Geometric Encoder Regularization for Autoencoders

Philipp Nazari [1]

Abstract

In this study we adapt existing decoder-based geometric regularization methods for autoencoders to depend only on the encoder. This shift aligns more naturally with the primary function of autoencoders in generating low-dimensional embeddings. Specifically, we replace the pullback metric with a pushforward. Although this pushforward does not account for all of the geometric information encoded in the data manifold, it is applicable to a much broader spectrum of models.

1. Introduction

Low-dimensional visualizations allow researchers to study even naturally high-dimensional datasets. It is thus crucial for these low-dimensional representations to carry as much information about the original dataset as possible.

Most visualizations are two-dimensional, and thus a certain amount of information loss is inevitable. However, by applying geometrically informed regularization, one can hope to minimize this distortion and maintain as much information of the dataset as possible.

Previous work (Nazari et al., 2023; Lee & Park, 2023; Lee, 2023; Lee et al., 2022; Arvanitidis et al., 2017) take a geometric perspective on the decoder to better understand the geometry of an autoencoders latent-space. Using techniques from differential geometry, they develop models which respect different aspects of the datasets geometry, making latent-space more interpretable. This is for example useful in bio-medicine (Kobak & Berens, 2019; Zheng et al., 2017; Packer et al., 2019), where low-dimensional representation of higher-dimensional datasets is a crucial part of data analysis.

More specifically, the aforementioned models metrize latent-space in a geometrically informed manner by using the decoder to pull back the ambient metric. The downside of this approach is that this regularization only indirectly affects the encoder via the reconstruction loss, even though it is the encoder which creates the embedding.

It is thus natural to ask whether these techniques can be adapted to directly affect the encoder. In this paper, we address this issue by shifting existing regularization methods from the decoder to the encoder, substituting the pullback with a pushforward approach.

This approach, however, also has a downside: due to mathematical constraints it suffers from a loss of information (more details in Section 2), which is related to the low dimensionality of latent-space.

The structure of this paper is as follows. In Section 2 we begin by providing some mathematical background on existing geometric regularization techniques and how we adapt them for our purpose. In Section 3 we specifically introduce the adapted techniques. In Section 4 we introduce novel metrics to measure the conservation of geometry, and the following sections will be focused on experiments.

To summarize, in this paper we

- introduce a technique called "pushforward" to metrize latent-space in a geometrically informed matter using only the encoder,
- qualitatively and quantitatively compare the two regularization approaches for several models.

The code can be found at `https://github.com/phnazari/GeometricEncoderRegularization`.

2. Mathematical Background

2.1. The Pullback Metric

Given an autoencoder, consisting of an encoder $E\colon \mathbb{R}^n \to \mathbb{R}^l$ and a decoder $D\colon \mathbb{R}^l \to \mathbb{R}^n$, Nazari et al. (2023) describe how one can interpret, under some mathematical assumptions, the image of the decoder as an l-dimensional manifold living in $\mathbb{R}^n$. It has an atlas consisting of precisely one chart, namely the decoders inverse D^{-1}.

Nazari et al. (2023) then use the decoder to equip latent-space with the pullback g of the ambient Euclidean metric of $\mathbb{R}^n$. Explicitly, given a latent point $p \in \mathbb{R}^l$ and two vectors

[1]Department of Mathematics, ETH Zurich, Switzerland. Correspondence to: Philip Nazari <pnazari@student.ethz.ch>.

PREPRINT

v, w in its tangent space, their inner product is defined as

$$g_p(v, w) = v^T J_p^T D J_p D w, \tag{1}$$

where $J_p D$ is the Jacobian of the decoder D evaluated at the point p (see Figure 1)

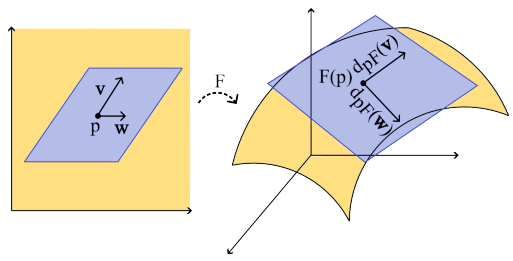


Figure 1. The figure illustrates the pullback metric along a smooth map $F\colon \mathbb{R}^2 \to \mathbb{R}^3$. Given a point $p \in \mathbb{R}^2$ and two vectors u and v from the tangent space at p (purple), their product in the pullback metric is defined as the product of their images under F's differential $J_p F$ (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 (taken from (Nazari et al., 2023)).

2.2. The Pushforward (Our Contribution)

While the above description allows a metrization of latent-space which respects the datasets geometry, it has the arguable disadvantage of relying on the existence of a decoder. One could say that this is somewhat artificial, since it is effectively the encoder which creates the embeddings. Indeed, there are embedding methods which only rely on an encoder (van der Maaten & Hinton, 2008; Damrich & Hamprecht, 2021).

In the following, assume we are given a differentiable encoder $E\colon \mathbb{R}^n \to \mathbb{R}^l$ which embeds an n-dimensional dataset into some lower (l-) dimensional latent-space. We will develop a method for first, measuring and second, relaxing the geometric distortion introduced by this encoder. While it will be not be as powerful as decoder-based methods, our method will rely on similar considerations and be less restrictive.

The Problem Our goal is to use the encoder to "push forward" the ambient Euclidean metric from $\mathbb{R}^n$ to $\mathbb{R}^l$. This will in general not be possible, as the encoder E can neither be injective nor an immersion for $n > l$. Intuitively, a point in latent-space can have multiple pre-images, and it is not clear which one to pick for the metrization. In the same way, a tangent vector can have multiple pre-images. Thus, a pushforward will in general not be an actual metric, as it need f.e. not be positive definite.

In mathematical terms, the fact that two tangent vectors of the data-manifold might be mapped to the same vector in latent-space is due to the Jacobian of the encoder having a non-trivial *kernel*.

The Solution Missing injectivity of the encoder is not a problem in our application, since we restrict our attention in latent-space to the embedding of the dataset. In particular, we know for each embedded point which pre-image to choose, namely the corresponding input. The fact that the encoder need not be an immersion (i.e. it's Jacobian can have a non-trivial kernel) is not as easy to resolve, and we dedicate the rest of this chapter towards this problem.

In the following, let X be a dataset and assume w.l.o.g. that $l = 2$, i.e. that the embedding is two-dimensional.

Let $x \in X \subset \mathbb{R}^n$ be a data point with embedding $z = E(x) \in \mathbb{R}^2$. Then, in general, $J_x E$ will have a non-trivial kernel. However, restricted to the orthogonal complement $\ker(J_x E)^\perp$ of that kernel, the Jacobian $J_x E|_{\ker(J_x E)^\perp}$ is an immersion. In the following, we will refer to $\ker(J_x E)$ as the *parallel* and to $\ker(J_x E)^\perp$ as the *horizontal tangent space* at x. Using this terminology, we can say that the restriction of the Jacobian to the horizontal tangent space is an immersion (See Figure 2 for a visual explanation).

By the nature of our problem, we are not be able to account for the information encoded in the vertical tangent space (which corresponds, by definition, to the kernel). However, we can still hope to metrize latent-space in a way that respects the horizontal tangent-space, on which the encoders' Jacobian is an immersion. And indeed, we develop such a metric in the following paragraphs.

By the Rank-Nullity Theorem (and assuming that $J_x E$ is surjective, which we can assume for $n >> 2$), we know that

$$\dim \ker(J_x E) = n - 2 \tag{2}$$

$$\dim \ker(J_x E)^\perp = 2. \tag{3}$$

Thus, we can not account for $n-2$ dimensions of the dataset.

The remaining two dimensions we can control the following way. Recall that for a matrix $A \in \mathbb{R}^{k,l}$, the "pseudo inverse" is defined as $A^+ := \left(A^T A\right)^{-1} A^T \in \mathbb{R}^{l,k}$ and satisfies $A^+ A = \mathbb{I}_l$. Intuitively, the pseudo inverse A^+ can be thought of as "the best we can do" inverting A. In the terminology developed above, we can only hope to invert the matrix on the horizontal tangent space (to see this, consider the SVD $A = U\Sigma V^T$ of A and acknowledge that $A^+ = V\Sigma^+ U^T$ where Σ^+ contains the reciprocals of all non-zero singular values).

As a next step towards developing the pushfoward, note that for any 2-form ω (for example a metric) and any diffeomorphism f, the pushforward $f_*\omega$ of ω under f is the same as the pullback $\left(f^{-1}\right)^* \omega$ of ω under f^{-1}.

We would like to apply this fact to the Jacobian $f = J_x E$ of the encoder and the Euclidean metric $\omega = \mathbb{I}_n$ of input-space. The problem that $J_x E$ is not a diffeomorphism, and thus not invertible, can be accounted for by picking the above-mentioned pseudo-inverse instead of the inverse.

Definition 2.1. Let $x \in X \subset \mathbb{R}^n$ be a data point and $v, w \in \mathbb{T}_x R^n \simeq \mathbb{R}^n$ two tangent-vectors located at x. We then define the *pushforward* of the Euclidean metric g_0 on $\mathbb{R}^n$ under the Jacobian $J_x E$ of the encoder to be

$$\left((J_x E)_* g_0\right)_x (v, w) \coloneqq g_0 \left(J_x E^+ v, J_x E^+ w\right). \quad (4)$$

In Euclidean coordinates, the pushfoward takes a rather simple form:

Proposition 2.2. *In Euclidean coordinates, the pushforward at $x \in X$ takes the form*

$$\left((J_x E)_* g_0\right)_x = \left(J_x E J_x^T E\right)^{-1} \in \mathbb{R}^{2,2}. \quad (5)$$

Proof. Note that for any matrix $A \in \mathbb{R}^{k,l}$, assuming that $A^T A$ is invertible (a necessary condition for the existence of A^+), we can write

$$\begin{aligned} \left(A^+\right)^T A^+ &= \left(A^T\right)^+ A^+ \\ &= \left(A A^T\right)^+ \\ &= \left(A A^T\right)^{-1}. \end{aligned}$$

Apply this fact to

$$\left((J_x E)_* g_0\right)_x (v, w) = v^T \left(J_x E^+\right)^T J_x E^+ w.$$

□

Note that, compared to the pullback metric in Equation (1), the pushforward metric in Equation (5) uses the Encoder instead of the decoder, the position of the transposed is switched and there is an inverse.

3. Regularizing the Encoder

Now that we have equipped latent-space with the pushforward, we can proceed analogously to previous work that used the pullback-metric, like f.e. (Nazari et al., 2023; Lee et al., 2022). However, instead of just replacing every occurrence of the pullback metric by the pushforward, we replace it by the inverse $J_x E J_x^T E$ of the pushforward.

This improves numerical stability and has a very similar affect to regularizing the pullback metric itself. Indeed, inverting the pushforward metric corresponds to inverting the two singular values (flipping the corresponding indicatrix (Section 5) by ninety degrees and scaling the main axes).

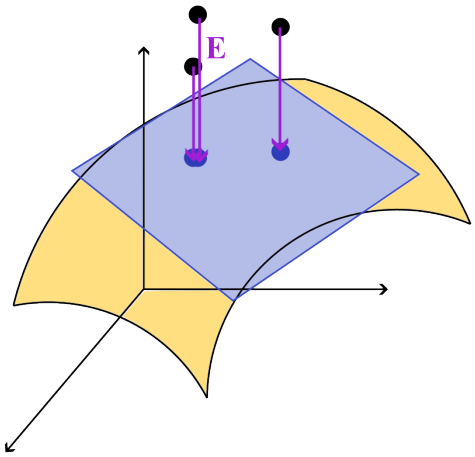


Figure 2. The figure illustrates a pushforward metric. The encoder (violet) maps points onto a manifold. While all information parallel to the direction of projection is lost, the information orthogonal to it is maintained.

In the following, we introduce three geometrically regularized models which should preserve more of the datasets geometry.

Volume Preservation In order to regularize the encoder to preserve relative volumes, we adapt the approach by (Nazari et al., 2023), replacing the pullback metric by the inverse of the pushforward. The regularization loss then becomes

$$\mathcal{L}_{\text{vol}} = \text{Var}_{x \sim \mathcal{U}(X)} \left(\log\left(\det\left(J_x E J_x^T E\right)\right)\right). \quad (6)$$

In words, the loss measures the variance of the generalized Jacobian determinant over the dataset.

Shape Preservation In order to regularize the encoder to preserve shapes (i.e. be conformal), we adapt the approach of an ongoing work, replacing the pullback metric by the inverse of the pushforward. The regularization loss then becomes

$$\mathcal{L}_{\text{shape}} = m^2 \mathbb{E}_{x \sim U(X)} \left[\frac{\text{Tr}\left(J_x E J_x^T E\right)^2}{\text{Tr}^2\, J_x D J_x^T E}\right] - m. \quad (7)$$

Volume and Shape Preservation In order to regularize the encoder to preserve both shapes and volumes up to a global factor (i.e. be a scaled isometry), we adapt the approach of Lee et al. (2022), replacing the pullback metric by a the inverse of the pushforward. The regularization loss then becomes

$$\mathcal{L}_{\text{shape}} = m^2 \frac{\mathbb{E}_{x \sim U(X)} \left[\text{Tr}\left(J_x E J_x^T E\right)^2\right]}{\mathbb{E}_{x \sim \mathcal{U}(X)} \left[\text{Tr}^2\, J_x E J_x^T E\right]} - m. \quad (8)$$

4. Geometric Metrics

In order to measure how well the models capture geometric information of the underlying dataset, we introduce the metrics VPres, CN and ISO.

In the following, assume we are given a family $\{G_i\}_{i=1,...,N} \subset \mathbb{R}^{l,l}$ of metric tensors, either coming from the decoder via a pullback or the encoder via a pushforward. The idea of the following metrics is to use these metric tensors to measure how well the corresponding part of the autoencoder (i.e. the encoder or the decoder) captures volumes and/or angles across a given embedding (corresponding to size and/or shape). A mapping which preserves volumes is called "volume preserving", one that accurately captures angles is called "conformal".

All of the given measures should be scale-invariant, meaning they should stay constant under a global scaling $G_i \mapsto \alpha \cdot G_i$ for some $\alpha \neq 0$ and all $i = 1, ..., N$.

In the following, given a G_i denote by $G_i = U_i \Sigma_i V_i^*$ the singular value decomposition of G_i with $\Sigma_i = \mathrm{diag}(\sigma_{i_1}, ..., \sigma_{i_l})$ the diagonal matrix containing the singular values σ_{i_j}.

Conformality Measure In order to measure conformality (i.e. the ability of the model to capture shapes) of the model, let $\sigma_{\min,i}, \sigma_{\max,i}$ be the minimal and maximal singular values of G_i. Then we define

$$\mathrm{CN} := \mathbb{E}_i \left[\left| 1 - \frac{\sigma_{\min,i}}{\sigma_{\max,i}} \right| \right] \tag{9}$$

Note that G comes from a conformal mapping if and only if $\sigma_{\min} = \sigma_{\max}$, which corresponds to $\mathrm{CN} = 0$. Furthermore, CN is scale invariant.

Volume Preservation Measure In order to measure the extent to which the model accurately captures volumes, we define

$$\mathrm{VPres} := \mathrm{Var}_i \left[|\log (\det G_i)| \right] \tag{10}$$

Note that G comes from a volume preserving map if and only if $\det G_i \equiv \det G$ for all $i = 1, ..., N$, which corresponds to $\mathrm{VPres} = 0$.

(Scaled) Isometry Measure In order to measure how close the model is to being a scaled isometry, i.e. preserving both shape and volume, we define

$$\mathrm{ISO} := \mathrm{Var}_{i_j} \left[\log \left(\sigma_{i_j} \right) \right], \tag{11}$$

where the variance is taken over the singular values of all metric tensors.

Note that G is a scaled isometry if and only if $G_i \equiv \alpha G$ for some $\alpha \in \mathbb{R} \setminus \{0\}$, which corresponds to $\mathrm{ISO} = 0$.

5. Experiments

We adapt the Conformal Autoencoder (ConfAE, ongoing research), the Geometric Autoencoder (GeomAE) (Nazari et al., 2023) and the Isometry Regularized Autoencoder (IRAE) (Lee et al., 2022) using encoder-based regularization techniques (see Table 2 to see what these models are optimized for).

In the following, we will refer to both encoder- and decoder regularized models by the same name, i.e. "GeomAE" might refer to a geometric autoencoder with either the encoder or the decoder regularized towards preserving volume. It will be clear from the context which case we are discussing.

While we regularize using the inverse $J_x E J_x^T E$ of the pushforward $\left(J_x E J_x^T E\right)^{-1}$ (Section 3), we visualize using the actual pushforward.

For our qualitative and quantitative evaluation, we train, inspect and compare encoder- and decoder-regularized models with respect to their ability to preserve the datasets geometry while embedding it into latent-space. To do so, we use the handwritten-digits dataset MNIST as well as the three mRNA-seq datasets Zilionis (Zilionis et al., 2019), PBMC (Zheng et al., 2017) and CElegans (Packer et al., 2019).

We train the autoencoders for 100 epochs using the ADAM optimizer (Kingma & Ba, 2014) with a learning-rate of 0.0005. We determine the regularization weights to be maximal under the constraint that the MSE of the regularized model is not bigger than 1.3 times that of the vanilla autoencoder. Table 4 lists the weights we used.

We visualize the geometric distortion introduced by the model using *indicatrices* (Nazari et al., 2023). Specifically, indicatrices are a way of visualizing a metric tensor field, in our case either the pullback- or the pushforward metric tensor field. An indicatrix at a point p is defined as the unit-ball in the corresponding metric.

Applied to the pushforward metric and evaluated at an embedding point $E(x)$, it can be interpreted the image under the encoder of the Euclidean unit-sphere around x (along the horizontal tangent space). Applied to the pullback metric, it can be interpreted as those vectors which are mapped to the unit-sphere under the decoder. An indicatrix thus captures information about which direction is squeezed and expanded locally by the encoder/decoder.

6. Related Work

Autoencoders ((Rumelhart et al., 1986)) are commonly used for visualizing otherwise unassessable, high-dimensional datasets (see e.g. (Hinton & Salakhutdinov, 2006)). Such datasets are common in f.e. bioinformatics, often times

		GeomAE (VPres)		ConfAE (CN)		IRAE (ISO)	
		encoder	decoder	encoder	decoder	encoder	decoder
MNIST	encoder	0.008 ± 0.003	1.6 ± 0.2	0.26 ± 0.2	0.637 ± 0.009	0.22 ± 0.02	1.02 ± 0.04
	decoder	2.1 ± 0.1	0.021 ± 0.003	0.6317 ± 0.0042	0.13 ± 0.01	1.1 ± 0.05	0.56 ± 0.12
PBMC	encoder	0.028 ± 0.006	2.7 ± 1.3	0.43 ± 0.1	0.39 ± 0.06	0.2 ± 0.2	0.8 ± 0.7
	decoder	1.2 ± 1.1	0.0099 ± 0.0042	0.62 ± 0.03	0.16 ± 0.03	0.44 ± 0.07	0.6 ± 0.8
Zilionis	encoder	0.012 ± 0.003	2.2 ± 1.2	0.48 ± 0.05	0.58 ± 0.07	0.1274 ± 0.003	2.7 ± 1.2
	decoder	1.65 ± 0.51	0.01 ± 0.01	0.734 ± 0.007	0.205 ± 0.022	0.57 ± 0.02	1.9 ± 1.3
CElegans	encoder	0.02 ± 0.009	0.5 ± 0.2	0.47 ± 0.06	0.47 ± 0.07	0.053 ± 0.007	0.63 ± 0.053
	decoder	0.51 ± 0.21	0.036 ± 0.007	0.611 ± 0.043	0.3 ± 0.04	0.5 ± 0.09	0.146 ± 0.092

Table 1. Geometric metrics for encoder- and decoder regularized autoencoder (column), evaluated for encoder- and decoder-based metrics (row). For GeomAE, we use the VPres metric, for ConfAE the CN metric and for IRAE the ISO metric. Metrics are evaluated on a 64×64 grid of embedded points. In order to deal with outliers, we only consider the 90% quantile. Values are averaged over three random seeds.

Model	Volume	Shape
ConfAE	$\times$	✓
GeomAE	✓	$\times$
IRAE	✓	✓

Table 2. This table provides an overview over which of the three geometrically regularized models ConfAE (ongoing work), GeomAE (Nazari et al., 2023) and IRAE (Lee et al., 2022) are trained to capture volume and/or shape.

consisting of mRNA sequences((Zheng et al., 2017; Zilionis et al., 2019; Packer et al., 2019)).

In order to minimize the the distortion in the embedding (Arvanitidis et al., 2017), previous work explored the topic of geometric regularization from the perspective of the decoder. Nazari et al. (2023) aim to regularize the decoder in a way that makes the embedding respect relative size, resulting in the geometric autencoder (GeomAE). Lee et al. (2022) take a similar approach with the goal of making the decoder respect both shape and size, giving rise to the isometry regularized autoencoder (IRAE). Ongoing research tries filling the gap by developing an autoencoder that respects only the shape of the dataset (conformal autoencoder, ConfAE).

7. Results

In this section we present our results on the MNIST dataset. For more datasets, see Appendix A.2.

7.1. Geometric Cross-Regularization

Our experiments provide useful insights about how decoder regularization affects the encoder and vice versa. Qualitatively, Figure 3 shows that the pushforward indicatrices (i.e. those derived from the *encoder*) in a *decoder*-regularized model exhibit roughly the expected shapes and volumes, demonstrating that geometric regularization of the decoder carries over to the encoder. The same holds true for the roles of encoder and decoder reversed. Furthermore, the elongations of the encoder- and decoder-based indicatrices, based at the same point in the same model, tend to align. This shows that the encoder contracts space wherever the decoder expands it (and the other way around), which is necessary for an accurate reconstruction of the dataset. A similar consideration explains why encoder-based indicatrices tend to have small volume wherever decoder-based ones have large volume, and the other way around.

Table 1 contains the data for a more quantitative study on how the geometric regularization carries over between encoder and decoder. It contains the value of the geometric metrics introduced in Section 4, where for each dataset we examine all four combinations of regularizing and visualizing using either the encoder or the decoder. By fixing a two-by-two submatrix of the table (corresponding to a geometric model, f.e. GeomAE, and a dataset), it is evident that encoder (decoder)-based metrics outperform decoder (encoder)-based ones when employing encoder- (decoder-) based regularization. This can be seen as a proof of concept, showing that the geometric regularization works as expected. However, comparing the off-diagonal entries of the two-by-two submatrices reveals that neither regularization approach has a clear edge over the other when it comes to cross-regularization.

We also notice that geometric cross-regularization seems to perform best wherever there is a lot of data (see for example Figure 3 or the plots in Appendix A.2), as measured by the alignment of indicatrices described above.

Table 3. In this table we compare well encoder- and decoder regularized autoencoder perform at preserving properties of the dataset as measured by various metrics used by (Nazari et al., 2023).

	LOCAL			GLOBAL				
	KL_{01}	KNN	TRUST	STRESS	KL_{100}	SPEAR	MSE (TEST)	MSE (TRAIN)
IRAE	ENCODER	DECODER	DECODER	DECODER	ENCODER	DECODER	DRAW	DRAW
GEOMAE	DRAW	DRAW	DECODER	DECODER	DRAW	DRAW	ENCODER	DECODER
CONFAE	ENCODER	DECODER	DRAW	DECODER	ENCODER	DRAW	DECODER	ENCODER
VANILLAAE	ENCODER	DRAW	DRAW	DECODER	ENCODER	DRAW	DECODER	DECODER

7.2. Comparing Regularization Approaches

Before we compare which regularization approach is better suited for geometrically meaninful reglarization, remember from Section 2 that encoder-based models only account for the parallel tangent space, whereas decoder-based models consider the entire tangent space. The inevitable loss of the information encoded in the horizontal tangent space weights heavier the bigger the difference in dimensionality of input- and latent-space (denoted by n and l, respectively). More quantitatively, by the considerations made in Section 2, the vertical tangent space will have dimensionality $n - l$, and thus the bigger $n - l$ the higher the dimensionality of the space that is dismissed during encoder-based regularization.

Also the decoder suffers from a big difference $n - l$, although this can be mitigated. To see why that is, think of the decoder as effectively folding latent-space into a very high dimensional output space. A sufficiently powerful decoder can effectively embed itself into, and thus accurately reconstruct, datasets even with a large difference $n - l$ is large (thinking in the aforementioned image, it can "twist and turn" a lot, i.e. have a lot of curvature). However, too much expressive power of the decoder can weaken its ability to accurately capture geometric information (overfitting), so there is a geometry-reconstruction trade-off.

Combining these two observations, i.e. that the decoder can, given enough expressive power, overcome a large difference in dimensionality while the encoder is always limited by a vertical tangent space of dimension $n - l$, we can explain why Figure 3 shows that the embeddings from decoder-regularized models appear to be more clustered and visually appealing/interpretable. As a consequence, while encoder regularization is advantageous due to its broader applicability, it lacks a bit the ability of decoder regularization of achieving well-clustered, visually appealing embeddings.

Overall, the quantitative analysis in Table 3 shows that both encoder and decoder regularized models perform comparably well on model-independent metrics, with a slight edge for decoder-based models.

8. Conclusion

In this work we study the visualization capacities of encoder- and decoder regularized autoencoders with a focus on their ability to preserve geometric information. Specifically, we adapt existing encoder-based regularization techniques for autoencoders to be encoder-based. The former is arguably the more natural approach, since it is the encoder which creates the embeddings. Furthermore, the proposed encoder-based regularization techniques can also be applied to architectures that do not have a decoder and are thus more generally applicable.

Our experiments show that, while encoder-based regularization techniques do indeed help preserving more of the datasets geometry, the embeddings created by decoder-regularized models have better seperated clusters. On a handful of quantitative metrics, however, encoder-based techniques are competitive when it comes to accurately representing the dataset.

Future Work Autoencoders are not just used for visualization, but also for data compression as a pre-processing step in a longer pipeline. It would thus be interesting to study the affect of the above techniques on downstream applications. Furthermore, a more regular latent-space is helpful for generative models, sampled points that are close in latent-space should create similar generative outputs. It would thus, for example, be interesting to study the impact of encoder-based regularization on the sample regularity of Variational Autoencoders (Kingma & Welling, 2013).

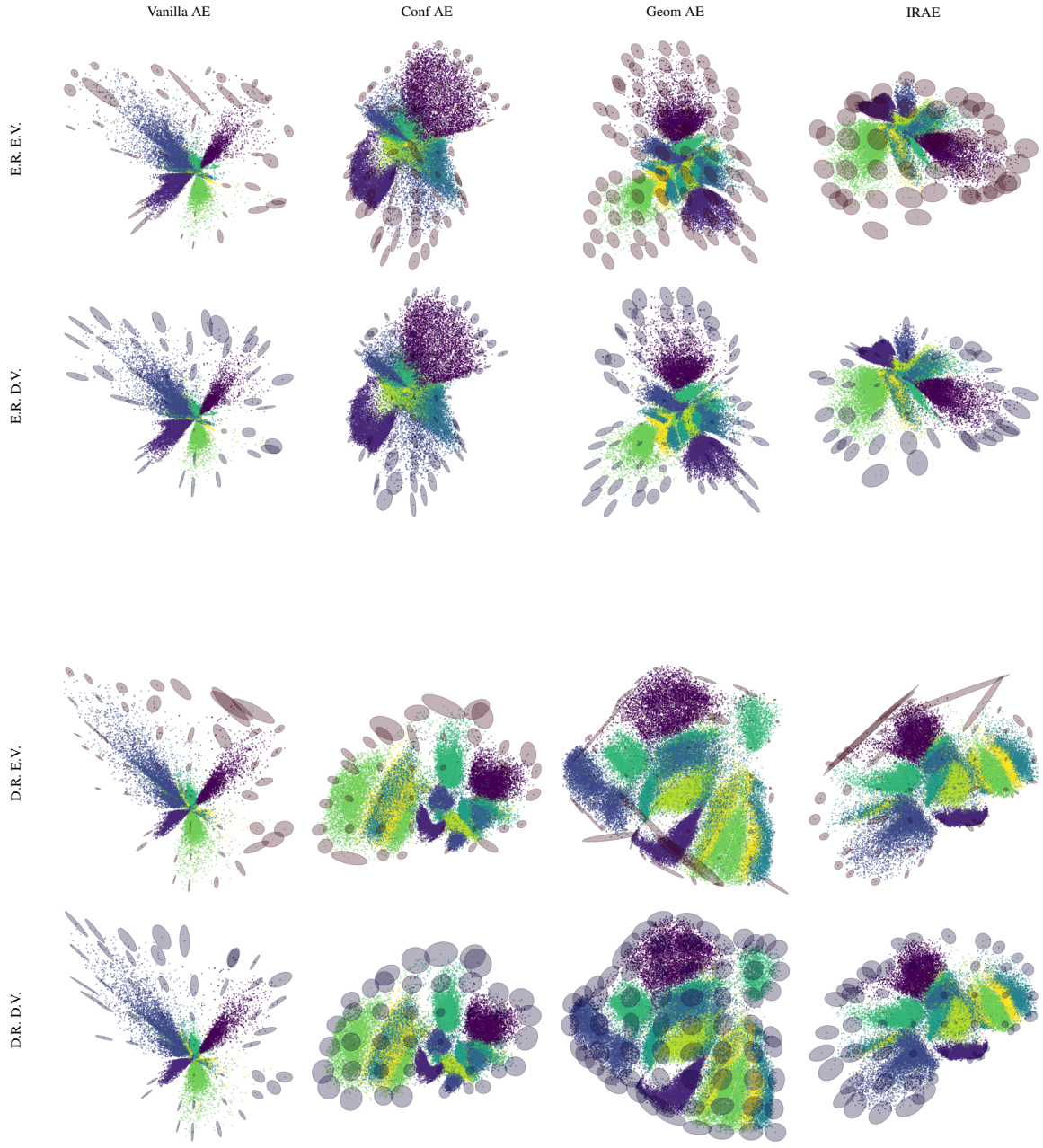


Figure 3. Indicatrices of geometrically regularized autoencoders trained on MNIST. The first block (first two rows) corresponds to encoder-regularized models, the second block (last two rows) corresponds to decoder-regularized models. Inside of each block, the top row shows encoder-based indicatrices (red) and the bottom row shows decoder-based indicatrices.

References

Arvanitidis, G., Hansen, L. K., and Hauberg, S. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017.

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

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.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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.

Lee, Y. A geometric perspective on autoencoders. *arXiv preprint arXiv:2309.08247*, 2023.

Lee, Y. and Park, F. C. On explicit curvature regularization in deep generative models. 2023.

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

Moor, M., Horn, M., Rieck, B., and Borgwardt, K. Topological autoencoders. In *International conference on machine learning*, pp. 7045–7054. PMLR, 2020.

Nazari, P., Damrich, S., and Hamprecht, F. A. Geometric autoencoders - what you see is what you decode. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25834–25857. PMLR, 23–29 Jul 2023.

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.

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.

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.

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. Experiments, Datasets and Metrics

A.1. Metrics

We evaluate each model on three random seeds. We evaluate the embeddings with metrics from (Nazari et al., 2023). The following section describes these metrics. It is copied from Nazari et al. (2023).

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).

A.2. Datasets

Table 4 shows the optimal regularization weights we found for all mode/dataset combinations.

Figure 4 shows the labels of the dataset we used.

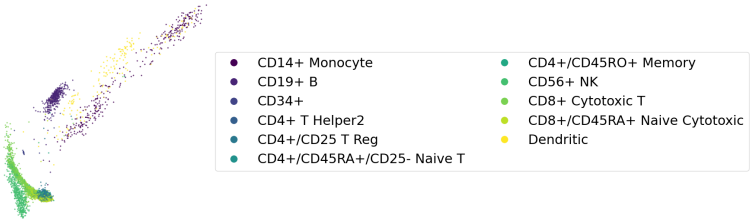
A.3. Results on More Datasets

Figure 5 - Figure 7 contain the indicatrices of the models trained on Zilionis, PBMC and CElegans.

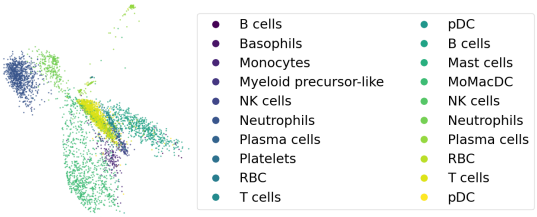
	MNIST		Zilionis		PBMC		CElegans	
model/reg	Encoder	Decoder	Encoder	Decoder	Encoder	Decoder	Encoder	Decoder
ConfAE	0.001	0.1	0.1	0.1	0.1	0.1	0.1	0.1
GeomAE	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
IRAE	0.1	0.1	10	1.0	10	1.0	10	10

Table 4. Regularization strengths for encoder- and decoder regularized models and all datasets used. Determined under the constraint that the corresponding MSE should not exceed 1.3 times that of the Vanilla MSE. Determined using a grid-search.

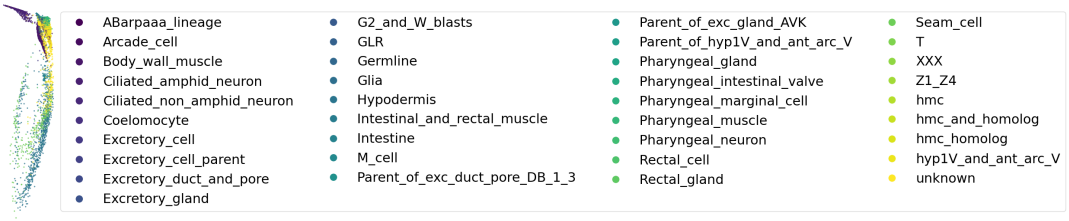
(a) MNIST



(b) PBMC



(c) Zilionis



(d) CElegans

Figure 4. Labels of the datasets used in this work.

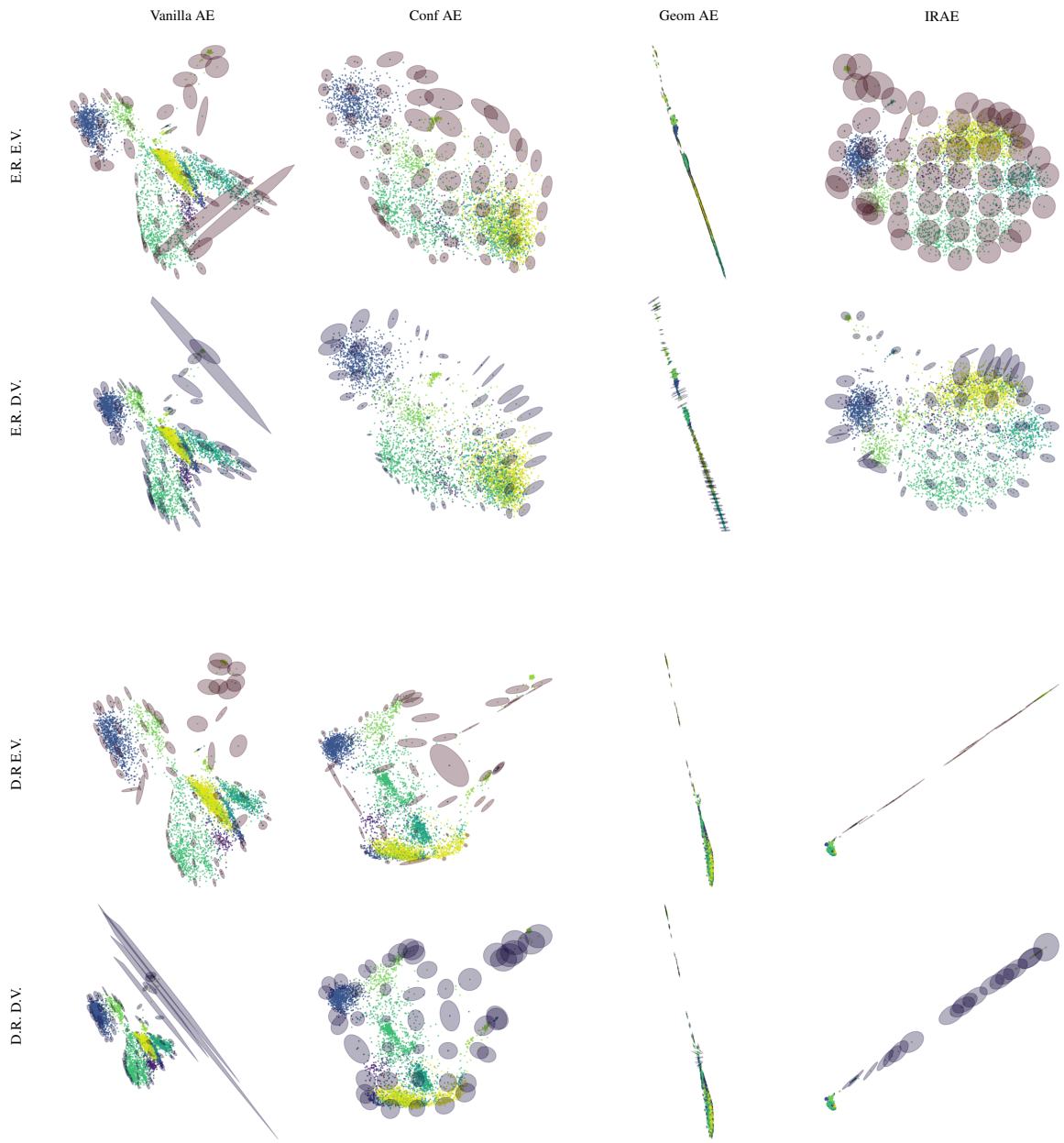


Figure 5. Indicatrices of geometrically regularized autoencoders trained on Zilionis. The first block (first two rows) corresponds to encoder-regularized models, the second block (last two rows) corresponds to decoder-regularized models. Inside of each block, the top row shows encoder-based indicatrices (red) and the bottom row shows decoder-based indicatrices.

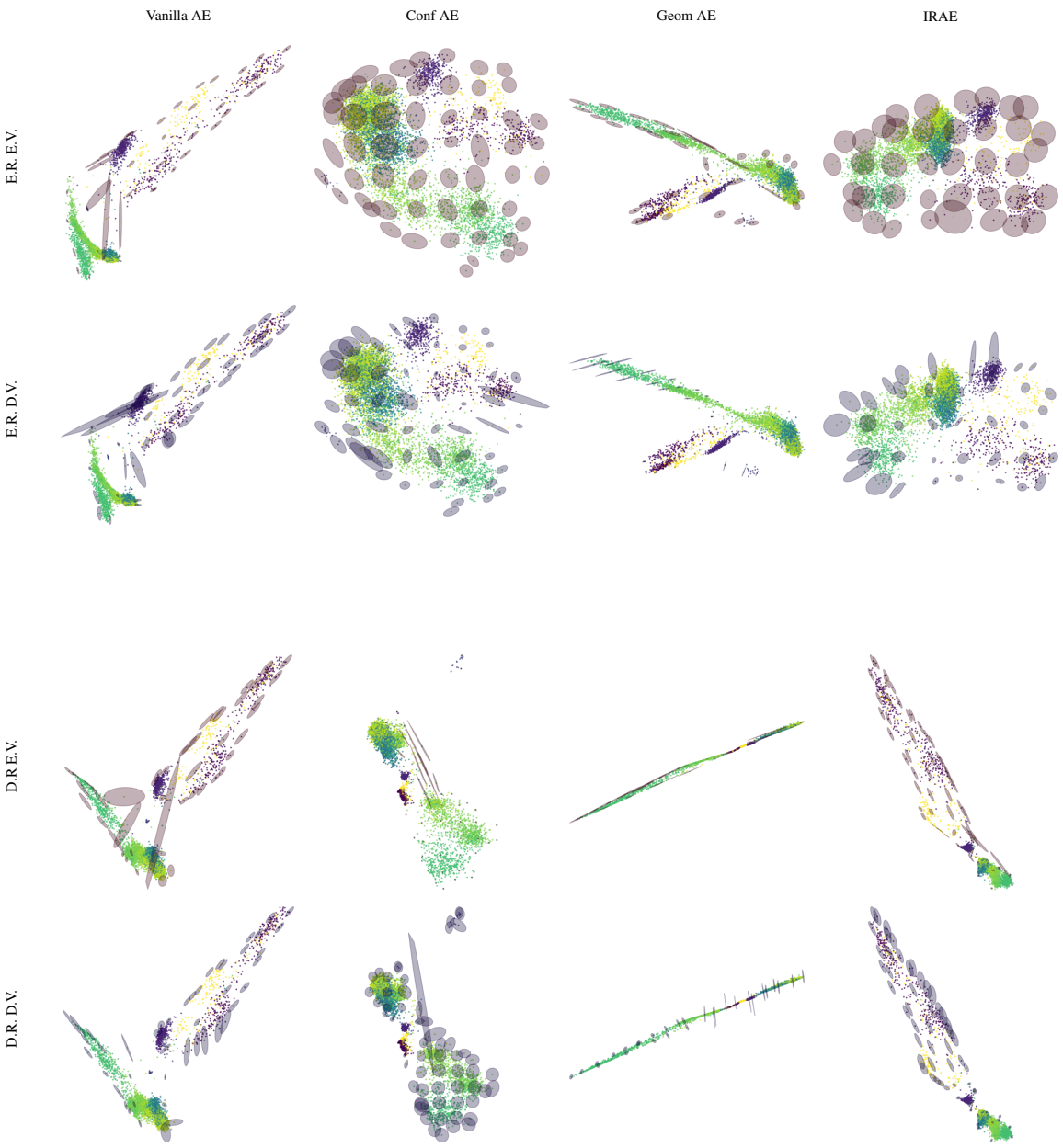


Figure 6. Indicatrices of geometrically regularized autoencoders trained on PBMC. The first block (first two rows) corresponds to encoder-regularized models, the second block (last two rows) corresponds to decoder-regularized models. Inside of each block, the top row shows encoder-based indicatrices (red) and the bottom row shows decoder-based indicatrices.

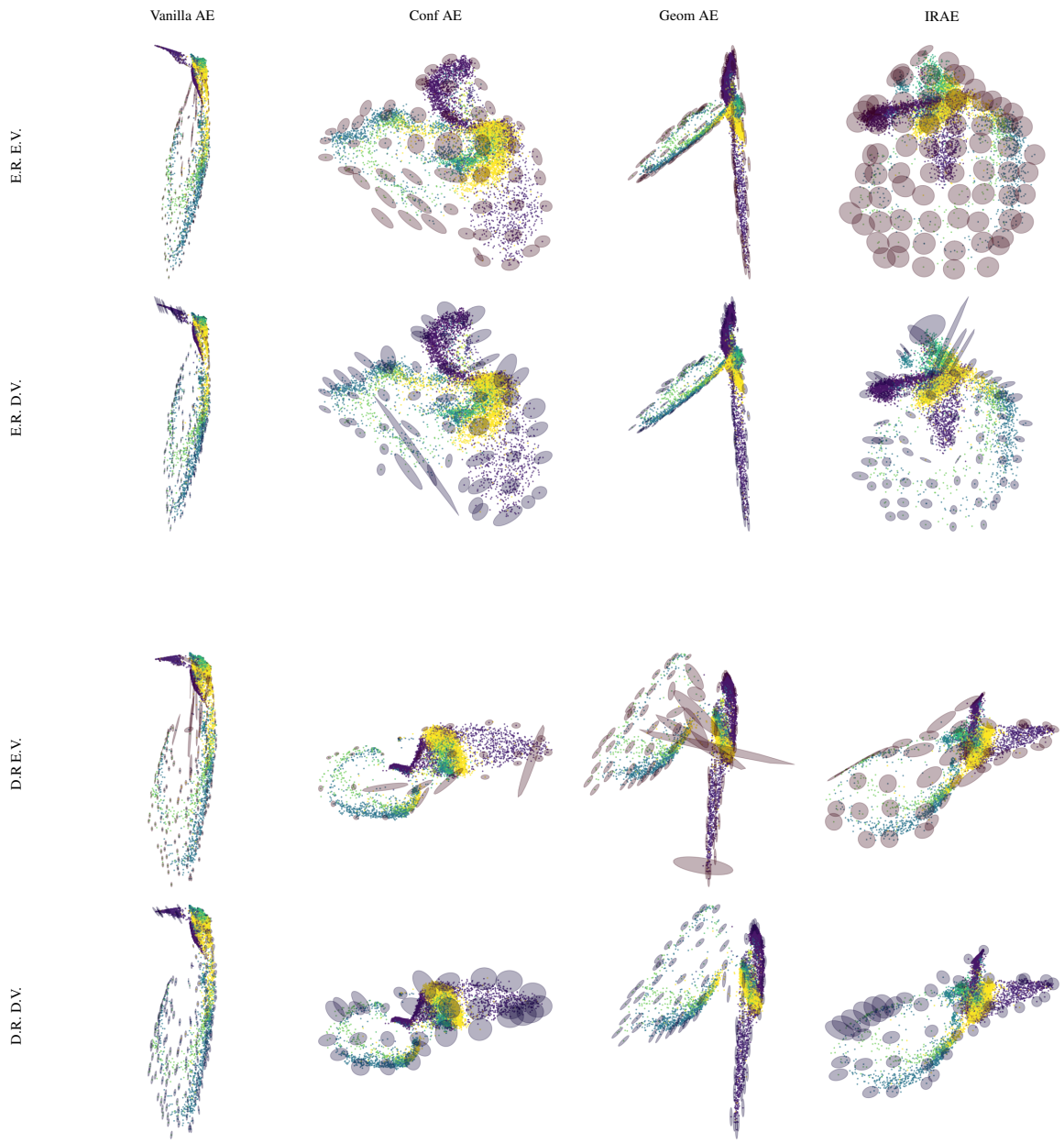


Figure 7. Indicatrices of geometrically regularized autoencoders trained on CElegans. The first block (first two rows) corresponds to encoder-regularized models, the second block (last two rows) corresponds to decoder-regularized models. Inside of each block, the top row shows encoder-based indicatrices (red) and the bottom row shows decoder-based indicatrices.

Table 5. Table underlying the metrics of Table 3, for encoder-regularized models. Averaged over three runs, bold+underlined indicates first, bold second place. The arrows point to the desirable direction of each metric.

		LOCAL			GLOBAL				
DATASET	MODEL	KL_{01}	KNN	TRUST	STRESS	KL_{100}	SPEAR	MSE (TEST)	MSE (TRAIN)
MNIST	IRAE	0.24 ± 0.03	0.66 ± 0.004	0.784 ± 0.007	**0.84 ± 0.02**	2.73E-07 ± 1.2E-08	0.331 ± 0.013	0.03568 ± 4.1E-05	**0.0999 ± 0.0005**
	GEOMAE	0.144 ± 0.032	**0.669 ± 0.007**	**0.798 ± 0.011**	**0.507 ± 0.011**	**2.6E-07 ± 5E-08**	0.4 ± 0.05	0.03359 ± 8E-05	0.10305 ± 0.00031
	CONFAE	**0.115 ± 0.021**	0.652 ± 0.012	0.79 ± 0.02	0.91 ± 0.04	3E-07 ± 2E-08	**0.397 ± 0.081**	**0.0334 ± 0.0002**	0.1022 ± 0.0007
	VANILLAAE	**0.12 ± 0.01**	**0.6665 ± 0.0051**	**0.807 ± 0.007**	0.882 ± 0.013	**2.6E-07 ± 7E-08**	**0.399 ± 0.032**	**0.0333 ± 0.0002**	**0.1021 ± 0.0002**
CELEGANS	IRAE	**0.067 ± 0.006**	**0.753 ± 0.012**	**0.87 ± 0.012**	14.7 ± 4.1	**1.6E-07 ± 3E-08**	**0.77 ± 0.011**	1.48 ± 0.02	**4.03 ± 0.02**
	GEOMAE	0.077 ± 0.006	**0.75 ± 0.01**	**0.87 ± 0.01**	7 ± 6	3.1E-07 ± 7E-08	**0.76 ± 0.01**	1.445 ± 0.013	4.06 ± 0.01
	CONFAE	0.07 ± 0.03	0.7 ± 0.02	0.82 ± 0.02	**0.75 ± 0.08**	**2.8E-07 ± 1.1E-07**	0.6 ± 0.2	**1.44 ± 0.03**	**4.04 ± 0.05**
	VANILLAAE	**0.07 ± 0.02**	0.7262 ± 0.0032	0.849 ± 0.005	**0.71 ± 0.04**	3.5E-07 ± 2E-07	0.72 ± 0.007	**1.4452 ± 0.0091**	4.05 ± 0.02
PBMC	IRAE	**0.04 ± 0.012**	**0.769 ± 0.008**	**0.923 ± 0.007**	15.4 ± 5	**1.7E-07 ± 2E-08**	0.879 ± 0.005	0.466 ± 0.002	**1.465 ± 0.003**
	GEOMAE	**0.04 ± 0.007**	0.76 ± 0.01	0.895 ± 0.022	15.8 ± 3.1	**2.3E-07 ± 1.3E-07**	**0.88 ± 0.02**	0.454 ± 0.0093	**1.475 ± 0.012**
	CONFAE	0.07 ± 0.01	0.759 ± 0.004	**0.9112 ± 0.0072**	**2.3 ± 1.3**	3E-07 ± 2E-07	0.82 ± 0.03	**0.4344 ± 0.0032**	1.4997 ± 0.00042
	VANILLAAE	0.07 ± 0.02	**0.7656 ± 0.0072**	0.909 ± 0.003	**1.61 ± 0.71**	2.79E-07 ± 8.2E-08	**0.887 ± 0.006**	**0.435 ± 0.003**	1.496 ± 0.006
ZILIONIS	IRAE	0.115 ± 0.006	**0.74 ± 0.02**	**0.888 ± 0.011**	27 ± 5	**2.1E-07 ± 3E-08**	**0.77 ± 0.04**	0.5 ± 0.01	**1.174 ± 0.01**
	GEOMAE	0.16 ± 0.03	**0.74 ± 0.009**	**0.878 ± 0.005**	9.1 ± 1.3	**2.1E-07 ± 3E-08**	0.731 ± 0.022	0.46 ± 0.01	**1.203 ± 0.011**
	CONFAE	**0.1146 ± 0.0081**	0.735 ± 0.012	0.86 ± 0.01	**0.53 ± 0.04**	4E-07 ± 2E-07	0.64 ± 0.07	**0.4417 ± 0.001**	1.2301 ± 0.0023
	VANILLAAE	**0.098 ± 0.031**	0.733 ± 0.01	0.88 ± 0.02	**0.54 ± 0.07**	2.7E-07 ± 1.3E-07	**0.733 ± 0.02**	**0.4444 ± 0.0043**	1.2262 ± 0.0032

Table 6. Table underlying the metrics of Table 3, for decoder-regularized models. Averaged over three runs, bold+underlined indicates first, bold second place. The arrows point to the desirable direction of each metric.

		LOCAL			GLOBAL				
DATASET	MODEL	KL_{01}	KNN	TRUST	STRESS	KL_{100}	SPEAR	MSE (TEST)	MSE (TRAIN)
MNIST	IRAE	0.148 ± 0.032	**0.678 ± 0.004**	**0.8036 ± 0.0011**	**0.68 ± 0.03**	**2.47E-07 ± 2.1E-08**	**0.42 ± 0.01**	0.03706 ± 0.00021	**0.09869 ± 0.00061**
	GEOMAE	**0.1193 ± 0.0042**	**0.6843 ± 0.0062**	**0.8095 ± 0.0073**	2.624 ± 0.071	**2.43E-07 ± 1.2E-08**	**0.46 ± 0.03**	0.03688 ± 0.00022	**0.0988 ± 0.0002**
	CONFAE	0.16 ± 0.03	0.648 ± 0.004	0.783 ± 0.004	0.88 ± 0.01	2.6E-07 ± 2E-08	0.316 ± 0.031	**0.0332 ± 0.0003**	0.1021 ± 0.0003
	VANILLAAE	**0.13 ± 0.03**	0.65235 ± 0.00051	0.796 ± 0.004	**0.878 ± 0.05**	3.2E-07 ± 8E-08	0.379 ± 0.041	**0.03367 ± 0.00032**	0.1019 ± 0.0011
CELEGANS	IRAE	**0.072 ± 0.01**	**0.76 ± 0.01**	**0.883 ± 0.007**	2.9 ± 1	**2.3E-07 ± 1.3E-07**	**0.74 ± 0.08**	1.565 ± 0.031	**3.89 ± 0.02**
	GEOMAE	**0.08 ± 0.02**	**0.72 ± 0.02**	**0.84 ± 0.02**	0.9 ± 0.2	**2.1E-07 ± 8E-08**	**0.673 ± 0.081**	1.56 ± 0.07	**3.91 ± 0.06**
	CONFAE	0.098 ± 0.008	0.71 ± 0.02	0.83 ± 0.01	**0.76 ± 0.07**	6.5E-07 ± 2.1E-07	0.65 ± 0.05	**1.442 ± 0.022**	4.05 ± 0.03
	VANILLAAE	0.1 ± 0.02	0.699 ± 0.002	0.82 ± 0.004	**0.85 ± 0.04**	4.23E-07 ± 5.2E-08	0.57 ± 0.11	**1.44 ± 0.02**	4.042 ± 0.008
PBMC	IRAE	0.06 ± 0.03	**0.793 ± 0.008**	**0.93 ± 0.006**	**1.4 ± 0.4**	**2.3E-07 ± 1E-07**	**0.92 ± 0.02**	0.445 ± 0.004	**1.482 ± 0.002**
	GEOMAE	**0.044 ± 0.021**	0.768 ± 0.004	**0.912 ± 0.008**	2.2 ± 1.2	**1.501E-07 ± 5.2E-09**	**0.8976 ± 0.0082**	**0.4323 ± 0.0013**	**1.4951 ± 0.0012**
	CONFAE	0.063 ± 0.006	0.75 ± 0.01	0.89 ± 0.02	1.6 ± 0.72	3E-07 ± 2E-07	0.866 ± 0.021	**0.433 ± 0.002**	1.4974 ± 0.0006
	VANILLAAE	**0.06 ± 0.02**	**0.77 ± 0.01**	0.91 ± 0.02	**1.4 ± 0.6**	3E-07 ± 1E-07	0.895 ± 0.007	0.4338 ± 0.002	1.498 ± 0.005
ZILIONIS	IRAE	0.15 ± 0.04	0.735 ± 0.012	0.88 ± 0.02	0.55 ± 0.03	**2E-07 ± 4E-08**	0.733 ± 0.031	0.443 ± 0.003	1.225 ± 0.005
	GEOMAE	**0.095 ± 0.021**	**0.7645 ± 0.0072**	**0.907 ± 0.002**	0.58 ± 0.06	2.23E-07 ± 3E-08	**0.796 ± 0.02**	**0.4407 ± 0.00081**	1.227 ± 0.003
	CONFAE	0.11 ± 0.02	0.739 ± 0.007	0.88 ± 0.01	**0.5 ± 0.05**	3.1E-07 ± 1.1E-07	0.71 ± 0.05	**0.442 ± 0.002**	**1.2222 ± 0.0071**
	VANILLAAE	**0.11 ± 0.02**	**0.752 ± 0.008**	**0.89 ± 0.01**	**0.52 ± 0.1**	**2.2E-07 ± 7.2E-08**	**0.75 ± 0.03**	0.443 ± 0.006	**1.225 ± 0.006**