A Geometric Perspective on Variational Autoencoders

Clément Chadebec
Université Paris Cité, INRIA, Inserm, SU
Centre de Recherche des Cordeliers
clement.chadebec@inria.fr

Stéphanie Allassonnière
Université Paris Cité, INRIA, Inserm, SU
Centre de Recherche des Cordeliers
stephanie.allassonniere@inria.fr

Abstract

This paper introduces a new interpretation of the Variational Autoencoder framework by taking a fully geometric point of view. We argue that vanilla VAE models unveil naturally a Riemannian structure in their latent space and that taking into consideration those geometrical aspects can lead to better interpolations and an improved generation procedure. This new proposed sampling method consists in sampling from the uniform distribution deriving intrinsically from the learned Riemannian latent space and we show that using this scheme can make a vanilla VAE competitive and even better than more advanced versions on several benchmark datasets. Since generative models are known to be sensitive to the number of training samples we also stress the method's robustness in the low data regime.

1 Introduction

Variational Autoencoders (VAE) [29, 50] are powerful generative models that map complex input data in a much lower dimensional space referred to as the latent space while driving the latent variables to follow a given prior distribution. Their simplicity to use in practice has made them very attractive models to perform various tasks such as high-fidelity image generation [48], speech modeling [5], clustering [59] or data augmentation [8].

Nonetheless, when taken in their simplest version, it was noted that these models produce blurry samples on image generation tasks most of the time. This undesired behavior may be due to several limitations of the VAE framework. First, the training of a VAE aims at maximizing the Evidence Lower BOund (ELBO) which is only a lower bound on the true likelihood and so does not ensure that we are always actually improving the true objective [6, 1, 24, 14, 60]. Second, the prior distribution may be too simplistic [15] leading to poor data generation and there exists no guarantee that the actual distribution of the latent code will match a given prior distribution inducing distribution mismatch [13]. Hence, trying to tackle those limitations through richer posterior distributions [53, 49] or better priors [57] represents a major part of the proposed improvements over the past few years. However, the tractability of the ELBO constrains the choice in distributions and so finding a trade-off between model expressiveness and tractability remains crucial. In this paper, we take a rather different approach and focus on the geometrical aspects a vanilla VAE is able to capture in its latent space. In particular, we propose the following contributions:

- We show that VAEs unveil naturally a latent space with a structure that can be modeled as a Riemannian manifold through the learned covariance matrices in the variational posterior distributions and that such modeling can lead to better interpolations.
- We propose a natural sampling scheme consisting in sampling from a uniform distribution defined on the learned manifold and given by the Riemannian metric. We show that this procedure improves the generation process from a *vanilla* VAE significantly without complexifying the model nor the training. The proposed sampling method outperforms more advanced VAE models in terms of Frechet Inception Distance [23] and Precision and

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Recall [52] scores on four benchmark datasets. We also discuss and show that it can benefit more recent VAEs as well.

- We show that the method appears robust to dataset size changes and outperforms even more strongly peers when only *smaller* sample sizes are considered.
- We discuss the link of the proposed metric to the *pull-back* metric.

2 Variational autoencoders

Considering that we are given $x \in \mathbb{R}^D$ a set of data points deriving from an unknown distribution $p(x)$, a VAE aims at inferring p with a parametric model $\{p_\theta, \theta \in \Theta\}$ using a maximum likelihood estimator. A key assumption behind the VAE is to assume that the generation process involves latent variables z living in a lower dimensional space such that the generative model writes

$$z \sim p(z) \quad ; \quad x \sim p_\theta(x|z)\,,$$

where p is a prior distribution over the latent variables often taken as a standard Gaussian and $p_\theta(x|z)$ is referred to as the decoder and is most of the time taken as a parametric distribution the parameters of which are estimated using neural networks. Hence, the likelihood p_θ writes:

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p(z)dz\,. \tag{1}$$

As this integral is most of the time intractable so is $p_\theta(z|x)$, the posterior distribution. Hence, Variational Inference [26] is used and a simple parametrized variational distribution $q_\phi(z|x)$ is introduced to approximate the posterior $p_\theta(z|x)$. $q_\phi(z|x)$ is referred to as the *encoder* and, in the vanilla VAE, q_ϕ is chosen as a multivariate Gaussian whose parameters μ_ϕ and $\mathbf{\Sigma}_\phi$ are again given by neural networks. An unbiased estimate $\hat{p}_\theta$ of the likelihood $p_\theta(x)$ can then be derived using importance sampling with $q_\phi(z|x)$ and the ELBO objective follows using Jensen's inequality:

$$\log\, p_\theta(x) = \log \mathbb{E}_{z\sim q_\phi}\left[\hat{p}_\theta\right] \geq \mathbb{E}_{z\sim q_\phi}\left[\log \hat{p}_\theta\right] \geq\, \mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z) - \mathrm{KL}(q_\phi(z|x)\|p(z)) = \mathcal{L} \tag{2}$$

The ELBO is now tractable since both $q_\phi(z|x)$ and $p_\theta(x|z)$ are known and so can be optimized with respect to the *encoder* and *decoder* parameters.

Remark 2.1. In practice, $p_\theta(x|z)$ is chosen depending on the modeling of the input data but is often taken as a simple distribution (*e.g* fixed variance Gaussian, Bernoulli ...) and a weight β can be applied to balance the weight of the KL term [24]. Hence, the ELBO can also be seen as a two terms objective [20]. The first one is a reconstruction term given by $p_\theta(x|z)$ while the second one is a regularizer given by the KL between the variational posterior q_ϕ and the prior p. For instance, in the case of a fixed variance Gaussian for $p_\theta(x|z)$ we have

$$\mathcal{L}_{\text{REC}} = \|x - \mu_\theta(z)\|_2^2, \;\; \mathcal{L}_{\text{REG}} = \beta \cdot \mathrm{KL}(q_\phi(z|x)\|p(z))\,. \tag{3}$$

3 Related work

A natural way to improve the generation from VAEs consists in trying to use more complex priors [25] than the standard Gaussian distribution used in the initial version such that they better match the true distribution of the latent codes. For instance, using a Mixture of Gaussian [41, 17] or a Variational Mixture of Posterior (VAMP) [57] as priors was proposed. In the same vein, hierarchical latent variable models [55, 31] or prior learning [12, 2] have recently emerged and aimed at finding the best suited prior distribution for a given dataset. Acceptance/rejection sampling method was also proposed to try to improve the expressiveness of the prior distribution [4]. Some recent works linking energy-based models (EBM) and VAEs [58] or modeling the prior as an EBM [45] have demonstrated promising results and are also worth citing.

On the ground that the latent space must adapt to the data as well, *geometry-aware* latent space modelings as hypershpere [16], torus [19] or Poincaré disk [40] or discrete latent representations [48] were proposed. Other recent contributions proposed to see the latent space as a Riemannian manifold where the Riemannian metric is given by the Jacobian of the generator function [3, 10, 54]. This metric was then used directly within the prior modeled by Brownian motions [27]. Others

proposed to learn the metric directly from the data throughout training thanks to *geometry-aware* normalizing flows [9] or learn the latent structure of the data using transport operators [13]. While these geometry-based methods show interesting properties of the learned latent space they either require the computation of a time consuming model-dependent function, the Jacobian, or add further parameters to the model to learn the metric or transport operators adding some computational burden.

Arguing that VAEs are essentially autoencoders regularized with a Gaussian noise, Ghosh et al. [20] proposed another interesting interpretation of the VAE framework and showed that other types of regularization may be of interest as well. Since the generation process from these autoencoders is no longer relying on the prior distribution, the authors proposed to use ex-post density estimation by fitting simple distributions such as a Gaussian mixture in the latent space. While this paves the way for consideration of other ways to generate data, it mainly reduces the VAE framework to an autoencoder while we believe that it can also unveil interesting geometrical aspects.

Another widely discussed improvement of the model consists in trying to tweak the approximate posterior in the ELBO so that it better matches the true posterior using MCMC methods [53] or normalizing flows [49]. For instance, methods using Hamiltonian equations in the flows to target the true posterior [7] were proposed.

Finally, while discussing the potential link between PCA and autoencoders some intuitions arose on the impact of both the intrinsic structure of the variance of the data [47] and the shape of the covariance matrices in the posterior distributions [51] on disentanglement in the latent space. We also believe that these covariance matrices indeed play a crucial role in the modeling of the latent space but in this paper, we instead propose to see their inverse as the value of a Riemannian metric.

4 Proposed method

In this section, we show that a vanilla VAE unveils naturally a Riemannian structure in its latent space through the learned covariance matrices in the variational posterior distribution. We then propose a new natural generation scheme guided by this estimated geometry and consisting in sampling from a uniform distribution deriving intrinsically from the learned Riemannian manifold.

4.1 A word on Riemannian geometry

First, we briefly recall some basic elements of Riemannian geometry needed in the rest of the paper. A more detailed discussion on Riemannian manifolds may be found in Appendix A. A d-dimensional manifold $\mathcal{M}$ is a manifold which is locally homeomorphic to a d-dimensional Euclidean space. If the manifold $\mathcal{M}$ is further differentiable it possesses a tangent space T_z at any $z \in \mathcal{M}$ composed of the tangent vectors of the curves passing by z. If $\mathcal{M}$ is equipped with a smooth inner product $g : z \to \langle \cdot | \cdot \rangle_z$ defined on its tangent space T_z for any $z \in \mathcal{M}$ then $\mathcal{M}$ is called a Riemannian manifold and g is the associated Riemannian metric. Then, a local representation of g at any $z \in \mathcal{M}$ is given by the positive definite matrix $\mathbf{G}(z)$ (See Appendix A). If $\mathcal{M}$ is connected, a Riemannian distance between two points z_1, z_2 of $\mathcal{M}$ can be defined

$$\mathrm{dist}_{\mathbf{G}}(z_1, z_2) = \inf_{\gamma} \int_a^b \sqrt{\dot{\gamma}(t)^\top \mathbf{G}(\gamma(t)) \dot{\gamma}(t)} dt = \inf_{\gamma} L(\gamma) \quad \text{s.t.} \quad z_1 = \gamma(a), z_2 = \gamma(b)\,, \tag{4}$$

where L is the length of curves $\gamma : \mathbb{R} \to \mathcal{M}$ traveling from z_1 to z_2. Curves minimizing L and parametrized proportionally to the arc length are *geodesic*. The manifold $\mathcal{M}$ is said to be *geodesically complete* if all geodesic curves can be extended to $\mathbb{R}$. In an Euclidean space, $\mathbf{G}$ reduces to I_d and the distance becomes the classic Euclidean one. A simple extension of this Euclidean framework consists in assuming that the metric is given by a constant positive definite matrix $\mathbf{\Sigma}$ different from I_d. In such a case the induced Riemannian distance is the well-known Mahalanobis distance $\mathrm{dist}_{\mathbf{\Sigma}}(z_1, z_2) = \sqrt{(z_2 - z_1)^\top \mathbf{\Sigma} (z_2 - z_1)}\,.$

4.2 The Riemannian Gaussian distribution

Given the Riemannian manifold $\mathcal{M}$ endowed with the Riemannian metric $\mathbf{G}$ and a chart z, an infinitesimal volume element may be defined on each tangent space T_z of the manifold $\mathcal{M}$ as follows

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)} dz\,, \tag{5}$$

with dz being the Lebesgue measure. This defines a canonical measure on the manifold and allows to extend the notion of random variables to Riemannian manifolds whose density can be defined with respect to that Riemannian measure (see Appendix A). Hence, a Riemannian Gaussian distribution on $\mathcal{M}$ can be defined using the Riemannian distance of Eq. (4) instead of the Euclidean one.

$$\mathcal{N}_{\text{riem}}(z|\sigma,\mu) = \frac{1}{C}\exp\Big(-\frac{\text{dist}_{\mathbf{G}}(z,\mu)^2}{2\sigma}\Big), \qquad C = \int_{\mathcal{M}} \exp\Big(-\frac{\text{dist}_{\mathbf{G}}(z,\mu)^2}{2\sigma}\Big) d\mathcal{M}_z\,, \tag{6}$$

where $d\mathcal{M}_z$ is the volume element defined in Eq. (5). Thus, a multivariate normal distribution with covariance matrix $\mathbf{\Sigma}$ is only a specific case of the Riemannian distribution with $\sigma = 1$ and defined on the manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ where $\mathbf{G}$ is the constant Riemannian metric $\mathbf{G}(z) = \mathbf{\Sigma}^{-1}$, $\forall z \in \mathcal{M}$.

4.3 Geometrical interpretation of the VAE framework

Within the VAE framework, the variational distribution $q_\phi(z|x)$ is often chosen as a simple multivariate Gaussian distribution defined on $\mathbb{R}^d$ with d being the latent space dimension. Hence, as explained in the previous section, given an input data point x_i, the posterior $q_\phi(z|x_i) = \mathcal{N}(\mu(x_i), \mathbf{\Sigma}(x_i))$ can also be seen as a Riemannian Gaussian distribution where the Riemannian distance is simply the distance with respect to the metric tensor $\mathbf{\Sigma}^{-1}(x_i)$. Hence, the VAE framework can be seen as follow: As with an autoencoder, the VAE provides a code $\mu(x_i)$ which is a lower dimensional representation of an input data point x_i. However, it also gives a tensor $\mathbf{\Sigma}^{-1}(x_i)$ depending on x_i which can be seen as the value of a Riemannian metric $\mathbf{G}$ at $\mu(x_i)$ *i.e.*

$$\mathbf{G}(\mu(x_i)) = \mathbf{\Sigma}^{-1}(x_i)\,.$$

This metric is crucial since it impacts the notion of distance in the latent space now seen as the Riemannian manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ and so changes the directions that are favored in the sampling from the posterior distribution $q_\phi(z|x_i)$. Then, a sample z is drawn from a standard (*i.e.* $\sigma = 1$ in Eq. (6)) Riemannian Gaussian distribution and fed to the decoder. Since we only have access to a finite number of metric tensors $\mathbf{\Sigma}^{-1}(x_i)$, as a first approximation the VAE model assumes that the metric is locally constant close to $\mu(x_i)$ and so the Riemannian distance reduces to the Mahalanobis distance in the posterior distribution. This drastically simplifies the training process since now Riemannian distances have closed form and so are easily computable. Interestingly, the VAE framework will impose through the ELBO expression given in Eq. (3), that z gives a sample $x \sim p_\theta(x|z)$ close to x_i when decoded. Since z has a probability density function imposing higher probability for samples having the smallest Riemannian distance to μ, the VAE imposes in a way that latent variables that are close in the latent space with respect to the metric $\mathbf{G}$ will also provide samples that are close in the data space $\mathcal{X}$ in terms of L2 distance as noticed in Remark. 2.1. Noteworthy is that the latter distance can be amended through the choice of the decoder $p_\theta(x|z)$. This is an interesting property since it allows the VAE to directly link the learned Riemannian distance in the latent space to the distance in the data space. The regularization term in Eq. (3) ensures that the covariance matrices do not collapse to $\mathbf{0}_d$ and constraints the latent codes to remain close to the origin easing optimization. Finally, at the end of training, we have a lower dimensional representation of the training data given by the means of the posteriors $\mu(x_i)$ and a family of metric tensors ($\mathbf{G}_i = \mathbf{\Sigma}^{-1}(x_i)$) corresponding to the value of a Riemannian metric defined locally on the latent space. Inspired from [22], we propose to build a smooth continuous Riemannian metric defined on the entire latent space as follows:

$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i)\cdot\omega_i(z) + \lambda\cdot e^{-\tau\|z\|_2^2}\cdot I_d\,, \quad \omega_i(z) = \exp\left(-\frac{\text{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z,\mu(x_i))^2}{\rho^2}\right), \tag{7}$$

where $\text{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z,\mu(x_i))^2 = (z-\mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z-\mu(x_i))$ is the Riemannian distance between z and $\mu(x_i)$ with respect to the locally constant metric $\mathbf{G}(\mu(x_i)) = \mathbf{\Sigma}^{-1}(x_i)$. Since the sum in Eq. (7) is made on the total number of training samples N, the number of centroids ($\mu(x_i)$) and so of reference metric tensors can be decreased for huge datasets by selecting only $k < N$ elements[1] and increasing ρ to reduce memory usage. We provide an ablation study on the impact of λ, the number of centroids and their choice along with a discussion on the choice for ρ in Appendix F. The parameter τ is only there to ensure that the volume of $(\mathbb{R}^d, \mathbf{G})$ is finite, a property that is needed in Sec. 4.5, and its value can be set as close as desired to zero so the norm of z does not influence the metric

[1]This may be performed with k-medoids algorithm.

close to the centroids. In practice, it is set below computer precision (*i.e.* $\tau \approx 0$). Rigorously, the metric defined in Eq. (7) should have been used during the training process. Nonetheless, this would have made the training longer and trickier since it would involve i) the computation of Riemannian distances that have no longer closed form and so make the resolution of the optimization problem in Eq. (4) needed, ii) the sampling from Eq. (6) which is not trivial and iii) the computation of the regularization term. Moreover, for small values of β in Eq. (3), the samples generated from the variational distribution $z \sim \mathcal{N}(\mu(x_i), \mathbf{\Sigma}(x_i))$ can be assumed to be concentrated around $\mu(x_i)$ and so we have the following first-order Taylor expansion around $\mu(x_i)$

$$\mathbf{G}(z) \approx \mathbf{\Sigma}^{-1}(x_i) + \sum_{j=1, j\neq i}^{N} \mathbf{\Sigma}^{-1}(x_j) \cdot \underbrace{\omega_j(\mu(x_i))}_{\approx 0} + \mathbf{\Sigma}^{-1}(x_i) \cdot \underbrace{\mathbf{J}_{\omega_i}(\mu(x_i))}_{=0}(z - \mu(x_i)) \,, \quad (8)$$

where $\mathbf{J}_{\omega_i}(\mu(x_i))$ is the Jacobian of the interpolant ω_i evaluated at $\mu(x_i)$. Note that we have further assumed small enough ρ and λ to neglect the influence of the other $\mathbf{\Sigma}(x_j)$ in Eq. (7). Hence by approximating the value of the metric during training by its value at $\mu(x_i)$ (*i.e.* $\mathbf{\Sigma}^{-1}(x_i)$), the VAE training remains unchanged, stable and computationally reasonable since Riemannian Gaussians become multivariate Gaussians in $q_\phi(z|x)$ as explained before. Noteworthy is the fact that following the discussion on the role of the KL loss in the VAE framework and the experiments conducted in [20], in our vision of the VAE, the prior distribution is only seen as a regularizer though the KL term and other latent space regularization schemes may have been also envisioned. In the following, we keep the proposed vision and do not amend the training.

4.4 Link with the *pull-back* metric

It has been shown that a natural Riemannian metric on the latent space of generative models can be the *pull-back* metric given by $\mathbf{G}(z) = \mathbf{J}_g(z)^\top \mathbf{J}_g(z)$ [3] and induced by the decoder mapping $g : \mathbb{R}^d \to \mathbb{R}^D$ outputing the parameters of the conditional distribution $p_\theta(x|z)$. Actually, there exists a strong relation linking the metric proposed in this paper to the *pull-back* metric. Indeed, assuming that samples from the variational posterior $z \sim q_\phi(z|x) = \mathcal{N}(\mu(x), \mathbf{\Sigma}(x))$ remain close to $\mu(x)$ (*e.g.* by setting a small β in Eq. (3)) allows to consider an approximation of the log density $h(z) := \log p_\theta(x|z)$ next to $\mu(x)$ for a given x [33].

$$h(z) \approx \; h(\mu(x)) + \mathbf{J}_h(\mu(x))(z - \mu(x)) \; + \; \frac{1}{2}(z - \mu(x))^\top \mathbf{H}_h(\mu(x))(z - \mu(x)) \,,$$

where $\mathbf{J}_h(\mu(x))$ is the Jacobian and $\mathbf{H}_h(\mu(x))$ is the Hessian of h. Using this and remarking that

$$\mathbb{E}_{z\sim q_\phi}\Big[\mathbf{J}_h(\mu)(z-\mu)\Big] = 0 \quad \text{and} \quad \mathbb{E}_{z\sim q_\phi}\Big[(z-\mu)^\top \mathbf{H}_h(\mu)(z-\mu)\Big] = \mathrm{Tr}(\mathbf{H}_h(\mu)\mathbf{\Sigma}) \,,$$

makes the ELBO in Eq. (3) write:

$$\mathcal{L} \approx h(\mu(x)) + \frac{1}{2}\mathrm{Tr}(\mathbf{H}_h(\mu(x))\mathbf{\Sigma}(x)) - \beta \, \mathrm{KL}(q_\phi(z|x) \| p(z)) \,. \quad (9)$$

Assuming a standard Gaussian prior, Kumar and Poole [33] showed that $\widetilde{\mathbf{\Sigma}}$ maximizing the ELBO is

$$\widetilde{\mathbf{\Sigma}}(x) = \Big(I_d - \frac{1}{\beta}\mathbf{H}_h(\mu(x))\Big)^{-1} \,, \quad (10)$$

and if we further assume some regularity on the neural networks used for the decoder mapping g (*e.g.* piece-wise linear activation functions) we have

$$\widetilde{\mathbf{\Sigma}}(x) = \Big(I_d - \frac{1}{\beta}\mathbf{J}_g(\mu(x))^\top \mathbf{H}_p(g(\mu(x)))\mathbf{J}_g(\mu(x))\Big)^{-1} \,, \quad (11)$$

where $\mathbf{H}_p(g(\mu(x)))$ is the Hessian of $\log p_\theta(x; g(z))$. A standard case for the VAE is to assume that $p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma \cdot I_D)$ and so gives $\mathbf{H}_p(g(\mu(x))) = -\frac{1}{\sigma} \cdot I_D$. If we further set $\sigma = \frac{1}{\beta}$, Eq. (11) gives a relation between the *pull-back* and the metric we propose

$$\widetilde{\mathbf{\Sigma}}^{-1}(x) = \mathbf{J}_g(\mu(x))^\top \mathbf{J}_g(\mu(x)) + I_d \,.$$

Hence, the proposed metric is closely linked to the *pull-back* metric and may be useful to approximate it (at least close to the $\mu(x)$) and so avoid the computation of a potentially costly function.

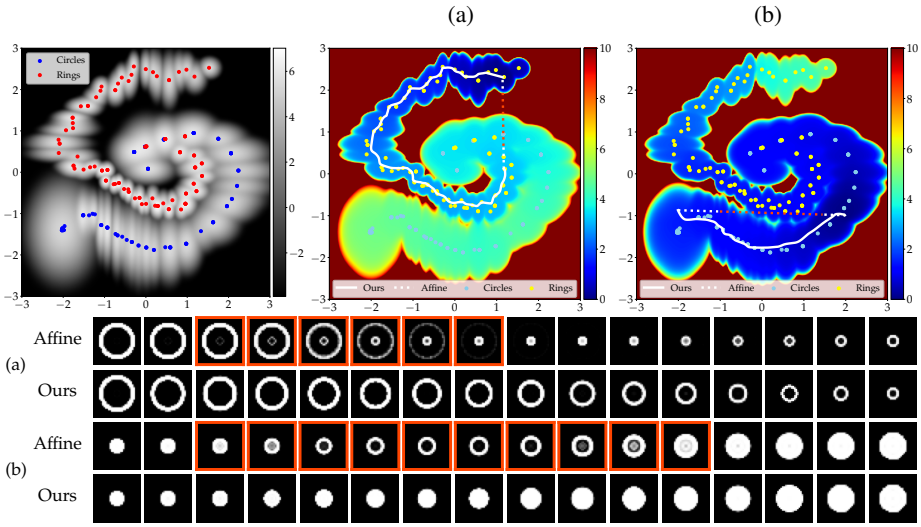


Figure 1: *Top left:* Visualization and interpolation in a 2D latent space learned by a VAE trained with binary images of rings and disks. The log of the metric volume element $\sqrt{\det \mathbf{G}(z)}$ (proportional to the log of the density we propose to sample from) is shown in gray scale. *Top middle and right*: Riemannian distance from a starting point (color maps). The dashed lines are affine interpolations between two points in the latent space and the solid ones are obtained by solving Eq. (12). *Bottom:* Decoded samples along the interpolation curves.

4.5 Geometry-aware sampling

Assuming that the VAE has learned a latent representation of the data in a space seen as a Riemannian manifold, we propose to exploit this strong property to enhance the generation procedure. A natural way to sample from such a latent space would consist in sampling from the uniform distribution intrinsically defined on the learned manifold. Similar to the Gaussian distribution presented in Sec. 4.2, the notion of uniform distribution can indeed be extended to Riemannian manifolds. Given a set $\mathcal{A} \subset \mathcal{M}$ having a finite volume, a Riemannian uniform distribution on $\mathcal{A}$ writes [46]

$$p_{\mathcal{A}}(z) = \frac{\mathbf{1}_{\mathcal{A}}(z)}{\mathrm{Vol}(\mathcal{A})} = \frac{\mathbf{1}_{\mathcal{A}}(z)}{\int_{\mathcal{M}} \mathbf{1}_{\mathcal{A}}(z) d\mathcal{M}_z} \,.$$

This density is taken with respect to $d\mathcal{M}_z$, the Riemannian measure but using Eq. (5) and a coordinate system z allows to obtain a pdf defined with respect to the Lebesgue one. Moreover, since the volume of the whole manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ is finite, we can now define a *uniform distribution* on $\mathcal{M}$

$$\mathcal{U}_{\text{Riem}}(z) = \frac{\sqrt{\det \mathbf{G}(z)}}{\int_{\mathbb{R}^d} \sqrt{\det \mathbf{G}(z) dz}} \,.$$

Since the Riemannian metric has a closed form expression given by Eq. (7) sampling from this distribution is quite easy and may be performed using the HMC sampler [42] for instance. Now we are able to sample from the intrinsic uniform distribution which is a natural way of exploring the estimated manifold and the sampling is guided by the geometry of the latent space. A discussion on practical outcomes can be found in Appendix B. Noteworthy is the fact that this approach can also be easily applied to more recent VAE models having a Gaussian posterior (*e.g.* [6, 34, 57, 38]). We detail this and show that the proposed method can also benefit these models in Appendix G.

4.6 Illustration on a toy dataset

The usefulness of such sampling procedure can be observed in Figure 1 where a vanilla VAE was trained with a toy dataset composed of binary images of disks and rings of different size and thickness (example inspired by [8]). On the left is presented the learned latent space along with the embedded training points given by the colored dots. The log of the metric volume element is given in gray scale.

In this example, we clearly see a geometrical structure appearing since the disks and rings seem to wrap around each other. Obviously, sampling using the prior (taken as a $\mathcal{N}(0, I_d)$) in such a case is far from being optimal since the sampling will be performed regardless of the underlying distribution of the latent variables and so will create irrelevant samples. To further illustrate this, we propose to interpolate between points in the latent space using different cost functions. Dashed lines represent affine interpolations while the solid ones show interpolations aiming at minimizing the potential $V(z) = (\sqrt{\det \mathbf{G}(z)})^{-1}$ all along the curve *i.e.* solving the minimization problem

$$\inf_{\gamma} \int_0^1 V(\gamma(t))dt \quad \text{s.t.} \quad \gamma(0) = z_1,\ \gamma(1) = z_2\,. \tag{12}$$

In Figure 1 are presented the decoded samples all along the interpolation curves. Thanks to those interpolations we can see that i) the latent space seems to really have a specific geometrical structure since decoding all along the interpolation curves obtained by solving Eq. (12) leads to qualitatively satisfying results, ii) certain locations of the latent space must be avoided since sampling there will produce irrelevant samples (see red frames and corresponding red dashes). Using the proposed sampling scheme will allow to sample in the light-colored areas and so ensure that the sampling remains close to the data *i.e.* where information is available and so does not produce irrelevant images when decoded while still proposing relevant variations from the input data.

5 Experiments

In this section, we conduct a comparison with other VAE models using other regularization schemes, more complex priors, richer posteriors, ex-post density estimation or trying to take into account geometrical aspects. In the following, all the models share the same auto-encoding neural network architectures and we used the code and hyper-parameters provided by the authors if available[2]. See Appendix D for models descriptions and the comprehensive experimental set-up.

5.1 Generation with benchmark datasets

First, we compare the proposed sampling method to several VAE variants such as a Wasserstein Autoencoder (WAE) [56], Regularized Autoencoders (RAEs) [20], a vamp-prior VAE (VAMP) [57], a Hamiltonian VAE (HVAE) [7], a geometry-aware VAE (RHVAE) [9] and an Autoencoder (AE). We elect these models since they use different ways to generate the data using either the prior or ex-post density estimation. For the latter, we fit a 10-component mixture of Gaussian in the latent space after training like [20] .

Figure 2 shows a qualitative comparison between the resulting generated samples for MNIST [35] and CELEBA [37], see Appendix C for SVHN [44] and CIFAR 10 [32]. Interestingly, using the non-prior based methods seems to produce qualitatively better samples (rows 7 to end). Nonetheless, the resulting samples seem even sharper when the sampling takes into account geometrical aspects of the latent space as we propose (last row). Additionally, even though the exact same model is used, we clearly see that using the proposed method represents a strong improvement of the generation process from a vanilla VAE when compared to the samples coming from a normal prior (second row). This confirms that even the simplest VAE model actually contains a lot of information in its latent space but the limited expressiveness of the prior impedes to access to it. Hence, using more complex priors such as the VAMP may be a tempting idea. However, one must keep in mind that the ELBO objective in Eq. (2) must remain tractable and so using more expressive priors may be impossible.

These observations are even more supported by Table 1 where we report the Frechet Inception Distance (FID) and the precision and recall (PRD) score against the test set to assess the sampling quality and diversity. Again, fitting a mixture of Gaussian (GMM) in the latent space appears to be an interesting idea since it allows for a better expressiveness and latent space prospecting. For instance, on MNIST the FID falls from 40.7 with the prior to 13.1 when using a GMM. Nonetheless, with the proposed method we are able to make it even smaller (8.5) and PRD scores higher without changing the model and performing post processing. This can also be observed on the 3 other datasets. Impressively, in almost all cases, the proposed generation method can either compete or outperform peers both in terms of FID and PRD scores.

[2]We also perform a wider hyper-parameter search on MNIST and CELEBA for each model in Appendix C

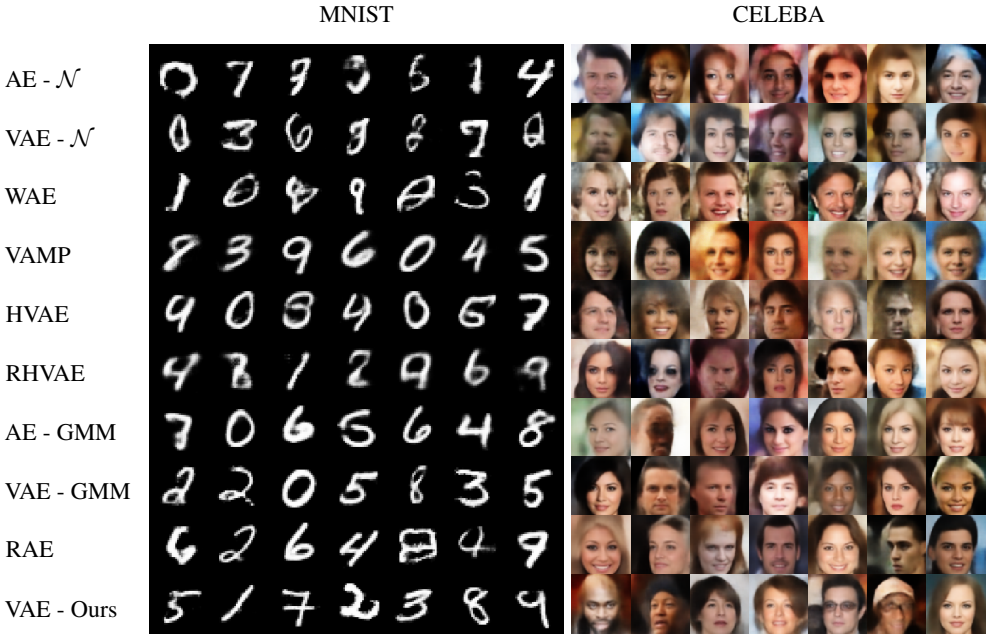
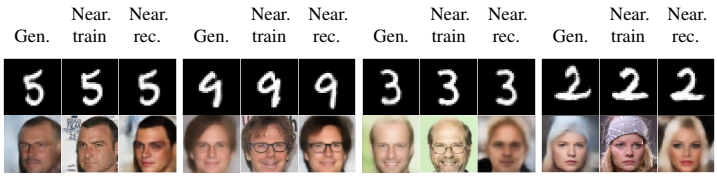


Figure 2: Generated samples with different models and generation methods. Results with RAE variants are provided in Appendix C.



reconstruction vs. generation		
	MNIST	CELEBA
FID	11.27	30.12

Figure 3: *Left:* Nearest train image (near. train) and nearest image in all reconstructions of train images (near. rec.) to the generated one (Gen.) with the proposed method. Note: the nearest reconstruction may be different from the reconstruction of the nearest train image. *Right*: The FID score between 10k generated images and 10k reconstructed train samples.

Finally, we check if the proposed method does not overfit the training data and is able to produce diverse samples by showing the nearest neighbor in the train set and the nearest image in all the reconstructions of the train images to a generated image in Figure 3 (left). We also provide the FID score between 10k generated samples and 10k train reconstructions in Figure 3 (right). These experiments show that the generated samples are not only resampled train images and that the sampling prospects quite well the manifold. To support even more this claim we provide in Appendix F an analysis in a case where only two centroids are selected in the metric. This also shows that the generated samples are not only an interpolation between the k selected centroids since some generated images contain attributes that are not present in the images of the decoded centroids.

The outcome of such an experiment is that using post training latent space processing such as ex-post density estimation or adding some geometrical consideration to the model allows to strongly improve the sampling without adding more complexity to the model. Generating 1k samples on CELEBA takes approx. 5.5 min for our method vs. 4 min for a 10-component GMM on a GPU V100-16GB.

5.2 Investigating robustness in low data regime

We perform a comparison using the same models and datasets as before but we progressively decrease the size of the training set to see the robustness of the methods according to the number of samples. Despite rarely performed in most generative models related papers, this setup appeared to us relevant

Table 1: FID (lower is better) and PRD score (higher is better) for different models and datasets. For the mixture of Gaussian (GMM), we fit a 10-component mixture of Gaussian in the latent space.

MODEL	MNIST (16)		SVHN (16)		CIFAR 10 (32)		CELEBA (64)	
	FID ↓	PRD ↑	FID ↓	PRD ↑	FID ↓	PRD ↑	FID ↓	PRD ↑
AE - $\mathcal{N}(0,1)$	46.41	0.86/0.77	119.65	0.54/0.37	196.50	0.05/0.17	64.64	0.29/0.42
WAE	20.71	0.93/0.88	49.07	0.80/**0.85**	132.99	0.24/0.52	54.56	**0.57**/0.55
VAE - $\mathcal{N}(0,1)$	40.70	0.83/0.75	83.55	0.69/0.55	162.58	0.10/0.32	64.13	0.27/0.39
VAMP	34.02	0.83/0.88	91.98	0.55/0.63	198.14	0.05/0.11	73.87	0.09/0.10
HVAE	15.54	0.97/0.95	98.05	0.64/0.68	201.70	0.13/0.21	52.00	0.38/0.58
RHVAE	36.51	0.73/0.28	121.69	0.55/0.41	167.41	0.12/0.22	55.12	0.45/0.56
AE - GMM	9.60	0.95/0.90	54.21	0.82/0.83	130.28	0.35/0.58	56.07	0.32/0.48
RAE (GP)	9.44	0.97/**0.98**	61.43	0.79/0.78	120.32	0.34/0.58	59.41	0.28/0.49
RAE (L2)	9.89	0.97/**0.98**	58.32	0.82/0.79	123.25	0.33/0.54	54.45	0.35/0.55
RAE (SN)	11.22	0.97/**0.98**	95.64	0.53/0.63	114.59	0.32/0.53	55.04	0.36/0.56
RAE	11.23	**0.98**/**0.98**	66.20	0.76/0.80	118.25	0.35/0.57	53.29	0.36/0.58
VAE - GMM	13.13	0.95/0.92	52.32	0.82/**0.85**	138.25	0.29/0.53	55.50	0.37/0.49
VAE - OURS	**8.53**	**0.98**/0.97	**46.99**	**0.84**/**0.85**	**93.53**	**0.71**/**0.68**	**48.71**	0.44/**0.62**

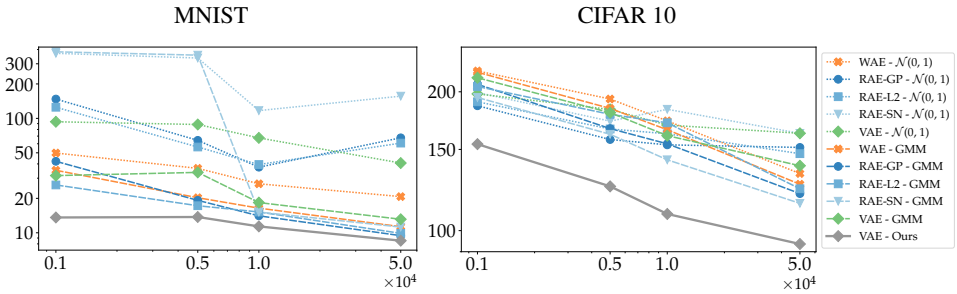


Figure 4: Evolution of the FID score according to the number of training samples.

since 1) it is well known that it may be challenging for these models, 2) in day-to-day applications collecting large databases may reveal costly if not impossible (*e.g.* medicine). Hence, we consider MNIST, CIFAR10 and SVHN and use either the full dataset size, 10k, 5k or 1k training samples. For each experiment, the best retained model is again the one achieving the best ELBO on the validation set the size of which is set as 20% of the train size. See Appendix D for further details about experiments set-up. Then, we report the evolution of the FID against the test set in Figure 4. Results obtained on SVHN are presented in Appendix E. Again, the proposed sampling method appears quite robust to the dataset size since it outperforms the other models' FID even when the number of training samples is smaller. This is made possible thanks to the proposed metric that allows to avoid regions of the latent space having poor information. Finally, our study shows that although using more complex generation procedures such as ex-post density estimation seems to still enhance the generation capability of the model when the number of training samples remains quite high ($\geq$5k), this gain seems to worsen when the dataset size reduces as illustrated on CIFAR. In addition, we also evaluate the model on a data augmentation task with neuroimaging data from OASIS [39] mimicking a day-to-day scenario where the limited data regime is very common in Appendix C.3.

6 Conclusion

In this paper, we provided a geometric understanding of the latent space learned by a VAE and showed that it can actually be seen as a Riemannian manifold. We proposed a new natural generation process consisting in sampling from the intrinsic uniform distribution defined on this learned manifold. The proposed method was empirically shown to be competitive with more advanced versions of the VAEs using either more complex priors, ex-post density estimation, normalizing flows or other regularization schemes. Interestingly, the proposed method revealed good robustness properties in complex settings such as high dimensional data or low sample sizes and appeared to benefit more recent VAE models as well. Future work would consist in trying to use this method to perform data augmentation in those challenging contexts and compare its reliability for such a task with state of the art methods or trying to use this metric to perform clustering in the latent space.

Acknowledgments and Disclosure of Funding

The research leading to these results has received funding from the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and reference ANR-10-IAIHU-06 (Agence Nationale de la Recherche-10-IA Institut Hospitalo-Universitaire-6). This work was granted access to the HPC resources of IDRIS under the allocation AD011013517 made by GENCI (Grand Equipement National de Calcul Intensif).

References

[1] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[2] Jyoti Aneja, Alexander Schwing, Jan Kautz, and Arash Vahdat. NCP-VAE: Variational autoencoders with noise contrastive priors. *arXiv:2010.02917 [cs, stat]*, 2020.

[3] Georgios Arvanitidis, Lars Kai Hansen, and Sören Hauberg. Latent space oddity: On the curvature of deep generative models. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[4] Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 66–75. PMLR, 2019.

[5] Merlijn Blaauw and Jordi Bonada. Modeling and transforming speech using variational autoencoders. *Morgan N, editor. Interspeech 2016; 2016 Sep 8-12; San Francisco, CA.[place unknown]: ISCA; 2016. p. 1770-4.*, 2016. Publisher: International Speech Communication Association (ISCA).

[6] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv:1509.00519 [cs, stat]*, 2016.

[7] Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, pages 8167–8177, 2018.

[8] Clément Chadebec, Elina Thibeau-Sutre, Ninon Burgos, and Stéphanie Allassonnière. Data augmentation in high dimensional low sample size setting using a geometry-based variational autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[9] Clément Chadebec, Clément Mantoux, and Stéphanie Allassonnière. Geometry-aware hamiltonian variational auto-encoder. *arXiv:2010.11518*, 2020.

[10] Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550. PMLR, 2018.

[11] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

[12] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[13] Marissa Connor, Gregory Canal, and Christopher Rozell. Variational autoencoder with learned latent structure. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2367. PMLR, 2021.

[14] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.

[15] Bin Dai and David Wipf. Diagnosing and Enhancing VAE Models. In *International Conference on Learning Representations*, 2018.

[16] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[17] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv:1611.02648 [cs, stat]*, 2017.

[18] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

[19] Luca Falorsi, Pim de Haan, Tim R. Davidson, Nicola De Cao, Maurice Weiler, Patrick Forré, and Taco S. Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv:1807.04689 [cs, stat]*, 2018.

[20] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[21] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[22] Søren Hauberg, Oren Freifeld, and Michael Black. A Geometric take on Metric Learning. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/ec5aa0b7846082a2415f0902f0da88f2-Paper.pdf`.

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

[24] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[25] Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2, 2016.

[26] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[27] Dimitrios Kalatzis, David Eklund, Georgios Arvanitidis, and Soren Hauberg. Variational autoencoders with riemannian brownian motion priors. In *International Conference on Machine Learning*, pages 5053–5066. PMLR, 2020.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114 [cs, stat]*, 2014.

[30] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[31] Alexej Klushyn, Nutan Chen, Richard Kurle, and Botond Cseke. Learning Hierarchical Priors in VAEs. *Advances in neural information processing systems*, page 10, 2019.

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[33] Abhishek Kumar and Ben Poole. On implicit regularization in β-vaes. In *International Conference on Machine Learning*, pages 5480–5490. PMLR, 2020.

[34] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[35] Yann LeCun. The MNIST database of handwritten digits. 1998.

[36] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[37] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[38] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[39] Daniel S. Marcus, Tracy H. Wang, Jamie Parker, John G. Csernansky, John C. Morris, and Randy L. Buckner. Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. *Journal of Cognitive Neuroscience*, 19(9):1498–1507, 2007.

[40] Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincaré variational auto-encoders. In *Advances in neural information processing systems*, pages 12565–12576, 2019.

[41] Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, volume 2, page 131, 2016.

[42] Radford M Neal. Hamiltonian importance sampling. In *talk presented at the Banff International Research Station (BIRS) workshop on Mathematical Issues in Molecular Dynamics*, 2005.

[43] Radford M Neal and others. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

[44] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[45] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. *Advances in Neural Information Processing Systems*, 33, 2020.

[46] Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006. ISSN 0924-9907, 1573-7683. doi: 10.1007/s10851-006-6228-4.

[47] Alexander Rakowski and Christoph Lippert. Disentanglement and local directions of variance. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–34. Springer, 2021.

[48] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in Neural Information Processing Systems*, 2020.

[49] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[50] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[51] Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue pca directions (by accident). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

[52] MSM Sajjadi, O Bachem, M Lucic, O Bousquet, and S Gelly. Assessing generative models via precision and recall. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, pages 5228–5237, 2019.

[53] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

[54] Hang Shao, Abhishek Kumar, and P. Thomas Fletcher. The riemannian geometry of deep generative models. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 428–4288. IEEE, 2018. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00071.

[55] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoder. In *29th Annual Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[56] I Tolstikhin, O Bousquet, S Gelly, and B Schölkopf. Wasserstein auto-encoders. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

[57] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

[58] Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*, 2020.

[59] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. Deep clustering by gaussian mixture variational autoencoders with graph embedding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6440–6449, 2019.

[60] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix C.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. 5.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendix D. We mention that we use code and data only when allowed by the license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We used well-known and publicly available datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We used well-known and publicly available datasets.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Further elements on Riemannian geometry

A d-dimensional Riemannian manifold $\mathcal{M}$ can be defined as a d-dimensional differentiable manifold equipped with is a smooth inner product $g : z \to \langle \cdot | \cdot \rangle_z$ defined on each tangent space $T_z\mathcal{M}$ of the manifold with $z \in \mathcal{M}$. A chart (or coordinate system) (U, ϕ) is a homeomorphism mapping an open set U of the manifold to an open set V of an Euclidean space. Given $z \in U$, a chart $\phi : (z^1, \dots, z^d)$ induces a basis $\left(\frac{\partial}{\partial z^1}, \dots, \frac{\partial}{\partial z^d}\right)_z$ on the tangent space $T_z\mathcal{M}$. Hence, the metric of a Riemannian manifold can be locally represented in the chart ϕ as a positive definite matrix as mentionned in Sec. 4.1.

$$\mathbf{G}(z) = (g_{i,j})_{z, 0 \leq i,j \leq d} = \left(\left\langle \frac{\partial}{\partial z^i} | \frac{\partial}{\partial z^j} \right\rangle_z \right)_{0 \leq i,j \leq d}, \tag{13}$$

for each point z of the manifold. That is for $v, w \in T_z\mathcal{M}$ and $z \in \mathcal{M}$, the inner product writes $\langle u | w \rangle_z = u^\top \mathbf{G}(z) w$. Assuming that the manifold is also connected, for any $z_1, z_2 \in \mathcal{M}$, two points of the manifold, we can consider a curve γ traveling in $\mathcal{M}$ and parametrized by $t \in [a, b]$ such that $\gamma(a) = z_1$ and $\gamma(b) = z_2$. Then, the length of γ is given by

$$L(\gamma) = \int_a^b \|\dot{\gamma}(t)\|_{\gamma(t)} dt = \int_a^b \sqrt{\langle \dot{\gamma}(t) | \dot{\gamma}(t) \rangle_{\gamma(t)}} dt$$

Curves γ that minimize L and are parameterized proportionally to the arc length are called *geodesic* curves. A distance $\mathrm{dist}_{\mathbf{G}}$ on the manifold $\mathcal{M}$ can then be derived and writes

$$\mathrm{dist}_{\mathbf{G}}(z_1, z_2) = \inf_\gamma L(\gamma) \qquad \text{s.t.} \qquad \gamma(a) = z_1, \gamma(b) = z_2 \tag{14}$$

The manifold $\mathcal{M}$ is said to be *geodesically complete* if all geodesic curves can be extended to $\mathbb{R}$. Given the Riemannian manifold $\mathcal{M}$ endowed with the Riemannian metric $\mathbf{G}$ and a chart z, an infinitesimal volume element may also be defined on each tangent space T_z of the manifold $\mathcal{M}$ as follows

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)} dz\,, \tag{15}$$

with dz being the Lebesgue measure. This defines a canonical measure on the manifold and allows to extend the notion of probability distributions to Riemannian manifolds. In particular, such a property allows to refer to random variables with a density defined with respect to the measure on the manifold. We recall such definition from [46] below

Definition A.1. Let $\mathcal{B}(\mathcal{M})$ be the Borel σ-algebra of $\mathcal{M}$. The random point $\mathbf{z}$ has a probability density function $\rho_{\mathbf{z}}$ if:

$$\forall \mathcal{Z} \in \mathcal{B}(\mathcal{M}), \quad \mathbb{P}(\mathbf{z} \in \mathcal{Z}) = \int_{\mathcal{Z}} \rho(z) d\mathcal{M}(z) \quad \text{and} \quad \int_{\mathcal{M}} \rho(z) d\mathcal{M}(z) = 1$$

Finally, given a chart ϕ defined on the whole manifold $\mathcal{M}$ and a random point $\mathbf{z}$ on $\mathcal{M}$, the point $\mathbf{p} = \phi(\mathbf{z})$ is a random point whose density $\rho'_{\mathbf{p}}$ may be written with respect to the Lebesgue measure as such [46]:

$$\rho'_{\mathbf{p}}(p) = \rho_{\mathbf{z}}(\phi^{-1}(p)) \sqrt{\det g(\phi^{-1}(p))} \tag{16}$$

B The generation process algorithm - Implementation details

In this appendix, we provide pseudo-code algorithms explaining how to build the metric from a trained VAE and how to use the proposed sampling process. Noteworthy is the fact that we do not amend the training process of the vanilla VAE which remains pretty simple and stable.

B.1 Building the metric

In this section, we explain how to build the proposed Riemannian metric. For the sake of clarity, we recall the expression of the metric below

$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d \,, \tag{17}$$

where

$$\omega_i(z) = \exp\left(- \frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2} \right) = \exp\left(- \frac{(z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i))}{\rho^2} \right),$$

Algorithm 1 Building the metric from a trained model

Input: A trained VAE model m, the training dataset $\mathcal{X}$, λ, τ ▷ In practice $\tau \approx 0$
for $x_i \in \mathcal{X}$ **do**
  $\mu_i, \mathbf{\Sigma}_i = m(x_i)$ ▷ Retrieve training embeddings and covariance matrices
end for
Select k centroids c_i in the μ_i ▷ e.g. with k-medoids
Get corresponding covariance matrices $\mathbf{\Sigma}_i$
$\rho \leftarrow \max_i \min_{j \neq i} \|c_i - c_j\|_2$ ▷ Set ρ to the max distance between two closest neighbors
Build the metric using Eq. (17)

$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}_i^{-1} \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d$$

Return G ▷ Return $\mathbf{G}$ as a function

As is standard in VAE implementations, we assume that the covariance matrices $\mathbf{\Sigma}_i$ given by the VAE are diagonal and that the encoder outputs a mean vector and the log of the diagonal coefficients. In the implementation, the exponential is then applied to recover the $\mathbf{\Sigma}_i$ so that no singular matrix arises.

B.2 Sampling process

Further to the description performed in the paper, we provide here a detailed algorithm stating the main steps of the generation process.

B.2.1 The HMC sampler

In the sampling process we propose to rely on the Hamiltonian Monte Carlo sampler to sample from the Riemanian uniform distribution. In a nutshell, the HMC sampler aims at sampling from a target distribution $p_{\text{target}}(z)$ with $z \in \mathbb{R}^d$ using Hamiltonian dynamics. The main idea behind such a sampler is to introduce an auxiliary random variable $v \sim \mathcal{N}(0, I_d)$ independent from z and mimic the behavior of a particle having z (resp. v) as location (resp. velocity). The Hamiltonian of the particle then writes

$$H(z, v) = U(z) + K(v) \,,$$

where $U(z)$ is the potential energy and $K(v)$ is its kinetic energy both given by

$$U(z) = -\log p_{\text{target}}(z), \qquad K(v) = \frac{1}{2} v^\top v$$

The following Hamilton's equations govern the evolution in time of the particle.

$$\begin{cases} \frac{\partial H(z,v)}{\partial v} = & v \,, \\ \frac{\partial H(z,v)}{\partial z} = & -\nabla_z \log p_{\text{target}}(z) \,. \end{cases} \tag{18}$$

In order to integrate these equations, recourse to the leapfrog integrator is needed and consists in applying n_{lf} times the following equations.

$$\left\{\begin{array}{ll} v(t+\frac{\varepsilon_{\mathrm{lf}}}{2}) & = v(t)+\frac{\varepsilon_{\mathrm{lf}}}{2}\cdot\nabla_z \log p_{\mathrm{target}}(z(t))\,, \\ z(t+\varepsilon_{\mathrm{lf}}) & = z(t)+\varepsilon_{\mathrm{lf}}\cdot v(t+\frac{\varepsilon_{\mathrm{lf}}}{2})\,, \\ v(t+\varepsilon_{\mathrm{lf}}) & = v(t+\frac{\varepsilon_{\mathrm{lf}}}{2})+\frac{\varepsilon_{\mathrm{lf}}}{2}\cdot\nabla_z \log p_{\mathrm{target}}(z(t+\varepsilon_{\mathrm{lf}}))\,, \end{array}\right. \tag{19}$$

where $\varepsilon_{\mathrm{lf}}$ is called the leapfrog step size. This algorithm produces a proposal $(\widetilde{z},\widetilde{v})$ that is accepted with probability α where

$$\alpha = \min\Big(1, \exp\Big(H(z,v)-H(\widetilde{z},\widetilde{v})\Big)\Big)\,.$$

This procedure is then repeated to create an ergodic Markov chain (z^n) converging to the distribution p_{target} [18, 36, 43, 21].

B.3 The proposed algorithm

In our setting the target density is given by the density of the Riemannian uniform distribution which writes with respect to Lebesgue measure as follows

$$p(z) = \mathcal{U}_{\mathrm{Riem}}(z) = \frac{1}{C}\sqrt{\det \mathbf{G}(z)} \qquad C = \int_{\mathbb{R}^d}\sqrt{\det \mathbf{G}(z)}dz\,. \tag{20}$$

Note that thanks to the shape of the metric, this distribution is well defined since $C < +\infty$. The log density follows

$$\log p(z) = \frac{1}{2}\log\det \mathbf{G}(z) - \log C\,,$$

Hence, the Hamiltonian writes

$$H(z,v) = -\log p(z) + \frac{1}{2}v^\top v\,,$$

and Hamilton's equations become

$$\left\{\begin{array}{lll} \frac{\partial H(z,v)}{\partial v} & = & v\,, \\ \frac{\partial H(z,v)}{\partial z_i} & = & -\frac{\partial \log p(z)}{\partial z_i} = -\frac{1}{2}\mathrm{tr}\Big(\mathbf{G}^{-1}(z)\frac{\partial \mathbf{G}(z)}{\partial z_i}\Big) \end{array}\right.$$

Since the covariance matrices are supposed to be diagonal as is standard in VAE implementations, the computation of the inverse metric is straightforward. Moreover, since $\mathbf{G}(z)$ is smooth and has a closed form, it can be differentiated with respect to z pretty easily. Now, the leapfrog integrator given in Eq. (19) can be used and the acceptance ratio α is easy to compute. Noteworthy is the fact that the normalizing constant C is never needed since it vanishes in the gradient computation and simplifies in the acceptance ratio α. We provide a pseudo-code of the proposed sampling procedure in Alg. 2. A typical choice in the sampler's hyper-parameters used in the paper is $N = 100$, $n_{\mathrm{lf}} = 10$ and $\varepsilon_{\mathrm{lf}} = 0.01$. The initialization of the chain can be done either randomly or on points that belong to the manifold (i.e. the centroids c_i or $\mu(x_i)$).

Algorithm 2 Proposed sampling process

Input: The metric function $\mathbf{G}$, hyper-parameters of the HMC sampler (chain length N, number of leapfrog steps n_{lf}, leapfrog step size $\varepsilon_{\mathrm{lf}}$)
Initialization: z ▷ Initialize the chain
for $i = 1 \rightarrow N$ **do**
  $v \sim \mathcal{N}(0, I_d)$ ▷ Draw a velocity
  $H_0 \leftarrow H(z, v)$ ▷ Compute the starting Hamiltonian
  $z_0 \leftarrow z$
  for $k = 1 \leftarrow n_{\mathrm{lf}}$ **do**
    $\bar{v} \leftarrow v - \frac{\varepsilon_{\mathrm{lf}}}{2} \cdot \nabla_z H(z, v)$
    $\widetilde{z} \leftarrow z + \varepsilon_{\mathrm{lf}} \cdot \bar{v}$ ▷ Leapfrog step Eq. (19)
    $\widetilde{v} \leftarrow \bar{v} - \frac{\varepsilon_{\mathrm{lf}}}{2} \cdot \nabla_z H(\widetilde{z}, \bar{v})$
    $v \leftarrow \widetilde{v}$
    $z \leftarrow \widetilde{z}$
  end for
  $H \leftarrow H(\widetilde{z}, \widetilde{v})$ ▷ Compute the ending Hamiltonian
  Accept $\widetilde{z}$ with probability $\alpha = \min\Big(1, \exp(H_0 - H)\Big)$
  if Accepted **then**
    $z \leftarrow \widetilde{z}$
  else
    $z \leftarrow z_0$
  end if
end for
Return z

C Other generation results

C.1 Some further samples on CELEBA and MNIST

In this section, we provide some further generated samples using the proposed method. Figure 5 and Figure 6 again support the fact that the method is able to generate sharp and diverse samples. We also add the other variants of the RAE model in Figure 7.

Figure 5: 100 samples with the proposed method on MNIST dataset.

Figure 6: 100 samples with the proposed method on CELEBA dataset.

Figure 7: Generated samples with different models and generation processes.

C.2 CIFAR and SVHN

In this appendix, we gather the resulting samplings from the different models considered for SVHN and CIFAR 10.

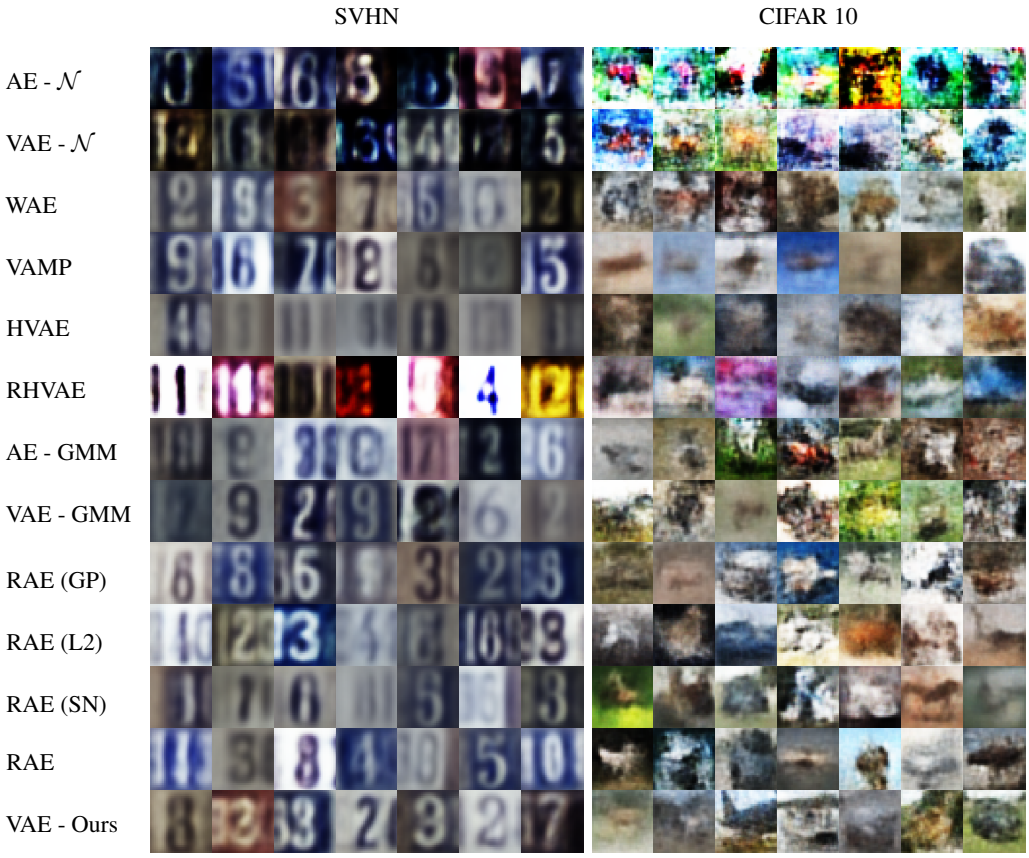


Figure 8: Generated samples with different models and generation processes.

Figure 9: Closest element in the training set (Near.) to the generated one (Gen.) with the proposed method.

C.3 Generation with complex data

Finally, we also propose to stress the proposed generation procedure in a day-to-day scenario where the limited data regime is more than common. To stress the model in such condition, we consider the publicly available OASIS database [39] composed of 416 MRI of patients, 100 of whom were diagnosed with Alzheimer disease (AD). Since both FID and PRD scores are not reliable when no large test set is available, we propose to assess quantitatively the generation quality with a data augmentation task. Hence, we split the dataset into a train (70%), a validation (10%) and a test set (20%). Each model is trained on each label of the train set and used to generate 2k samples per class. Then a CNN classifier is trained on i) the original train set and ii) the 4k generated samples from the generative models and tested on the test set. Table 2 shows classification results averaged across 20 runs for each considered model. The line *raw (resampled)* corresponds to a case where the train set is obtained by balancing the classes with simple repetitions of the samples from the under-represented class. These metrics provide a way to assess i) if the model can generate data adding relevant information for classification and ii) allows to quantify the amount of overfitting. The proposed method is the only one allowing to achieve higher balanced accuracy and F1 scores for both labels than on the original (unbalanced) data meaning that the samples are relevant to the classifier and this is also sign of a good generalization. Moreover, we provide generated samples using each generation procedure in Figure 10. Again, the proposed method appears to produce visually the sharpest samples. However, such augmentation method for medical data requires caution and needs further assessment on the possibly induced biases before being used on a *real-life* application case.

Table 2: Classification results averaged on 20 independent runs. For the VAEs, the classifier is trained on 2K generated samples per class.

Generation method	Balanced Accuracy	F1		Precision		Recall	
		AD	CN	AD	CN		
Original*	66.2 ± 7.6	47.6 ± 15.8	87.3 ± 2.0	74.7 ± 8.4	80.3 ± 4.0	35.7 ± 16.3	95.7 ± 1.5
Original (resampled)	81.8 ± 2.6	72.1 ± 3.6	$\mathbf{88.0 \pm 2.3}$	67.0 ± 5.3	91.4 ± 1.8	78.5 ± 5.2	85.1 ± 4.2
AE - $\mathcal{N}$	50.0 ± 0.0	0.0 ± 0.0	84.1 ± 0.0	0.0 ± 0.0	72.6 ± 0.0	0.0 ± 0.0	100.0 ± 0.0
WAE	57.4 ± 9.7	21.0 ± 24.5	84.4 ± 2.3	48.5 ± 42.8	76.7 ± 6.1	19.3 ± 27.5	95.4 ± 9.3
VAE - $\mathcal{N}$	51.8 ± 3.8	6.1 ± 11.8	84.6 ± 1.1	38.0 ± 47.3	73.4 ± 1.7	3.7 ± 7.8	99.8 ± 0.7
VAMP	83.1 ± 2.6	70.4 ± 3.6	82.2 ± 4.7	56.3 ± 5.2	97.5 ± 2.1	94.8 ± 4.7	71.5 ± 7.4
HVAE	56.3 ± 7.9	19.6 ± 21.7	85.4 ± 1.7	48.7 ± 41.7	75.5 ± 3.8	13.9 ± 17.6	98.6 ± 2.2
RHVAE	68.0 ± 10.9	47.0 ± 24.2	85.1 ± 3.3	56.1 ± 25.3	83.0 ± 7.5	46.7 ± 30.2	89.2 ± 10.6
AE - GMM	82.4 ± 2.3	69.5 ± 3.1	82.0 ± 3.6	55.8 ± 4.9	96.8 ± 2.4	93.3 ± 5.6	71.5 ± 6.2
RAE (GP)	63.9 ± 9.8	46.5 ± 15.9	70.6 ± 19.6	45.3 ± 18.5	84.2 ± 8.6	60.9 ± 28.6	67.0 ± 24.9
RAE (L2)	74.1 ± 6.0	60.6 ± 9.5	82.1 ± 5.9	57.8 ± 10.1	88.3 ± 5.2	70.0 ± 18.7	78.3 ± 11.7
RAE (SN)	62.3 ± 8.9	37.8 ± 22.6	80.1 ± 7.9	43.1 ± 24.9	80.6 ± 6.6	41.7 ± 30.1	82.9 ± 16.4
RAE	69.3 ± 8.1	53.8 ± 12.9	80.0 ± 10.7	56.2 ± 13.5	85.2 ± 6.2	60.0 ± 24.0	78.5 ± 17.5
VAE - GMM	83.0 ± 3.6	71.4 ± 4.3	85.3 ± 3.0	60.7 ± 5.4	94.9 ± 3.7	88.0 ± 9.5	77.9 ± 5.9
VAE - Ours	$\mathbf{85.4 \pm 2.5}$	$\mathbf{74.7 \pm 3.5}$	87.3 ± 2.7	64.0 ± 5.3	95.8 ± 2.2	90.4 ± 5.6	80.3 ± 5.1

*unbalanced

C.4 Wider hyper-parameter search

As stated in the paper, for the experiments, we used the official implementation and hyper-parameters provided by the authors when available. However, we also propose to perform a hyper-parameter search for the models considered in the benchmark *i.e.* WAE, VAMP-VAE, RAE-GP and RAE-L2 [3] on MNIST and CELEBA. Since both HVAE and RHVAE models have a very time consuming training, we propose to replace these approaches with models having the same objective (i.e. enriching the posterior distribution). Do to so we consider a VAE with inverse autoregressive flows [30] (VAE-IAF) and a VAE with normalizing flows with radial/planar invertible transformations [49] (VAE-NF).

We train these models with 10 different hyper-parameter configurations on MNIST and CELEBA. For the WAE, we vary the kernel bandwidth in {0.01, 0.1, 0.5, 1, 2, 5} and change the regularization factor weighting the reconstruction and regularization in {0.01, 0.1, 1, 10, 100}. For the RAEs, we vary the L2 latent code regularization factor and the factor before the explicit regularization in $\{1e^{-6}, 1e^{-4}, 1e^{-3}, 0.01, 0.1, 1\}$. For the VAMP we vary the number of pseudo-inputs in {10, 20, 30, 50, 100, 150, 200, 250, 300,189 500}. Finally, for the flow-based VAEs we vary the complexity of the flows with different number of IAF blocks (VAE-IAF) or different flow lengths (VAE-NF). To assess the influence of the neural architecture, the experiment is performed twice each time with a different neural network architecture (CNN in Table. 4 or a simpler ResNet). In Table. 3, we show the generation vs. test FID of the model achieving the lowest FID on the validation set.

Table 3: FID (lower is better) for different models and datasets. For the mixture of Gaussian (GMM), we fit a 10-component mixture of Gaussian in the latent space.

MODELS	MNIST		CELEBA	
NETS	CNN	RESNET	CNN	RESNET
AE - N(0,1)	46.4	221.8	64.6	275.0
WAE	18.9	20.3	54.6	67.1
VAE - N(0,1)	40.7	47.8	64.1	69.5
VAMP	34.0	34.5	56.0	67.2
VAE-NF	29.3	32.5	55.4	67.1
VAE-IAF	27.5	30.6	56.5	66.2
AE - GMM	9.6	11.0	56.1	57.4
RAE-GP	9.4	11.4	52.5	59.0
RAE-L2	9.1	11.5	54.5	58.3
VAE - GMM	13.1	12.4	55.5	59.9
OURS	**8.5**	**10.7**	**48.7**	**53.2**

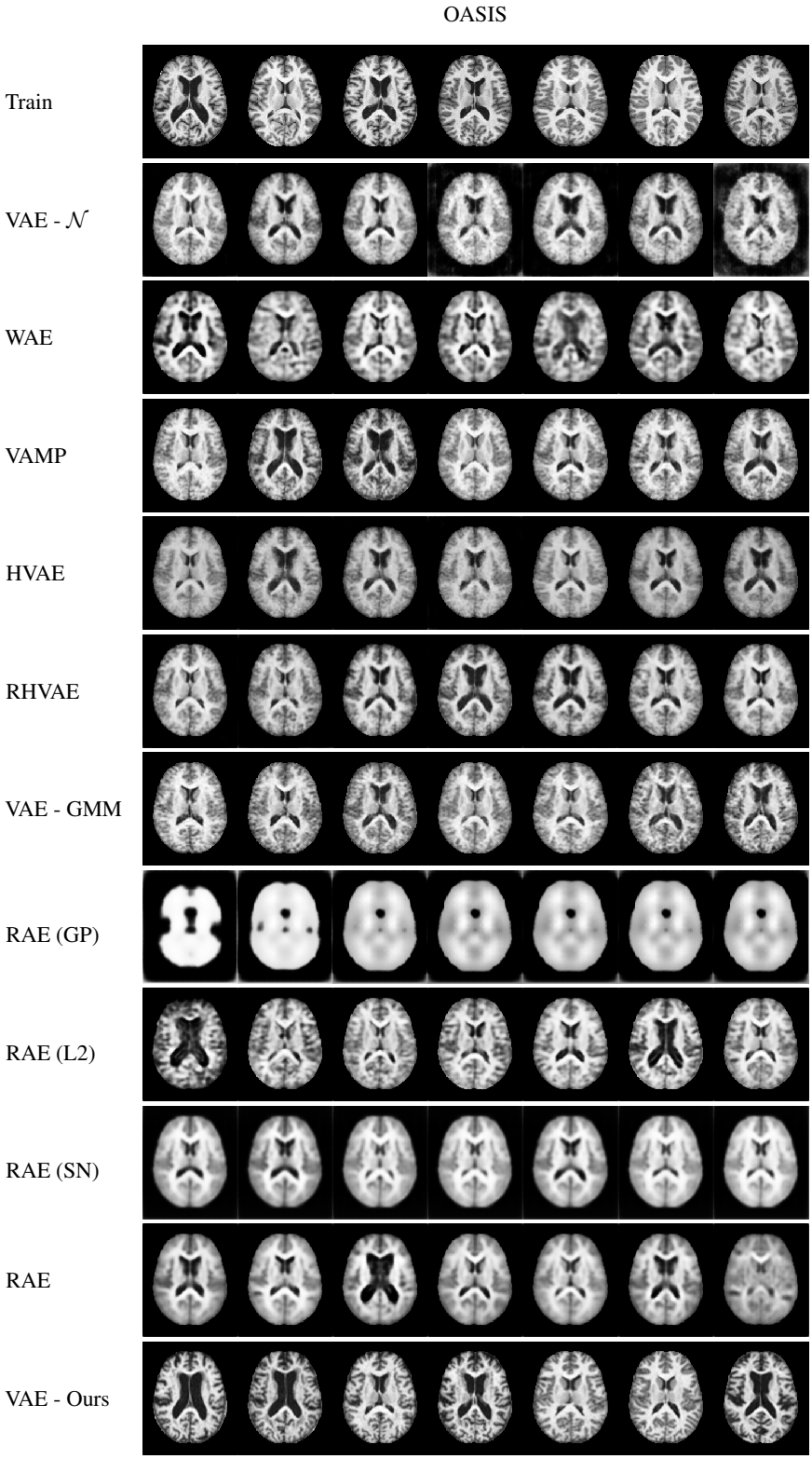


Figure 10: Generated samples with different models and generation processes.

D Experimental set-up

We compare the proposed sampling method to several VAE variants such as a Wasserstein Autoencoder (WAE) [56], Regularized Autoencoders [20] with either L2 decoder's parameters regularization (RAE-L2), gradient penalty (RAE-GP), spectral normalization (RAE-SN) or simple L2 latent code regularization (RAE), a vamp-prior VAE (VAMP) [57], a Hamiltonian VAE (HVAE) [7], a geometry-aware VAE (RHVAE) [9] and an Autoencoder (AE). The RAEs, VAEs and AEs are trained for 100 epochs for SVHN, MNIST[3] and CELEBA and 200 on CIFAR10. Each time we use the official train and test split of the data. For MNIST and SVHN, 10k samples out of the train set are reserved for validation and 40k for CIFAR10. As to CELEBA, we use the official validation set for validation. The model that is kept at the end of training is the one achieving the best validation loss. All the models are trained with a batch size of 100 and starting learning rate of $1e{-}3$ (but CIFAR where the learning rate is set to $5e{-}4$) with an Adam optimizer [28]. We also use a scheduler decreasing the learning rate by half if the validation loss stops increasing for 5 epochs. For the experiments on the sensitivity to the training set size, we keep the same set-up. For each dataset we ensure that the validation set is $1/5^{\text{th}}$ the size of the train set but for CIFAR where we select the best model on the train set. The neural networks architectures can be found in Table 4 and are inspired by [20]. The metrics (FID and PRD scores) are computed with 10000 samples against the test set (for CELEBA we selected only the 10000 first samples of the official test set). The factor ρ is set to $\rho = \max_i \min_{j \neq i} \|c_i - c_j\|_2$ to ensure some *smoothness* of the manifold. For models coming from peers, we use the parameters and code provided by the authors when available and allowed by licenses.

For the data augmentation task, the generative models are trained on each class for 1000 epochs with a batch size of 100 and a starting learning rate of $1e{-}4$. Again a scheduler is used and the learning rate is cut by half if the loss does not improve for 20 epochs. All the models have the autoencoding architecture described in Table 4. As to the classifier, it is trained with a batch size of 200 for 50 epochs with a starting learning rate of $1e{-}4$ and Adam optimizer. A scheduler reducing the learning rate by half every 5 epochs if the validation loss does not improve is again used. The best kept model is the one achieving the best balanced accuracy on the validation set. Its neural network architecture may be found in Table 5. MRIs are only pre-processed such that the maximum value of a voxel is 1 and the minimum 0 for each data point.

[3]MNIST images are re-scaled to 32x32 images with a 0 padding.

Table 4: Neural networks used for the encoder and decoders of VAEs in the benchmarks

	MNIST [CIFAR10]	SVHN	CELEBA	OASIS
ENCODER	(1[3], 32, 32)	(3, 32, 32)	(3, 64, 64)	(1, 208, 176)
LAYER 1	CONV(128, (4, 4), STRIDE=2) BATCH NORMALIZATION RELU	LINEAR(1000) RELU	CONV(128, (5, 5), STRIDE=2) BATCH NORMALIZATION RELU	CONV(64, (5, 5), STRIDE=2) RELU
LAYER 2	CONV(256, (4, 4), STRIDE=2) BATCH NORMALIZATION RELU	LINEAR(500) RELU	CONV(256, (5, 5), STRIDE=2) BATCH NORMALIZATION RELU	CONV(128, (5, 5), STRIDE=2) RELU
LAYER 3	CONV(512, (4, 4), STRIDE=2) BATCH NORMALIZATION RELU	LINEAR(500, 16)	CONV(512, (5, 5), STRIDE=2) BATCH NORMALIZATION RELU	CONV(256, (5, 5), STRIDE=2) RELU
LAYER 4	CONV(1024, (4, 4), STRIDE=2) BATCH NORMALIZATION RELU	-	CONV(1024, (5, 5), STRIDE=2) BATCH NORMALIZATION RELU	CONV(512, (5, 5), STRIDE=2) RELU
LAYER 5	LINEAR(4096, 16)	-	LINEAR(16384, 64)	CONV(1024, (5, 5), STRIDE=2) RELU
LAYER 6	-	-	-	LINEAR(4096, 16)
DECODER	(16 [32])	(16)	(64)	(16)
LAYER 1	LINEAR(65536) RESHAPE(1024, 8, 8)	LINEAR(500) RELU	LINEAR(65536) RESHAPE(1024, 8, 8)	LINEAR(65536) RESHAPE(1024, 8, 8)
LAYER 2	CONVT(512, (4, 4), STRIDE=2) BATCH NORMALIZATION RELU	LINEAR (1000) RELU	CONVT(512, (5, 5), STRIDE=2) BATCH NORMALIZATION RELU	CONVT(512, (5, 5), STRIDE=(3, 2)) RELU
LAYER 3	CONVT(256, (4, 4), STRIDE=2) BATCH NORMALIZATION RELU	LINEAR(3072) RESHAPE(3, 32, 32) SIGMOID	CONVT(256, (5, 5), STRIDE=2) BATCH NORMALIZATION RELU	CONVT(256, (5, 5), STRIDE=2) RELU
LAYER 4	CONVT(3, (4, 4), STRIDE=1) BATCH NORMALIZATION SIGMOID	-	CONVT(128, (5, 5), STRIDE=2) BATCH NORMALIZATION RELU	CONVT(128, (5, 5), STRIDE=2) RELU
LAYER 5	-	-	CONVT(3, (5, 5), STRIDE=1) BATCH NORMALIZATION SIGMOID	CONVT(64, (5, 5), STRIDE=2) RELU
LAYER 6	-	-	-	CONVT(1, (5, 5), STRIDE=1) RELU

Table 5: Neural Network used for the classifier in Sec. C.3

	OASIS CLASSIFIER
INPUT SHAPE	(1, 208, 176)
LAYER 1	CONV(8, (3, 3), STRIDE=1) BATCH NORMALIZATION LEAKYRELU MAXPOOL(2, STRIDE=2)
LAYER 2	CONV(16, (3, 3), STRIDE=1) BATCH NORMALIZATION LEAKYRELU MAXPOOL(2, STRIDE=2)
LAYER 3	CONV(32, (3, 3), STRIDE=2) BATCH NORMALIZATION LEAKYRELU MAXPOOL(2, STRIDE=2)
LAYER 4	CONV(64, (3, 3), STRIDE=2) BATCH NORMALIZATION LEAKYRELU MAXPOOL(2, STRIDE=2)
LAYER 5	LINEAR(256, 100) RELU
LAYER 6	LINEAR(100, 2) SOFTMAX

E Dataset size sensibility on SVHN

In Figure 11, we show the same plot for SVHN as in Sec. 5.2. Again the proposed method appears to be part of the most robust generation procedures to dataset size changes.

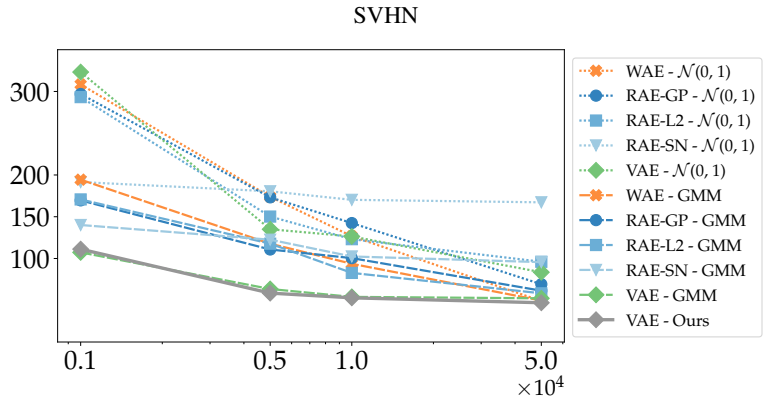


Figure 11: FID score evolution according to the number of training samples.

F Ablation study

F.1 Influence of the number of centroids in the metric

In order to assess the influence of the number of centroids and their choice in the metric in Eq. (7), we show in Figure 12 the evolution of the FID according to the number of centroids in the metric (left) and the variation of FID according to the choice in the centroids (right). As expected, choosing a small number of centroids will increase the value of the FID since it reduces the variability of the generated samples that will remain *close* to the centroids. Nonetheless, as soon as the number of centroids is higher than 1000 the FID score is either competitive or better than peers and continues decreasing as the number of centroids increases.

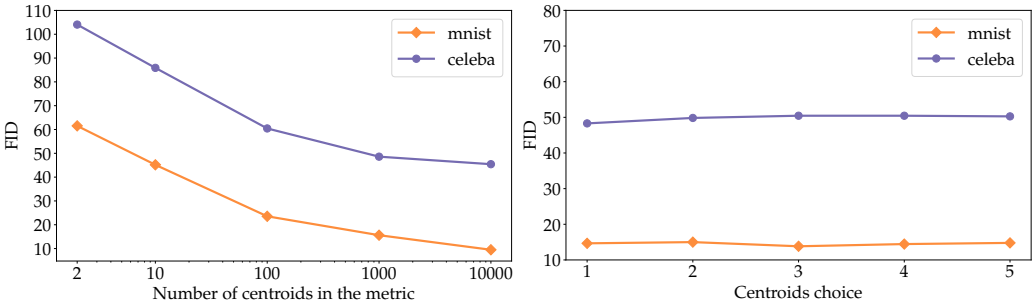


Figure 12: *Left:* FID score evolution according to the number of centroids in the metric (Eq. (7)). *Right*: The FID variation with respect to the choice in centroids. We generate 10000 samples by selecting each time different centroids ($k = 1000$).

To assess the variability of the generated samples, we propose to analyze some generated samples when only 2 centroids are considered. In Figure 13, we display on the left the decoded centroids along with the closest image to these decoded centroids in the train set. On the right are presented some generated samples. We place these samples in the top row if they are closer to the first decoded centroid and in the bottom row otherwise. Interestingly, even with a small number of centroids the proposed sampling scheme is able to access to a relatively good diversity of samples. These samples are not simply resampled train images or a simple interpolation between selected centroids as some of the generated samples have attributes such as glasses that are not present in the images of the decoded centroids.

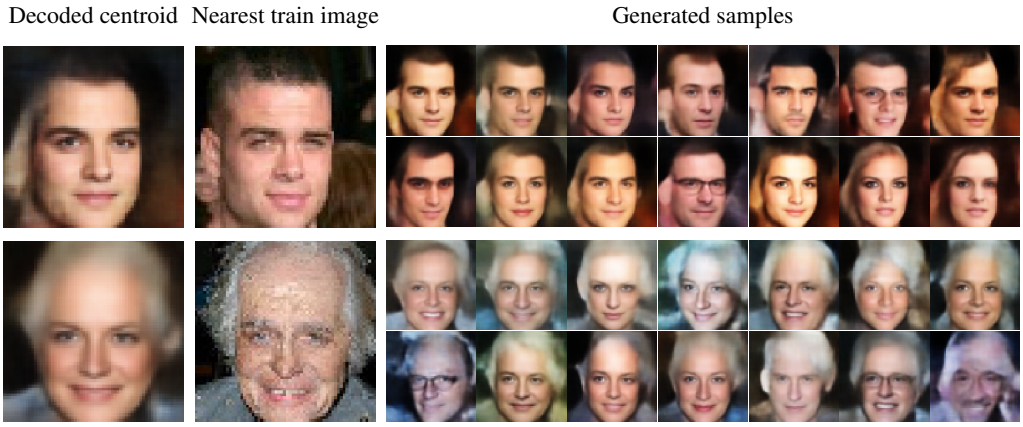


Figure 13: Variability of the generated samples when only two centroids are considered in the metric. *Left:* The image obtained by decoding the centroids. *Middle*: The nearest image in the train set to the decoded centroids. *Right:* Some generated samples. Each generated sample is assigned to the closest decoded centroid (top row for the first centroid and bottom row for the second one).

F.2 Influence of λ in the metric

In this section, we also assess the influence of the regularization factor λ in Eq. (7) on the resulting sampling. To do so, we generate 10k samples using the proposed method on both MNIST and CELEBA datasets for values of $\lambda \in [1e^{-6}, 1e^{-4}, 1e^{-2}, 1e^{-1}, 1]$. Then, we compute the FID against the test set. Each time, we consider $k = 1000$ centroids in the metric. As shown in Figure 14, the influence of λ remains limited. In the implementation, a typical choice for λ is $1e^{-2}$.

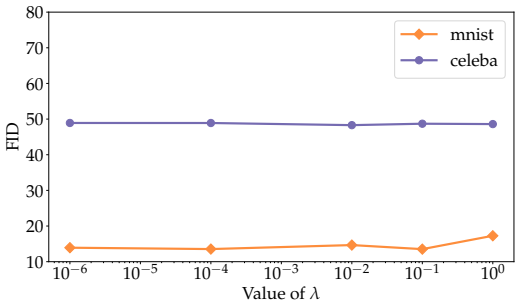


Figure 14: FID score evolution according to the value of λ in the metric (Eq. (7)).

F.3 The choice of ρ

In the experiments presented, the smoothing factor ρ in Eq. (7) is set to the value of the maximum distance between two closest centroids $\rho = \max_i \min_{j \neq i} \|c_j - c_i\|_2$. This choice is motivated by the fact that we wanted to build a smooth metric and so ensure some *smoothness* of the manifold while trying to interpolate faithfully between the metric tensors $\mathbf{G}_i = \mathbf{\Sigma}(x_i)^{-1}$. In particular, a too small value of ρ would have allowed disconnected regions and the sampling may have not prospected well the learned manifold and would have only become a resampling of the centroids. On the other hand, setting a high value for ρ would have biased the interpolation and the value of the metric at a $\mu(x_i)$. As a result, $\mathbf{G}(\mu(x_i))$ might have been very different from the one observed $\mathbf{\Sigma}(x_i)^{-1}$ since the other $\mu(x_j)$ would have had a strong influence on its value. The proposed value for ρ appeared to work well in practice.

G Can the method benefit more recent models ?

Our method proposes to build a Riemannian metric using the covariances in the posterior distributions. Thus, it can be easily plugged into more recent models provided that they have a Gaussian posterior distribution. In order to assess how it would benefit to more recent VAE models, we train a VAMP-VAE [57], a VAEGAN [34], an Adversarial AE [38] and an IWAE [6] and compare the generation FID obtained 1) with the prior or 2) when plugging our method. For this experiment, we conduct a hyper-parameter search consisting in training each model with 10 different configurations. For the VAMP we vary the number of pseudo-inputs in {10, 20, 30, 50, 100, 150, 200, 250, 300, 500}. For the VAEGAN, we use a discriminator similar to the encoder described in Table. 4 and vary the layer depth considered for the reconstruction loss in {2, 3, 4} and the factor balancing reconstruction/generation for the decoder's loss in {0.3, 0.5, 0.7, 0.8, 0.9, 0.99, 0.999}. For the AAE, we change the factor balancing the reconstruction loss and the regularization in {0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99, 0.999}. Finally, for the IWAE, we vary the number of importance samples in {2, 3, 4, 5, 6, 7, 8, 9, 10, 12}. For each model and generation scheme, we report the results of the model achieving the lowest FID on the validation set. According to Table. 6, the proposed generation method seems to benefit these models in almost all cases since the FID decreases when compared to the prior-based generation.

Table 6: FID (lower is better) vs. the test set using either the prior (classic approach) or by plugging our generation method.

MODEL	GENERATION	MNIST	CELEBA
VAMP	PRIOR	34.5	67.2
	OURS	**32.7**	**60.9**
IWAE	PRIOR	**32.4**	67.6
	OURS	33.8	**60.3**
AAE	PRIOR	19.1	64.8
	OURS	**11.7**	**51.4**
VAEGAN	PRIOR	8.7	39.7
	OURS	**6.1**	**31.4**

Another approach that is interesting to compare to is the 2-stage VAE model proposed in [15]. Our method can indeed be seen as part of the methods trying to counterbalance the poor expressiveness of the prior distribution. In [15], the authors argue that the actual distribution of the latent codes (i.e. the aggregated posterior) is "likely not close to a standard Gaussian distribution" [15] leading to a distribution mismatch degrading the generation capability of the model. To address this issue, they propose to use a second VAE to estimate the learned distribution of the latent variables. Our approach starts with the same observation that the latent codes have no reason to follow the prior. However, it differs since we propose to adopt a fully geometric perspective and propose instead a sampling scheme using the intrinsic uniform distribution defined on the learned Riemannian manifold.

We nonetheless compare our method with models obtained with the official implementation provided by the authors of [15] on MNIST and CELEBA. To allow a fair comparison, we simply plug our method to the obtained trained models and build the metric using the posteriors coming from the 1st stage VAE. In Table. 7, we compare the FID obtained 1) with the first stage VAE (*i.e.* prior), 2) with the second stage VAE [15] and 3) with our method. Again, our proposed generation method allows to achieve lower FID results.

Table 7: FID (lower is better) vs. the test set using the 2-stage VAE implementation [15] for either the reconstructed samples (recon.), using the prior (1st stage), using the 2-stage approach (2nd stage) or by plugging our generation method.

DATASET	NETS	RECON.	1st STAGE	2nd STAGE	OURS
MNIST	SIMILAR TO [11]	14.8	20.0	12.9	**9.9**
CELEBA	SIMILAR TO [11]	44.9	67.8	53.3	**49.6**
CELEBA	SIMILAR TO [56]	34.3	70.8	40.7	**37.9**