A Geometric Perspective on Variational Autoencoders

Clément Chadebec and Stéphanie Allassonnière

Université de Paris - INRIA (HeKA team) - INSERM

NeurIPS 2022

November 3, 2022

Overview

1. Variational Autoencoder - The Idea
   - Autoencoder
   - VAE framework
   - Mathematical foundations

2. Toward a Geometric Perspective on VAEs
   - Some Elements of Riemannian Geometry
   - A Geometric view of the Model
   - A new Sampling Scheme
   - Results

3. Some Resources on VAE

Autoencoder

- The objective $\Longrightarrow$ Dimensionnality Reduction

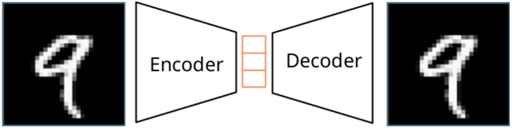


Figure: Simple Autoencoder

- Need for a representation of the image $\Longrightarrow$ vectors

Autoencoder

- The objective $\Longrightarrow$ Dimensionnality Reduction

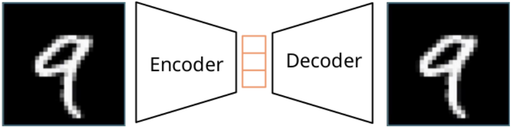


Figure: Simple Autoencoder

- Need for a representation of the image $\Longrightarrow$ vectors

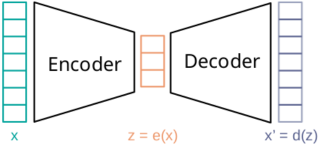


Figure: Simple Autoencoder

AutoEncoder

Assumptions:

- Let $x \in \mathcal{X}$ be a set a data. We assume that there exists $z \in \mathcal{Z}$ such that z is a low dimensional representation of x
- The encoder e_θ and decoder d_ϕ are functions modelled by neural networks (NNs) such that θ and ϕ are the weights of the NNs
- Let x' be the reconstructed samples, the objective is to have $x \simeq x'$

The Objective function writes:

$$\mathcal{L} = \|x - x'\|^2 = \|x - d_\phi(z)\|^2 = \|x - d_\phi(e_\theta(x))\|^2$$

$\Longrightarrow$ The networks are optimised using stochastic gradient descent

$$\phi \leftarrow \phi - \varepsilon \cdot \nabla_\phi \mathcal{L}$$
$$\theta \leftarrow \theta - \varepsilon \cdot \nabla_\theta \mathcal{L}$$

AutoEncoder

Assumptions:

- Let $x \in \mathcal{X}$ be a set a data. We assume that there exists $z \in \mathcal{Z}$ such that z is a low dimensional representation of x
- The encoder e_θ and decoder d_ϕ are functions modelled by neural networks (NNs) such that θ and ϕ are the weights of the NNs
- Let x' be the reconstructed samples, the objective is to have $x \simeq x'$

The Objective function writes:

$$\mathcal{L} = \|x - x'\|^2 = \|x - d_\phi(z)\|^2 = \|x - d_\phi(e_\theta(x))\|^2$$

$\Longrightarrow$ The networks are optimised using stochastic gradient descent

$$\phi \leftarrow \phi - \varepsilon \cdot \nabla_\phi \mathcal{L}$$
$$\theta \leftarrow \theta - \varepsilon \cdot \nabla_\theta \mathcal{L}$$

AutoEncoder

Assumptions:

- Let $x \in \mathcal{X}$ be a set a data. We assume that there exists $z \in \mathcal{Z}$ such that z is a low dimensional representation of x
- The encoder e_θ and decoder d_ϕ are functions modelled by neural networks (NNs) such that θ and ϕ are the weights of the NNs
- Let x' be the reconstructed samples, the objective is to have $x \simeq x'$

The Objective function writes:

$$\mathcal{L} = \|x - x'\|^2 = \|x - d_\phi(z)\|^2 = \|x - d_\phi(e_\theta(x))\|^2$$

$\Longrightarrow$ The networks are optimised using stochastic gradient descent

$$\phi \leftarrow \phi - \varepsilon \cdot \nabla_\phi \mathcal{L}$$
$$\theta \leftarrow \theta - \varepsilon \cdot \nabla_\theta \mathcal{L}$$

AutoEncoder

Assumptions:

- Let $x \in \mathcal{X}$ be a set a data. We assume that there exists $z \in \mathcal{Z}$ such that z is a low dimensional representation of x
- The encoder e_θ and decoder d_ϕ are functions modelled by neural networks (NNs) such that θ and ϕ are the weights of the NNs
- Let x' be the reconstructed samples, the objective is to have $x \simeq x'$

The Objective function writes:

$$\mathcal{L} = \|x - x'\|^2 = \|x - d_\phi(z)\|^2 = \|x - d_\phi(e_\theta(x))\|^2$$

$\Longrightarrow$ The networks are optimised using stochastic gradient descent

$$\phi \leftarrow \phi - \varepsilon \cdot \nabla_\phi \mathcal{L}$$
$$\theta \leftarrow \theta - \varepsilon \cdot \nabla_\theta \mathcal{L}$$

AutoEncoder

Assumptions:

- Let $x \in \mathcal{X}$ be a set a data. We assume that there exists $z \in \mathcal{Z}$ such that z is a low dimensional representation of x
- The encoder e_θ and decoder d_ϕ are functions modelled by neural networks (NNs) such that θ and ϕ are the weights of the NNs
- Let x' be the reconstructed samples, the objective is to have $x \simeq x'$

The Objective function writes:

$$\mathcal{L} = \|x - x'\|^2 = \|x - d_\phi(z)\|^2 = \|x - d_\phi(e_\theta(x))\|^2$$

$\Longrightarrow$ The networks are optimised using stochastic gradient descent

$$\phi \leftarrow \phi - \varepsilon \cdot \nabla_\phi \mathcal{L}$$
$$\theta \leftarrow \theta - \varepsilon \cdot \nabla_\theta \mathcal{L}$$

AutoEncoder - Shortcomings

- How to generate new data ?

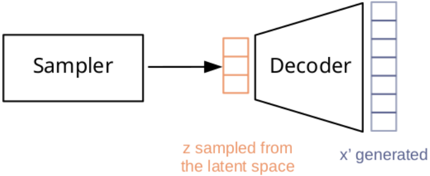


Figure: Generation procedure ?

- How to sample form the latent space?
- The AutoEncoder was just trained to encode and decode the **input data** without information on its structure or distribution.

$\Longrightarrow$ Need for a new framework

AutoEncoder - Shortcomings

- How to generate new data ?

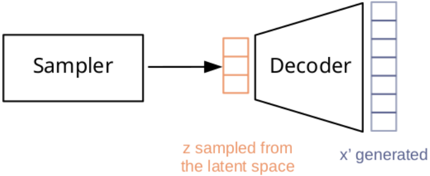


Figure: Generation procedure ?

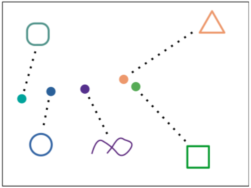


Figure: Potential latent space

- How to sample form the latent space?
- The AutoEncoder was just trained to encode and decode the **input data** without information on its structure or distribution.

$\Longrightarrow$ Need for a new framework

AutoEncoder - Shortcomings

- How to generate new data ?

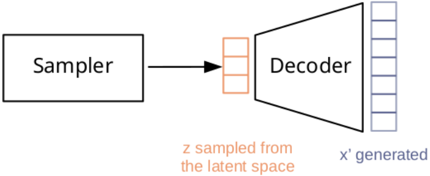


Figure: Generation procedure ?

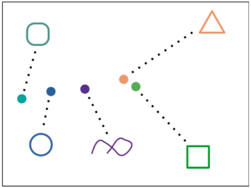


Figure: Potential latent space

- How to sample form the latent space?
- The AutoEncoder was just trained to encode and decode the **input data** without information on its structure or distribution.

$\Longrightarrow$ Need for a new framework

AutoEncoder - Shortcomings

- How to generate new data ?

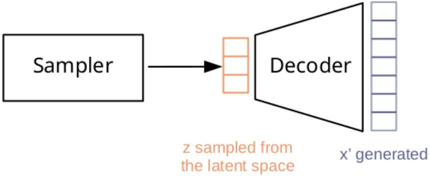


Figure: Generation procedure ?

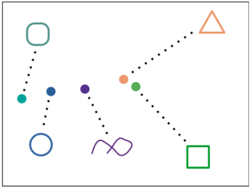


Figure: Potential latent space

- How to sample form the latent space?
- The AutoEncoder was just trained to encode and decode the **input data** without information on its structure or distribution.

$\Longrightarrow$ Need for a new framework

VAE - The Idea

- An autoencoder based model...

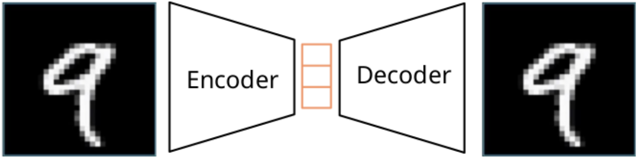


Figure: Simple autoencoder

- ... but where an input data point is encoded as a **distribution** defined over the latent space [2, 4]

VAE - The Idea

- An autoencoder based model...

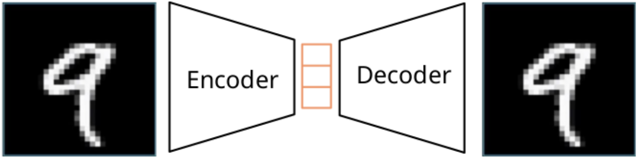


Figure: Simple autoencoder

- ... but where an input data point is encoded as a **distribution** defined over the latent space [2, 4]

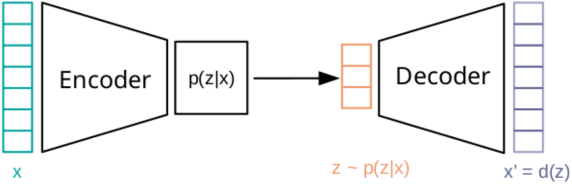


Figure: VAE framework

VAE - Mathematical Considerations

- Let $x \in \mathcal{X}$ be a set of data and $\{P_\theta, \theta \in \Theta\}$ be a parametric model
- We assume there exists latent variables $z \in \mathcal{Z}$ living in a smaller space such that the marginal likelihood writes

$$p_\theta(x) = \int p_\theta(x|z) q_{\text{prior}}(z) dz \,,$$

where q_{prior} is a prior distribution over the latent variables and $p_\theta(x|z)$ is referred to as the decoder

- Example:

$$q_{\text{prior}} = \mathcal{N}(0, I), \qquad p_\theta(x|z) = \prod_{i=1}^{D} \mathcal{B}(\pi_{\theta_i(z)})$$

Objective:

- Maximizing the likelihood of the model

Problem: The integral is often intractable.

VAE - Mathematical Considerations

- Let $x \in \mathcal{X}$ be a set of data and $\{P_\theta, \theta \in \Theta\}$ be a parametric model
- We assume there exists latent variables $z \in \mathcal{Z}$ living in a smaller space such that the marginal likelihood writes

$$p_\theta(x) = \int p_\theta(x|z) q_{\text{prior}}(z) dz \,,$$

where q_{prior} is a prior distribution over the latent variables and $p_\theta(x|z)$ is referred to as the decoder

- Example:

$$q_{\text{prior}} = \mathcal{N}(0, I), \qquad p_\theta(x|z) = \prod_{i=1}^{D} \mathcal{B}(\pi_{\theta_i(z)})$$

Objective:

- Maximizing the likelihood of the model

Problem: The integral is often intractable.

VAE - Mathematical Considerations

- Let $x \in \mathcal{X}$ be a set of data and $\{P_\theta, \theta \in \Theta\}$ be a parametric model
- We assume there exists latent variables $z \in \mathcal{Z}$ living in a smaller space such that the marginal likelihood writes

$$p_\theta(x) = \int p_\theta(x|z) q_{\text{prior}}(z) dz \,,$$

where q_{prior} is a prior distribution over the latent variables and $p_\theta(x|z)$ is referred to as the decoder

- Example:

$$q_{\text{prior}} = \mathcal{N}(0, I), \qquad p_\theta(x|z) = \prod_{i=1}^{D} \mathcal{B}(\pi_{\theta_i(z)})$$

Objective:

- Maximizing the likelihood of the model

Problem: The integral is often intractable.

VAE - Mathematical Considerations

- Let $x \in \mathcal{X}$ be a set of data and $\{P_\theta, \theta \in \Theta\}$ be a parametric model
- We assume there exists latent variables $z \in \mathcal{Z}$ living in a smaller space such that the marginal likelihood writes

$$p_\theta(x) = \int p_\theta(x|z) q_{\text{prior}}(z) dz \, ,$$

where q_{prior} is a prior distribution over the latent variables and $p_\theta(x|z)$ is referred to as the decoder

- Example:

$$q_{\text{prior}} = \mathcal{N}(0, I), \qquad p_\theta(x|z) = \prod_{i=1}^{D} \mathcal{B}(\pi_{\theta_i(z)})$$

Objective:

- Maximizing the likelihood of the model

Problem: The integral is often intractable.

VAE - Mathematical Considerations

- Let $x \in \mathcal{X}$ be a set of data and $\{P_\theta, \theta \in \Theta\}$ be a parametric model
- We assume there exists latent variables $z \in \mathcal{Z}$ living in a smaller space such that the marginal likelihood writes

$$p_\theta(x) = \int p_\theta(x|z) q_{\text{prior}}(z) dz \,,$$

  where q_{prior} is a prior distribution over the latent variables and $p_\theta(x|z)$ is referred to as the decoder
- Example:

$$q_{\text{prior}} = \mathcal{N}(0, I), \qquad p_\theta(x|z) = \prod_{i=1}^{D} \mathcal{B}(\pi_{\theta_i(z)})$$

Objective:

- Maximizing the likelihood of the model

Problem: The integral is often intractable.

Variational inference

We have to use Variational Inference:

$$
\begin{aligned}
\log p_\theta(x) &= \log\left(\int p_\theta(x|z)q_{\text{prior}}(z)dz\right) \\
&= \log\left(\int p_\theta(x,z)dz\right) \\
&= \log\left(\int p_\theta(x,z)\frac{q(z)}{q(z)}dz\right), \text{ for any pdf } q \\
&\geq \int\left(\log\frac{p_\theta(x,z)}{q(z)}\right)q(z)dz, \text{ using Jensen's inequality} \\
&\geq \int\left(\log p_\theta(x,z)\right)q(z)dz - H(q(z))
\end{aligned}
$$

with H the entropy of $q(z)$.

The equality holds for $q(z) = p_\theta(z|x)$.

Variational inference

We have to use Variational Inference:

$$
\begin{aligned}
\log p_\theta(x) &= \log\left(\int p_\theta(x|z) q_{\text{prior}}(z) dz\right) \\
&= \log\left(\int p_\theta(x,z) dz\right) \\
&= \log\left(\int p_\theta(x,z) \frac{q(z)}{q(z)} dz\right), \text{ for any pdf } q \\
&\geq \int \left(\log \frac{p_\theta(x,z)}{q(z)}\right) q(z) dz, \text{ using Jensen's inequality} \\
&\geq \int \left(\log p_\theta(x,z)\right) q(z) dz - H(q(z))
\end{aligned}
$$

with H the entropy of $q(z)$.

The equality holds for $q(z) = p_\theta(z|x)$.

Variational inference

We have to use Variational Inference:

$$
\begin{aligned}
\log p_\theta(x) &= \log\left(\int p_\theta(x|z)q_{\text{prior}}(z)dz\right) \\
&= \log\left(\int p_\theta(x,z)dz\right) \\
&= \log\left(\int p_\theta(x,z)\frac{q(z)}{q(z)}dz\right), \text{ for any pdf } q \\
&\geq \int\left(\log\frac{p_\theta(x,z)}{q(z)}\right)q(z)dz, \text{ using Jensen's inequality} \\
&\geq \int\left(\log p_\theta(x,z)\right)q(z)dz - H(q(z))
\end{aligned}
$$

with H the entropy of $q(z)$.

The equality holds for $q(z) = p_\theta(z|x)$.

Variational inference

We have to use Variational Inference:

$$
\begin{aligned}
\log p_\theta(x) &= \log\left(\int p_\theta(x|z)q_{\text{prior}}(z)dz\right)\\
&= \log\left(\int p_\theta(x,z)dz\right)\\
&= \log\left(\int p_\theta(x,z)\frac{q(z)}{q(z)}dz\right)\,, \text{ for any pdf } q\\
&\geq \int\left(\log\frac{p_\theta(x,z)}{q(z)}\right)q(z)dz\,, \text{ using Jensen's inequality}\\
&\geq \int\left(\log p_\theta(x,z)\right)q(z)dz - H(q(z))
\end{aligned}
$$

with H the entropy of $q(z)$.

The equality holds for $q(z) = p_\theta(z|x)$.

Variational inference

We have to use Variational Inference:

$$
\begin{aligned}
\log p_\theta(x) &= \log\left(\int p_\theta(x|z)q_{\text{prior}}(z)dz\right) \\
&= \log\left(\int p_\theta(x,z)dz\right) \\
&= \log\left(\int p_\theta(x,z)\frac{q(z)}{q(z)}dz\right)\ , \text{ for any pdf } q \\
&\geq \int\left(\log\frac{p_\theta(x,z)}{q(z)}\right)q(z)dz\,, \text{ using Jensen's inequality} \\
&\geq \int\left(\log p_\theta(x,z)\right)q(z)dz - H(q(z))
\end{aligned}
$$

with H the entropy of $q(z)$.

The equality holds for $q(z) = p_\theta(z|x)$.

Variational inference

We have to use Variational Inference:

$$\begin{aligned}
\log p_\theta(x) &= \log\left(\int p_\theta(x|z)q_{\text{prior}}(z)dz\right) \\
&= \log\left(\int p_\theta(x,z)dz\right) \\
&= \log\left(\int p_\theta(x,z)\frac{q(z)}{q(z)}dz\right)\text{ , for any pdf } q \\
&\geq \int\left(\log\frac{p_\theta(x,z)}{q(z)}\right)q(z)dz\text{ , using Jensen's inequality} \\
&\geq \int\left(\log p_\theta(x,z)\right)q(z)dz - H(q(z))
\end{aligned}$$

with H the entropy of $q(z)$.

The equality holds for $q(z) = p_\theta(z|x)$.

Variational inference: The ELBO

- Well-know issue: the posterior $q(z) = p_\theta(z|x)$ is intractable.
  $\longrightarrow$ use Expectation-Maximization algorithms (up to the MCMC-SAEM version)
- **OR** approximate this posterior $\rightarrow$ ELBO
- Introduce a parametric approximation:

$$q_\phi(z|x) \simeq p_\theta(z|x)\,,$$

  where $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$
- This leads to an unbiased estimate of the log-likelihood

$$\widehat{p_\theta}(x) = \frac{p_\theta(x,z)}{q_\phi(z|x)}, \qquad \mathbb{E}_{z\sim q_\phi(z|x)}[\widehat{p_\theta}(x)] = p_\theta(x)\,,$$

- and the definition of the **Evidence Lower Bound** (ELBO):

$$\begin{aligned}\log p_\theta(x) &\geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log(p_\theta(x,z)) - \log(q_\phi(z|x))]\\ &\geq ELBO\end{aligned}$$

Variational inference: The ELBO

- Well-know issue: the posterior $q(z) = p_\theta(z|x)$ is intractable.
  $\longrightarrow$ use Expectation-Maximization algorithms (up to the MCMC-SAEM version)
- **OR** approximate this posterior $\to$ ELBO
- Introduce a parametric approximation:

$$q_\phi(z|x) \simeq p_\theta(z|x)\,,$$

where $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$

- This leads to an unbiased estimate of the log-likelihood

$$\widehat{p_\theta}(x) = \frac{p_\theta(x,z)}{q_\phi(z|x)}, \qquad \mathbb{E}_{z\sim q_\phi(z|x)}[\widehat{p_\theta}(x)] = p_\theta(x)\,,$$

- and the definition of the **Evidence Lower Bound** (ELBO):

$$\begin{aligned}\log p_\theta(x) &\geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log(p_\theta(x,z)) - \log(q_\phi(z|x))]\\ &\geq ELBO\end{aligned}$$

Variational inference: The ELBO

- Well-know issue: the posterior $q(z) = p_\theta(z|x)$ is intractable.
  $\longrightarrow$ use Expectation-Maximization algorithms (up to the MCMC-SAEM version)
- **OR** approximate this posterior $\rightarrow$ ELBO
- Introduce a parametric approximation:

$$q_\phi(z|x) \simeq p_\theta(z|x)\,,$$

  where $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$
- This leads to an unbiased estimate of the log-likelihood

$$\widehat{p_\theta}(x) = \frac{p_\theta(x,z)}{q_\phi(z|x)}, \qquad \mathbb{E}_{z\sim q_\phi(z|x)}[\widehat{p_\theta}(x)] = p_\theta(x)\,,$$

- and the definition of the **Evidence Lower Bound** (ELBO):

$$\begin{aligned}\log p_\theta(x) &\geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log(p_\theta(x,z)) - \log(q_\phi(z|x))] \\ &\geq ELBO\end{aligned}$$

Variational inference: The ELBO

- Well-know issue: the posterior $q(z) = p_\theta(z|x)$ is intractable.
  $\longrightarrow$ use Expectation-Maximization algorithms (up to the MCMC-SAEM version)
- **OR** approximate this posterior $\rightarrow$ ELBO
- Introduce a parametric approximation:

$$q_\phi(z|x) \simeq p_\theta(z|x) \, ,$$

  where $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$
- This leads to an unbiased estimate of the log-likelihood

$$\widehat{p_\theta}(x) = \frac{p_\theta(x,z)}{q_\phi(z|x)}, \qquad \mathbb{E}_{z\sim q_\phi(z|x)}[\widehat{p_\theta}(x)] = p_\theta(x) \, ,$$

- and the definition of the **Evidence Lower Bound** (ELBO):

$$\begin{aligned} \log p_\theta(x) &\geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log(p_\theta(x,z)) - \log(q_\phi(z|x))] \\ &\geq ELBO \end{aligned}$$

Variational inference: The ELBO

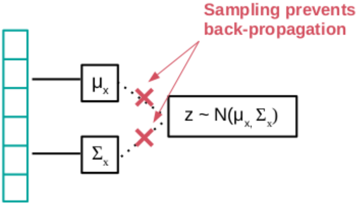
Objective:

1. Optimize the ELBO **as a function** instead of the target distribution

Use stochastic gradient descent in both θ and ϕ

The Reparametrization Trick for stochastic gradient descent

- Since $z \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, the model is not amenable to gradient descent



(a) Back-propagation impossible

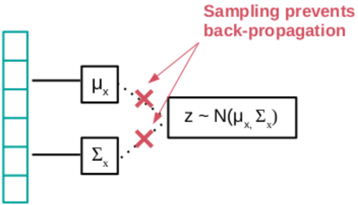
$\Longrightarrow$

$\Longrightarrow$ Optimization with respect to encoder and decoder parameters made possible !

Objective $\Longrightarrow$ OK

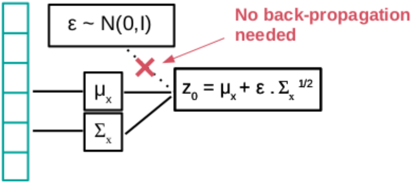
The Reparametrization Trick for stochastic gradient descent

- Since $z \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, the model is not amenable to gradient descent



(a) Back-propagation impossible

$\Longrightarrow$



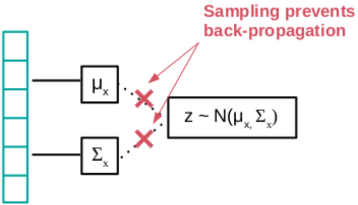
(b) Back-propagation possible

$\Longrightarrow$ Optimization with respect to encoder and decoder parameters made possible !

Objective $\Longrightarrow$ OK

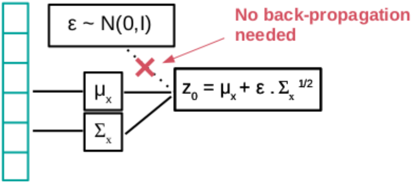
The Reparametrization Trick for stochastic gradient descent

- Since $z \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, the model is not amenable to gradient descent



(a) Back-propagation impossible

$\Longrightarrow$



(b) Back-propagation possible

$\Longrightarrow$ Optimization with respect to encoder and decoder parameters made possible !

Objective $\Longrightarrow$ OK

Generating new samples

- We only need to sample $z \sim \mathcal{N}(0, I)$ and feed it to the decoder.

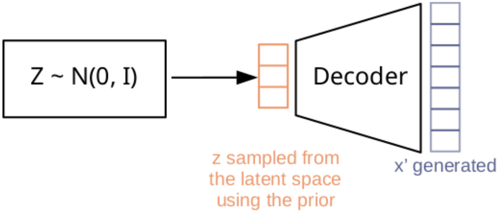


Figure: Generation procedure using prior

Pros:

- Very simple to use in practice

Cons:

- The prior and posterior are not expressive enough to capture complex distributions
- Poor latent space prospecting

Generating new samples

- We only need to sample $z \sim \mathcal{N}(0, I)$ and feed it to the decoder.

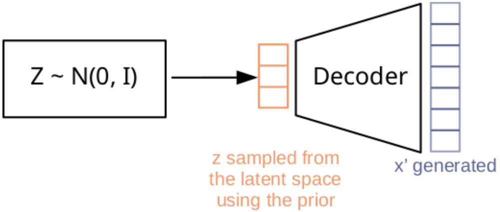


Figure: Generation procedure using prior

Pros:

- Very simple to use in practice

Cons:

- The prior and posterior are not expressive enough to capture complex distributions
- Poor latent space prospecting

Generating new samples

- We only need to sample $z \sim \mathcal{N}(0, I)$ and feed it to the decoder.

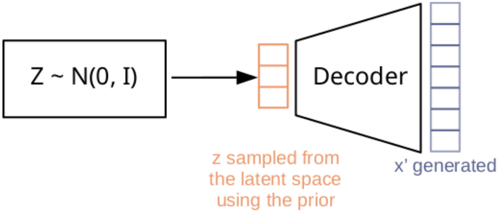


Figure: Generation procedure using prior

Pros:

- Very simple to use in practice

Cons:

- The prior and posterior are not expressive enough to capture complex distributions
- Poor latent space prospecting

Defining a new framework

Assumptions:

- As of now the latent space structure was supposed to be Euclidean (i.e. $\mathcal{Z} = \mathbb{R}^d$)
- Let us now relax this hypothesis and assume that $\mathcal{Z}$ is a Riemannian manifold endowed with a metric $\mathbf{G}$.

Defining a new framework

Assumptions:

- As of now the latent space structure was supposed to be Euclidean (i.e. $\mathcal{Z} = \mathbb{R}^d$)
- Let us now relax this hypothesis and assume that $\mathcal{Z}$ is a Riemannian manifold endowed with a metric $\mathbf{G}$.

Riemannian geometry principles

- Riemannian manifold: (reduced to our model) $\mathbb{R}^d$ endowed with a metric $\mathbf{G}$: $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$.
  $\Longrightarrow \mathbb{R}^d$ not flat anymore, curved space (as montains)
- Geodesic curves:
  - Length of a curve $\gamma : [0,1] \to \mathcal{M}$ from z_1 to z_2 living in a Riemannian manifold $\mathcal{M}$

$$\begin{aligned} L(\gamma) &= \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)}} dt \qquad \gamma(0) = z_1, \gamma(1) = z_2 \\ &= \int_0^1 \sqrt{\gamma'(t)^\top \mathbf{G}(\gamma(t)) \gamma'(t)} dt \,. \end{aligned} \tag{1}$$

  - Geodesic paths = curve γ minimizing Eq. (1)
  - or equivalently minimizing the curve energy

$$E(\gamma) = \int_0^1 \langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)} dt \qquad \gamma(0) = z_1, \gamma(1) = z_2 \,.$$

Riemannian geometry principles

- Riemannian manifold: (reduced to our model) $\mathbb{R}^d$ endowed with a metric $\mathbf{G}$: $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$.
  $\Longrightarrow$ $\mathbb{R}^d$ not flat anymore, curved space (as montains)
- Geodesic curves:
  - Length of a curve $\gamma : [0, 1] \to \mathcal{M}$ from z_1 to z_2 living in a Riemannian manifold $\mathcal{M}$

$$\begin{aligned} L(\gamma) &= \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)}} dt \qquad \gamma(0) = z_1, \gamma(1) = z_2 \\ &= \int_0^1 \sqrt{\gamma'(t)^\top \mathbf{G}(\gamma(t)) \gamma'(t)} dt \,. \end{aligned} \tag{1}$$

  - Geodesic paths = curve γ minimizing Eq. (1)
  - or equivalently minimizing the curve energy

$$E(\gamma) = \int_0^1 \langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)} dt \qquad \gamma(0) = z_1, \gamma(1) = z_2 \,.$$

Riemannian geometry principles

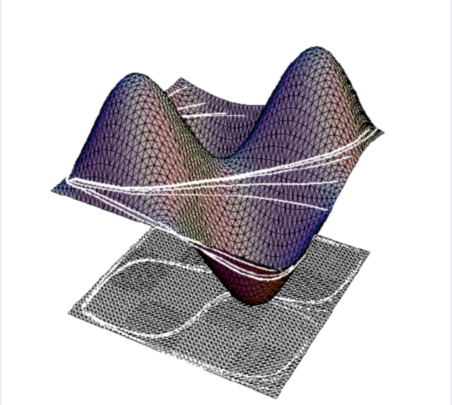
- Riemannian manifold: (reduced to our model) $\mathbb{R}^d$ endowed with a metric $\mathbf{G}$: $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$.
  $\implies \mathbb{R}^d$ not flat anymore, curved space (as montains)
- Geodesic curves:
  - Length of a curve $\gamma : [0,1] \to \mathcal{M}$ from z_1 to z_2 living in a Riemannian manifold $\mathcal{M}$

$$\begin{aligned} L(\gamma) &= \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)}} dt \qquad \gamma(0) = z_1, \gamma(1) = z_2 \\ &= \int_0^1 \sqrt{\gamma'(t)^\top \mathbf{G}(\gamma(t)) \gamma'(t)} dt \,. \end{aligned} \tag{1}$$

  - Geodesic paths = curve γ minimizing Eq. (1)
  - or equivalently minimizing the curve energy

$$E(\gamma) = \int_0^1 \langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)} dt \qquad \gamma(0) = z_1, \gamma(1) = z_2 \,.$$

Riemannian geometry principles



Shortest path geodesic on sinusoidal surface. See Ref. 1 below.

Figure: Image taken from: Fast Marching Methods on Triangulated Domains : Kimmel, R., and Sethian, J.A., Proceedings of the National Academy of Sciences, 95, pp. 8341-8435, 1998

Riemannian Gaussian Distribution

Given a 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 $\mathcal{M}$

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)}dz\,, \tag{2}$$

with dz being the Lebesgue measure.

A Riemannian Gaussian distribution on $\mathcal{M}$ can be defined using this canonical measure and the Riemannian distance.

$$\mathcal{N}_{\text{riem}}(z|\sigma,\mu) = \frac{1}{C}\exp\left(-\frac{\text{dist}_{\mathbf{G}}(z,\mu)^2}{2\sigma}\right). \tag{3}$$

So,

$$\mathcal{N}(z|\mu,\Sigma) = \mathcal{N}_{\text{riem}}(z|\sigma=1,\mu)\,,$$

where $\mathbf{G}$ is the constant Riemannian metric $\mathbf{G}(z) = \mathbf{\Sigma}^{-1}$, $\forall z \in \mathcal{M}$.

Riemannian Gaussian Distribution

Given a 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 $\mathcal{M}$

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)}dz\,, \tag{2}$$

with dz being the Lebesgue measure.

A Riemannian Gaussian distribution on $\mathcal{M}$ can be defined using this canonical measure and the Riemannian distance.

$$\mathcal{N}_{\text{riem}}(z|\sigma,\mu) = \frac{1}{C}\exp\Big(-\frac{\text{dist}_{\mathbf{G}}(z,\mu)^2}{2\sigma}\Big)\,. \tag{3}$$

So,

$$\mathcal{N}(z|\mu,\Sigma) = \mathcal{N}_{\text{riem}}(z|\sigma=1,\mu)\,,$$

where $\mathbf{G}$ is the constant Riemannian metric $\mathbf{G}(z) = \Sigma^{-1}$, $\forall z \in \mathcal{M}$.

Riemannian Gaussian Distribution

Given a 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 $\mathcal{M}$

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)} dz \,, \tag{2}$$

with dz being the Lebesgue measure.

A Riemannian Gaussian distribution on $\mathcal{M}$ can be defined using this canonical measure and the Riemannian distance.

$$\mathcal{N}_{\text{riem}}(z|\sigma, \mu) = \frac{1}{C} \exp\left(- \frac{\text{dist}_{\mathbf{G}}(z, \mu)^2}{2\sigma} \right) . \tag{3}$$

So,

$$\mathcal{N}(z|\mu, \mathbf{\Sigma}) = \mathcal{N}_{\text{riem}}(z|\sigma = 1, \mu) \,,$$

where $\mathbf{G}$ is the constant Riemannian metric $\mathbf{G}(z) = \mathbf{\Sigma}^{-1}$, $\forall z \in \mathcal{M}$.

The Idea

The main idea is to see the posterior $q_\phi(z|x_i) = \mathcal{N}(\mu(x_i), \mathbf{\Sigma}(x_i))$ as a **Riemannian** Gaussian distribution where the **Riemannian** distance is simply the distance with respect to the metric tensor $\mathbf{\Sigma}^{-1}(x_i)$.

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

$\Longrightarrow$ Only defined at $\mu(x_i)$

Inspired from [1], we propose to build a smooth continuous Riemannian metric defined on the entire latent space as follows:

$$\begin{aligned}\mathbf{G}(z) &= \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d\,,\\ \omega_i(z) &= \exp\left(-\frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2}\right),\end{aligned} \tag{4}$$

where $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i))$.

The Idea

The main idea is to see the posterior $q_\phi(z|x_i) = \mathcal{N}(\mu(x_i), \mathbf{\Sigma}(x_i))$ as a **Riemannian** Gaussian distribution where the **Riemannian** distance is simply the distance with respect to the metric tensor $\mathbf{\Sigma}^{-1}(x_i)$.

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

$\Longrightarrow$ Only defined at $\mu(x_i)$

Inspired from [1], we propose to build a smooth continuous Riemannian metric defined on the entire latent space as follows:

$$\begin{aligned}\mathbf{G}(z) &= \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d\,,\\ \omega_i(z) &= \exp\left(-\frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2}\right),\end{aligned} \tag{4}$$

where $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i))$.

The Idea

The main idea is to see the posterior $q_\phi(z|x_i) = \mathcal{N}(\mu(x_i), \mathbf{\Sigma}(x_i))$ as a **Riemannian** Gaussian distribution where the **Riemannian** distance is simply the distance with respect to the metric tensor $\mathbf{\Sigma}^{-1}(x_i)$.

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

$\Longrightarrow$ Only defined at $\mu(x_i)$

Inspired from [1], we propose to build a smooth continuous Riemannian metric defined on the entire latent space as follows:

$$\begin{aligned}\mathbf{G}(z) &= \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d\,,\\ \omega_i(z) &= \exp\left(- \frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2} \right),\end{aligned} \tag{4}$$

where $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i))$.

Algorithm to Build the Metric

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 $\mathbf{\Sigma}_i$

$\rho \leftarrow \max\limits_{i} \min\limits_{j \neq i} \|c_i - c_j\|_2$ ▷ Set ρ to the max distance between two closest neighbors

Build the metric using Eq. (17)

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

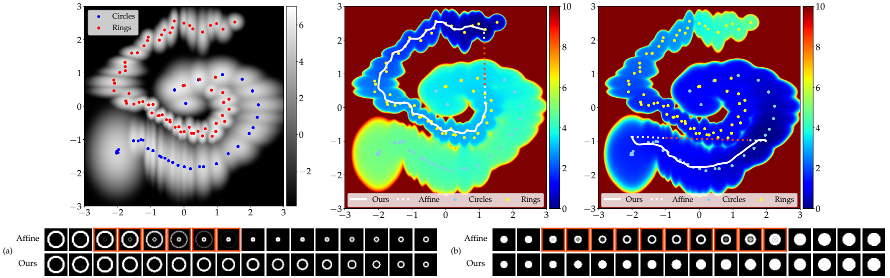
Return G ▷ Return **G** as a function

Building the metric from a trained model

A Riemannian Latent Space

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.

Learned latent space (a) Distance map (b) Distance Maps



New Sampling Procedure

Sampling for the intrinsic uniform Riemannian distribution Since the volume of the whole manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ is finite, we can now define a *uniform distribution* on $\mathcal{M}$

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

Since the Riemannian metric has a closed form expression sampling from this distribution is quite easy and may be performed using the HMC sampler [3].

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

New Sampling Procedure

Sampling for the intrinsic uniform Riemannian distribution Since the volume of the whole manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ is finite, we can now define a *uniform distribution* on $\mathcal{M}$

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

Since the Riemannian metric has a closed form expression sampling from this distribution is quite easy and may be performed using the HMC sampler [3].

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

New Sampling Procedure

Sampling for the intrinsic uniform Riemannian distribution Since the volume of the whole manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ is finite, we can now define a *uniform distribution* on $\mathcal{M}$

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

Since the Riemannian metric has a closed form expression sampling from this distribution is quite easy and may be performed using the HMC sampler [3].

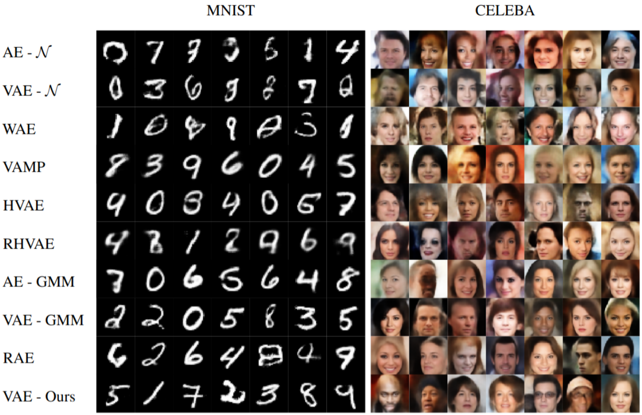
$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d \,,$$

Generation results

Model	MNIST (16) FID ↓	MNIST (16) PRD ↑	SVHN (16) FID ↓	SVHN (16) PRD ↑	CIFAR 10 (32) FID ↓	CIFAR 10 (32) PRD ↑	CELEBA (64) FID ↓	CELEBA (64) 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**

Generation results

Generation results



Generation samples

Generation results - Sampling Diversity

Recall the shape of the metric:

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

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

where $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i))$.

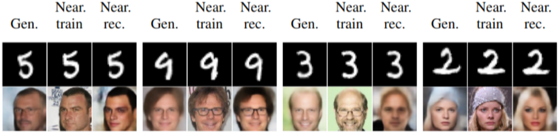
Generation results - Sampling Diversity

Recall the shape of the metric:

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

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

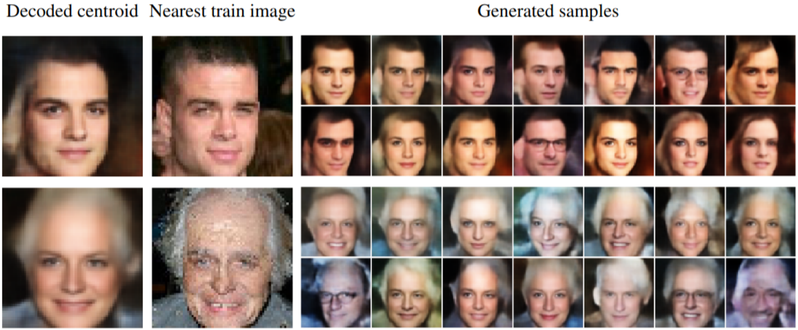
where $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i))$.



reconstruction vs. generation		
	MNIST	CELEBA
FID	11.27	30.12

Sampling diversity

Generation results - Sampling Diversity

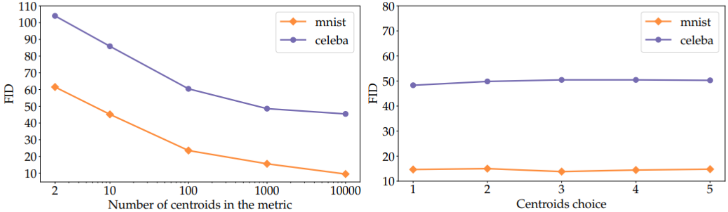


Case with 2 centroids

Generation results - Influence of the number of centroids

Recall the shape of the metric:

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

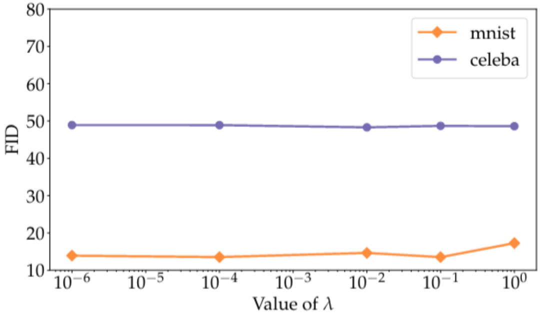


Influence of the centroids

Generation results - Influence of λ

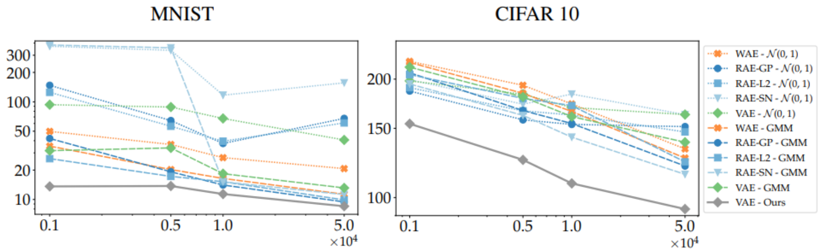
Recall the shape of the metric:

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



Influence of λ

Generation results - Influence of the Number of Training Samples



Influence of the number of training samples (FID)

Can the method benefit more recent models

Can the method be applied to more recent models and benefit them?

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**

Method applied to more recent models

Interested in VAEs ?

Check out Pythae, a Python library that unifies Generative Autoencoder implementations in Python.

pypl package 0.0.8 | python 3.7|3.8|3.9+ | docs passing | license Apache-2.0
code style black | codecov 98% | Open in Colab

Documentation

pythae

This library implements some of the most common (Variational) Autoencoder models under a unified implementation. In particular, it provides the possibility to perform benchmark experiments and comparisons by training the models with the same autoencoding neural network architecture. The feature *make your own autoencoder* allows you to train any of these models with your own data and own Encoder and Decoder neural networks. It integrates experiment monitoring tools such wandb and mlflow 🥄 and allows model sharing and loading from the HuggingFace Hub 🤗 in a few lines of code.

Quick access:

- Installation
- Implemented models / Implemented samplers
- Reproducibility statement / Results flavor
- Model training / Data generation / Custom network architectures
- Model sharing with 🤗 Hub / Experiment tracking with `wandb` / Experiment tracking with `mlflow`
- Tutorials / Documentation
- Contributing 🚀 / Issues 🛠️
- Citing this repository

Interested in VAEs ?

GAE Model	Pythae model
Autoencoder	AE
Variational Autoencoder	VAE
Beta Variational Autoencoder	BetaVAE
VAE with Linear Normalizing Flows	VAE_LinNF
VAE with Inverse Autoregressive Flows	VAE_IAF
Disentangled β-VAE	DisentangledBetaVAE
Disentangling by Factorising	FactorVAE
Beta-TC-VAE	BetaTCVAE
Importance Weighted Autoencoder	IWAE
Multiply Importance Weighted Autoencoder	MIWAE
Partially Importance Weighted Autoencoder	PIWAE
Combination Importance Weighted Autoencoder	CIWAE
VAE with perceptual metric similarity	MSSSIM_VAE
Wasserstein Autoencoder	WAE
Info Variational Autoencoder	INFOVAE_MMD
VAMP Autoencoder	VAMP
Hyperspherical VAE	SVAE
Poincaré Disk VAE	PoicaréVAE
Adversarial Autoencoder	Adversarial_AE
Variational Autoencoder GAN	VAEGAN
Vector Quantized VAE	VQVAE
Hamiltonian VAE	HVAE
Regularized AE with L2 decoder param	RAE_L2
Regularized AE with gradient penalty	RAE_GP
Riemannian Hamiltonian VAE	RHVAE

Pythae - Resources

- ✓ Github: `https://github.com/clementchadebec/benchmark_VAE`
- ✓ Online documentation: `https://pythae.readthedocs.io/en/latest/`
- ✓ Pypi project page: `https://pypi.org/project/pythae/`
- ✓ Open to contributors!

```
pip install pythae
```

Thank you

Thank you!

Code of the paper:
https://github.com/clementchadebec/geometric_perspective_on_vaes

Code for Pythae: https://github.com/clementchadebec/benchmark_VAE

Bibliography

[1] 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`.

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

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

[4] 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.