

Geometrically Enriched Latent Spaces

Georgios Arvanitidis[†] Søren Hauberg[‡] Bernhard Schölkopf[†]

[†]MPI for Intelligent Systems, Tübingen, Germany

[‡]Technical University of Denmark, Lyngby, Denmark

Abstract

A common assumption in generative models is that the generator immerses the latent space into a Euclidean ambient space. Instead, we consider the ambient space to be a Riemannian manifold, which allows for encoding domain knowledge through the associated Riemannian metric. Shortest paths can then be defined accordingly in the latent space to both follow the learned manifold and respect the ambient geometry. Through careful design of the ambient metric we can ensure that shortest paths are well-behaved even for deterministic generators that otherwise would exhibit a misleading bias. Experimentally we show that our approach improves interpretability of learned representations both using stochastic and deterministic generators.

1 Introduction

Unsupervised representation learning has made tremendous gains with generative models such as *variational autoencoders (VAEs)* [Kingma and Welling, 2014, Rezende et al., 2014] and *generative adversarial networks (GANs)* [Goodfellow et al., 2014]. These, and similar, models provide a flexible and efficient parametrization of the density of observations in an ambient space \mathcal{X} through a typically lower dimensional latent space \mathcal{Z} .

While the latent space \mathcal{Z} constitutes a compressed representation of the data, it is by no means unique. Like most other latent variable models, these generative models are subject to *identifiability problems*, such that different representations can give rise to identical densities [Bishop, 2006]. This implies that straight lines in \mathcal{Z} are not shortest paths in any meaningful sense, and therefore do not constitute natural interpolants. To overcome this issue, it has been proposed to endow the latent space with a *Riemannian metric* such that curve lengths are measured in the ambient observation space \mathcal{X} [Tosi et al., 2014, Arvanitidis et al., 2018]. This approach immediately solves the identifiability problem. In other words, this ensures that any smooth invertible transformation of \mathcal{Z} does not change the distance between a pair of points, as long as the ambient path in \mathcal{X} remains the same.

While distances in \mathcal{X} are well-defined and give rise to an identifiable latent representation, they need not be particularly useful. We take inspiration from *metric learning* [Weinberger et al., 2006, Arvanitidis et al., 2016] and propose to equip the ambient observation space \mathcal{X} with a Riemannian metric and measure curve lengths in latent space accordingly. With this approach it is straight-forward to steer shortest paths in latent space to avoid low-density regions, but also to incorporate higher level information. For instance, Fig. 1 shows a shortest path interpolant under an ambient metric that favors images of *blond* people. Hence, we get both identifiable and useful latent representations.

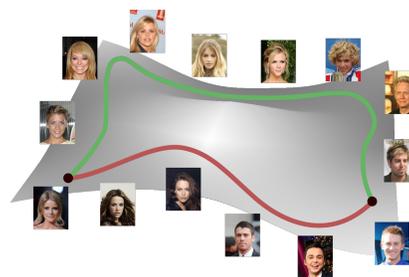


Figure 1: The proposed shortest path (—) favors the *blond* class, while the standard shortest path (—) merely minimizes the distance on the manifold.

2 A compact introduction to applied Riemannian geometry

We are interested in Riemannian manifolds [do Carmo, 1992], which constitute well-defined metric spaces, where the inner product is defined only locally and changes smoothly throughout space. In a nutshell, these are smooth spaces where we can compute shortest paths, which prefer to cross regions where the magnitude of the inner product is small. In this work, we show how to use such structures in machine learning, where is commonly assumed that data lie near a low dimensional manifold.

Definition 1. A Riemannian manifold is a smooth manifold \mathcal{M} , equipped with a positive definite Riemannian metric $\mathbf{M}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{M}$, which changes smoothly and defines a local inner product on the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ at each point $\mathbf{x} \in \mathcal{M}$ as $\langle \mathbf{v}, \mathbf{u} \rangle_{\mathbf{x}} = \langle \mathbf{v}, \mathbf{M}(\mathbf{x})\mathbf{u} \rangle$ with $\mathbf{v}, \mathbf{u} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$.

A smooth manifold is a topological space, which locally is homeomorphic to a Euclidean space. An intuitive way to think of a d -dimensional smooth manifold is as an embedded non-intersecting surface \mathcal{M} in an ambient space \mathcal{X} for example \mathbb{R}^D with $D > d$ (see Fig. 2 top). In this case, the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ is a d -dimensional vector space tangential to \mathcal{M} at the point $\mathbf{x} \in \mathcal{M}$. Hence, $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ is a vector $\mathbf{v} \in \mathbb{R}^D$ and actually the Riemannian metric is $\mathbf{M}_{\mathcal{X}} : \mathcal{M} \rightarrow \mathbb{R}_{>0}^{D \times D}$. The simplest approach is to assume that \mathcal{X} is equipped with the Euclidean metric $\mathbf{M}_{\mathcal{X}}(\mathbf{x}) = \mathbb{I}_D$ and its restriction is utilized as the Riemannian metric on $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Since the choice of $\mathbf{M}_{\mathcal{X}}(\cdot)$ has a direct impact on \mathcal{M} , we can utilize other metrics designed to encode high-level information (see Sec. 3).

Another view is to consider as smooth manifold the whole ambient space $\mathcal{X} = \mathbb{R}^D$. Hence, the $\mathcal{T}_{\mathbf{x}}\mathcal{X} = \mathbb{R}^D$ is centered at the point $\mathbf{x} \in \mathbb{R}^D$ and the simplest Riemannian metric is the Euclidean $\mathbf{M}_{\mathcal{X}}(\mathbf{x}) = \mathbb{I}_D$. However, we are able to use other suitable metrics that simply change the way we measure distances in \mathcal{X} (see Sec. 3). For instance, given a set of points in \mathcal{X} we can construct a metric with small magnitude near the data, such that to pull the shortest paths towards them (see Fig. 2 bottom).

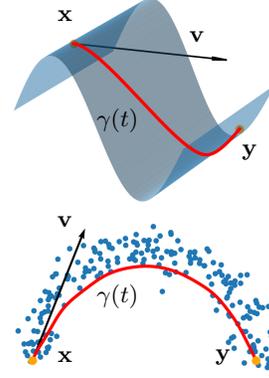


Figure 2: Examples of tangent vector (\rightarrow) and shortest path (\rightarrow) on embedded $\mathcal{M} \subset \mathcal{X}$ (top), ambient \mathcal{X} (bottom).

For a d -dimensional embedded manifold $\mathcal{M} \subset \mathcal{X}$, a collection of chart maps $\phi_i : \mathcal{U}_i \subset \mathcal{M} \rightarrow \mathbb{R}^d$ is used to assign local intrinsic coordinates to neighborhoods $\mathcal{U}_i \subset \mathcal{M}$, and for simplicity, we assume that a global chart map $\phi(\cdot)$ exists. By definition, when \mathcal{M} is smooth the $\phi(\cdot)$ and its inverse $\phi^{-1} : \phi(\mathcal{M}) \subset \mathbb{R}^d \rightarrow \mathcal{M} \subset \mathcal{X}$ exist and are smooth maps. Thus, a $\mathbf{v}_{\mathbf{x}} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ can be expressed as $\mathbf{v}_{\mathbf{x}} = \mathbf{J}_{\phi^{-1}}(\mathbf{z})\mathbf{v}_{\mathbf{z}}$, where $\mathbf{z} = \phi(\mathbf{x}) \in \mathbb{R}^d$ and $\mathbf{v}_{\mathbf{z}} \in \mathbb{R}^d$ are the representations in the intrinsic coordinates. Also, the Jacobian $\mathbf{J}_{\phi^{-1}}(\mathbf{z}) \in \mathbb{R}^{D \times d}$ defines a basis that spans the $\mathcal{T}_{\mathbf{x}}\mathcal{M}$, and thus, we can represent the metric $\mathbf{M}_{\mathcal{X}}(\cdot)$ in the intrinsic coordinates as

$$\langle \mathbf{v}_{\mathbf{x}}, \mathbf{v}_{\mathbf{x}} \rangle_{\mathbf{x}} = \langle \mathbf{v}_{\mathbf{z}}, \mathbf{J}_{\phi^{-1}}(\mathbf{z})^T \mathbf{M}_{\mathcal{X}}(\phi^{-1}(\mathbf{z})) \mathbf{J}_{\phi^{-1}}(\mathbf{z}) \mathbf{v}_{\mathbf{z}} \rangle = \langle \mathbf{v}_{\mathbf{z}}, \mathbf{M}(\mathbf{z}) \mathbf{v}_{\mathbf{z}} \rangle = \langle \mathbf{v}_{\mathbf{z}}, \mathbf{v}_{\mathbf{z}} \rangle_{\mathbf{z}}, \quad (1)$$

with smooth $\mathbf{M}(\mathbf{z}) = \mathbf{J}_{\phi^{-1}}(\mathbf{z})^T \mathbf{M}_{\mathcal{X}}(\phi^{-1}(\mathbf{z})) \mathbf{J}_{\phi^{-1}}(\mathbf{z}) \in \mathbb{R}_{>0}^{d \times d}$. As we discuss below, we should be able to evaluate the intrinsic $\mathbf{M}(\mathbf{z})$ in order to find length minimizing curves on \mathcal{M} . But, when \mathcal{M} is embedded the chart maps are usually unknown, as well as a global chart rarely exists. In contrast, for ambient like manifolds the global chart is $\phi(\mathbf{x}) = \mathbf{x}$, which is convenient to use in practice.

Generally, one of the main utilities of a Riemannian manifold $\mathcal{M} \subseteq \mathcal{X}$ is to enable us compute shortest paths therein. Intuitively, the norm $\sqrt{\langle d\mathbf{x}, d\mathbf{x} \rangle_{\mathbf{x}}}$ represents how the infinitesimal displacement vector $d\mathbf{x} \approx \mathbf{x}' - \mathbf{x}$ on \mathcal{M} is locally scaled. Thus, for a curve $\gamma : [0, 1] \rightarrow \mathcal{M}$ that connects two points $\mathbf{x} = \gamma(0)$ and $\mathbf{y} = \gamma(1)$, the length on \mathcal{M} or equivalently in $\phi(\mathcal{M})$ using Eq. 1, is measured as

$$\text{length}[\gamma(t)] = \int_0^1 \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)}} dt = \int_0^1 \sqrt{\langle \dot{c}(t), \mathbf{M}(c(t)) \dot{c}(t) \rangle} dt = \text{length}[c(t)] \quad (2)$$

where $\dot{\gamma}(t) = \partial_t \gamma(t) \in \mathcal{T}_{\gamma(t)}\mathcal{M}$ is the velocity of the curve and accordingly $\dot{c}(t) \in \mathcal{T}_{c(t)}\phi(\mathcal{M})$. The minimizers of this functional are the shortest paths, also known as geodesics. We find them by solving a system of 2nd order nonlinear ordinary differential equations (ODEs) defined in the intrinsic coordinates. Notably, for ambient like manifolds the trivial chart map enables us to compute the shortest paths in practice by solving the ODEs system. In some sense, the behavior of the shortest paths is to avoid high magnitude $\sqrt{|\mathbf{M}(c(t))|}$ regions. In general, the analytical solution is intractable, so we rely on approximate solutions [Hennig and Hauberg, 2014, Yang et al., 2018, Arvanitidis et al., 2019]. For further information on Riemannian geometry and the ODEs system see Appendix A.

2.1 Unifying the two manifold perspectives

In all related works, the ambient space \mathcal{X} is considered as a Euclidean space. Instead, we propose to consider \mathcal{X} as a Riemannian manifold. This allows us to encode high-level information through the associated metric, which constitutes an interpretable way to control the shortest paths. In order to find such a path on an embedded $\mathcal{M} \subset \mathcal{X}$, we have to solve the system of ODEs defined in the intrinsic coordinates. But, when \mathcal{M} is embedded the chart maps are mostly unknown. Hence, the usual trick is to utilize another manifold \mathcal{Z} having a trivial chart map and to represent the geometry of \mathcal{M} therein. Thus, we find the curve in \mathcal{Z} , which corresponds to the actual shortest path on \mathcal{M} .

In particular, assume an embedded d -dimensional manifold $\mathcal{M} \subset \mathcal{X}$ within a Riemannian manifold $\mathcal{X} = \mathbb{R}^D$ with metric $\mathbf{M}_{\mathcal{X}}(\cdot)$, a Euclidean space $\mathcal{Z} = \mathbb{R}^{d_{\mathcal{Z}}}$ called as *latent space* and a smooth function $g : \mathcal{Z} \rightarrow \mathcal{X}$ called as *generator*. Since $g(\cdot)$ is smooth, $\mathcal{M}_{\mathcal{Z}} = g(\mathcal{Z}) \subset \mathcal{X}$ is an immersed $d_{\mathcal{Z}}$ -dimensional smooth (sub)manifold¹. In general, we assume that $d_{\mathcal{Z}} = d$ and also that $g(\cdot)$ approximates closely the true embedded $\mathcal{M}_{\mathcal{Z}} \approx \mathcal{M}$, while if $d_{\mathcal{Z}} < d$ then $g(\cdot)$ can only approximate a submanifold on \mathcal{M} . Consequently, the corresponding Jacobian matrix $\mathbf{J}_g(\mathbf{z}) \in \mathbb{R}^{D \times d_{\mathcal{Z}}}$ is a basis that spans the $\mathcal{T}_{\mathbf{x}=g(\mathbf{z})}\mathcal{M}_{\mathcal{Z}}$, and maps a tangent vector $\mathbf{v}_{\mathbf{z}} \in \mathcal{T}_{\mathbf{z}}\mathcal{Z}$ to a tangent vector $\mathbf{v}_{\mathbf{x}} \in \mathcal{T}_{\mathbf{x}=g(\mathbf{z})}\mathcal{M}_{\mathcal{Z}}$. Thus, as before the restriction of $\mathbf{M}_{\mathcal{X}}(\cdot)$ on $\mathcal{T}_{\mathbf{x}}\mathcal{M}_{\mathcal{Z}}$ induces a metric in the latent space \mathcal{Z} as

$$\langle \mathbf{v}_{\mathbf{x}}, \mathbf{v}_{\mathbf{x}} \rangle_{\mathbf{x}} = \langle \mathbf{v}_{\mathbf{z}}, \mathbf{J}_g(\mathbf{z})^{\top} \mathbf{M}_{\mathcal{X}}(g(\mathbf{z})) \mathbf{J}_g(\mathbf{z}) \mathbf{v}_{\mathbf{z}} \rangle = \langle \mathbf{v}_{\mathbf{z}}, \mathbf{M}(\mathbf{z}) \mathbf{v}_{\mathbf{z}} \rangle. \quad (3)$$

This Riemannian metric $\mathbf{M}(\mathbf{z})$ is known as the *pull-back* metric, and essentially, captures the intrinsic geometry of the immersed $\mathcal{M}_{\mathcal{Z}}$, while taking into account the geometry of \mathcal{X} . The space \mathcal{Z} together with $\mathbf{M}(\mathbf{z})$ constitutes a Riemannian manifold, but since $\mathcal{Z} = \mathbb{R}^{d_{\mathcal{Z}}}$ the chart map and $\mathcal{T}_{\mathbf{z}}\mathcal{Z}$ are trivial. Therefore, we can evaluate the metric $\mathbf{M}(\mathbf{z})$ in intrinsic coordinates, which enables us to compute shortest paths on \mathcal{Z} by solving the ODEs system. Intuitively, these paths in \mathcal{Z} move optimally on $\mathcal{M}_{\mathcal{Z}}$, while simultaneously respecting the geometry of the ambient space \mathcal{X} . Also, note that $g(\cdot)$ is not a chart map, and hence, it is easier to learn. For further discussion see Appendix B.

3 Data learned Riemannian manifolds

We discuss some usages of differential geometry in machine learning, which largely inspire our work. Briefly, we present previous Riemannian metric learning methods and we propose a simple technique to construct such metrics in the ambient space \mathcal{X} . This is a principled and interpretable way to encode domain knowledge in our models. Also, we present the related work where the structure of an embedded data manifold is properly captured in the latent space of stochastic generators.

3.1 Learning Riemannian Metrics in the Ambient Space

Assume that a set of points $\{\mathbf{x}_n\}_{n=1}^N$ in $\mathcal{X} = \mathbb{R}^D$ is given. The Riemannian metric learning task is to learn a positive definite metric tensor $\mathbf{M}_{\mathcal{X}} : \mathcal{X} \rightarrow \mathbb{R}_{>0}^{D \times D}$ that changes smoothly across the space. The actual behavior of the metric depends on the problem we want to model. For example, when we want the shortest paths to stay on the data manifold (see Fig. 2 bottom) the meaningful behavior for the metric is that the magnitude $\sqrt{|\mathbf{M}_{\mathcal{X}}(\cdot)|}$ should be small near the data manifold and large as we move away. Similarly, in Fig. 1 the ambient metric is designed such that its magnitude is small near the data points with blond hair, and thus, the shortest paths tend to follow this semantic constraint.

One of the first approaches to learn such a Riemannian metric was presented by [Haugberg et al. \[2012\]](#), where $\mathbf{M}_{\mathcal{X}}(\mathbf{x})$ is the convex combination of a predefined set of metrics, using a smooth weighting function. In particular, at first K metrics are estimated $\{\mathbf{M}_k\}_{k=1}^K \in \mathbb{R}_{>0}^{D \times D}$ centered at the locations $\{\mathbf{c}_k\}_{k=1}^K \in \mathbb{R}^D$. Then, we can evaluate the metric at a new point as

$$\mathbf{M}_{\mathcal{X}}(\mathbf{x}) = \sum_{k=1}^K w_k(\mathbf{x}) \mathbf{M}_k, \quad \text{with } w_k(\mathbf{x}) = \frac{\tilde{w}_k(\mathbf{x})}{\sum_{j=1}^K \tilde{w}_j(\mathbf{x})} \quad \text{and } \tilde{w}_k(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_k\|_2^2}{2\sigma^2}\right), \quad (4)$$

where the kernel $\tilde{w}_k(\mathbf{x})$ with bandwidth $\sigma \in \mathbb{R}_{>0}$ is a smooth function, and thus, the metric is smooth as a linear combination of smooth functions. A practical example for the base metrics \mathbf{M}_k is the

¹A function $g : \mathcal{Z} \rightarrow \mathcal{M} \subset \mathcal{X}$ is an immersion if the $\mathbf{J}_g : \mathcal{T}_{\mathbf{z}}\mathcal{Z} \rightarrow \mathcal{T}_{g(\mathbf{z})}\mathcal{M}$ is injective (full rank) $\forall \mathbf{z} \in \mathcal{Z}$. Intuitively, \mathcal{M} is a $\dim(\mathcal{Z})$ -dimensional surface and intersections are allowed. While $g(\cdot)$ is an embedding if it is an injective function, which means that intersections are not allowed. An immersion locally is an embedding.

local Linear Discriminant Analysis (LDA), where local metrics are learned using labeled data such that to separate well the classes locally [Hastie and Tibshirani, 1994]. Also, a related approach is the Large Margin Nearest Neighbor (LMNN) classifier [Weinberger et al., 2006]. Note that the domain of metric learning provides a huge list of options that can be considered [Suárez et al., 2018].

Similarly, in an unsupervised setting Arvanitidis et al. [2016] proposed to construct the Riemannian metric in a non-parametric fashion as the the inverse of the local diagonal covariance. In particular, for a given point set $\{\mathbf{x}_n\}_{n=1}^N$ at a point \mathbf{x} the diagonal elements of the metric $M_{\mathcal{X}}(\cdot)$ are equal to

$$M_{\mathcal{X}_{dd}}(\mathbf{x}) = \left(\sum_{n=1}^N w_n(\mathbf{x})(x_{nd} - x_d)^2 + \varepsilon \right)^{-1} \quad \text{with} \quad w_n(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}\|_2^2}{2\sigma^2}\right), \quad (5)$$

where $\sigma \in \mathbb{R}_{>0}$ controls the curvature of the Riemannian manifold i.e., how fast the metric changes, and $\varepsilon > 0$ is a small scalar to upper bound the metric. Although these are quite flexible and intuitive metrics, selecting the parameter σ is a challenging task [Arvanitidis et al., 2017], especially due to the curse of dimensionality [Bishop, 2006] and the sample size N due to the non-parametric regime.

The proposed Riemannian metrics. Inspired by the approaches described above and Peyré et al. [2010], we propose a general and simple technique to easily construct metrics in \mathcal{X} , which allows to encode information depending on the problem. An unsupervised diagonal metric can be defined as

$$M_{\mathcal{X}}(\mathbf{x}) = (\alpha \cdot h(\mathbf{x}) + \varepsilon)^{-1} \cdot \mathbb{I}_D \quad (6)$$

where $h(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}_{>0}$ with behavior $h(\mathbf{x}) \rightarrow 1$ when \mathbf{x} is near the data manifold, otherwise $h(\mathbf{x}) \rightarrow 0$, and $\alpha, \varepsilon > 0$ are scaling factors to lower and upper bound the metric, respectively. One simple but very effective approach is to use a positive Radial Basis Function (RBF) network [Que and Belkin, 2016] as $h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ where $\mathbf{w} \in \mathbb{R}_{>0}^K$ and $\phi_k(\mathbf{x}) = \exp(-0.5 \cdot \lambda_k \cdot \|\mathbf{x} - \mathbf{x}_k\|_2^2)$ with bandwidth $\lambda_k > 0$. Similarly, $h(\mathbf{x})$ can be the probability density function of the given data. Usually, the true density function is unknown and difficult to learn, but we can approximate it roughly by utilizing a simple model as the Gaussian Mixture Model (GMM) [Bishop, 2006]. Such a Riemannian metric pulls the shortest paths towards areas of \mathcal{X} with high $h(\mathbf{x})$ (see Fig. 2 bottom).

In a similar context, a supervised version can be defined where the function $h(\mathbf{x})$ represents cost, while in Eq. 6 we do not use the inversion. In this way, shortest paths will tend to avoid regions of the ambient space \mathcal{X} where the cost function is high. For instance, in Fig. 1 we can think of a cost function that is high over all the non-blonde data points. Of course, such a cost function can be learned as an independent regression or classification problem, while in other cases can even be given by the problem or a domain expert. For further details about all the metrics above see Appendix C.

3.2 Learning Riemannian Metrics in the Latent Space

As discussed in Sec. 2.1, we can capture the geometry of the given embedded data manifold $\mathcal{M} \subset \mathcal{X}$ by learning a smooth generator $g : \mathcal{Z} \rightarrow \mathcal{M}_{\mathcal{Z}} \subset \mathcal{X}$ such that $\mathcal{M}_{\mathcal{Z}} \approx \mathcal{M}$. In previous works has been shown how to learn in practice such a function $g(\cdot)$, and also, the mild conditions it has to follow so that the induced Riemannian metric to capture properly the structure of \mathcal{M} . In the latent space $\mathcal{Z} = \mathbb{R}^d$ we call as latent codes or representations the points $\{\mathbf{z}_n \in \mathcal{Z} \mid \mathbf{x}_n = g(\mathbf{z}_n), n = 1, \dots, N\}$.

First Tosi et al. [2014] considered the Gaussian Process Latent Variable Model (GP-LVM) [Lawrence, 2005], where $g(\cdot)$ is a stochastic function defined as a multi-output Gaussian process $g \sim \mathcal{GP}(\mathbf{m}(\mathbf{z}), k(\mathbf{z}, \mathbf{z}'))$. Since the generator is stochastic, it induces a random Riemannian metric in \mathcal{Z} , and in practice, the expected metric is used for the computation of shortest paths. The advantage of such a stochastic generator is that the metric magnitude increases analogous to the uncertainty of $g(\cdot)$, which happens in regions of \mathcal{Z} where there are no latent codes. Apart from this desired behavior, this metric is not very practical due to the GPs computational cost.

Another set of approaches known as deep generative models, parameterizes $g(\cdot)$ as a deep neural network (DNN). On the one hand are the explicit density models, where the marginal likelihood can be computed, with main representatives the Variational Auto-Encoder (VAE) [Kingma and Welling, 2014, Rezende et al., 2014] and the normalizing flow models [Dinh et al., 2016, Rezende and Mohamed, 2015]. On the other hand are the implicit density models for which the marginal likelihood is intractable, as is the Generative Adversarial Networks (GAN) [Goodfellow et al., 2014].

Recently, Arvanitidis et al. [2018] showed that we are able to properly capture the structure of the data manifold $\mathcal{M} \subset \mathcal{X}$ in the latent space \mathcal{Z} of a VAE under the condition of having meaningful uncertainty quantification for the generative process. In particular, the standard VAE assumes a Gaussian likelihood $p(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\mathbf{x} | \mu(\mathbf{z}), \mathbb{I}_D \cdot \sigma^2(\mathbf{z}))$ with a prior $p(\mathbf{z})$. Hence, the generator can be written as $g(\mathbf{z}) = \mu(\mathbf{z}) + \text{diag}(\varepsilon) \cdot \sigma(\mathbf{z})$ where $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_D)$ and $\mu : \mathcal{Z} \rightarrow \mathcal{X}$, $\sigma : \mathcal{Z} \rightarrow \mathbb{R}_{>0}^D$ are usually parametrized with DNNs. However, parametrizing $\sigma(\cdot)$ with a DNN does not directly imply meaningful uncertainty quantification, because it extrapolates arbitrarily to regions of \mathcal{Z} with no latent codes. Thus, the proposed solution in Arvanitidis et al. [2018] is to model the inverse variance $\beta(\mathbf{z}) = (\sigma^2(\mathbf{z}))^{-1}$ with a positive Radial Basis Function (RBF) network [Que and Belkin, 2016], which implies that moving further from the latent codes increases the uncertainty. Under this stochastic generator, the expected Riemannian metric in \mathcal{Z} is equal to

$$\mathbf{M}(\mathbf{z}) = \mathbb{E}_{p(\varepsilon)}[\mathbf{J}_{g_\varepsilon}(\mathbf{z})^\top \mathbf{J}_{g_\varepsilon}(\mathbf{z})] = \mathbf{J}_\mu(\mathbf{z})^\top \mathbf{J}_\mu(\mathbf{z}) + \mathbf{J}_\sigma(\mathbf{z})^\top \mathbf{J}_\sigma(\mathbf{z}), \quad (7)$$

where $g_\varepsilon(\cdot)$ implies that ε is kept fixed $\forall \mathbf{z} \in \mathcal{Z}$, such that to ensure a smooth mapping. Here, we observe that the metric increases when the generator becomes uncertain due to the second term. This constitutes a desired behavior, as the metric informs us to avoid regions of \mathcal{Z} where there are no latent codes, which directly implies that these regions do not correspond to parts of the data manifold in \mathcal{X} . In some sense, we can think of modeling the topology of \mathcal{M} too [Hauberg, 2018].

Clearly, the deterministic generators like the Auto-Encoder (AE) and the GAN, capture poorly the structure of \mathcal{M} in \mathcal{Z} since the second term in Eq. 7 does not exist. The reason is that these models are trained based on the likelihood $p(\mathbf{x} | \mathbf{z}) = \delta(\mathbf{x} - g(\mathbf{z}))$, and hence, the uncertainty is not quantified. Of course, for the AE one potential *heuristic* solution is to use the latent codes of the training data to fit *post-hoc* a meaningful variance estimator under the Gaussian likelihood and the maximum likelihood principle as $\theta^* = \text{argmax}_\theta \prod_{n=1}^N \mathcal{N}(\mathbf{x}_n | g(e(\mathbf{x}_n)), \mathbb{I}_D \cdot \sigma_\theta^2(e(\mathbf{x}_n)))$, with encoder $e : \mathcal{X} \rightarrow \mathcal{Z}$. In principle, we could follow the same procedure for the GAN by learning an encoder [Donahue et al., 2016, Dumoulin et al., 2016]. However, it is still unclear if the encoder for a GAN learns meaningful representations or if the powerful generator ignores the inferred latent codes [Arora et al., 2018].

Therefore, in order to properly capture the structure of \mathcal{M} in \mathcal{Z} we mainly rely on stochastic generators with increasing uncertainty as we move further from the latent codes. Even if the RBF based approach is a meaningful way to get the desired behavior, in general, uncertainty quantification with parametric models is still considered as an open problem [MacKay, 1992, Gal and Ghahramani, 2015, Lakshminarayanan et al., 2017, Detlefsen et al., 2019, Arvanitidis et al., 2018]. Nevertheless, Eklund and Hauberg [2019] showed that the expected Riemannian metric in Eq. 7 is a reasonable approximation to use in practice. Obviously, when $g(\cdot)$ is deterministic, like the GAN, the second term in Eq. 7 disappears, since these models do not quantify the uncertainty of the generative process. This directly means that deterministic generators are not able, by construction, to properly capture the geometric structure of \mathcal{M} in the latent space, and hence, exhibit a misleading bias [Hauberg, 2018].

4 Enriching the Latent Space with Geometric Information

Here, we unify the approaches presented in Sec. 3.1 and Sec. 3.2, in order to provide extra structure in the latent space of a generative model. This is the first time that these two fundamentally different Riemannian views are combined. Their difference is that the metric induced by $g(\cdot)$ merely tries to capture the intrinsic geometry of the given data manifold \mathcal{M} , while $\mathbf{M}_\mathcal{X}(\cdot)$ allows to directly encode high-level information in \mathcal{X} based on domain knowledge. Moreover, we provide in the stochastic case a relaxation for efficient computation of the expected metric. While in the deterministic case we combine a carefully designed ambient metric with a new architecture for $g(\cdot)$ to extrapolate meaningfully, which is one way to ensure well-behaved shortest paths that respect the structure of \mathcal{M} .

Stochastic generators. Assuming that an ambient $\mathbf{M}_\mathcal{X}(\cdot)$ is given (see Sec. 3.1), we learn a VAE with Gaussian likelihood, so the stochastic mapping is $g(\mathbf{z}) = \mu(\mathbf{z}) + \text{diag}(\varepsilon) \cdot \sigma(\mathbf{z})$, while using a positive RBF for meaningful estimation of the uncertainty $\sigma(\cdot)$. As before, we assume that ε is constant for each $\mathcal{M}_\varepsilon = g_\varepsilon(\mathcal{Z})$ to ensure smoothness. Therefore, we can apply Eq. 3 to derive the new stochastic more informative pull-back metric in \mathcal{Z} , which is equal to

$$\mathbf{M}_\varepsilon(\mathbf{z}) = [\mathbf{J}_\mu(\mathbf{z}) + \mathbf{J}_\sigma(\mathbf{z})\varepsilon]^\top \mathbf{M}_\mathcal{X}(\mu(\mathbf{z}) + \text{diag}(\varepsilon) \cdot \sigma(\mathbf{z})) [\mathbf{J}_\mu(\mathbf{z}) + \mathbf{J}_\sigma(\mathbf{z})\varepsilon]. \quad (8)$$

Since this is a random metric, in principle, we can compute the expectation simply by sampling $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_D)$, as $\mathbf{M}(\mathbf{z}) = \mathbb{E}_{\varepsilon \sim p(\varepsilon)}[\mathbf{M}_\varepsilon(\mathbf{z})]$. However, in practice this expectation will increase

dramatically the cost, especially, since we need to evaluate the expected metric many times when computing a shortest path. Hence, we consider only the $\mathbb{E}_{\varepsilon \sim p(\varepsilon)}[\mathcal{M}_\varepsilon] = \mu(\mathbf{z})$ for the evaluation of the ambient metric $\mathbf{M}_{\mathcal{X}}(\cdot)$ in Eq. 8, which simplifies the expected Riemannian metric to

$$\mathbf{M}(\mathbf{z}) \triangleq \mathbf{J}_\mu(\mathbf{z})^\top \mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z})) \mathbf{J}_\mu(\mathbf{z}) + \mathbf{J}_\sigma(\mathbf{z})^\top \mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z})) \mathbf{J}_\sigma(\mathbf{z}). \quad (9)$$

Essentially, the realistic underlying assumption is that near the latent codes $\sigma(\mathbf{z}) \rightarrow \mathbf{0}$ so $g(\mathbf{z}) \rightarrow \mu(\mathbf{z})$ and the first term dominates. But, as we move in regions of \mathcal{Z} with no codes the $\sigma(\mathbf{z}) \gg 0$, and for this reason, we need the second term in the equation. In particular, $\mathbf{J}_\sigma(\cdot)$ dominates when moving further from the latent codes, and hence, the behavior of $\mathbf{M}_{\mathcal{X}}(\cdot)$ will be less important there. Thus, we are allowed to consider this relaxation, for which the meaningful uncertainty estimation is still necessary. We further analyze and check empirically this relaxation in Appendix D.

Deterministic generators. As regards the deterministic generators, we propose a simple solution that ensures well-behaved shortest paths which respect the structure of the given data manifold \mathcal{M} . The idea is to learn a Riemannian metric $\mathbf{M}_{\mathcal{X}}(\cdot)$ in \mathcal{X} that only roughly represents the structure of \mathcal{M} , for instance, by using an RBF or a GMM (Eq. 6). Essentially, this ambient metric informs us how close the generated $\mathcal{M}_{\mathcal{Z}}$ is to the given data. Hence, we additionally need the $g(\cdot)$ to extrapolate meaningfully. That means $g(\cdot)$ should learn to generate well the given data from a prior $p(\mathbf{z})$, but as we move further from the support of $p(\mathbf{z})$, the generated $\mathcal{M}_{\mathcal{Z}}$ should also move further from the given data in \mathcal{X} . Consequently, since $\mathbf{M}_{\mathcal{X}}(\cdot)$ is designed to increase far from the given data, the induced Riemannian metric in \mathcal{Z} properly captures the structure of the data manifold.

One of the simplest deterministic generators with this desirable behavior is the probabilistic Principal Component Analysis (pPCA) [Tipping and Bishop, 1999]. This is a very basic model with a Gaussian prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$ and a generator that is simply a linear map. The generator is constructed by the top d eigenvectors of the empirical covariance matrix, scaled by their eigenvalues. Inspired by this simple model, we propose for the deterministic generator the following architecture

$$g(\mathbf{z}) = f(\mathbf{z}) + \mathbf{U} \cdot \text{diag}([\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}]) \cdot \mathbf{z} + \mathbf{b}, \quad (10)$$

where $f : \mathcal{Z} \rightarrow \mathcal{X}$ is a deep neural network, $\mathbf{U} \in \mathbb{R}^{D \times d}$ the top d eigenvectors with their corresponding eigenvalues λ_d computed from the empirical data covariance, and $\mathbf{b} \in \mathbb{R}^D$ is the data mean. This interpretable model can be seen as a residual network (ResNet) [He et al., 2015]. In particular, the desired behavior is that as we move further from the $p(\mathbf{z})$ the linear part of Eq. 10 becomes the dominant one, especially, when bounded activation functions are utilized for $f(\cdot)$. Hence, the $\mathcal{M}_{\mathcal{Z}}$ will extrapolate meaningfully as we move further from the support of the prior. However, we need again the generated $\mathcal{M}_{\mathcal{Z}}$ to be a valid immersion. For further discussion see Appendix B.

5 Experiments

5.1 Demonstrations with Deterministic Generators

Synthetic experiment. Usually, in GANs the $g(\cdot)$ is a continuous function, so we expect some generated points to fall off the given data support. Mainly, when the data lie near a disconnected \mathcal{M} and irrespective of training optimality. This is known as the distribution mismatch problem. We generate a synthetic dataset (Fig. 4) and we train a Wasserstein GAN [Arjovsky et al., 2017] with a latent space $\mathcal{Z} = \mathbb{R}^2$ and $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbb{I}_2)$. For the ambient metric we used a positive RBF (Eq. 6). For implementation details see Appendix E.

Then, we define a density function in \mathcal{Z} using the learned Riemannian metric as $q(\mathbf{z}) \propto \tilde{p}_r(\mathbf{z}) \cdot (1 + \sqrt{|\mathbf{M}(\mathbf{z})|})^{-1}$, where $\tilde{p}_r(\mathbf{z})$ is a uniform density within a ball of radius r [Lebanon, 2002, Le and Cuturi, 2015]. The behavior of $q(\mathbf{z})$ is interpretable, since the density is high wherever $\mathbf{M}(\cdot)$ is small and that happens only in the regions of \mathcal{Z} that $g(\cdot)$ learns to map near the given data manifold in \mathcal{X} . Also, the $\tilde{p}(\mathbf{z})$ simply ensures that the support of $q(\mathbf{z})$ is within the region where the $g(\cdot)$ is trained. Thus, we compare samples generated from $q(\mathbf{z})$ using Markov Chain Monte Carlo (MCMC) and the prior $p(\mathbf{z})$.

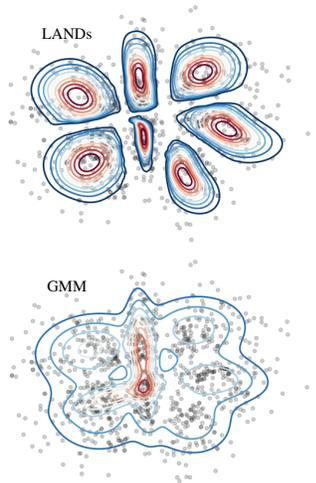


Figure 3

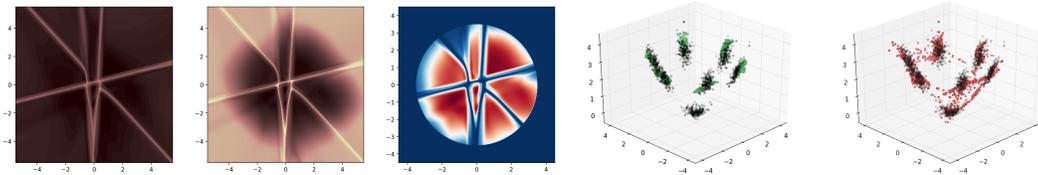


Figure 4: From *left to right*: The measure $\sqrt{|\mathbf{M}(\mathbf{z})|}$ without and with the ambient Riemannian metric, the approximate aggregated posterior $q(\mathbf{z})$, generated samples by the $q(\mathbf{z})$ (●) and the prior $p(\mathbf{z})$ (●). Comparing the first two panels, we see that the proposed model properly captures the structure of \mathcal{M} .

We see in Fig. 4 that our samples align better with the data manifold, so we can think of $q(\mathbf{z})$ as an approximate aggregated posterior without using an encoder. In a similar spirit, Tanielian et al. [2020] proposed to reject samples from $p(\mathbf{z})$ based only on the norm of generator’s Jacobian. Moreover, using the sampled latent points we fit a mixture of LANDs (Fig. 3), which are locally adaptive normal distributions on Riemannian manifolds [Arvanitidis et al., 2016]. Hence, we can sample from each component individually, without training a conditional GAN [Mirza and Osindero, 2014].

MNIST data. Similarly, we performed an experiment with the MNIST digits 0,1,2 and $\mathcal{Z} = \mathbb{R}^5$ and we show the results in Fig. 5. Since this is a high dimensional dataset, the RBF used for the $\mathbf{M}_{\mathcal{X}}(\cdot)$ will be a poor fit, and also, the Euclidean distance between images is not meaningful. Therefore, after the Wasserstein GAN training, we used PCA to project linearly the data in a d' -dimensional subspace $D > d' > d$ with $d' = 10$, where we define the ambient $\mathbf{M}_{\mathcal{X}'}(\cdot)$. This step removes the non informative dimensions from the data, while keeping the global structure of the data manifold unchanged. So the intrinsic geometry is approximately preserved. We discuss this linear projection step in Appendix B, and provide further implementation details and results in Appendix E.

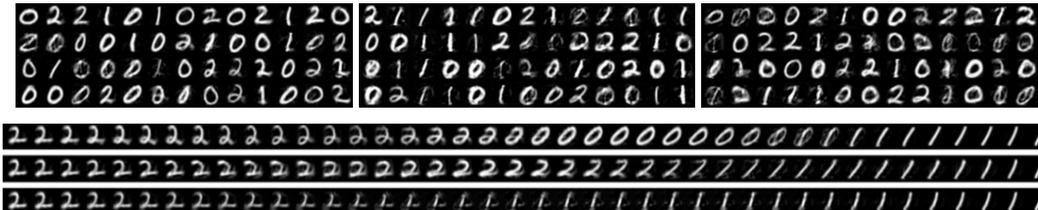


Figure 5: *Top panels, left to right*: samples from $q(\mathbf{z})$, from $q(\mathbf{z})$ without the $\mathbf{M}_{\mathcal{X}'}(\cdot)$ and from $p(\mathbf{z})$. *Bottom panels, top to bottom row*: interpolations under the proposed Riemannian metric, without the $\mathbf{M}_{\mathcal{X}'}(\cdot)$ and the linear interpolation. We see that $\mathbf{M}_{\mathcal{X}'}(\cdot)$ improves the sampling, and also, our path (top row) respects the manifold structure avoiding off-the-manifold “shortcuts” (middle row).

Pre-trained generator. Finally, we use as $g(\cdot)$ a pre-trained Progressive GAN on the CelebA dataset [Karras et al., 2018], and we also train a classifier that distinguishes the blond people. As before, we use a linear projection to $d' = 1000$, where we define a cost based ambient metric $\mathbf{M}_{\mathcal{X}'}(\cdot)$ based on a positive RBF (Eq. 6). This metric is designed to penalize regions in \mathcal{X} that correspond to blonds and the goal is to avoid these regions when interpolating in \mathcal{Z} . As we discussed in Sec. 3, it is not guaranteed that this deterministic generator properly captures the structure of the data manifold in \mathcal{Z} , but even so, we test our ability to control the shortest paths.

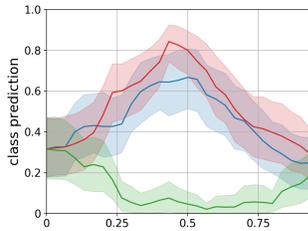


Figure 6

As we observe in Fig. 7 only our path that utilizes the informative $\mathbf{M}_{\mathcal{X}'}(\cdot)$ successfully avoids crossing regions with blond hair. In particular, it corresponds to the optimal path on the generated manifold, while taking into account the high-level semantic information. In contrast, the shortest path without the $\mathbf{M}_{\mathcal{X}'}(\cdot)$ passes through the high cost region in \mathcal{X} , as it merely minimizes the distance on the manifold and does not utilize the additional information. Also, we show in Fig. 6 the classifier prediction along several interpolants. Clearly, we see that only our shortest paths (—) respect the ambient geometry, while both the straight line (—) and the naive shortest path (—), interpolate through regions classified as blond. For further details and interpolation results see Appendix E.



Figure 7: *Top to bottom*: interpolations under the proposed Riemannian metric, without the $M_{\mathcal{X}'}(\cdot)$ and linear interpolation. Only our path successfully avoids the high cost regions (blond hair). This high-level semantic information is encoded into the ambient metric which only our path utilizes.

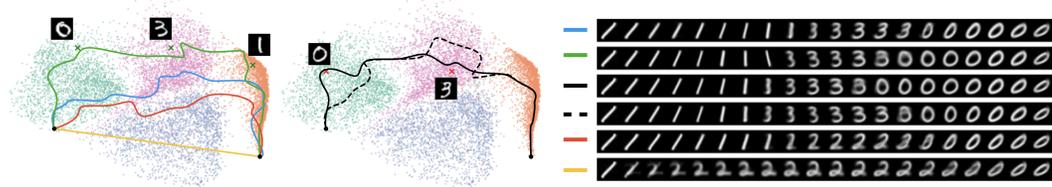


Figure 8: Comparing interpolants under different ambient metrics and their generated images. We can control effectively the shortest paths to follow high-level information incorporated in $M_{\mathcal{X}'}(\cdot)$.

5.2 Demonstrations with Stochastic Generators

Controlling Shortest Paths. We compare in Fig. 8 the effect of several interpretable ambient metrics on the shortest paths in the latent space of a VAE trained with $\mathcal{Z} = \mathbb{R}^2$ on the MNIST digits 0,1,2,3. As before, we project linearly the data in $d' = 10$ to construct there the ambient metrics. At first, we observe that under the Euclidean metric in \mathcal{X} the path (—) merely follows the structure of the generated $\mathcal{M}_{\mathcal{Z}}$ since it avoids regions with no data. Then, we construct an LDA metric in \mathcal{X}' (Eq. 4) by considering the digits 0,1,3 to be in the same class. Hence, the resulting path (—) avoids crossing the regions in \mathcal{Z} that correspond to digit 2, while simultaneously respects the geometry of $\mathcal{M}_{\mathcal{Z}}$. Also, we select 3 data points (\times) and using their 100 nearest neighbors in \mathcal{X}' we construct the local covariance based metric (Eq. 5), such that the path (—) to move closer to the selected points.

Moreover, we linearly combine these two metrics, such that to enforce the path (—) to pass through 0,1,3 while moving closer to the selected points (\times). Finally, we include to this linear combination a cost related metric (Eq. 6) based on a positive RBF that increases near the points (\times) in \mathcal{X}' . Therefore, the resulting path (- -) avoids these neighborhoods, while respecting the other ambient metrics and the geometry of $\mathcal{M}_{\mathcal{Z}}$. Hence, we can effectively control the shortest paths by designing and combining the ambient metrics accordingly.

Kernel Density estimation. Finally, we show in Fig. 9 the kernel density estimation in \mathcal{Z} comparing the straight line to the shortest path. For $M_{\mathcal{X}'}(\cdot)$ we linearly combine an LDA metric where each digit is a separate class and a cost based metric that increases near the points (\times) in \mathcal{X}' . We see that $M(\cdot)$ helps to distinguish better the classes due to the LDA metric, while the density is reduced near the regions with high cost. For further discussion see Appendix E.

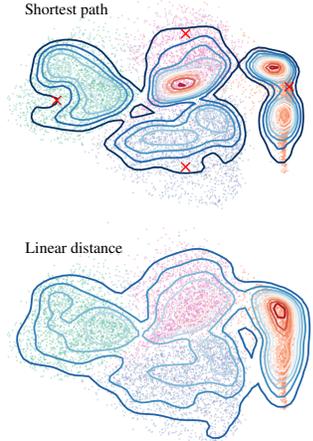


Figure 9

6 Conclusion

We considered the ambient space of generative models as a Riemannian manifold. This allows for encoding domain knowledge through the metric and we proposed an easy way to construct suitable metrics. In order to capture the geometry into the latent space, proper uncertainty estimation is essential in stochastic generators, while in the deterministic case one way is through the proposed meaningful extrapolation. Thus, we get interpretable shortest paths that respect the ambient space geometry, while moving optimally on the learned manifold. In the future, it will be interesting to see if and how we can use the ambient space geometry during the learning phase of the generative model.

Broader Impact

We have proposed a framework that allows for solving the identifiability problem associated with latent variable models, while retaining flexibility with regards to the metric behavior of the latent space. This is valuable wherever a *faithful representation* is of use, such as to ensure a *fair* and *interpretable* model. The approach also carries potential value for *causal inference* where the identifiability issue is a paramount concern.

The model does carry a risk of inappropriate usage, as an end-user (most likely a data scientist) can easily manipulate empirical findings by changing the ambient metric. Misleading results can, thus, be presented by a manipulation of an ambient metric, which may not be transparent to recipients of the data analysis. Conceptually, this is the “same old” issue that occurs when empirical findings are overly sensitive to data pre-processing.

Acknowledgments

SH was supported by a research grant (15334) from VILLUM FONDEN. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement n° 757360).

References

- M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
- S. Arora, A. Risteski, and Y. Zhang. Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations (ICLR)*, 2018.
- G. Arvanitidis, L. K. Hansen, and S. Hauberg. A locally adaptive normal distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.
- G. Arvanitidis, L. K. Hansen, and S. Hauberg. Maximum likelihood estimation of riemannian metrics from euclidean data. In *Geometric Science of Information (GSI)*, 2017.
- G. Arvanitidis, L. K. Hansen, and S. Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations (ICLR)*, 2018.
- G. Arvanitidis, S. Hauberg, P. Hennig, and M. Schober. Fast and robust shortest paths on manifolds learned from data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.
- C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.
- N. Chen, F. Ferroni, A. Klushyn, A. Paraschos, J. Bayer, and P. van der Smagt. Fast approximate geodesics for deep generative models. *Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning*, 2019.
- N. S. Detlefsen, M. Jørgensen, and S. Hauberg. Reliable training and estimation of variance networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.
- M. do Carmo. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992.
- J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning, 2016.
- V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference, 2016.
- D. Eklund and S. Hauberg. Expected path length on random manifolds. In *arXiv preprint*, 2019.
- Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2015.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

- T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification, 1994.
- S. Hauberg. Only bayes should learn a manifold. 2018.
- S. Hauberg, O. Freifeld, and M. Black. A Geometric Take on Metric Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.
- P. Hennig and S. Hauberg. Probabilistic Solutions to Differential Equations and their Application to Riemannian Statistics. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.
- T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.
- D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.
- B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 2005.
- T. Le and M. Cuturi. Unsupervised riemannian metric learning for histograms using aitchison transformations. In *International Conference on International Conference on Machine Learning (ICML)*, 2015.
- G. Lebanon. Learning riemannian metrics. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.
- D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 1992.
- M. Mirza and S. Osindero. Conditional generative adversarial nets. 2014.
- G. Peyré, M. Péchaud, R. Keriven, and L. D. Cohen. Geodesic methods in computer vision and graphics. *Found. Trends. Comput. Graph. Vis.*, 2010.
- Q. Que and M. Belkin. Back to the future: Radial basis function networks revisited. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Cadiz, Spain, 2016.
- D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 2015.
- D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.
- J. L. Suárez, S. García, and F. Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms and experiments, 2018.
- U. Tanielian, T. Issenhuth, E. Dohmatob, and J. Mary. Learning disconnected manifolds: a no gans land. In *International Conference on Machine Learning (ICML)*, 2020.
- J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 2000.
- M. E. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 1999.
- A. Tosi, S. Hauberg, A. Vellido, and N. D. Lawrence. Metrics for Probabilistic Geometries. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, July 2014.
- K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2006.
- T. Yang, G. Arvanitidis, D. Fu, X. Li, and S. Hauberg. Geodesic clustering in deep generative models. In *arXiv preprint*, 2018.

Appendix

A Further information on Riemannian geometry

Let us assume a d -dimensional smooth manifold \mathcal{M} embedded in an ambient space $\mathcal{X} = \mathbb{R}^D$ with $d < D$, where it is defined a Riemannian metric $\mathbf{M}_{\mathcal{X}} : \mathcal{X} \rightarrow \mathbb{R}_{>0}^{D \times D}$. Therefore, the space \mathcal{X} is a Riemannian manifold, since \mathcal{X} is a smooth manifold. This directly implies that the simple Euclidean space is a Riemannian manifold as well. Due to the embedding of \mathcal{M} a Riemannian metric is induced in the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ by the restriction of the Riemannian ambient metric $\mathbf{M}_{\mathcal{X}}(\cdot)$, even for the simple case $\mathbf{M}_{\mathcal{X}}(\mathbf{x}) = \mathbb{I}_D$. For simplicity, we further assume that a global chart map $\phi(\mathcal{M})$ exists.

Generally, one of the main utilities of a Riemannian manifold $\mathcal{M} \subseteq \mathcal{X}$ is to enable us compute shortest paths therein. Intuitively, the norm $\sqrt{\langle d\mathbf{x}, d\mathbf{x} \rangle_{\mathbf{x}}}$ represents how the infinitesimal displacement vector $d\mathbf{x} \approx \mathbf{x}' - \mathbf{x}$ on \mathcal{M} is locally scaled. Thus, for a curve $\gamma : [0, 1] \rightarrow \mathcal{M}$ that connects two points $\mathbf{x} = \gamma(0)$ and $\mathbf{y} = \gamma(1)$, the length on \mathcal{M} or equivalently in $\phi(\mathcal{M})$ using Eq. 2 is measured as

$$\text{length}[\gamma(t)] = \int_0^1 \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)}} dt = \int_0^1 \sqrt{\langle \dot{c}(t), \mathbf{M}(c(t))\dot{c}(t) \rangle} dt = \text{length}[c(t)] \quad (11)$$

where $\dot{\gamma}(t) = \partial_t \gamma(t) \in \mathcal{T}_{\gamma(t)}\mathcal{M}$ is the velocity of the curve and accordingly $\dot{c}(t) \in \mathcal{T}_{c(t)}\phi(\mathcal{M})$. The length is an invariant quantity under reparametrization i.e., for any continuous monotonic function $s : [a, b] \rightarrow [0, 1]$ the curve $\tilde{\gamma}(t') = \gamma(s(t'))$, $t' \in [a, b]$ has the same length. Instead, in order to find the *shortest path* we minimize the corresponding energy functional, which is a non-invariant quantity,

$$\gamma^*(t) = \underset{\gamma(t)}{\operatorname{argmin}} E[\gamma(t)] = \underset{\gamma(t)}{\operatorname{argmin}} \frac{1}{2} \int_0^1 \langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)} dt, \quad (12)$$

and similarly the energy can be written for the $c(t) \in \phi(\mathcal{M})$ in the intrinsic coordinates as Eq. 11. The minimizers of this energy have constant speed $\|\dot{\gamma}(t)\|_{\gamma(t)}$ and are known as *geodesics*.

In theory, instead of solving the problem directly on \mathcal{M} , we utilize the intrinsic coordinates. Assuming the global chart $\phi(\cdot)$, we search for a curve $c(t) = \phi(\gamma(t)) \in \phi(\mathcal{M})$ that minimizes the corresponding energy functional $E[c(t)] = \frac{1}{2} \int_0^1 \langle \dot{c}(t), \mathbf{M}(c(t))\dot{c}(t) \rangle dt$. Here, we used the fact that $\gamma(t) = \phi^{-1}(c(t)) \Rightarrow \dot{\gamma}(t) = \mathbf{J}_{\phi^{-1}}(c(t))\dot{c}(t)$, since by the definition of a smooth manifold $\phi(\cdot)$, $\phi^{-1}(\cdot)$ exist and are smooth maps. Now, we can find the minimizers by directly applying the Euler-Lagrange equations to the energy $E[c(t)]$, which results to a system of 2^{nd} order non-linear ordinary differential equations (ODEs) written as in [Arvanitidis et al. \[2018\]](#)

$$\ddot{c}(t) = -\frac{1}{2} \mathbf{M}^{-1}(c(t)) \left[2(\dot{c}(t)^\top \otimes \mathbb{I}_d) \frac{\partial \operatorname{vec}[\mathbf{M}(c(t))]}{\partial c(t)} \dot{c}(t) - \frac{\partial \operatorname{vec}[\mathbf{M}(c(t))]}{\partial c(t)}^\top (\dot{c}(t) \otimes \dot{c}(t)) \right], \quad (13)$$

where $\operatorname{vec}[\cdot]$ stacks the columns of a matrix into a vector and \otimes is the Kronecker product. This is solved as a boundary value problem (BVP) with boundary conditions $c(0) = \mathbf{x}$ and $c(1) = \mathbf{y}$. Note that this ODEs system is a standard result in differential geometry, and intuitively, the resulting shortest paths tend to avoid areas where the metric magnitude $\sqrt{|\mathbf{M}(c(t))|}$ is high.

In order to perform computations on a Riemannian manifold we need to define two operations analogous to the “plus” and “minus” of the Euclidean space. First, the *exponential map* is an operator $\operatorname{Exp}_{\mathbf{x}}(\mathbf{v}t) = \gamma(t)$ that takes two inputs, a point $\mathbf{x} \in \mathcal{M}$ and a $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$, and generates a geodesic with $\gamma(1) = \mathbf{y} \in \mathcal{M}$ and initial velocity $\dot{\gamma}(0) = \mathbf{v}$. The inverse operator is called the *logarithmic map* $\operatorname{Log}_{\mathbf{x}}(\mathbf{y}) = \mathbf{v}$ that takes two inputs $\mathbf{x}, \mathbf{y} \in \mathcal{M}$ to return the tangent vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$. Note that these two operators are dual in a small neighborhood around $\mathbf{x} \in \mathcal{M}$. Moreover, the logarithmic map provides *coordinates* for the points in a neighborhood on \mathcal{M} with respect to the base point \mathbf{x} , but only the distances between the center \mathbf{x} and the points are meaningful and not the ones between the points. Also, by definition the $\langle \operatorname{Log}_{\mathbf{x}}(\mathbf{y}), \mathbf{M}_{\mathcal{X}}(\mathbf{x})\operatorname{Log}_{\mathbf{x}}(\mathbf{y}) \rangle = \operatorname{dist}^2(\mathbf{x}, \mathbf{y}) = \operatorname{length}^2[\gamma(t)]$, but we can rescale the logarithmic map such that the $\operatorname{dist}^2(\mathbf{x}, \mathbf{y}) = \langle \overline{\operatorname{Log}}_{\mathbf{x}}(\mathbf{y}), \overline{\operatorname{Log}}_{\mathbf{x}}(\mathbf{y}) \rangle$. The rescaled coordinates $\overline{\operatorname{Log}}_{\mathbf{x}}(\mathbf{y})$ are known as *normal coordinates* and are the ones that we use in practice.

B Theoretical analysis of the generator

In this section we analyze the properties that the proposed generator $g(\cdot)$ should have. In particular, in order to have a theoretically sound model the generator has to be at least twice differentiable, and additionally, an immersion. Also, we need a specific behavior such that to properly capture the structure of the data manifold in the latent space, both in the stochastic and deterministic generator case. Of course, the basic assumption is that the data lie near an *embedded* smooth manifold \mathcal{M} in the ambient space \mathcal{X} . Intuitively, an embedded d -dimensional manifold can be considered as a surface that is everywhere homeomorphic to a d -dimensional Euclidean space, which implies that contractions and intersections are now allowed. In contrast, an *immersion* is a relative simpler condition, since intersections are allowed but again no contractions. In theory, the generator has to be at least an immersion such that to pull-back the Riemannian metric of the manifold.

Stochastic generator. We consider as generator the function $g(\mathbf{z}) = \mu(\mathbf{z}) + \text{diag}(\varepsilon) \cdot \sigma(\mathbf{z})$, where $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_D)$ and $\mu : \mathcal{Z} \rightarrow \mathcal{X}$ is a DNN and $\sigma : \mathcal{Z} \rightarrow \mathbb{R}_{>0}^{\dim(\mathcal{X})}$ is based on a positive RBF. Note that in principle we model the precision $\beta(\mathbf{z}) = (\sigma^2(\mathbf{z}))^{-1}$ with the positive RBF, so the $\sigma(\mathbf{z}) = \beta^{-1/2}(\mathbf{z})$. From the theory we know that $g(\cdot)$ has to be smooth. At first, we can achieve smoothness easily for $\mu(\cdot)$ and $\sigma(\cdot)$. In particular, $\sigma(\cdot)$ is smooth as a linear combination of smooth functions. For the DNN $\mu(\cdot)$ we can use smooth activation functions as the $\tanh(\cdot)$, $\text{softplus}(\cdot)$, etc. But the stochasticity of ε makes $g(\cdot)$ non-smooth, and hence, non differentiable with respect to \mathbf{z} . Instead, if ε is fixed $\forall \mathbf{z} \in \mathcal{Z}$ denoted as $g_\varepsilon(\cdot)$, then this is a smooth nonlinear map, and consequently, differentiable. A different perspective on the smoothness of $g(\cdot)$ has been given by Eklund and Hauberg [2019]. There it is shown that $g_\varepsilon(\cdot)$ is actually the random projection of the deterministic smooth nonlinear map $\mathbf{z} \mapsto [\mu(\mathbf{z}), \sigma(\mathbf{z})]$ under the random projection matrix $\mathbf{P}_\varepsilon = [\mathbb{I}_D, \text{diag}(\varepsilon)]$. In both views, fixing ε implies that the *sampled* $\mathcal{M}_\varepsilon = g_\varepsilon(\mathcal{Z})$ is a smooth immersed manifold in \mathcal{X} . Obviously, the $\mathbb{E}_\varepsilon[\mathcal{M}_\varepsilon] = \mathbb{E}_\varepsilon[g_\varepsilon(\mathcal{Z})] = \mathbb{E}_\varepsilon[g(\mathcal{Z})] = \mu(\mathcal{Z})$, which shows that the expected manifold, as well as the likelihood of the individual points $p(\mathbf{x}|\mathbf{z})$ do not change.

The $g_\varepsilon(\cdot)$ is an immersion if $\mathbf{J}_{g_\varepsilon}(\mathbf{z}) = \mathbf{J}_\mu(\mathbf{z}) + \mathbf{J}_\sigma(\mathbf{z})\varepsilon$ has full rank $\forall \mathbf{z} \in \mathcal{Z}$. For $\mu(\cdot)$ (DNN) this can be true within the support of $p(\mathbf{z})$ where the activation functions usually do not reach their limit behavior. For instance, with $\tanh(\cdot)$ as activation, we expect within the support of $p(\mathbf{z})$ the hidden units output to not be constant ± 1 . In addition, we need each hidden layer to have greater or equal number of units to the previous layer while all the weight matrices are full rank. While for $\sigma(\cdot)$ (inverse positive RBF) at least $\dim(\mathcal{Z})$ basis functions has to be active and the weight matrix has to be full rank. The conventions above define an immersed manifold $\mathcal{M}_\varepsilon = g_\varepsilon(\mathcal{Z})$ in \mathcal{X} , since we avoid *contractions*. Of course, the two matrices $\mathbf{J}_\mu(\mathbf{z}), \mathbf{J}_\sigma(\mathbf{z})$ should not cancel any of their columns.

Generator with linear extrapolation. We analyze the behavior of the proposed architecture $g(\mathbf{z}) = f(\mathbf{z}) + \mathbf{U} \cdot \text{diag}([\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}]) \cdot \mathbf{z} + \mathbf{b}$ where $f : \mathcal{Z} \rightarrow \mathcal{X}$ is a nonlinear map and $\mathcal{Z} = \mathbb{R}^d, \mathcal{X} = \mathbb{R}^D$ with $D \gg d$. Note that for the stochastic generator case and for fixed ε the function $f_\varepsilon(\mathbf{z}) = \mu(\mathbf{z}) + \text{diag}(\varepsilon) \cdot \sigma(\mathbf{z})$ can be simply seen as the addition of two nonlinear functions (see above). The linear map is constructed using the top d -eigenvectors scaled by their eigenvalues, coming from the eigen-decomposition of the data empirical covariance matrix. More specifically, the empirical data covariance $\mathbf{C} = \frac{1}{N-1} \sum_{n=1}^N (\mathbf{x}_n - \mathbf{b})(\mathbf{x}_n - \mathbf{b})^\top$, where $\mathbf{b} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$, which can be decomposed as $\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$. We use for \mathbf{U} the first d columns of $\mathbf{V} \in \mathbb{R}^{D \times D}$ and the corresponding eigenvalues. In particular, we check if and when $g(\cdot)$ satisfies the properties:

Smoothness. We need the generator to be sufficiently smooth, which means in our case at least twice differentiable. This condition can be easily satisfied by selecting the activation functions accordingly as $\tanh(\cdot)$, $\text{softplus}(\cdot)$, etc. In practice, this is necessary since in the geodesic ODEs system we need to compute the derivative of the metric tensor, which in our case is implemented by first taking the derivative of the Jacobian $\mathbf{J}_g(\cdot)$. Obviously, by including in $g(\cdot)$ the linear map $\mathbf{A} = \mathbf{U} \cdot \text{diag}([\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}])$ and \mathbf{b} , the smoothness property will not change.

Immersion. In theory a mapping $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$ with $D \gg d$ is an immersion if the corresponding $\mathbf{J}_g(\mathbf{z}) \in \mathbb{R}^{D \times d}$ is everywhere injective or in other words full rank. In our case, the Jacobian includes a neural network and for an example we consider the simple function $f(\mathbf{z}) = \mathbf{W}_1 \cdot \psi(\mathbf{W}_0\mathbf{z} + \mathbf{b}_0) + \mathbf{b}_1$ with $\psi(\cdot)$ the activation function, and thus, the Jacobian is $\mathbf{J}_f(\mathbf{z}) = \mathbf{W}_1 \cdot \psi'(\mathbf{W}_0\mathbf{z} + \mathbf{b}_0) \odot \mathbf{W}_0$. In order to be this quantity an immersion, first we need each hidden layer to have more or equal number of hidden units from the previous layer and the weight matrices to have full rank. Additionally, since the derivative of the activation functions $\psi'(\cdot)$ appears, we need this to be non-zero. Otherwise, this will directly affect the total rank of the Jacobian, because it will reduce the rank of the corresponding weight matrix. We *conjecture* that for generative models which are trained using a compact support prior $p(\mathbf{z})$ like the Gaussian, the trained model uses the activation functions $\psi(\cdot)$ closer to the center of their domain, where their corresponding derivative $\psi'(\mathbf{z})$ is not zero, and not towards the domain limits. This basically implies that the corresponding hidden unit is active and is used by the model.

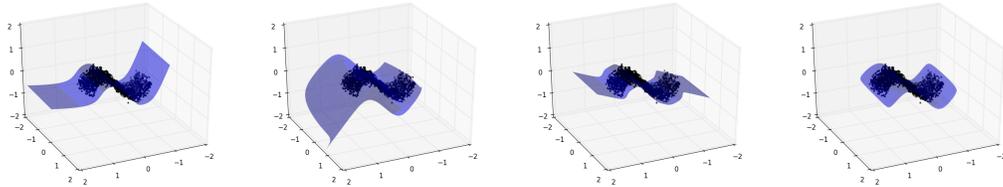


Figure 11: We trained an MLP with 2 hidden layers from $g : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, with hidden layer sizes 2, 3. From *left to right*: The extrapolation result with activation function $\text{softplus}(\cdot)$ and including the proposed linear map, without the linear map, with activation function $\tanh(\cdot)$ and including the proposed linear map, without the linear map. We see that the linear map improves the extrapolation of $g(\cdot)$, and the change is faster along the direction with the largest eigenvalue.

However, the Jacobian in our case is $\mathbf{J}_g(\mathbf{z}) = \mathbf{J}_f(\mathbf{z}) + \mathbf{A}$, which means that in theory there are cases where the two matrices could cancel some of their columns. This will directly break the full rank condition, and thus, the mapping at this point will not be an immersion. Practically, this means that the corresponding Riemannian metric tensor in \mathcal{Z} , computed as the $\mathbf{J}_g(\mathbf{z})^\top \mathbf{J}_g(\mathbf{z})$, will degenerate since it will not have full rank. However, even if in theory this is a case that could happen, in practice, we *conjecture* that this is a relatively unrealistic scenario. Instead, if the $\mathbf{J}_f(\mathbf{z})$ has low rank the linear part \mathbf{A} could even fix the problem, of course, if any of the rest columns do not cancel each other.

Extrapolation. The proposed meaningful extrapolation for a deterministic generator is one way to properly capture in \mathcal{Z} the structure or topology of the data manifold, and thus, the geometry of the ambient space. Especially, this is necessary in the case where the behavior of the ambient metric is small only close to the data, which pulls the shortest paths towards the data manifold. Similarly, in the stochastic generator case meaningful uncertainty quantification is utilized in order to properly capture in \mathcal{Z} the structure of the data manifold or in some sense its topology [Hauberg, 2018].

Thus, let us consider the deterministic generator case where $g(\mathbf{z})$ is simply a neural network $f(\mathbf{z})$ and let us pick a direction \mathbf{z} so that we move on the line $t\mathbf{z}$ for $t \in \mathbb{R}$. When the $\tanh(\cdot)$ activation function is used, as we move further from the support of the prior, the units of the first hidden layer will tend to output always a constant value $\rightarrow +1$ or $\rightarrow -1$. This means that the extrapolation will not be meaningful since it is gonna be always a constant. Similarly, for the $\text{softplus}(\cdot)$ as we move to the boundaries of the domain of t , the output of the activation will be either a constant $\rightarrow 0$ or a linear function. However, for each output dimension $f_j(t\mathbf{z})$ if the $t \rightarrow +\infty$ corresponds to a linear extrapolation the $t \rightarrow -\infty$ will extrapolate to zero. Therefore, in the $\text{softplus}(\cdot)$ case, even if the generator will potentially extrapolate meaningfully in some parts, in general, the behavior is arbitrary and hard to interpret $\forall \mathbf{z} \in \mathcal{Z}$. We show the behavior on a synthetic example in Fig. 11.

So including the linear map \mathbf{A} , \mathbf{b} could potentially fix the extrapolation issue, since the map $g(\cdot)$ after some threshold t becomes solely linear. However, as regards the immersion condition, when $t \rightarrow \pm\infty$ if all the dimension $f_j(t\mathbf{z})$ cancel out the corresponding rows of $\mathbf{A}t\mathbf{z} + \mathbf{b}$, then the $g(t\mathbf{z})$ output will be a constant value. However, we argue again that this is quite unrealistic to happen on the same time for all the output D dimensions.

Above, we only describe the theoretical conditions and the properties that a generator has to respect. Nevertheless, proper guarantees and analysis should be provided in the future.

Linear projection of the ambient space. Additionally, we discuss the case where we linearly project the data manifold in $\mathcal{X}' = \mathbb{R}^{d'}$, a lower dimensional space $D > d' > d$, and we learn the ambient metric $\mathbf{M}_{\mathcal{X}'}(\cdot)$ therein. Intuitively, instead of finding the shortest path $\gamma(t)$ on the $\mathcal{M} \subset \mathcal{X}$ we find the path on the projected manifold in \mathcal{X}' and we expect that the actual structure of \mathcal{M} is preserved. The reason of this step is to remove the non very informative extra dimensions from the data e.g. high frequency context, which do not provide any significant information regarding the structure of that data manifold or simply if they just correspond to noise. In other words, this step helps us to reduce the dimensionality, such that to construct the “ambient” metric using the projected data. Of course, this is only acceptable if the linear projection does not change the structure of the data manifold, for instance by introducing self intersections or contractions. Note that still the generator is trained between the space \mathcal{Z} and \mathcal{X} , prior to the linear projection, so $g(\cdot)$ is still able to capture the high frequency context of the given data.

The practical reason for this step is that for high dimensional data e.g. images, due the curse of dimensionality [Bishop, 2006], we need to reduce the dimension of \mathcal{X} , especially when the learned ambient metric is based on pairwise Euclidean distances. Also, we know that, even locally, Euclidean distance makes not too much sense for images. Hence, the linear projection to a lower dimensional space \mathcal{X}' helps us to ensure that at least locally straight lines will be more meaningful.

Therefore, the linear projection of the data, helps us to learn easier an ‘‘ambient’’ Riemannian metric that provides information regarding the structure of the actual data manifold. However, we note again that it is necessary this step to not change the structure of the data manifold. Thus, the Riemannian metric $\mathbf{M}_{\mathcal{X}'}(\cdot)$ that is learned from the projected data is defined in $\mathbb{R}^{d'}$, and hence, the pull-back in the latent space takes the following form

$$\langle \mathbf{v}', \mathbf{M}_{\mathcal{X}'}(\mathbf{x}')\mathbf{v}' \rangle = \langle \mathbf{P}\bar{\mathbf{v}}, \mathbf{M}_{\mathcal{X}'}(\mathbf{P}(\bar{\mathbf{x}} - \bar{\mathbf{c}}))\mathbf{P}\bar{\mathbf{v}} \rangle = \langle \mathbf{v}, \mathbf{J}_g(\mathbf{z})^\top \mathbf{P}^\top \mathbf{M}_{\mathcal{X}'}(\mathbf{P}(g(\mathbf{z}) - \bar{\mathbf{c}}))\mathbf{P}\mathbf{J}_g(\mathbf{z})\mathbf{v} \rangle, \quad (14)$$

where $\mathbf{v}', \mathbf{x}' \in \mathbb{R}^{d'}$ the point and the tangent vector in \mathcal{X}' , $\mathbf{P} \in \mathbb{R}^{d' \times D}$ is the projection matrix derived from PCA with $\bar{\mathbf{c}} \in \mathbb{R}^D$ the center of the data, $\bar{\mathbf{x}}, \bar{\mathbf{v}} \in \mathbb{R}^D$ the point and the tangent vector in \mathcal{X} and $\mathbf{z}, \mathbf{v} \in \mathbb{R}^d$ the latent space inputs with the Jacobian $\mathbf{J}_g(\mathbf{z}) \in \mathbb{R}^{D \times d}$. Note that we can directly use the same setting when $g(\cdot)$ is a stochastic generator.

A simple constructive example is to consider the data in Fig. 11, and expand the dimensions by concatenating 100 columns with noise sampled from $\varepsilon_i \sim \mathcal{N}(0, 0.001^2)$ as $[\mathbf{x}, \varepsilon_1, \dots, \varepsilon_{100}]$. Obviously, the structure of the actual data manifold will not be different in $\mathcal{X} = \mathbb{R}^{103}$, and also, we can ‘‘project’’ it in $\mathcal{X}' = \mathbb{R}^3$ by excluding the last 100 columns. Therefore, we can construct the ‘‘ambient’’ metric $\mathbf{M}_{\mathcal{X}'}(\cdot)$ in \mathcal{X}' , which will be induced on the 3-dimensional subspace in \mathcal{X} where the actual data manifold lies. Thus, the shortest path computed in \mathcal{X}' actually corresponds to the path on \mathcal{M} in \mathcal{X} that lies on a 3-dimensional subspace. As regards the real data, the extra dimensions might not be just noise, but high frequency context, which commonly does not affect the underlying structure of the manifold. Hence, the shortest paths computed in \mathcal{X}' are able to approximate closely the true paths on the actual data manifold $\mathcal{M} \subset \mathcal{X}$, as long as the linear projection step does not change the structure of \mathcal{M} in \mathcal{X}' .

C Details for the construction of ambient Riemannian metrics

In this section we provide the details for constructing the metrics that have been used in the paper. As we discussed above, the ambient metrics can be either constructed in \mathcal{X} or in lower dimensional space \mathcal{X}' where we project linearly the given data manifold.

C.1 Local linear discriminant analysis based Riemannian metric

To compute the ambient metric for a test point $\mathbf{x} \in \mathcal{X} = \mathbb{R}^D$ using the local LDA we have first to learn the base metrics for a set of points $\mathcal{S} = \{\mathbf{x}_s\}_{s=1}^S$ following the approach of Hastie and Tibshirani [1994], and then, compute the weighted average (see Sec 3.1, Eq. 4). Based on a given labeled set $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ the metric at each \mathbf{x}_s is defined as

$$\mathbf{M}_s = \mathbf{W}_s^{-1} \mathbf{B}_s \mathbf{W}_s^{-1} + \varepsilon \mathbf{W}_s^{-1}, \quad (15)$$

where $\varepsilon > 0$ a small scalar to avoid degenerate metrics, the $\mathbf{W}_s \in \mathbb{R}^{D \times D}$ is called the within covariance matrix and $\mathbf{B}_s \in \mathbb{R}^{D \times D}$ the in-between covariance matrix. Let K be the number of the k -nearest neighbors denoted with the set $\text{knn}(\mathbf{x}_s)$ computed under the initial $\mathbf{M}_s = \mathbb{I}_D$ and $d_s = \left\| \mathbf{M}_s^{1/2}(\mathbf{x} - \mathbf{x}_s) \right\|_2$. We use a weighting function $w_s(\mathbf{x}) = [1 - (d_s/\sigma_s)^3]^3 \cdot \mathbb{1}_{\{d_s < \sigma_s\}}$ where $\sigma_s = \max_{k \in \text{knn}(\mathbf{x}_s)} d_k$. Then, from the labeled point set we consider only the ones that are within the $\text{knn}(\mathbf{x}_s)$, and thus, the matrices within and in-between are defined as

$$\mathbf{W}_s = \frac{1}{\sum_{k=1}^K w_s(\mathbf{x}_k)} \sum_{c \in \mathcal{C}} \sum_{n: y_n=c} w_s(\mathbf{x}_n) (\mathbf{x}_n - \mathbf{m}_c) (\mathbf{x}_n - \mathbf{m}_c)^\top, \quad (16)$$

$$\mathbf{m}_c = \frac{1}{\sum_{k: y_k=c} w_s(\mathbf{x}_k)} \sum_{n: y_n=c} w_s(\mathbf{x}_n) \mathbf{x}_n, \quad (17)$$

$$\mathbf{B} = \sum_{c \in \mathcal{C}} \pi_c (\mathbf{m}_c - \mathbf{m}) (\mathbf{m}_c - \mathbf{m})^\top, \quad (18)$$

$$\pi_c = \frac{\sum_{n: y_n=c} w_s(\mathbf{x}_n)}{\sum_{k=1}^K w_s(\mathbf{x}_k)}. \quad (19)$$

Using the updated metrics \mathbf{M}_s we iterate the procedure i.e. finding the $\text{knn}(\mathbf{x}_s)$, computing d_s , etc, until either a fixed point is found i.e., the \mathbf{M}_s matrices do not change, or if we exceed a pre-specified number of iterations. Moreover, we use only the diagonal \mathbf{W}_s since in higher dimensions is easier to get degenerate metrics when this matrix is full.

As we have discussed in the main paper the construction of the base metrics \mathbf{M}_k is problem dependent. Hence, these can be constructed in any meaningful way, such that to provide the essential high-level information or domain knowledge for the problem we want to model. For further examples, we could construct these metrics based on ordinal information between points or triplet constraints. In general, this is a metric learning related problem Suárez et al. [2018].

C.2 Density and data support based Riemannian metric

In order to construct a probability density function based ambient metric, essentially, we want to roughly estimate the density of the high dimensional data. A relatively simple, easy and robust model to learn such a density is the Gaussian Mixture Model (GMM). So in practice, we want to learn a $h(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, with $\sum_{k=1}^K \pi_k = 1$. However, we have to pay attention to some details. First we want to avoid centers $\boldsymbol{\mu}_k$ with huge covariance $\boldsymbol{\Sigma}_k$ that are placed outside of the data distribution. For that reason we chose to use the same covariance matrix for all the data $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$. Intuitively, we want this covariance to be roughly a spherical one, in order to cover the whole data manifold with balls or ellipsoids. So we chose $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_D^2)$.

A second problem is that in higher dimensions the $|\boldsymbol{\Sigma}|$ can be tricky. In particular, the normalization constant will be an issue, since if many $\sigma_d < 1$ the $|\boldsymbol{\Sigma}| \rightarrow 0$. For that reason we use the un-normalized Gaussian mixture model and this is not a problem, because all of the components share the same covariance. Of course, we are still able to set the parameters α, ε such that to lower and upper bound the metric. In some sense, these parameters define one aspect of the manifold's curvature, since they define how big is the difference of the metric between the points where $h(\mathbf{x}) \rightarrow 0$ and $h(\mathbf{x}) \rightarrow 1$.

One drawback of this method, is that the metric will shrink the distances accordingly to the data density in the ambient space is high. Obviously, in some cases this might be a meaningful behavior. However, we might want to simply move near the data and not necessarily analogous to the corresponding density. So a close related approach is to utilize a positive function $h(\mathbf{x}) = \sum_k w_k \phi_k(\mathbf{x})$, with $w_k > 0$ and $\phi_k(\mathbf{x}) = \exp(-0.5 \cdot \lambda_k \|\mathbf{x} - \mathbf{c}_k\|_2^2)$, that is trained in such a way that the output near the given data is $h(\mathbf{x}) \rightarrow 1$, otherwise $h(\mathbf{x}) \rightarrow 0$. One way to train the parameters is to fix \mathbf{c}_k using k -means, setting the bandwidth $\lambda_k = \frac{1}{2} \left[\kappa \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x} \in \mathcal{C}_k} \|\mathbf{x} - \mathbf{c}_k\|_2 \right]^{-2}$ where $\kappa > 0$ a scaling factor, \mathcal{C}_k the points in the cluster of \mathbf{c}_k , and the w_k can be found using a closed form solution or gradient descent under the mean squared error $L(\mathbf{w}) = \sum_{n=1}^N \|1 - h(\mathbf{x}_n)\|_2^2$. Obviously, this is a relatively simple model, however, it models very well the desired behavior of the ambient metric.

C.3 Cost based Riemannian metric

The cost related ambient Riemannian metric essentially pulls the shortest paths towards regions of the ambient space \mathcal{X} where the cost is low. For our experiments we used a relatively simple and interpretable cost function utilizing again the RBF network $h(x) = \sum_k y_k \phi_k(\mathbf{x})$ with basis functions $\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{c}_k\|_2^2\right)$ and $y_k > 0$ some given values. Apart from the simplicity, this type of cost function has a very interpretable behavior, since it defines regions in \mathcal{X} where the cost is high and the corresponding regions in \mathcal{Z} will be avoided by the shortest paths. Intuitively, these can be neighborhoods of points in \mathcal{X} that we want to avoid as we move on the data manifold.

C.4 Can we construct the $M_{\mathcal{X}}(\cdot)$ in \mathcal{Z} ?

A logical question is, why we do not construct the informative metric directly in \mathcal{Z} using the latent codes, and simply, combine it linearly with the pull-back metric that is induced by the generator? The answer is quite straight forward though. The metric $M_{\mathcal{X}}(\cdot)$ is mainly based on Euclidean distances. Therefore, the definition of this Riemannian metric in the latent space \mathcal{Z} is impossible, since using the Euclidean distance in \mathcal{Z} is fundamentally wrong and misleading.

D Approximation of the expected Riemannian metric in the latent space

Here we discuss the approximation to the true expected Riemannian metric, where we evaluate the ambient metric only on the expected generated $\mathcal{M}_{\mathcal{Z}} = \mu(\mathcal{Z})$. In particular, the true stochastic Riemannian metric in the latent space is written as

$$\mathbf{M}_{\varepsilon}(\mathbf{z}) = [\mathbf{J}_{\mu}(\mathbf{z}) + \mathbf{J}_{\sigma}(\mathbf{z})\varepsilon]^{\top} \mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z}) + \text{diag}(\varepsilon) \cdot \sigma(\mathbf{z})) [\mathbf{J}_{\mu}(\mathbf{z}) + \mathbf{J}_{\sigma}(\mathbf{z})\varepsilon], \quad (20)$$

for which we can approximate the expectation in the latent space as $\mathbf{M}(\mathbf{z}) = \mathbb{E}_{\varepsilon \sim p(\varepsilon)}[\mathbf{M}_{\varepsilon}(\mathbf{z})]$ with $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_D)$. Even if this is a doable computation, in practice, we need to estimate this metric, as well as, its derivative for all the computations on a Riemannian manifold. This directly means that the computational cost will be extremely high, and hence, prohibited. For this reason we provide the following relaxation

$$\mathbf{M}(\mathbf{z}) = \mathbf{J}_{\mu}(\mathbf{z})^{\top} \mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z})) \mathbf{J}_{\mu}(\mathbf{z}) + \mathbf{J}_{\sigma}(\mathbf{z})^{\top} \mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z})) \mathbf{J}_{\sigma}(\mathbf{z}). \quad (21)$$

Here we are based on the realistic assumption that the generator’s uncertainty in the regions of the latent space with representations of the training data to be $\sigma(\mathbf{z}) \rightarrow \mathbf{0}$. The reason is that $\mu(\mathbf{z})$ is trained to reconstruct sufficiently well the training data \mathbf{x} , and we are also based on the main assumption that the training data lie *near* a manifold $\mathcal{M} \subset \mathcal{X}$. This essentially implies that the corresponding deviation of \mathbf{x} from \mathcal{M} will be negligible and our $g(\mathcal{Z}) = \mathcal{M}_{\mathcal{Z}} \approx \mathcal{M}$. Therefore, the Eq. 20 will become first

$$\widetilde{\mathbf{M}}_{\varepsilon}(\mathbf{z}) = [\mathbf{J}_{\mu}(\mathbf{z}) + \mathbf{J}_{\sigma}(\mathbf{z})\varepsilon]^{\top} \mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z})) [\mathbf{J}_{\mu}(\mathbf{z}) + \mathbf{J}_{\sigma}(\mathbf{z})\varepsilon], \quad (22)$$

and we will compute the expectation upon this to get the $\mathbf{M}(\mathbf{z}) = \mathbb{E}_{\varepsilon}[\widetilde{\mathbf{M}}_{\varepsilon}(\mathbf{z})]$ that is shown in Eq. 21. As regards the regions far from the latent codes where $\sigma(\mathbf{z}) \gg 0$, the $\mathbf{J}_{\sigma}(\cdot)$ will be the dominant term, and hence, the contribution of $\mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z}))$ or even $\mathbf{M}_{\mathcal{X}}(\mu(\mathbf{z}) + \text{diag}(\varepsilon) \cdot \sigma(\mathbf{z}))$ will be negligible there anyways.

In order to demonstrate this behavior, we generate a dataset near \mathcal{M} as $\mathbf{x} = [x_1, x_2, \sin(x_1)]$ and we add noise using $\mathcal{N}(0, \sigma^2)$ with two different $\sigma = 0.1, 0.2$. For the ambient metric we use the cost based RBF approach by selecting 3 points and their 10 nearest neighbors in \mathcal{X} with $y_k = 10$. We train two VAEs and we show in the latent space the resulting Riemannian metric with and without the stochasticity of the generator for the evaluation of the ambient metric $\mathbf{M}_{\mathcal{X}}(\cdot)$.

From the results in Fig. 12 we observe that by considering the true expected Riemannian metric, the captured structure does not differ significantly from the one we get using the proposed relaxation, especially, near the latent codes. Therefore, by taking into account the trade off, we argue that it is sufficient to use the expected generated manifold $\mathcal{M}_{\mathcal{Z}} = \mathbb{E}_{\varepsilon \sim p(\varepsilon)}[\mathcal{M}_{\varepsilon}]$ such that to evaluate the ambient Riemannian metric $\mathbf{M}_{\mathcal{X}}(\cdot)$ as it is shown in Eq. 21.

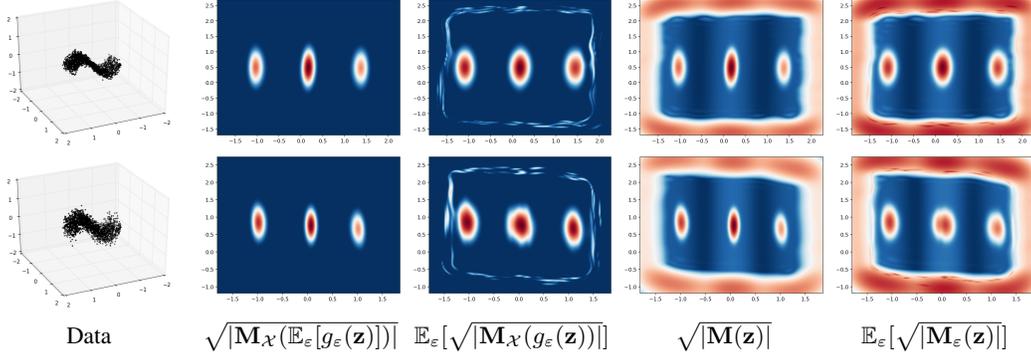


Figure 12: In the *top* row we added noise with $\sigma = 0.1$ and in the *bottom* row $\sigma = 0.2$. We see that in the two last columns and per row, that the structure does not change significantly when we use the proposed relaxation.

E Experiments

In this section we provide further details and discussion regarding the conducted experiments.

E.1 Details for the Generative Adversarial Network demonstrations

Synthetic data. The synthetic data are generated as follows. First, we pick the centers of 6 Gaussian distributions $\mathcal{N}(\mu, 0.2^2 \cdot \mathbb{I}_2)$ uniformly on a circle with radius 3 and one in the center. Then, we generate 300 points from each Gaussian that can be seen as the actual latent representations, and thus, we construct the data $\mathbf{x} = [z_1, z_2, 0.3 \cdot (z_1^2 + z_2^2) + \varepsilon]$, where $\varepsilon \sim \mathcal{N}(0, 0.1^2)$. We used a Wasserstein GAN with latent space $\mathcal{Z} = \mathbb{R}^2$ and ambient space $\mathcal{X} = \mathbb{R}^3$, with functions

Function	Layer 1	Layer 2	Output
$f(\mathbf{z})$	$\tanh(2)$	$\tanh(3)$	linear(3)
$d(\mathbf{x})$	LeakyReLU(3) + Dropout(0.3)	LeakyReLU(3) + Dropout(0.3)	linear(1)

We trained the model using Adam optimizer for 1000 epochs with stepsize $1e^{-2}$ and batch sizes of size 128, and also, we used ℓ_2 regularization for the weights with parameter $1e^{-5}$. The discriminator is trained for 5 more steps within each epoch and the weights are clamped into the interval $(-0.01, 0.01)$ to satisfy the Lipschitz constraint of the Wasserstein GAN. For the sampling of the latent codes we experimented both with standard Markov Chain Monte Carlo (MCMC), as well as rejection sampling [Bishop, 2006]. For the mixture of LAND we used the default training procedure with 10 epochs and full covariance matrices per component. In order to construct the RBF ambient metric we used 20 components and the scaling factor of the bandwidth was set to $\kappa = 1$ as discussed in Appendix C.2.

MNIST data. We used the digits 0,1,2, we scaled them in the interval $[-1, 1]$ and we added point-wise noise $\varepsilon \sim \mathcal{N}(0, 0.02^2)$, such that the data do not lie exactly on \mathcal{M} . Thus, it is easier to train the generator without utilizing the bounded $\tanh(\cdot)$ in the output layer to clip the values. Because, in such a case the meaningful extrapolation is not anymore useful, since the linear part will be also clipped. However, when we show or pass the images into the critic $d(\mathbf{x})$ first we apply the $\tanh(\cdot)$ function. Specifically, the latent space is $\mathcal{Z} = \mathbb{R}^5$ and $\mathcal{X} = \mathbb{R}^{784}$ and the functions are defined as

Function	Layer 1	Layer 2	Output
$f(\mathbf{z})$	$\tanh(128)$	$\tanh(256)$	linear(784)
$d(\mathbf{x})$	$\tanh + \text{LeakyReLU}(128) + \text{Drop}(0.3)$	$\text{LeakyReLU}(128) + \text{Drop}(0.3)$	linear(1)

The discriminator is trained for 5 more steps within each epoch and the weights are clamped into the interval $(-0.01, 0.01)$ to satisfy the Lipschitz constraint of the Wasserstein GAN. The model is trained using Adam optimizer for 10000 epochs and batch size 64 with stepsize $1e^{-4}$ and ℓ_2 regularization of the weights with parameter $1e^{-7}$. For the sampling of the latent codes we experimented both with standard Markov Chain Monte Carlo, as well as rejection sampling. The ambient Riemannian metric is constructed with the RBF method discussed in Appendix C.2 and we used 100 centers and $\kappa = 0.33$ which decreases the bandwidth of the RBF kernels. Moreover, we projected linearly the data to a lower dimensional space $\mathcal{X}' = \mathbb{R}^{10}$ using principal components analysis (PCA), where we construct the metric $\mathbf{M}_{\mathcal{X}'}(\cdot)$ (see Appendix B). Also, to stabilize training and to prevent mode collapse, we include a VAE loss when we train the generator with a regularization parameter $1e^{-5}$.

We see that using the ambient metric $\mathbf{M}_{\mathcal{X}'}(\cdot)$ improves the sampling, and some additional results are shown in Fig. 13. The resulting samples due to the $\mathbf{M}_{\mathcal{X}'}(\cdot)$ lie closer to the support of the given data manifold, and also, we avoid samples in-between the disconnected components in \mathcal{X} . Moreover, we show some additional interpolations (see Fig. 14) where we again see that using the ambient metric improves the interpolations. In particular, the difference between our proposed approach and the standard shortest paths is that the ambient Riemannian metric pulls the paths towards the data manifold and avoids “shortcuts”. Intuitively, shortcut means that the path moves optimally on the generated $\mathcal{M}_{\mathcal{Z}}$, but not necessarily always near the given data manifold. Note that $\mathcal{M}_{\mathcal{Z}}$ is a continuous smooth surface and some parts are not near the given data points/manifold, but without considering the $\mathbf{M}_{\mathcal{X}'}(\cdot)$ it might be cheap to move there which is a misleading behavior.

Pre-trained model. We used as generator a Progressive GAN (PGAN) [Karras et al., 2018] which utilizes a latent space $\mathcal{Z} = \mathbb{R}^d$ with $d = 512$ and has ambient space $\mathcal{X} = \mathbb{R}^D$ with $D = 256 \times 256 \times 3$, while the labeled training dataset is not directly provided. Note that in this generator it is not included the linear map to ensure meaningful extrapolation, and also, due to $\text{ReLU}(\cdot)$ activation the $\mathbf{M}(\cdot)$ is not sufficiently smooth.

However, we tested how the additional consideration of an ambient metric can affect the shortest paths, and additionally, we use a heuristic that we describe below for computing approximate shortest paths where a smooth metric $\mathbf{M}(\cdot)$ is not necessary. Moreover, we upscaled the standard CelebA labeled dataset of size $128 \times 128 \times 3$ to D in order to be able to compute the linear projection matrix $\mathbf{P} \in \mathbb{R}^{d' \times D}$ from \mathcal{X} to \mathcal{X}' with $d' = 1000$ and the linear mean $\mathbf{c} \in \mathbb{R}^D$. See discussion in Appendix B regarding this linear projection step.

Obviously, the computation of the Jacobian matrix for this $g(\cdot)$ is prohibited due to the size of the latent space and the complexity of the model, even with finite differences. So we relied on some tricks that we explain below, in order to be able to compute relatively efficient shortest paths. First, we define a new latent space $\tilde{\mathcal{Z}} = \mathbb{R}^{\tilde{d}}$ of dimensionality $\tilde{d} < d$ with $\tilde{d} = 10$ and we construct an ortho-normal random projection matrix $\tilde{\mathbf{U}} \in \mathbb{R}^{d \times \tilde{d}}$. In such a way, we can compute shortest paths in $\tilde{\mathcal{Z}}$ that correspond to shortest paths on a \tilde{d} -dimensional sub-space in \mathcal{Z} . Hence, in total we have

$$\tilde{\mathcal{Z}} \xrightarrow{\tilde{\mathbf{U}}} \mathcal{Z} \xrightarrow{g(\cdot)} \mathcal{X} \xrightarrow{\mathbf{P}(\cdot - \mathbf{c})} \mathcal{X}'. \quad (23)$$

Clearly, this tactic constraints the shortest paths to lie on the linear subspace spanned by $\tilde{\mathbf{U}}$ in \mathcal{Z} , and hence, they are not be able to move freely in the whole \mathcal{Z} . However, this approximation allows us to compute shortest paths in reasonable time. Essentially, we induce the pull-back Riemannian metric in a lower dimensional latent space $\tilde{\mathcal{Z}}$, while the $\tilde{\mathbf{U}}$ matrix does not introduce further distortions.

The main reason for using the $\tilde{\mathbf{U}}$ is that when \tilde{d} is relatively small, we are able to compute the Jacobian matrix $\tilde{\mathbf{J}}_g(\tilde{\mathbf{z}}) \in \mathbb{R}^{D \times \tilde{d}}$ using finite differences. In particular, in the latent space \tilde{d} we approximate the j -th column of the Jacobian from $\tilde{\mathcal{Z}} \rightarrow \mathcal{X}$ with finite differences as

$$\tilde{\mathbf{J}}_g^j(\tilde{\mathbf{z}}) = \lim_{\lambda \rightarrow 0} \frac{g(\tilde{\mathbf{U}} \cdot (\tilde{\mathbf{z}} + \lambda \mathbf{e}_j)) - g(\tilde{\mathbf{U}} \cdot \tilde{\mathbf{z}})}{\lambda}, \quad (24)$$

where $\tilde{\mathbf{z}} \in \mathbb{R}^{\tilde{d}}$ and $\mathbf{e}_j = [0, \dots, 1, \dots, 0]$ a \tilde{d} -dimensional vector of zeros with 1 at the j -th location. Furthermore, we can exploit the forward pass to compute simultaneously all the columns of the Jacobian, by using a batch of inputs that we truncate using the identity matrix $\lambda \mathbb{I}_{\tilde{d}}$. In such a way, we can compute the Jacobian at a point with only one forward pass with batch size $\tilde{d} + 1$. Nevertheless, even in an approximate ODE solver this is still very computationally expensive. So we implemented one heuristic to compute the shortest path based on the idea of ISOMAP [Tenenbaum et al., 2000].

We start by sampling 10000 points in \tilde{d} uniformly inside a hyper-sphere of radius 4 and using k -means we find 100 prototypes. Note that we do not have access to latent codes, so we want to introduce some artificial codes in $\tilde{\mathcal{Z}}$. Then, using these prototypes we construct the K -nearest neighbor graph with $K = 7$ by using the Euclidean distance to find the neighbors. But, for the weight of the edges we use the straight line distance measured under pull-back Riemannian metric that we can evaluate using the finite differences based Jacobian as

$$\text{length}[l(t)] = \int_0^1 \sqrt{\langle \dot{l}(t_n), \mathbf{M}_{\text{fd}}(l(t_n)) \dot{l}(t_n) \rangle} dt \approx \sum_{t_n=1}^N \sqrt{\langle \dot{l}(t_n), \mathbf{M}_{\text{fd}}(l(t_n)) \dot{l}(t_n) \rangle} \Delta t_n, \quad (25)$$

where $l(t)$ is the line between two latent points in $\tilde{\mathcal{Z}}$. For the metric $\mathbf{M}_{\text{fd}}(\cdot)$ first we compute the Jacobian of the total map $\mathbf{P}(g(\tilde{\mathbf{U}} \cdot l(t)) - \mathbf{c})$ with respect to $l(t)$, which can be achieved by using the finite differences for the Jacobian computation of the map $g(\tilde{\mathbf{U}} \cdot l(t))$, and then, we use the $\mathbf{M}_{\mathcal{X}'}(\mathbf{P}(g(\tilde{\mathbf{U}} \cdot l(t)) - \mathbf{c}))$. In particular the metric is equal to

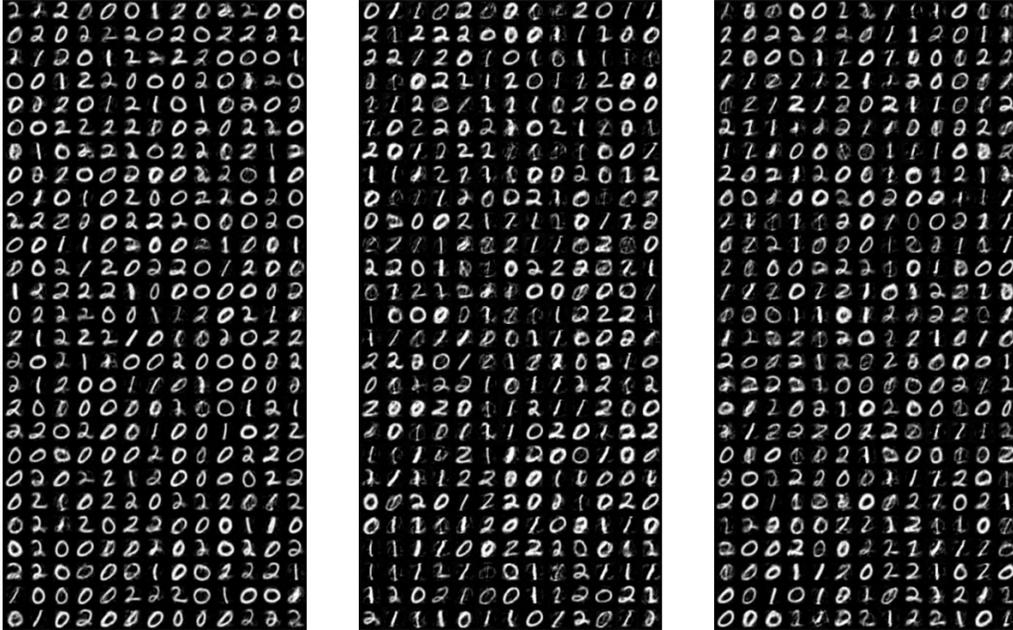
$$\mathbf{M}_{\text{fd}}(\tilde{\mathbf{z}}) = \left[\mathbf{P} \frac{\partial g(\tilde{\mathbf{U}} \cdot \tilde{\mathbf{z}})}{\partial \tilde{\mathbf{z}}} \right]^\top \mathbf{M}_{\mathcal{X}'}(\mathbf{P}(g(\tilde{\mathbf{U}} \cdot \tilde{\mathbf{z}}) - \mathbf{c})) \left[\mathbf{P} \frac{\partial g(\tilde{\mathbf{U}} \cdot \tilde{\mathbf{z}})}{\partial \tilde{\mathbf{z}}} \right]. \quad (26)$$

Essentially, the straight line in $\tilde{\mathcal{Z}}$ measured under the Riemannian metric will inform us how far on the manifold in the space \mathcal{X}' and under the metric $\mathbf{M}_{\mathcal{X}'}(\cdot)$ are the decoded latent points that seem to be close in the \tilde{d} -dimensions.

For two test points in $\tilde{\mathcal{Z}}$ that we want to compute the shortest path, first we find their closest K -neighbors from the points on the graph using the Euclidean metric, and then, we assign the corresponding edge weights using the Riemannian distances. Finally, we chose two auxiliary points, one per kNN set with the smallest Riemannian distance. Thus, we can find the discrete shortest path using Dijkstra's algorithm on the graph using the auxiliary nodes as the boundaries. Note that the path prefers edges with low weight i.e., the edge corresponds to a curve on \mathcal{M} with small length. Ultimately, the continuous path is the a cubic spline interpolation through the points of the discrete path on the graph replacing the two auxiliary points with the test points. Obviously, this is a heuristic methodology to approximate the true shortest path which is inspired by ISOMAP, and also, a very similar heuristic approach that has been proposed in Chen et al. [2019].

The task that we want our ambient Riemannian metric to model, is to avoid regions with blond people when interpolating between two latent codes. As we described above we linearly project in \mathcal{X}' the implicitly given data manifold $\mathcal{M} \subset \mathcal{X} = \mathbb{R}^D$, by using the standard labeled CelebA dataset. In \mathcal{X}' , we construct the $\mathbb{M}_{\mathcal{X}'}(\cdot)$ which is based on a simple RBF cost based metric (see Appendix C.3) with $y_k = 1e^9$ and $\sigma = 5$. Therefore, we have to define the centers $c_k \in \mathcal{X}'$. In order to do that, first we train on the labeled CelebA dataset of size $128 \times 128 \times 3$ a simple convolutional neural network classifier $c(\mathbf{x})$ (see table below). Once the classifier is trained, we decode the nodes of the graph and samples from the prior, which we classify after resizing from $\mathbb{R}^{256 \times 256 \times 3}$ to $\mathbb{R}^{128 \times 128 \times 3}$. With these steps, we are able to define the centers of the metric in \mathcal{X}' , by using the points that are classified as blond. Note that this is a very simple to implement metric, but rather informative, since the shortest path is penalized heavily when moves close to the high cost regions in \mathcal{X}' . Essentially, the (discrete) shortest path avoids the nodes which after decoding fall near the high cost regions in \mathcal{X}' . We show some further interpolation results in Fig. 15 using different projection matrices $\tilde{\mathbf{U}}$, which means that we explore different subspaces in \mathcal{Z} , and consequently, on $\mathcal{M}_{\mathcal{Z}}$.

Function	Layer 1,2,3	Output
$c(\mathbf{x})$	$3 \times [\text{conv}(16,5,1) + \text{MaxPool}(2) + \text{ReLU}]$	Sigmoid(Linear(265, 1))



(a) From $q(\mathbf{z})$ with $\mathbb{M}_{\mathcal{X}}(\cdot)$. (b) From $q(\mathbf{z})$ without $\mathbb{M}_{\mathcal{X}}(\cdot)$. (c) From prior $p(\mathbf{z})$.

Figure 13: Additional samples for the MNIST data, GAN experiment with 5-dimensional latent space. Note that our proposed method does not generate a lot of ghostly samples, which fall on parts of the ambient space with no given data nearby.

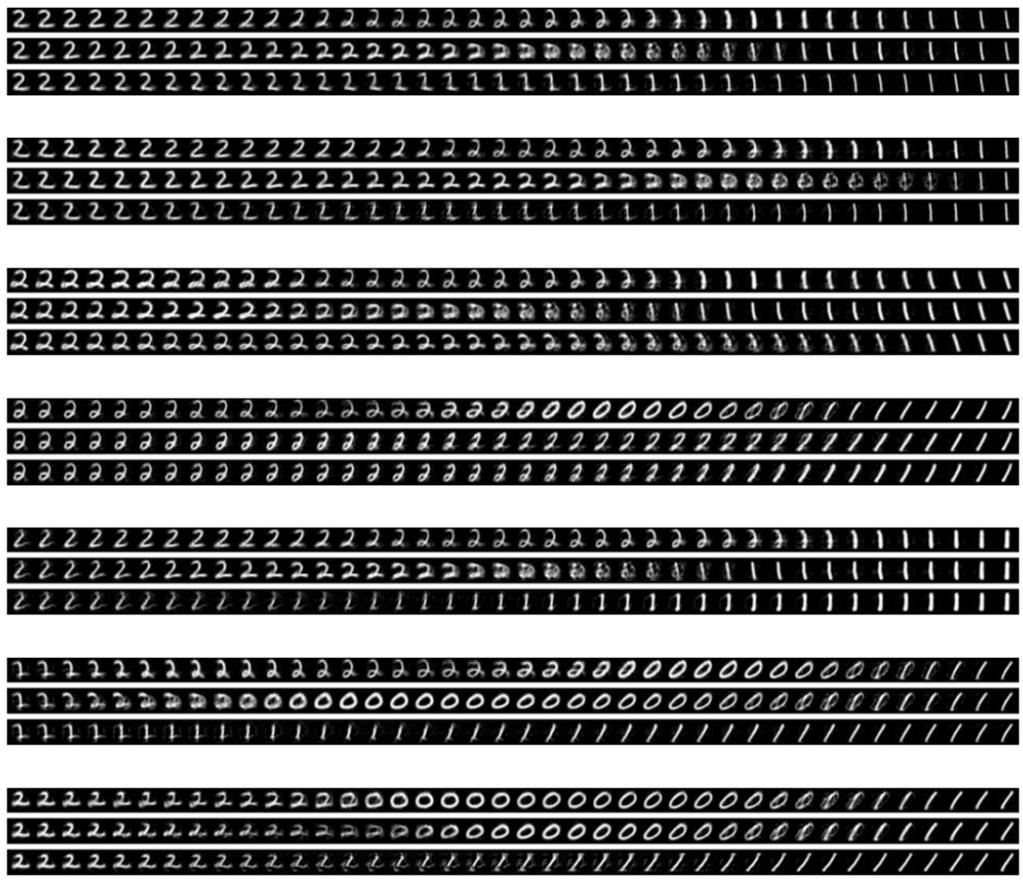


Figure 14: Additional interpolation results. From *top* to *bottom*: our interpolation, interpolation without using the $M_{\chi}(\cdot)$, linear interpolation. Note that our interpolation (top rows) avoids the “shortcuts” of the simple shortest path interpolant (middle rows), while the linear path is arbitrary.

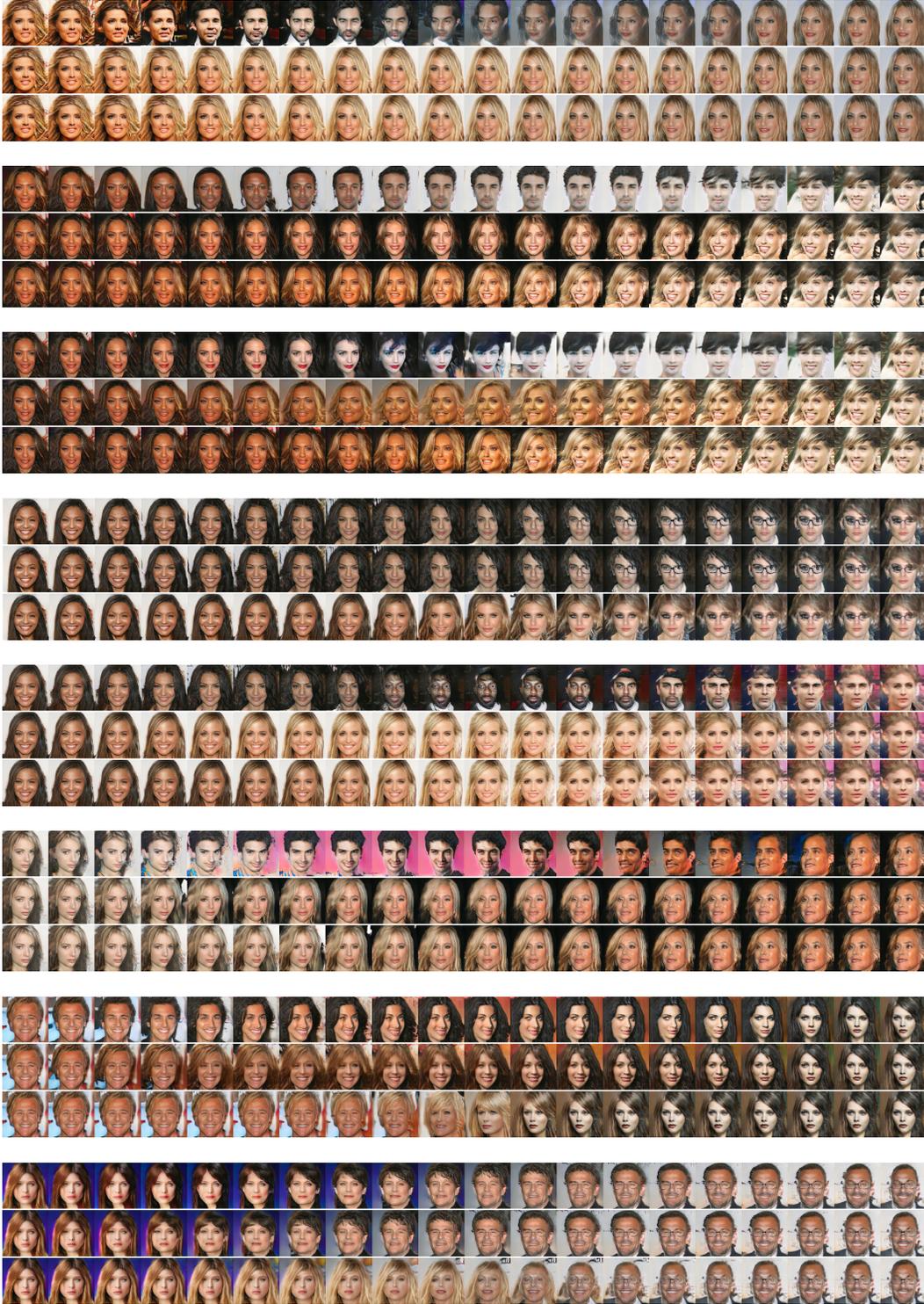


Figure 15: Additional interpolation results for the PGAN. From *top* to *bottom*: our interpolation, interpolation without using the $M_{\mathcal{X}'}(\cdot)$, linear interpolation. Note that our interpolation (top rows) provides a smooth transition between the images, while it avoids the high cost regions (people with blond hair). The shortest path without the ambient metric still provides smooth changes, however, it often crosses high cost regions. The relatively smooth behavior of the straight line is due to the nature of the generator and not due to the actual interpolant.

E.2 Details for the Variational Auto Encoder experiment

We used the MNIST digits 0,1,2,3 scaled in the interval $[-1, 1]$ and then we added point-wise noise $\varepsilon \sim \mathcal{N}(0, 0.02^2)$. As we explained before, we add the noise such that the data to not lie exactly on \mathcal{M} , so that we can train the generator without utilizing the bounded $\tanh(\cdot)$ in the output layer to clip the values. However, when we show the images first we apply the $\tanh(\cdot)$ function. Note that in the stochastic generator case the meaningful extrapolation is not necessary, even if we use it in our experiments, since the uncertainty quantification helps to properly capture the data manifold structure. The ambient space is $\mathcal{X} = \mathbb{R}^{784}$ and the latent space $\mathcal{Z} = \mathbb{R}^5$. We used the following functions

Function	Layer 1	Layer 2	Output
decoder: $f(\mathbf{z})$	softplus(128)	softplus(256)	linear(784)
decoder: $\beta(\mathbf{z})$	RBF(100)		linear(784)
encoder: $\mu_\phi(\mathbf{x})$	softplus(256)	softplus(128)	linear $_{>0}$ (5)
encoder: $\sigma_\phi^2(\mathbf{x})$	softplus(256)	softplus(128)	softplus(linear(5))

where the $\beta(\mathbf{z}) + \zeta = \frac{1}{\sigma^2(\mathbf{z})}$ with $\zeta = 1e^{-6}$ and $\beta(\mathbf{z})$ is an RBF with 100 centers and only positive weights. We trained the model using Adam optimizer for 1000 epochs and batch sizes of size 64 with stepsize $1e^{-4}$ and also ℓ_2 regularization of the weight with parameter $\lambda = 1e^{-5}$.

For the interpolation experiment, the LDA metric is constructed by considering the digits 0,1,3 in the same class, while in the kernel density estimation experiment every class is separated. We used 2000 randomly chosen training points as the base points \mathbf{x}_s , the $\varepsilon = 1e^{-3}$, the number of nearest neighbors is $K = 50$ and we used a fixed number of iterations 20. See Appendix C.1 for details.

For the cost function based ambient metrics we use the RBF cost discussed in Appendix C.3, and we start by picking 3 latent codes in \mathcal{Z} . Then, we decode these points and by using the closest 100 neighbors per decoded point in \mathcal{X}' we constructed the metric with parameters $y_k = 100$ and $\sigma = 0.2$. So in total we have 300 RBF basis functions in \mathcal{X}' . We used the same approach both in the interpolations and the KDE experiment.

For the linear combination of the ambient metrics we used the weights 1 for the LDA, 0.001 for the local diagonal inverse covariance and 0.1 for the cost metric. We used the same coefficients both in the interpolations and the KDE experiment. Also, the reason for so different coefficients is the scaling of each individual metric. Of course, choosing carefully the parameters of each ambient metric could regularize the scaling differences. However, a principled method to estimate the mixture coefficients is a future problem.

As an additional experiment we examine if the proposed meaningful extrapolation technique is useful. Therefore, using a set of points \mathbf{z}_s on a uniform grid in the latent space, we generate the points in \mathcal{X} on the expected manifold $\mathcal{M}_{\mathcal{Z}}$ as $\mathbf{x}_s = \mu(\mathbf{z}_s) = f(\mathbf{z}_s) + \mathbf{U} \cdot \text{diag}([\sqrt{\lambda_1}, \sqrt{\lambda_2}]) \cdot \mathbf{z}_s + \mathbf{b}$. Here, the $f : \mathcal{Z} \rightarrow \mathcal{X}$ is a DNN and the linear part is defined as explained in the main paper. In Fig. 16 we show for each \mathbf{z}_s the Euclidean distance measured in \mathcal{X} between the center of the training data and the corresponding point \mathbf{x}_s . Indeed, we see that as we move further from the prior $p(\mathbf{z})$ support, the distance between the points on the generated surface and the center of the data increases. However, we observe that the distance on the x_1 -axis increases faster than the x_2 -axis. The reason is that the corresponding eigenvalue of the linear map is higher, so the generated $\mathcal{M}_{\mathcal{Z}}$ extrapolates linearly faster along this latent dimension. Note that in this example we used the `softplus` activation function, for which the extrapolation behavior is more difficult to analyze than the `tanh`.

Even so we get a meaningful extrapolation due to the linear part of the function $\mu(\cdot)$.

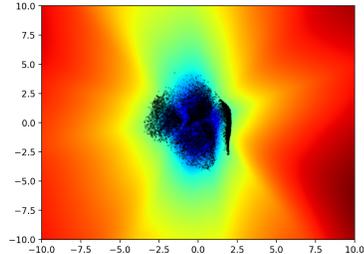


Figure 16: Meaningful extrapolation. The distance $\|\mathbf{b} - \mu(\mathbf{z})\|_2$ in \mathcal{X} , where \mathbf{b} is the center of the data.

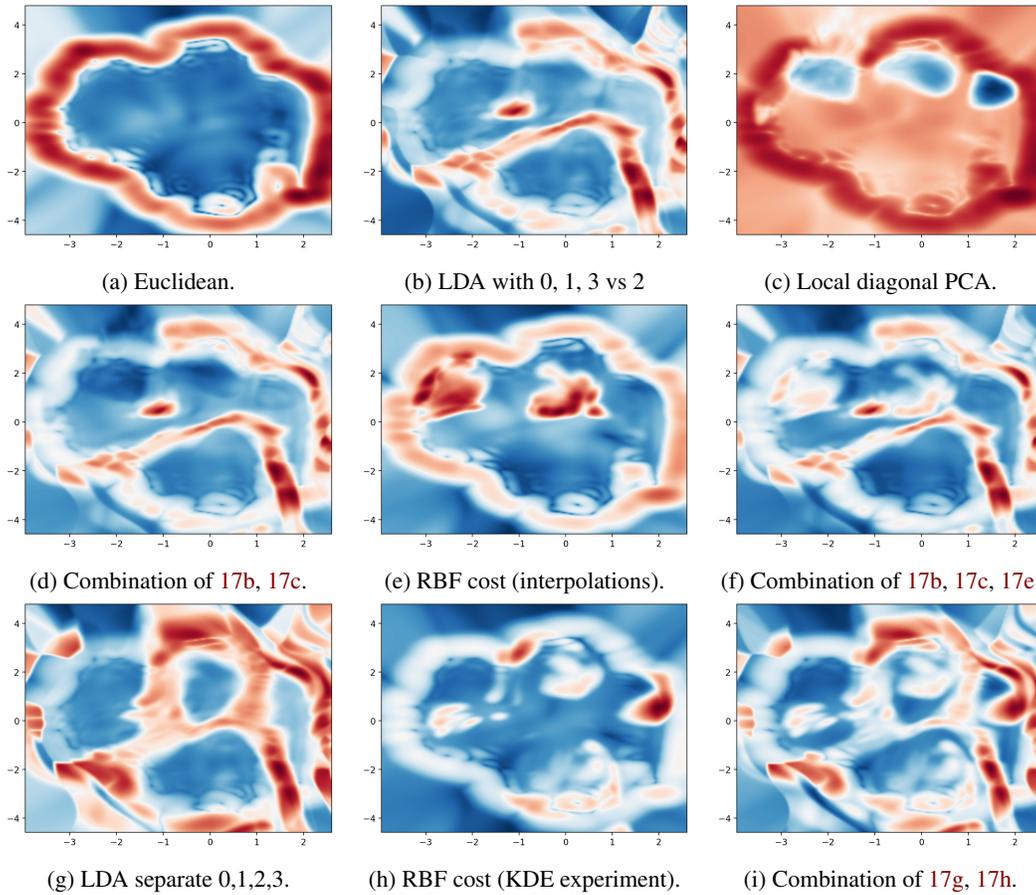


Figure 17: The Riemannian measure for the VAE experiments and several ambient metrics $M_{\mathcal{X}}(\cdot)$. In each caption we mention briefly the form and the details of the ambient metric.