result is shown in Table S1. Indeed, the geometric autoencoder is the most isotropic, and the vanilla autoencoder the least (note the huge standard deviation).

D. Datasets, Metrics, Training

D.1. Datasets

We evaluate the models using the image datasets MNIST (LeCun et al., 1998) and FashionMNIST (Xiao et al., 2017), both of which we normalize to the unit interval as proposed by Moor et al. (2020), as well as the single-cell datasets Zilionis (Zilionis et al., 2019), CElegans (Packer et al., 2019) and PBMC (Zheng et al., 2017).

The single-cell datasets where obtained from `http://cb.csail.mit.edu/cb/densvis/datasets/`, where they are already preprocessed (Narayan et al., 2021). On Zilionis, we additionally normalize each feature to have a mean of zero and a standard deviation of one. This is necessary since two features dominated the dataset, as indicated by the PCA embedding of the non-normalized dataset (see Figure S6a, b). Our analysis furthermore revealed an artefact in the preprocessed PBMC data, see Figure S6c. We downloaded the original data from `https://support.10xgenomics.com/single-cell-gene-expression/datasets` and preprocessed the dataset following the procedure in Kobak et al. (2020): First, we selected the 1000 most variable genes. Then, we normalized the library sizes to the median library size in the dataset. Next, we log-transformed with $\log_2(x + 1)$, and finally applied PCA to reduce that dataset to 50 dimensions. This second version of the PBMC dataset did not show the artefact anymore, see Figure S6d.

We created the Earth dataset using the python package "mpl_toolkits". It consists of 100 000 points randomly sampled from the S^2, wherever there is landmass on earth, excluding Antarctica. Every point is labeled by the continent to which it belongs. Furthermore, in the python package we used, Europe and Russia have the same label. The labelled datasets can be found in Figure S5.

D.2. Training

All of the autoencoders except for the UMAP autoencoder are optimized using ADAM (Kingma & Ba, 2015), and trained using a batch size of 125, learning rate 10^{-3} and a weight decay of 10^{-5}. For the UMAP autoencoder, we used the standard settings with an additional weight decay of 10^{-5}. The vanilla, topological and geometric autoencoders are trained for 100 epochs. For the UMAP autoencoder we use the standard settings of the TensorFlow implementation, which in particular trains for only one epoch. However, an epoch of UMAP autoencoder is much longer than an epoch of the other autoencoders. Usually, an epoch iterates once over the entire dataset. The UMAP autoencoder iterates over pairs of datapoints that are incident in the kNN graph. Additionally, this graph is upsampled to reflect a weighing of the kNN graph. We measured the size of the resulting dataset.

Table S2. In the standard implementation of Parametric UMAP, the number of training samples (# samples) varies from the datasets' size (# dataset). This table provides an overview of how many "actual epochs" one "Parametric UMAP epoch" corresponds to.

DATASET	# DATASET	# SAMPLES	EPOCHS
MNIST	70 000	182 278 632	2600
FASHIONMNIST	70 000	191 104 176	2800
PBMC	68 551	191 021 188	2800
CELEGANS	86 024	236 600 132	2800
ZILIONIS	48 969	142807948	3000

Table S3. Training runtimes on MNIST, averaged over five random initializations. Including one standard-deviation.

MODEL	TIME [MIN]
GEOM AE	39.00 ± 1.00
VANILLA AE	24.40 ± 0.40
TOPO AE	82.10 ± 2.10
UMAP AE[2]	108.40 ± 4.9
PCA	00.20 ± 0.01
t-SNE	05.70 ± 0.50
UMAP	03.51 ± 0.03

For MNIST experiment, we found it to be more than 2600 times larger than the normal MNIST dataset. Therefore, a single epoch of UMAP autoencoder corresponds to even more than 100 epochs, which we use for the other autoencoder models. An overview for all datasets can be found in Table S2.

D.3. Implementation

The vanilla and the geometric autoencoder were implemented by us in PyTorch. In data loading, training schedule and quantitative evaluation we follow the PyTorch implementation of the topological autoencoder[1], as referenced in Moor et al. (2020). For the differential geometry involved, we use the Geomstats package (Miolane et al., 2020).

We plot the indicatrices on a regular grid in latent space, intersected with the convex hull of the dataset. We furthermore scale them globally such that the elongated ones do not cover each other too heavily. This is justified because we only care about the relative size of the indicatrices, not their absolute size.

Table S3 summarizes the training runtimes on MNIST, averaged over five random initializations.

Figure S7 shows the geometric loss curve for the vanilla autoencoder, the topological autoencoder and the geometric

[1] https://github.com/BorgwardtLab/topological-autoencoders

[2] Even though we train with the default batch size of 1000, the speed test uses 125 for consistency.

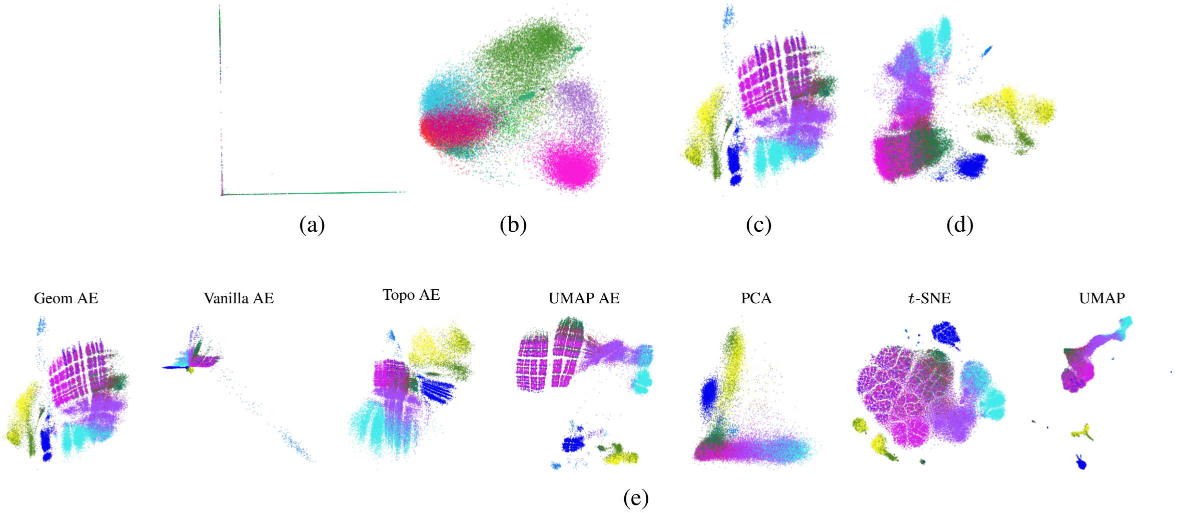


Figure S6. **First row:** Our changes in the preprocessing of the Zilionis and the PBMC dataset. Panel (a) shows the PCA embedding of the original preprocessed Zilionis data, which is dominated by two features. Normalizing each feature yields the PCA in Panel (b). Panel (c) shows the geometric autoencoder's embedding of the original preprocessed PBMC dataset, with artefacts originating from the preprocessing. Redoing the preprocessing ourselves yields the Geometric autoencoder embedding in Panel (d). **Second row:** In Panel (e) we show how different models spot the artefacts in preprocessed PBMC differently well. While the geometric autoencoder exposes the regular structure, UMAP, Topo AE, PCA, and the vanilla autoencoder disguise it almost completely.

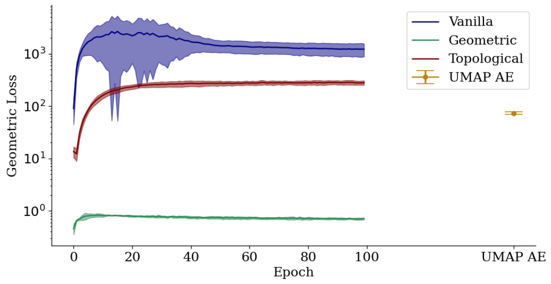


Figure S7. The geometric loss curves for the autoencoder models.

autoencoder, averaged over 4 random initialization. The geometric loss of the geometric autoencoder is roughly three orders of magnitude smaller than that of the vanilla and the topological autoencoder. It furthermore decreases with training, as opposed to the geometric loss of the topological autoencoder. We do not show the graph for the UMAP autoencoder, since it comes pre-implemented in TensorFlow, and our geometric loss is implemented in PyTorch. To compute diagnostics of the final model, we transferred the trained network weights from TensorFlow to PyTorch.

We use the automatic differentiation abilities of PyTorch, namely the function *jacrev* in the *functorch* library, for efficiently calculating the decoders pullback metric (see Algorithm 1). Since we are using ELU activations, the decoder $D\colon \mathbb{R}^2 \to \mathbb{R}^n$ is differentiable everywhere, and thus has a well defined Jacobian matrix $J_x D \in \mathbb{R}^{n \times 2}$ for all $x \in \mathbb{R}^2$.

Algorithm 1 Calculating the Generalized Jacobian Determinant

```
from functorch import jacrev
import torch

$J$ = jacrev(decoder)(batch_of_embeddings)
metric = J.T @ J
gen_jac_det = torch.det(metric)
return gen_jac_det
```

D.4. Metrics

We evaluate each model on five random seeds. As mentioned in the main paper, we evaluate the embeddings with metrics from Moor et al. (2020) and Kobak & Berens (2019). Namely, there are the three local metrics $KL_{0.1}$, *kNN*, *Trust* and the three global metrics *Stress*, KL_{100}, *Spear*.

- The *kNN* metric measures which ratio of nearest neighbors in the the embedding are also nearest neighbors in the original dataset (Kobak et al., 2020; Sainburg et al., 2021).
- The *Trust* metric is a metric based on nearest neighbor ranks (Venna & Kaski, 2006).
- The *Stress* metric coincides with the loss of multidimensional scaling, and measures the sum of the squared

differences of the distances between all pairs of embedding points and the corresponding differences of all pairs of input points (Moor et al., 2020).

- The *Spear* metric measures the Spearman correlation between the distances between all pairs of embedding and input points (Kobak & Berens, 2019).
- The KL_σ metrics ($\sigma = 0.1, 100$) measure the Kullback-Leibler divergence between a density estimate f_σ^X of the dataset X and the corresponding estimate f_σ^Z of the embedding Z. As a density estimate, we use the *distance to a measure* density estimator (Chazal et al., 2011) defined as $f_\sigma^X(x) = \sum_{y\in X} \exp\left(-\sigma^{-1}\frac{\|x-y\|_2^2}{\max_{y',x'\in X}\|y'-x'\|_2^2}\right)$, where the parameter σ defines a length scale. Ideally, the KL_σ value is small, which means that the density estimation in latent space is similar to the density estimation of the actual dataset.

The metrics depending on the number of nearest neighbors are averaged over a range of values from 10 to 200 in steps of 10, as proposed by Moor et al. (2020).

D.5. The Geometric Loss

In this subsection we further investigate how the geometric loss $\mathcal{L}_{\text{det}}$ affects encoder and decoder.

In our implementation both encoder and decoder accumulate gradients during one step of backpropagation. This may seem a bit counterintuitive, since at first glance the secondary objective only depends on the decoders' Jacobian matrix, hence only on the decoder (see Equations (2) and (3)). The reason for the encoder to accumulate gradients is that the the pullback metric tensor is also function of latent space point at which it is evaluated. Evaluating it at a given embedding, as we do in our geometric objective (Equation (3)), gives us the set of pullback metric tensors we use.

Consequently, the autoencoder can reduce the geometric loss in two ways. First, it can change the decoder's weights in order to achieve more uniform contraction for a fixed set of embedding points. Second, it can push around the embedding into areas where the decoder contracts more homogeneously. In practice, a geometric autoencoder will pursue both approaches simultaneously.

A simple gradient stop layer could prevent the encoder from receiving gradients from the geometric regularizer. We consider this an interesting avenue for future work, but believe that such a gradient stop layer might impede the autoencoder's ability to achieve homogeneous stretching of the embedding by the decoder.

E. Additional Experiments and Insights

E.1. Autoencoders as Orthogonal Projectors

As described in Appendix B.1, a linear autoencoder reduces to PCA. Viewed through the geometric lens, the linear decoder leads to a linear subspace as reconstruction manifold M, while the encoder's job is to place the reconstruction of an input point onto this reconstruction manifold. When trained with the usual mean squared error (MSE), the optimal encoder for a given linear decoder projects each input point orthogonally to the linear reconstruction manifold. To see this, consider the sphere around an input point x through some point z on M. If the vector $z - x$ is not orthogonal to the linear subspace M, the sphere intersects M and there exists some $y \in M$ inside the sphere. So z does not yield optimal mean squared error. Now, consider the general, non-linear case. If for a given decoder and thus a given reconstruction manifold (not necessarily a linear subspace anymore), there exists a data point x and a point $y \in M$ such that any point $z \in M$ is at least as far from x as y, i.e., $\|x - y\| \leq \|x - z\| \; \forall z \in M$, then the vector $x - y$ is orthogonal to any tangent vector at y by the same argument as in the PCA case.

Note, however, that there are some subtleties in the non-linear case. For instance, the encoder is limited by its architecture. So it might not be able to express the function that maps each input point in such a way to latent space that their corresponding positions on the reconstruction manifold are the desired orthogonal projections. Moreover, it might not be possible to orthogonally project an input point to the reconstruction manifold in the first place, e.g., if $x = (2, 0, ..., 0)$ and $\text{im}(D)$ is the open unit ball in the first l dimensions of $\mathbb{R}^n$. Also, there might be multiple closest points to a data point, e.g., if the data point is the center of a sphere and M the sphere.

E.2. Flexibility of our Regularizer

We expect our regularizer to work for different kinds of autoencoder models, as long as the decoder is differentiable (at least almost everywhere). As an example, we present its effect on convolutional autoencoders.

Application to Convolutional Autoencoders The proposed regularizer as well as the diagnostic methods also work for convolutional autoencoders. Adapting the architecture proposed by Moor et al. (2020) to ensure a two-dimensional latent space, we train convolutional autoencoders on the image datasets MNIST and FashionMNIST. Indeed, the geometric regularizer still ensures more homogeneous contraction (see Figure S4). The corresponding quantitative comparisons can be found in Tables S1 and S6.

Pulling Back other Metrics In the main paper, we have discussed how pulling back the Euclidean metric from output space yields a metric on latent space which respects the geometry of the reconstruction manifold M.

The method we describe, however, is not limited to the Euclidean metric. Rather, it is applicable if the output space is equipped with a Riemannian metric g' or a scalar product. Let such a Riemannian metric g' be represented by a matrix $A \in \mathbb{R}^{n,n}$. Then it follows directly from the proof of Proposition 2.2, that the pullback D^*g' is in coordinates given by

$$D^*g'_p = (J_p D)^t \, A_D(p) J_p D \tag{41}$$

for any point $p \in \mathbb{R}^l$. In the case of a general scalar product, we can simply pullback the scalar product.

Note that for instance the ℓ^1 distance on euclidean space is not induced by a scalar product and hence not a Riemannian metric. Nevertheless, it would be possible, albeit somewhat inconsistent, to use the ℓ^1 metric in the reconstruction loss, but pull back the standard scalar product in Euclidean space.

Training with other Loss Functions Even though all the autoencoders considered in this paper are trained with main ℓ^2 loss, our method is non-restrictive in the choice of loss function. This is because the regularizer depends only on the decoder's architecture. In particular, geometric autoencoders can also be trained with ℓ^1 or cross-entropy loss.

E.3. Effect on Downstream Applications

Since autoencoders are often part of a larger pipeline, a possible question is how the geometric regularizer affects downstream applications. While the focus of our work is on visualization and representations for downstream tasks typically have more than two dimensions, we did compare the class separation in the two-dimensional vanilla and geometric embeddings. First, we cluster both the vanilla and the geometric embedding with HDBSCAN as in Böhm et al. (2023). We then evaluate these clusters against the existing class labels by computing the adjusted Rand index. Looking at Figure S1, we would expect our geometric autoencoder to outperform the vanilla autoencoder, as it creates visually better separated clusters. For HDBSCAN we use the sklearn implementation with default parameters. Table S4 shows that our method has a higher score on all datasets and thus outperforms the vanilla model.

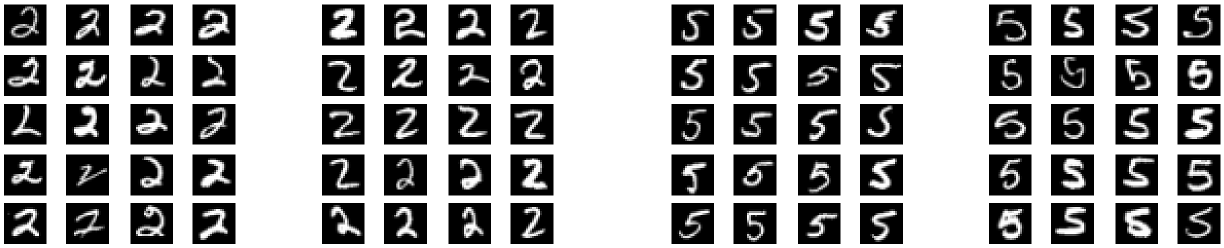
Table S4. Adjusted Rand index of class labels after performing an HDBSCAN clustering on the vanilla and the geometric embeddings with default parameters. Averaged over five random initializations of the network. Bold indicates first place.

DATASET	VANILLA SCORE	GEOMETRIC SCORE
MNIST	0.21 ± 0.09	$\mathbf{0.23 \pm 0.15}$
FASHIONMNIST	0.18 ± 0.06	$\mathbf{0.21 \pm 0.07}$
PBMC	0.047 ± 0.012	$\mathbf{0.074 \pm 0.011}$
CELEGANS	0.036 ± 0.004	$\mathbf{0.095 \pm 0.010}$
ZILIONIS	0.47 ± 0.13	$\mathbf{0.57 \pm 0.14}$

E.4. Semantic Information in MNIST embeddings

Comparing the vanilla and the geometric embeddings of MNIST (Figures 3a,b), one can see that the geometric autoencoder creates two clusters for the digit 2 (pink). We manually inspected these clusters and found that the lower left cluster of the geometric embedding contains digits with loop and curved lower stroke, the upper right cluster contains digits without loop and curved lower stroke (samples can be found in Figure S8). The vanilla autoencoder is also able to pick up on this signal, but not as well as the geometric autoencoder, because one of the two clusters is in the densely packed region of the vanilla autoencoder embedding and thus barely visible. Hence, the geometric autoencoder is able to pick up and display some semantic better that the vanilla autoencoder. In Figure 3c, one can see that the topological autoencoder also creates two clusters for the digit 2. Investigating them more closely we find that they too differentiate between more curly and more straight 2's. Similar to the vanilla autoencoder, the second cluster for the topological autoencoder is in the dense region of the embedding, so that it is hard to spot.

A similar phenomenon holds for the digit 5. The geometric autoencoder differentiates between slanted digits in the left cluster and straight digits in the right cluster. Again, so does the vanilla autoencoder, but due to the heavy overlapping in the contracted area we can only see one cluster.



(a) Lower Cluster

(b) Upper Cluster

(c) Left Cluster

(d) Right Cluster

Figure S8. Random ground-truth samples from the two clusters of each of the digit 2 and 5 in the geometric MNIST embedding (Figure 3a). While the upper right cluster of the digit 2 contains mainly straight digits, the lower left cluster contains mostly ones with a loop. For the digit 5, we observe that the digits corresponding to the left cluster are slanted, while the ones corresponding to the right cluster are not.

Table S5. Table underlying the aggregated metrics of Table 1. Additionally contains the reconstruction loss wherever existent (MSE). Averaged over five runs, bold+underlined indicates first, bold second place. The arrows point to the desirable direction of each metric. ConvNet results are considered separately, see Appendix E.2 and Table S6.

		LOCAL			GLOBAL			
DATASET	MODEL	$KL_{0.1}$ (↓)	KNN (↑)	TRUST (↑)	STRESS (↓)	KL_{100} (↓)	SPEAR (↑)	MSE (↓)
MNIST	GEOM AE (OURS)	0.169 ± 0.023	0.356 ± 0.007	0.938 ± 0.003	**6.2 ± 1.2**	2.2E-07 ± 3E-08	0.4 ± 0.02	0.0357 ± 0.0003
	VANILLA AE	**0.133 ± 0.007**	0.322 ± 0.01	0.93 ± 0.004	11 ± 3	1.8E-07 ± 2E-08	0.44 ± 0.05	**0.0356 ± 0.0007**
	TOPO AE	**0.094 ± 0.003**	0.311 ± 0.005	0.925 ± 0.002	8.91 ± 0.05	**9.3E-08 ± 6E-09**	**0.64 ± 0.01**	0.03701 ± 8E-05
	UMAP AE	0.18 ± 0.007	**0.4104 ± 0.0011**	**0.9483 ± 0.0003**	7.3 ± 0.6	3.1E-07 ± 2E-08	0.34 ± 0.02	**0.0335 ± 0.0003**
	UMAP	0.19 ± 0.002	0.4013 ± 0.0003	**0.94638 ± 0.00051**	**4.79 ± 0.03**	4.1E-07 ± 1E-08	0.3377 ± 0.0042	-
	t-SNE	0.168 ± 0.031	**0.404 ± 0.003**	0.9443 ± 0.0005	39.8 ± 0.1	2.9E-07 ± 4E-08	0.3 ± 0.03	-
	PCA	0.16276402 ± 2.2E-07	0.117955 ± 1.1E-06	0.7456815 ± 5E-07	6.5830853 ± 8E-07	**1.636274E-07 ± 2E-13**	**0.5246475 ± 7E-07**	0.055636764 ± 8E-09
FASHIONMNIST	GEOM AE (OURS)	**0.0407 ± 0.0052**	0.37 ± 0.03	**0.971 ± 0.003**	7 ± 1	**9.6E-08 ± 1.1E-08**	0.75 ± 0.03	**0.02562 ± 0.00013**
	VANILLA AE	0.069 ± 0.031	0.34 ± 0.02	0.9666 ± 0.002	14 ± 2	1.6E-07 ± 1E-07	0.66 ± 0.12	**0.0253 ± 0.0003**
	TOPO AE	**0.049 ± 0.01**	0.366 ± 0.003	0.9686 ± 0.00073	9.569 ± 0.081	1.1E-07 ± 2E-08	**0.82 ± 0.02**	0.0261 ± 0.0002
	UMAP AE	0.0925 ± 0.0073	0.4147 ± 0.0072	**0.971 ± 0.003**	10.86 ± 0.51	5.36E-07 ± 3.1E-08	0.595 ± 0.012	0.0258 ± 0.001
	UMAP	0.0947 ± 0.0021	**0.422 ± 0.002**	0.971 ± 0.001	**4.416 ± 0.022**	3.01E-07 ± 1E-08	0.603 ± 0.002	-
	t-SNE	0.072 ± 0.004	**0.441 ± 0.002**	0.96872 ± 0.0006	39 ± 0.2	2.5E-07 ± 1.2E-08	0.56 ± 0.03	-
	PCA	0.052010267 ± 8E-09	0.2076921 ± 4E-07	0.91678396 ± 5E-08	**4.5253376 ± 4E-07**	**7.084261E-08 ± 2E-14**	**0.88169565 ± 1E-08**	0.046092747 ± 2E-09
CELEGANS	GEOM AE (OURS)	**0.047 ± 0.009**	0.464 ± 0.01	0.956 ± 0.002	17.6 ± 1.2	**1.4E-07 ± 3.1E-08**	**0.683 ± 0.091**	0.73 ± 0.02
	VANILLA AE	0.09 ± 0.03	0.42 ± 0.02	0.943 ± 0.007	36 ± 9	2.8E-07 ± 1E-07	0.5 ± 0.1	**0.71 ± 0.02**
	TOPO AE	**0.056 ± 0.004**	0.47 ± 0.007	**0.9561 ± 0.0013**	19.8 ± 0.3	**1.5E-07 ± 2E-08**	**0.72 ± 0.02**	0.724 ± 0.004
	UMAP AE	0.067 ± 0.011	**0.506 ± 0.004**	**0.963 ± 0.002**	**13.27 ± 0.21**	2E-07 ± 6E-08	0.554 ± 0.05	**0.6751 ± 0.004**
	UMAP	0.058 ± 0.003	**0.4853 ± 0.001**	0.946 ± 0.002	**13.35 ± 0.05**	1.6E-07 ± 8E-09	0.599 ± 0.01	-
	t-SNE	0.057 ± 0.006	0.4697 ± 0.0031	0.93 ± 0.004	29.8 ± 0.2	1.81E-07 ± 2.3E-08	0.494 ± 0.02	-
	PCA	0.08170186 ± 6E-08	0.16197602 ± 2.1E-07	0.8143107 ± 3E-07	14.1533186 ± 1E-06	2.501011E-07 ± 2E-13	0.6426984 ± 2E-07	2.06 ± 0
ZILIONIS	GEOM AE (OURS)	0.11 ± 0.013	0.3945 ± 0.0072	**0.943 ± 0.002**	17 ± 2	**2.3E-07 ± 1E-07**	0.71 ± 0.04	0.338 ± 0.004
	VANILLA AE	0.14 ± 0.01	0.361 ± 0.008	0.939 ± 0.003	24 ± 8	2.7E-07 ± 6E-08	0.64 ± 0.12	**0.3332 ± 0.0022**
	TOPO AE	0.124 ± 0.003	0.353 ± 0.003	0.924 ± 0.003	19.32 ± 0.06	2.81E-07 ± 3.1E-08	0.734 ± 0.021	0.3431 ± 0.0004
	UMAP AE	**0.085 ± 0.01**	**0.407 ± 0.002**	**0.9451 ± 0.0012**	**10.36 ± 0.07**	3E-07 ± 1.2E-07	0.72 ± 0.04	**0.332 ± 0.001**
	UMAP	0.099 ± 0.006	0.387 ± 0.002	0.93717 ± 0.00033	12.48 ± 0.23	3E-07 ± 3E-08	**0.74 ± 0.03**	-
	t-SNE	**0.0977 ± 0.0051**	**0.3967 ± 0.0041**	0.938 ± 0.002	27.09 ± 0.11	**2.2E-07 ± 1.2E-08**	0.516 ± 0.05	-
	PCA	0.11343118 ± 1E-08	0.2175087 ± 3.3E-06	0.86533776 ± 1.1E-07	**12.2618798 ± 3E-07**	2.944264E-07 ± 3E-13	**0.80789925 ± 7E-08**	0.59147804 ± 2E-08
PBMC	GEOM AE (OURS)	**0.0163 ± 0.0023**	**0.2435 ± 0.0009**	**0.9084 ± 0.0006**	6.4 ± 0.4	**1.1E-07 ± 2E-08**	0.847 ± 0.011	**0.3703 ± 0.0011**
	VANILLA AE	0.0653 ± 0.0032	0.221 ± 0.003	0.902 ± 0.002	15 ± 8	1.98E-07 ± 4.3E-08	0.72 ± 0.09	**0.371 ± 0.002**
	TOPO AE	0.022 ± 0.002	0.23222 ± 0.00092	**0.9037 ± 0.0007**	7.37 ± 0.06	**7.5E-08 ± 1E-09**	**0.871 ± 0.011**	0.3731 ± 0.0008
	UMAP AE	0.026 ± 0.005	**0.2382 ± 0.0007**	0.90174 ± 0.0003	**4.1 ± 0.3**	1.7E-07 ± 7E-08	0.82 ± 0.01	0.37955 ± 0.00081
	UMAP	0.027 ± 0.004	0.21599 ± 0.00051	0.8858 ± 0.0003	**3.84 ± 0.12**	1.61E-07 ± 6.2E-08	0.84 ± 0.02	-
	t-SNE	0.038 ± 0.002	0.237 ± 0.001	0.8946 ± 0.0011	24.4 ± 0.09	1.3E-07 ± 2E-08	0.674 ± 0.022	-
	PCA	**0.012270422 ± 3E-09**	0.12920468 ± 3E-07	0.82435367 ± 1E-07	4.91943945 ± 4.1E-07	1.1969531E-07 ± 9E-14	**0.91108369 ± 2E-08**	0.59559689 ± 3E-08

Table S6. Quantitative comparison of a convolutional vanilla autoencoder and a convolutional geometric autoencoder. Averaged over five runs, bold indicates first place. The arrows point to the desirable direction of each metric.

		LOCAL			GLOBAL			
DATASET	MODEL	$KL_{0.1}$ (↓)	KNN (↑)	TRUST (↑)	STRESS (↓)	KL_{100} (↓)	SPEAR (↑)	MSE (↓)
MNIST	CONV GEOMAE	0.16 ± 0.01	0.189 ± 0.005	0.8396 ± 0.009	**4.98 ± 0.32**	1.9E-07 ± 2E-08	0.52 ± 0.03	0.0467 ± 0.0008
	CONV VANILLA	**0.1304 ± 0.0043**	**0.195 ± 0.005**	**0.847 ± 0.005**	6 ± 2	**1.7E-07 ± 4E-08**	**0.57 ± 0.03**	**0.0456 ± 0.0005**
FASHIONMNIST	CONV GEOMAE (OURS)	0.051 ± 0.005	**0.283 ± 0.004**	**0.951 ± 0.0008**	**5.27 ± 0.92**	**9.7E-08 ± 1.3E-08**	0.82 ± 0.03	0.0322 ± 0.0004
	CONV VANILLA	**0.046 ± 0.006**	0.2809 ± 0.0021	0.95089 ± 0.00053	14.3 ± 1.1	1.1E-07 ± 2E-08	**0.85 ± 0.02**	**0.0312 ± 0.0002**