

Introduction to dimension reduction methods from PCA to VAEs and their applications.

Stéphanie Allassonnière

Université Paris Cité - INRIA HeKA - INSERM

Contributions here are extracted from joined work with Laurent Younes, **Clément Chadebec** and
applications conducted with Ninon Burgos & Elina Thibeau-Sutre

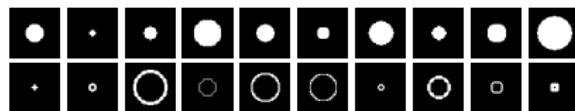


Overview

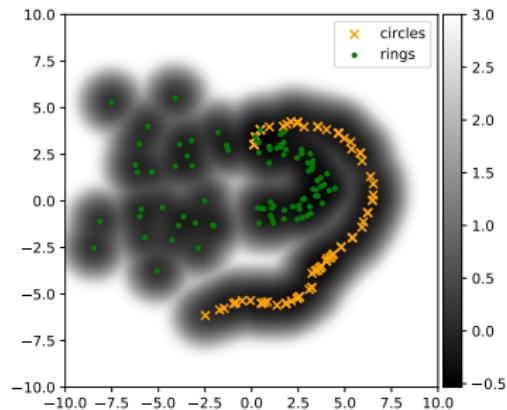
- 1 Introduction
- 2 PCA
- 3 ICA general model
 - General hierarchical model
 - Experiments
- 4 Variational Auto-Encoder - The Idea
 - Auto-Encoder
- 5 VAE framework
 - The idea
 - Mathematical foundations
 - Changing the model
 - Tweaking the approximate posterior distribution
- 6 Toward a Geometry-Aware VAE
 - The framework
 - The proposed model
 - A new way to generate data
 - Sensitivities and robustness on toy data
- 7 Results on Neuroimaging data

Why dimension reduction:

Training samples:

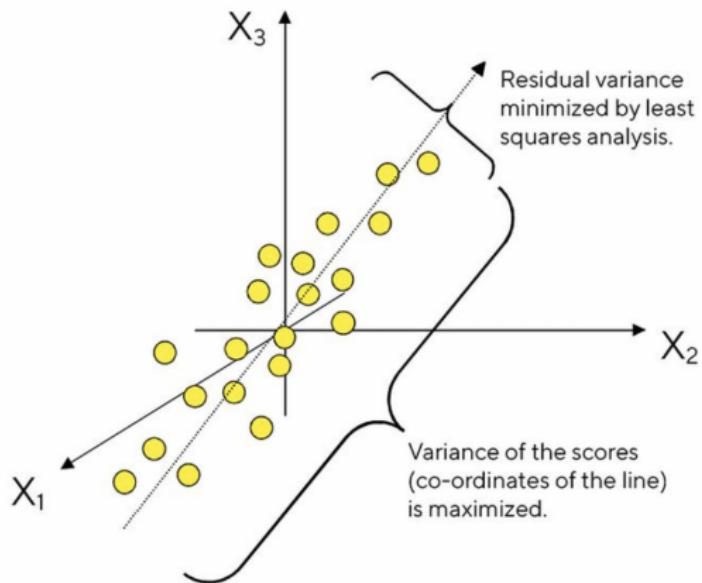


Smaller dimension representation:



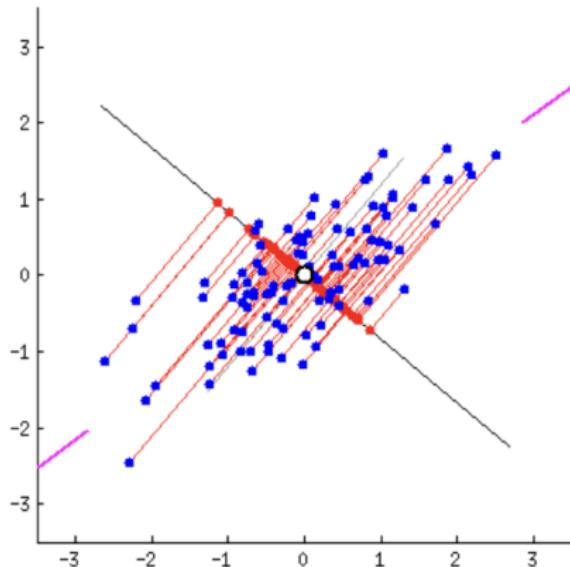
Principal Component Analysis (PCA)

PCA creates a visualization of data that minimizes residual variance in the least squares sense and maximizes the variance of the projection coordinates.



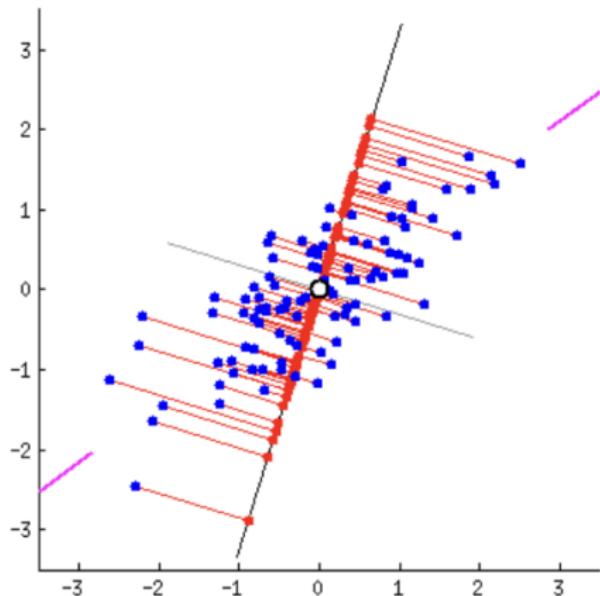
Principal Component Analysis (PCA)

PCA creates a visualization of data that minimizes residual variance in the least squares sense and maximizes the variance of the projection coordinates.



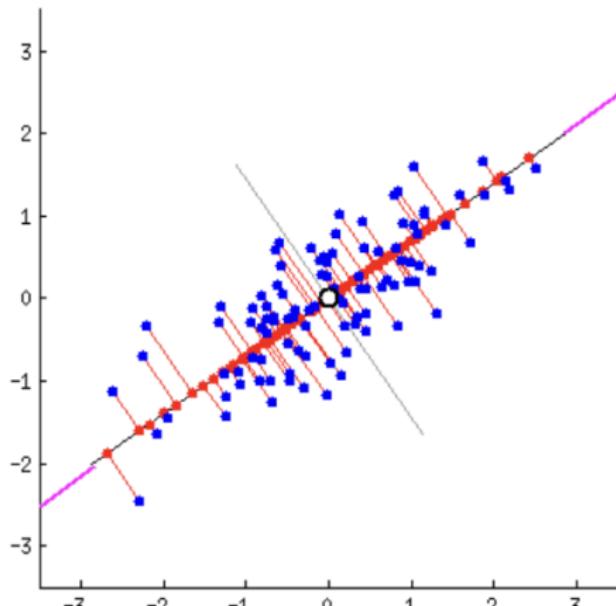
Principal Component Analysis (PCA)

PCA creates a visualization of data that minimizes residual variance in the least squares sense and maximizes the variance of the projection coordinates.



Principal Component Analysis (PCA)

PCA creates a visualization of data that minimizes residual variance in the least squares sense and maximizes the variance of the projection coordinates.



Principal Component Analysis (PCA)

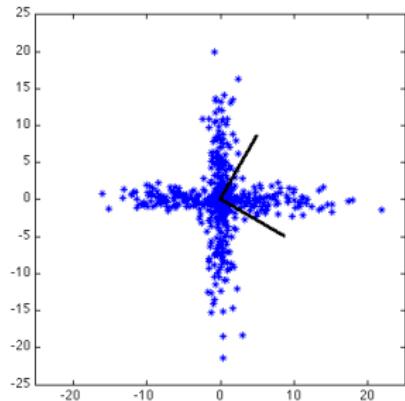
- Geometrical interpretation **Orthogonal** direction of maximum variance:

Principal Component Analysis (PCA)

- Geometrical interpretation **Orthogonal** direction of maximum variance:
- Probabilistic interpretation Assumes a **Gaussian** distribution: $X = \mu + \Sigma^{1/2}\varepsilon$, $\varepsilon \sim \mathcal{N}(0, Id)$.

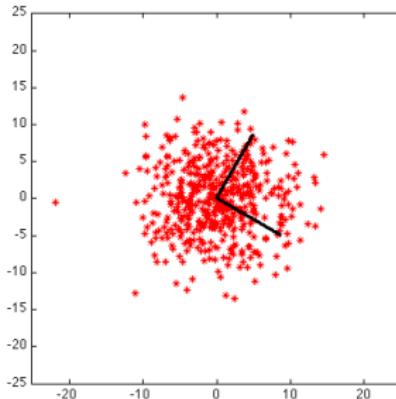
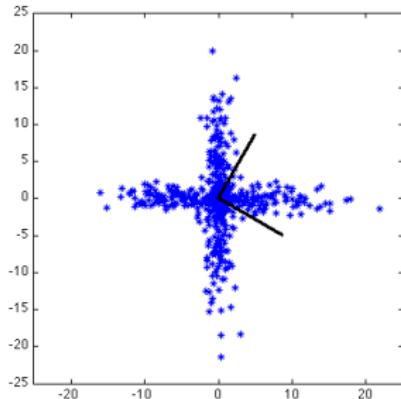
Principal Component Analysis (PCA)

- Geometrical interpretation **Orthogonal** direction of maximum variance:
- Probabilistic interpretation Assumes a **Gaussian** distribution: $X = \mu + \Sigma^{1/2}\varepsilon$, $\varepsilon \sim \mathcal{N}(0, Id)$.
- Issue: Finds here the two orthogonal axis for which the cloud is the most spread (black lines)



Principal Component Analysis (PCA)

- Geometrical interpretation **Orthogonal** direction of maximum variance:
- Probabilistic interpretation Assumes a **Gaussian** distribution: $X = \mu + \Sigma^{1/2}\varepsilon$, $\varepsilon \sim \mathcal{N}(0, Id)$.
- Issue: Finds here the two orthogonal axis for which the cloud is the most spread (black lines)
- W.r.t PCA **the two point clouds are equivalent**



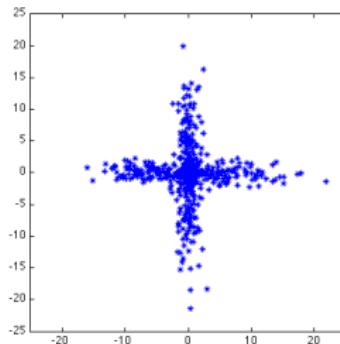
Principal Component Analysis (PCA)

PCA decomposition:

- Pros
 - Easy decomposition in practice: only requires finding the eigen vectors of the empirical covariance matrix
 - Interpretable
 - Visually understandable

- Cons
 - Too simple model which reduces the interpretation

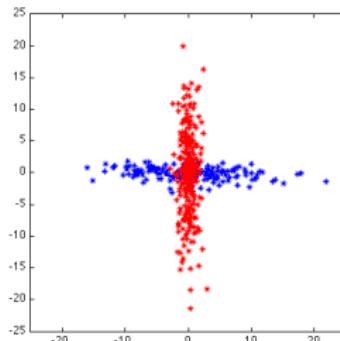
Independant Component Analysis:



Data point cloud

- Data: one point cloud
- Goal: explain these data

Independant Component Analysis:

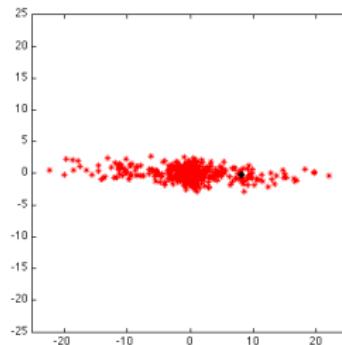


*Data point cloud
Mixture of two Gaussian distributions*

- Data: one point cloud
- Goal: explain these data
- How: Extracting the **sources** which had generated the data

Independent Component Analysis (ICA)

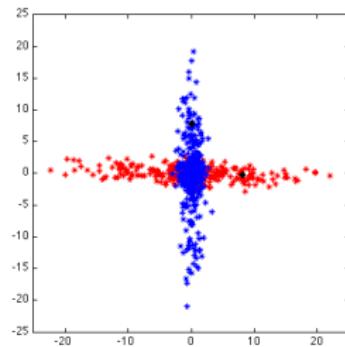
- **Interpretation** of the data (\neq Description)
- Finds **sources** which may have generated the cloud
- Accounts for **non Gaussian** distributions (more flexible model)



First estimated source

Independent Component Analysis (ICA)

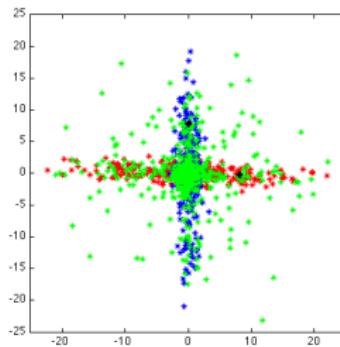
- **Interpretation** of the data (\neq Description)
- Finds **sources** which may have generated the cloud
- Accounts for **non Gaussian** distributions (more flexible model)



First estimated source
Second estimated source

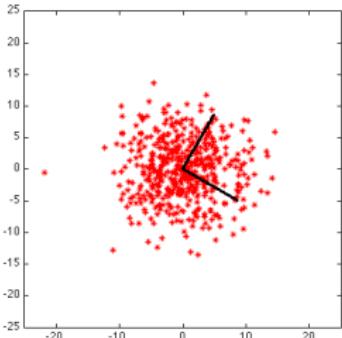
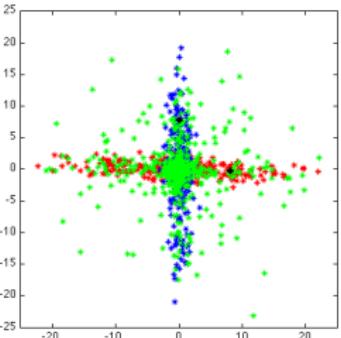
Independent Component Analysis (ICA)

- **Interpretation** of the data (\neq Description)
- Finds **sources** which may have generated the cloud
- Accounts for **non Gaussian** distributions (more flexible model)



First estimated source
Second estimated source
Resampling from the model

Difference between PCA and ICA

PCA	ICA
Maximum variance axes	Source separation
Geometrical (orthogonal axis)	Statistical (source points in the plane)
Description of data	Explanation and interpretation
Gaussian distribution only	Many other possible distributions e.g. mixtures, continuous or discrete (see later)
	

General hierarchical model:

$$X_i = A\beta_i + \sigma\varepsilon_i$$

- * Observations: X_1^n in $(\mathbb{R}^d)^n$
- * Source matrix: A called decomposition matrix
- * Gaussian noise: $\sigma\varepsilon_i$
- * Independent components: $\beta_i \in \mathbb{R}^p$, $p \ll d$
→ random vector with **independent** coordinates
 β_1^n **hidden variables**.

General hierarchical model:

- $X_i = A\beta_i + \sigma\varepsilon_i$
- * Observations: X_1^n in $(\mathbb{R}^d)^n$
 - * Source matrix: A called decomposition matrix
 - * Gaussian noise: $\sigma\varepsilon_i$
 - * Independent components: $\beta_i \in \mathbb{R}^p$, $p \ll d$
 \rightarrow random vector with **independent** coordinates β_1^n **hidden variables**.
 - Model: for all images X_i , $1 \leq i \leq n$

$$\begin{cases} \beta_{i,j} & \sim \nu_\eta \mid \eta, \forall 1 \leq j \leq p \\ X_i & \sim \mathcal{N}(A\beta_i, \sigma^2 Id) \mid A, \sigma^2, \beta_i. \end{cases}$$

General hierarchical model:

$$X_i = A\beta_i + \sigma\varepsilon_i$$

- * Observations: X_1^n in $(\mathbb{R}^d)^n$
- * Source matrix: A called decomposition matrix
- * Gaussian noise: $\sigma\varepsilon_i$
- * Independent components: $\beta_i \in \mathbb{R}^p$, $p \ll d$
 \rightarrow random vector with **independent** coordinates
 β_1^n **hidden variables**.

- Model: for all images X_i , $1 \leq i \leq n$

$$\begin{cases} \beta_{i,j} & \sim \nu_\eta \mid \eta, \forall 1 \leq j \leq p \\ X_i & \sim \mathcal{N}(A\beta_i, \sigma^2 Id) \mid A, \sigma^2, \beta_i. \end{cases}$$

- **Various choices of the distribution ν_η on the independent components**

Various examples of distributions:

Independent Factor analysis

$$\begin{cases} \beta_{i,j} & \sim \nu_{\eta} \mid \eta, \forall 1 \leq j \leq p \\ X_i & \sim \mathcal{N}(A\beta_i, \sigma^2 Id) \mid A, \sigma^2, \beta_i \end{cases}$$

- For identifiability, ν_{η} cannot be Gaussian
- ν_{η} is a **mixture of K 1D Gaussian distributions** $\mathcal{N}(m_k, 1)$, $k = 1, \dots, K$ with weights $(w_k)_{1 \leq k \leq K}$.
- $\eta = (m_k, w_k)_{1 \leq k \leq K}$
- $\theta = (A, \sigma^2, (m_k, w_k)_{1 \leq k \leq K})$

Various examples of distributions:

Continuous distributions

$$\begin{cases} \beta_{i,j} & \sim \nu_\eta \mid \eta, \forall 1 \leq j \leq p \\ X_i & \sim \mathcal{N}(A\beta_i, \sigma^2 Id) \mid A, \sigma^2, \beta_i \end{cases}$$

- ν_η is either:
 - Logistic $\text{Log}(1/2)$,
 - Laplacian,
 - Exponentially scaled Gaussian(EG): $\beta_i^j = s_i^j Y_i^j$ where $\mathbf{Y} \sim \mathcal{N}(\mu, Id)$ and $\mu = (\mu, \dots, \mu)$; s_i^1, \dots, s_i^p are independent $\text{Exp}(1)$, also independent from \mathbf{Y} (sub-exponential tail)
- $\eta = \emptyset$ or μ
- $\theta = (A, \sigma^2)$ or $\theta = (A, \sigma^2, \mu)$

Various examples of distributions:

Discrete distributions

$$\begin{cases} \beta_{i,j} & \sim \nu_\eta \mid \eta, \forall 1 \leq j \leq p \\ X_i & \sim \mathcal{N}(A\beta_i, \sigma^2 Id) \mid A, \sigma^2, \beta_i \end{cases}$$

Idea: Introduce a switch to cancel some of the decomposition vectors
→ Either binary ("on/off") or ternary (activate, inhibit, remove)

Various examples of distributions:

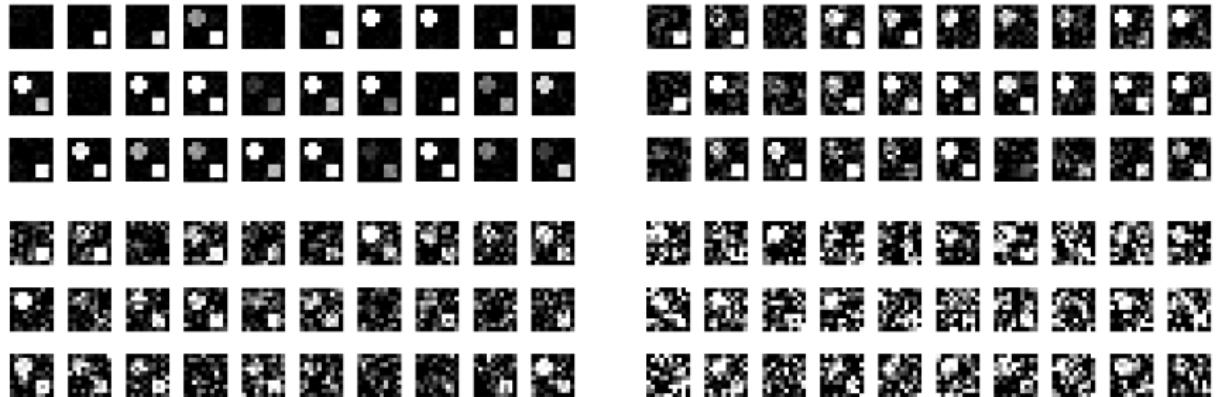
Discrete distributions

$$\begin{cases} \beta_{i,j} & \sim \nu_\eta \mid \eta, \forall 1 \leq j \leq p \\ X_i & \sim \mathcal{N}(A\beta_i, \sigma^2 Id) \mid A, \sigma^2, \beta_i \end{cases}$$

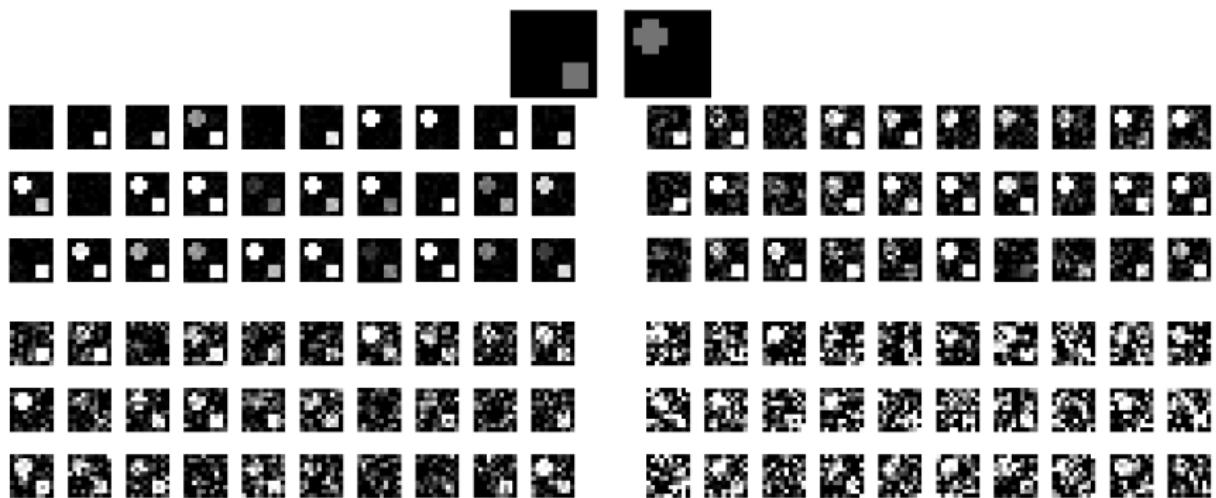
Idea: Introduce a switch to cancel some of the decomposition vectors

→ Either binary ("on/off") or ternary (activate, inhibit, remove)

- **Bernoulli-censored Gaussian (BG):** $\beta^j = b^j Y^j$ with $b^j \sim \mathcal{B}(\alpha)$, Y is a Gaussian vector with distribution $\mathcal{N}(\mu, Id)$.
- **Exponentially scaled Bernoulli-censored Gaussian (EBG):** mix of EG and BG
- **Exponentially-scaled ternary distribution (ET):** $\beta^j = s^j Y^j$, where s^1, \dots, s^p are i.i.d. $\mathcal{Exp}(1)$. $\gamma = P(Y^j = -1) = P(Y^j = 1)$, providing a symmetric distribution for the components of Y .
- $\theta = (A, \sigma^2, \mu, \alpha, \gamma)$

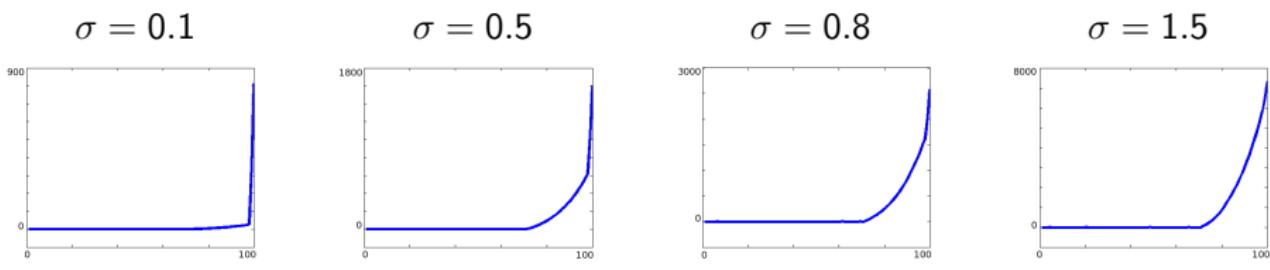


Samples of the four training sets different level of noise. From left to right and top to bottom: $\sigma = 0.1, 0.5, 0.8, 1.5$



Samples of the four training sets different level of noise. From left to right and top to bottom: $\sigma = 0.1, 0.5, 0.8, 1.5$

Results of the PCA decomposition

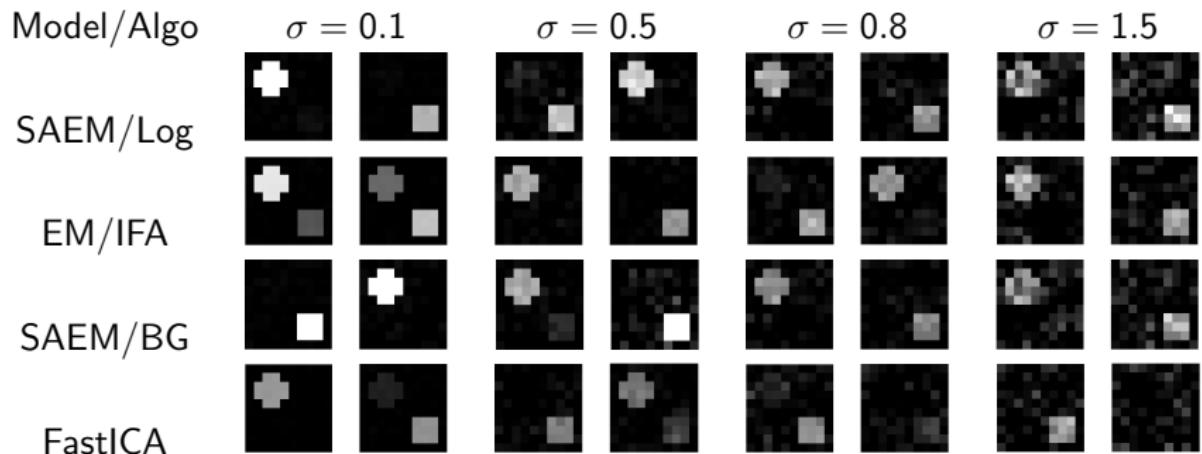


Cumulative eigen values of the PCA decomposition

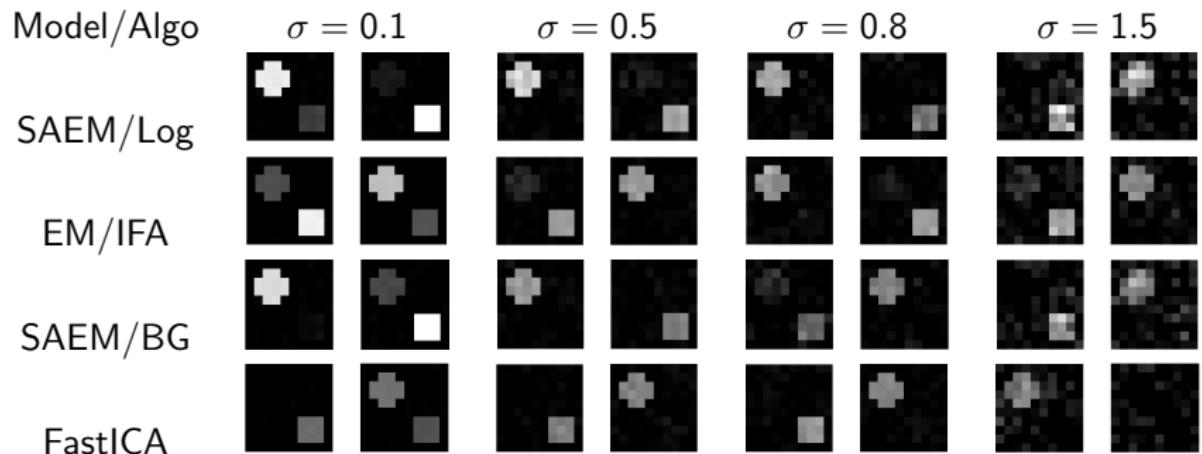


Two first Principal Components (orthogonal images).

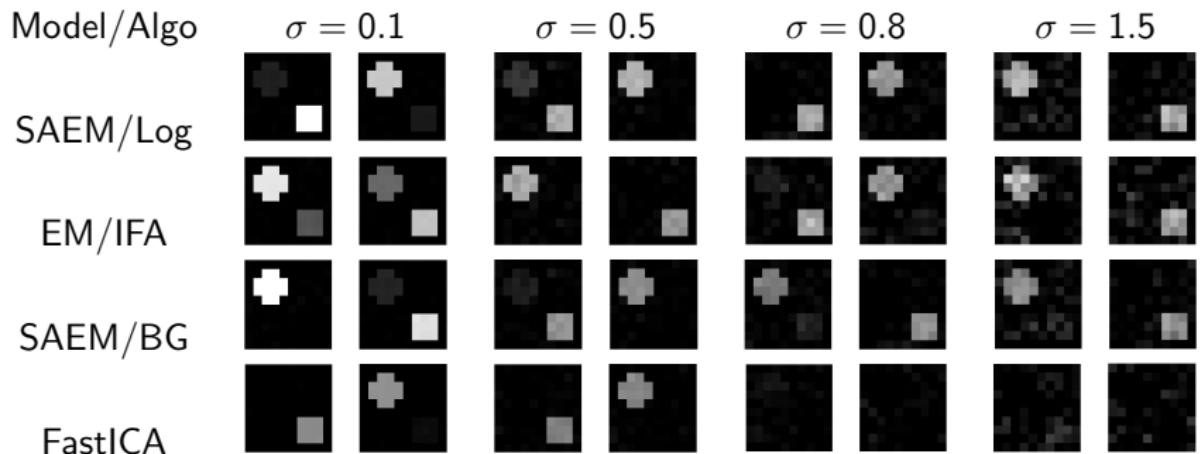
Comparison: 30 images per training set



Comparison: 50 images per training set



Comparison: 100 images per training set

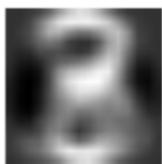
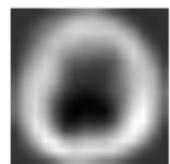
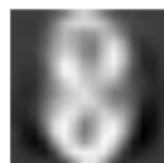
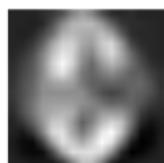
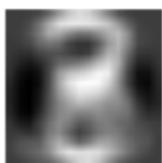
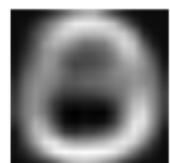


Handwritten digits from the USPS database



Examples from the training sample

Handwritten digits from the USPS database

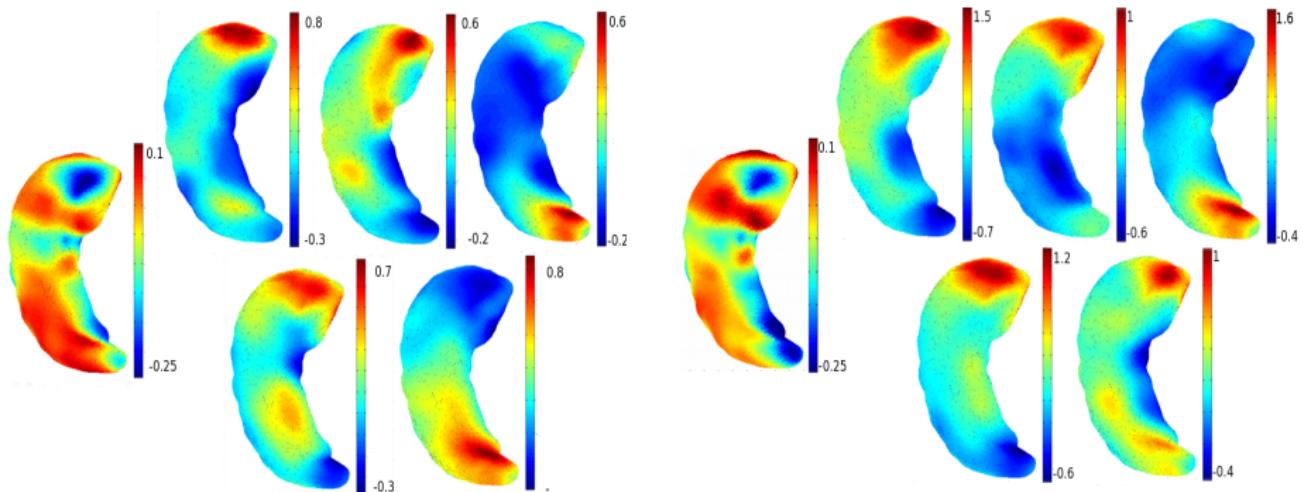


(a) SAEM

(b) tempering SAEM

IFA estimated sources

101 hippocampus deformations (3 populations: Ctrl - Mild AD - AD)



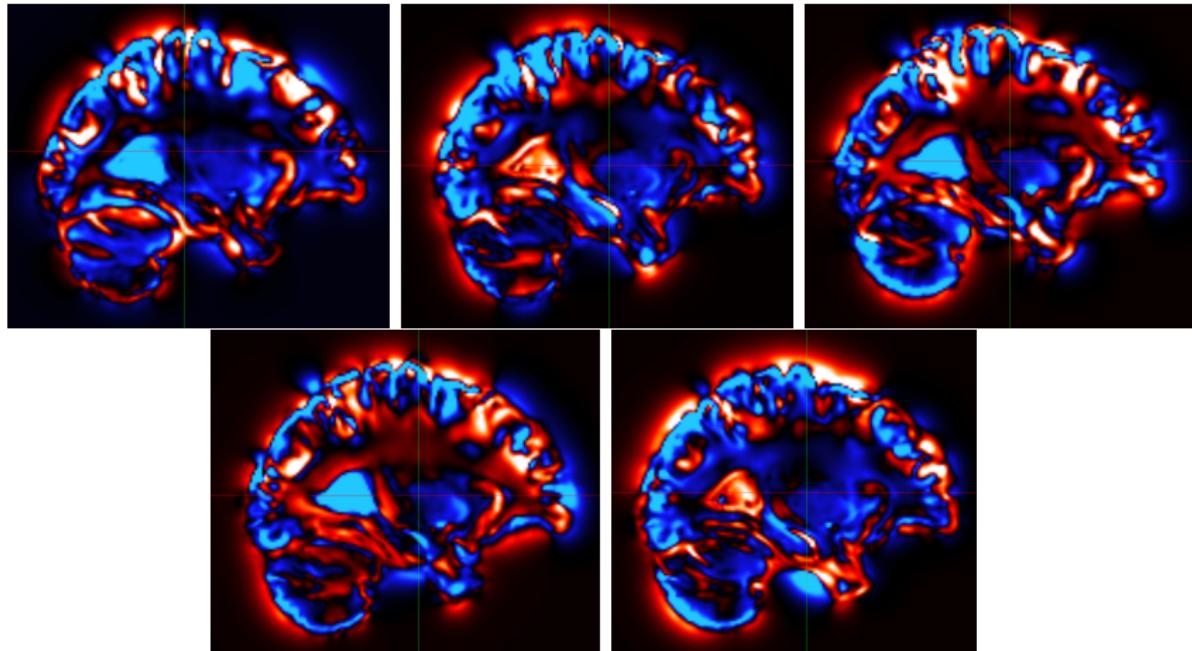
*Mean and five decomposition vectors estimated with L-ICA (left) and ET-ICA (right).
Each image has its own colorbar to highlight the major patterns.*

101 hippocampus deformations (3 populations: Ctrl - Mild AD - AD)

	Ctrl/AD		Ctrl/mild AD	
Model	L-ICA	BG-ICA	L-ICA	BG-ICA
Mean	0.31×10^{-3}	0.33×10^{-3}	9.0×10^{-3}	1.09×10^{-2}
Std dev.	0.16×10^{-3}	0.25×10^{-3}	3.8×10^{-3}	$4.6 \quad 7.6 \times 10^{-3}$

Table: Mean and standard deviation of the p-values for the two models with the decomposition vectors. Means and standard deviations are computed over 50 runs to separate the Controls from the AD group (left columns) and to separate the Controls from the mild AD group (right columns). PCA p-values: 0.3×10^{-3} and 7.7×10^{-3} using 95% of the cumulative variance.

Preliminary Results on the ADNI database



Can we go further?

- PCA model: too simple
- ICA model: interesting but still reduced to linear interpretation
- → may not be able to capture correct features,
- → may not describe the population well.

Need for non linear methods! Auto-encoders emerged as one possible solution...

With many other applications!

Auto-Encoder

- The objective \implies Dimensionality Reduction

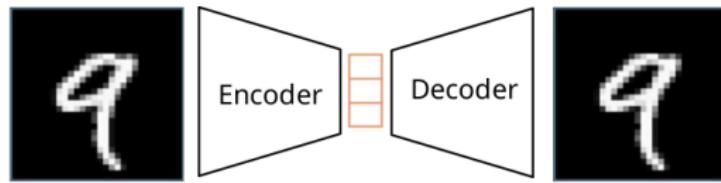


Figure: Simple Auto-Encoder

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:

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

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:

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

⇒ The networks are optimised using stochastic gradient descent

$$\begin{aligned}\phi &\leftarrow \phi - \varepsilon \cdot \nabla_\phi \mathcal{L} \\ \theta &\leftarrow \theta - \varepsilon \cdot \nabla_\theta \mathcal{L}\end{aligned}$$

AutoEncoder - Shortcomings

- What is represented in the latent space?

AutoEncoder - Shortcomings

- What is represented in the latent space?

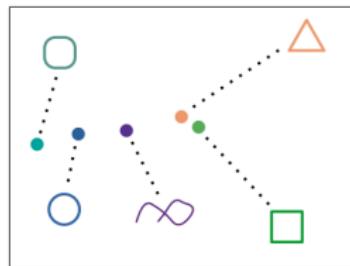


Figure: Potential latent space

- The AutoEncoder was just trained to **encode and decode the input data** without information on its structure or distribution.

AutoEncoder - Shortcomings

- What is represented in the latent space?

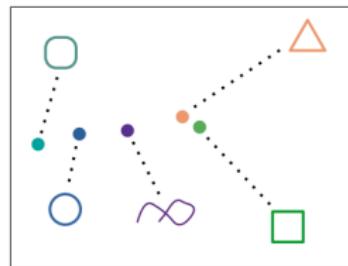


Figure: Potential latent space

- The AutoEncoder was just trained to **encode and decode the input data** without information on its structure or distribution.
- Can we get more information about the original **population**?

AutoEncoder - Shortcomings

- What is represented in the latent space?

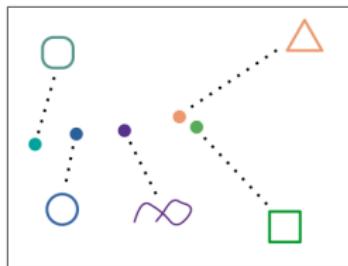


Figure: Potential latent space

- The AutoEncoder was just trained to **encode and decode the input data** without information on its structure or distribution.
- Can we get more information about the original **population**?

⇒ Need for a new framework

VAE - The Idea

- An auto-encoder based model...

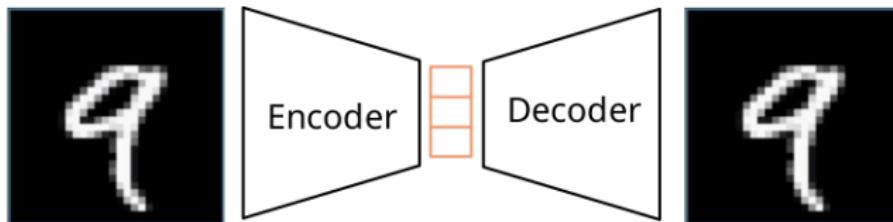


Figure: Simple Auto-Encoder

- ... but where an input data point is encoded as a **distribution** defined over the latent space [KW14, RMW14]

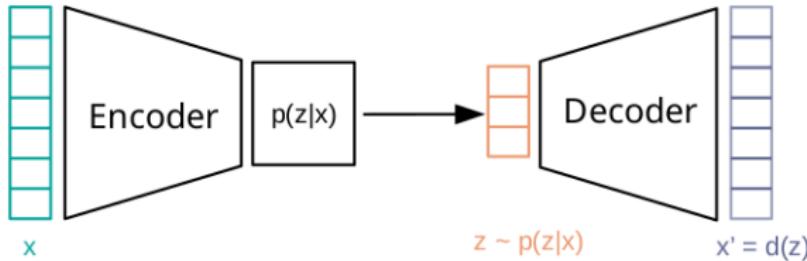


Figure: VAE framework

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

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:

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

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:

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

Objective:

- Maximizing the likelihood of the model

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:

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

Objective:

- Maximizing the likelihood of the model

Problem: The integral is often intractable.

Variational inference

We have to use Variational Inference:

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

Variational inference

We have to use Variational Inference:

$$\begin{aligned}\log p_{\theta}(x) &= \log \left(\int p_{\theta}(x|z)q_{\text{prior}}(z)dz \right) \\ &= \log \left(\int p_{\theta}(x, z)dz \right)\end{aligned}$$

Variational inference

We have to use Variational Inference:

$$\begin{aligned}\log p_{\theta}(x) &= \log \left(\int p_{\theta}(x|z)q_{\text{prior}}(z)dz \right) \\ &= \log \left(\int p_{\theta}(x, z)dz \right) \\ &= \log \left(\int p_{\theta}(x, z) \frac{q(z)}{q(z)} dz \right), \text{ for any pdf } q\end{aligned}$$

Variational inference

We have to use Variational Inference:

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

Variational inference

We have to use Variational Inference:

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

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

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

Variational inference: The ELBO

- Well-known issue: the posterior $q(z) = q_\theta(z|x)$ is intractable.
→ use Expectation-Maximization like algorithms (MCMC-SAEM version if needed)
- **OR** approximate this posterior:

Variational inference: The ELBO

- Well-known issue: the posterior $q(z) = q_\theta(z|x)$ is intractable.
→ use Expectation-Maximization like algorithms (MCMC-SAEM version if needed)
- OR** approximate this posterior:
Introduce a parametric approximation:

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

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

Variational inference: The ELBO

- Well-known issue: the posterior $q(z) = q_\theta(z|x)$ is intractable.
→ use Expectation-Maximization like algorithms (MCMC-SAEM version if needed)
- OR** approximate this posterior:
Introduce a parametric approximation:

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

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

This leads to an unbiased estimate of the log-likelihood

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

Variational inference: The ELBO

- Well-known issue: the posterior $q(z) = q_\theta(z|x)$ is intractable.
→ use Expectation-Maximization like algorithms (MCMC-SAEM version if needed)
- OR** approximate this posterior:
Introduce a parametric approximation:

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

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

This leads to an unbiased estimate of the log-likelihood

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

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

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

Variational inference: The ELBO

Objectives:

1. Optimize the ELBO **as a function** instead of the target distribution
Use stochastic gradient descent in both θ and ϕ

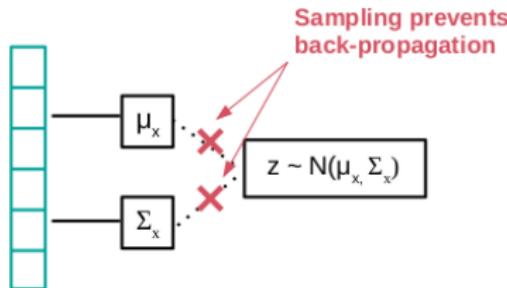
Variational inference: The ELBO

Objectives:

1. Optimize the ELBO **as a function** instead of the target distribution
Use stochastic gradient descent in both θ and ϕ
2. Optimize the ELBO **as a bound** to get closer to the target
Use sampling methods to produce samples $z \sim q_\theta(z|x)$

The Reparametrization Trick for stochastic gradient descent

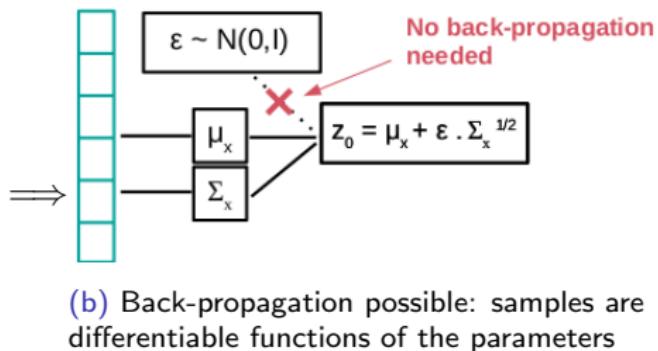
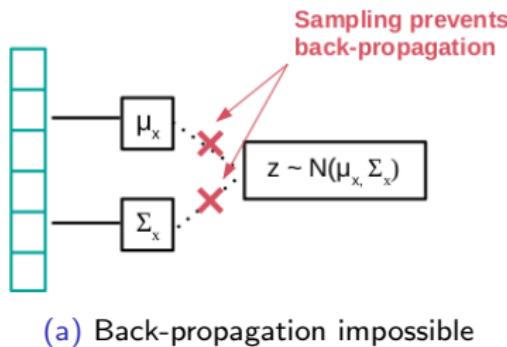
- Since $z \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, the model is not amenable to gradient descent



(a) Back-propagation impossible

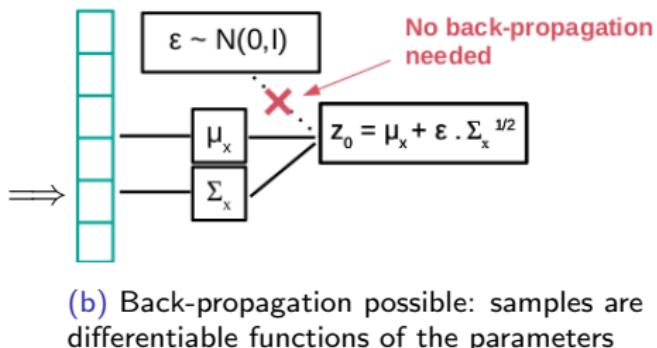
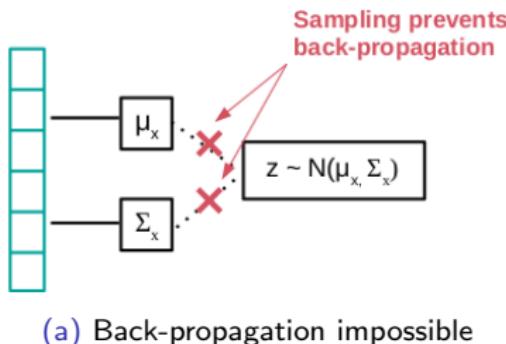
The Reparametrization Trick for stochastic gradient descent

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



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



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

Objective 1.



Changing the model

General VAE model requires the choice of $q_{prior}(z)$.

→ Most common choice: $q_{prior}(z) = \mathcal{N}(0, 1)$.

Pros:

- good initial guess,
- enables easy computations

Cons:

- very restrictive as it is isotropic and monomodal

Can we improve the model itself choosing other distributions?

Other VAE models changing the prior

Literature is full of propositions:

- the first one being a mixture of Gaussian distribution ("Approximate inference for deep latent gaussian mixtures", "Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders")
- VAE with a VampPrior
- up to prior leaning (VQVAE, *RAE_L2*)

Requires to be able optimize the ELBO: many limitations!

Other VAE models changing the distance in the ELBO

One may prefer to change the definition of the distance between probability distributions.

- Weighting the KL–divergence in the ELBO (β – VAE)
- Changing from KL–divergence to Wassenstein distance between the data distribution and the model one (WAE).

Other VAE models changing the Reconstruction metric

One may not want to use the standard L^2 norm on images.

- metric by patch and no more pixelwise (MSSSIM_VAE)
- Discriminator extract features and the similarity is between extracted features (VAEGAN)

Tweaking the Approximate Posterior Distribution

Concerning Objective 2.

- The ELBO can be written as

$$\text{ELBO} = \log p_\theta(x) - \underbrace{\text{KL}(q_\phi(z|x)||p_\theta(z|x))}_{\approx 0 \text{ if } q_\phi(z|x) \approx p_\theta(z|x)} .$$

- Kullback-Leibler divergence $\geq 0 \Rightarrow$ make it vanish by tweaking the approximate posterior $q_\phi(z|x)$
- Produce variables z which targets the true posterior $p_\theta(z|x)$ using a sample $z_0 \sim q_{init}$

Tweaking the Approximate Posterior Distribution

Concerning Objective 2.

- The ELBO can be written as

$$\text{ELBO} = \log p_\theta(x) - \underbrace{\text{KL}(q_\phi(z|x)||p_\theta(z|x))}_{\approx 0 \text{ if } q_\phi(z|x) \approx p_\theta(z|x)} .$$

- Kullback-Leibler divergence $\geq 0 \Rightarrow$ make it vanish by tweaking the approximate posterior $q_\phi(z|x)$
- Produce variables z which targets the true posterior $p_\theta(z|x)$ using a sample $z_0 \sim q_{init}$
- How? and how to ensure that the model would still be amenable to the back-propagation ?

Solution 1: Normalizing Flows

- Use smooth invertible parametrized mappings f_ψ to “sample” z [RM15]
- Apply K transformations to $z_0 \sim q_{init}$ (here $q_{init} = q_\phi$)
- Final random variable $z_K = f_x^K \circ \dots \circ f_x^1(z_0) \sim q_\phi(z_K|x)$ with

$$q_\phi(z_K|x) = q_\phi(z_0|x) \prod_{k=1}^K |\det \mathbf{J}_{f_x^k}|^{-1}, \quad (1)$$

Solution 1: Normalizing Flows

- Use smooth invertible parametrized mappings f_ψ to “sample” z [RM15]
- Apply K transformations to $z_0 \sim q_{init}$ (here $q_{init} = q_\phi$)
- Final random variable $z_K = f_x^K \circ \dots \circ f_x^1(z_0) \sim q_\phi(z_K|x)$ with

$$q_\phi(z_K|x) = q_\phi(z_0|x) \prod_{k=1}^K |\det \mathbf{J}_{f_x^k}|^{-1}, \quad (1)$$

Objective 2.



Many references (VAE_LinNF, VAE_IAF)

although difficult to compute the Jacobian of these maps f_1^K

Solution 2: Hamiltonian VAE

- Idea = Hybrid Monte Carlo Sampler [No11, DMS17, LBB⁺19],
- Target density

$$p_{\theta}(z|x) = \frac{p_{\theta}(x, z)}{p_{\theta}(x)} \propto p_{\theta}(x, z) = \pi_x(z).$$

- Introduce an auxiliary random variable $\rho \sim \mathcal{N}(0, \mathbf{M})$ called “momentum”
- Write the Hamiltonian:

$$\begin{aligned} H_x(z, \rho) &= -\log \pi_x(z, \rho) \\ &= -\log \pi_x(z) + \frac{1}{2} \log((2\pi)^d |\mathbf{M}|) + \rho^{\top} \mathbf{M}^{-1} \rho \\ &= U_x(z) + \kappa(\rho). \end{aligned}$$

- Sample (z, ρ) with this dynamic.

Solution 2: Hamiltonian VAE

- Use a discretization scheme

$$\begin{aligned}\rho(t + \varepsilon/2) &= \rho(t) - \frac{\varepsilon}{2} \cdot \nabla_z H(z(t), \rho(t)), \\ z(t + \varepsilon) &= z(t) + \varepsilon \cdot \nabla_\rho(H(z(t), \rho(t + \varepsilon/2))), \\ \rho(t + \varepsilon) &= \rho(t + \varepsilon/2) - \frac{\varepsilon}{2} \cdot \nabla_z H(z(t + \varepsilon), \rho(t + \varepsilon/2)),\end{aligned}\tag{2}$$

- A proposal $(\tilde{z}, \tilde{\rho})$ is accepted with probability:

$$\alpha = \min\left(1, \exp\left(-H(\tilde{z}, \tilde{\rho}) + H(z, \rho)\right)\right)$$

⇒ Creates an ergodic, time-reversible Markov Chain having π_x as stationary distribution.

Note that the Metropolis Hastings' acceptance step has to be removed for the back propagation to be possible.

Hamiltonian VAE

- The graphical scheme [CDS18]

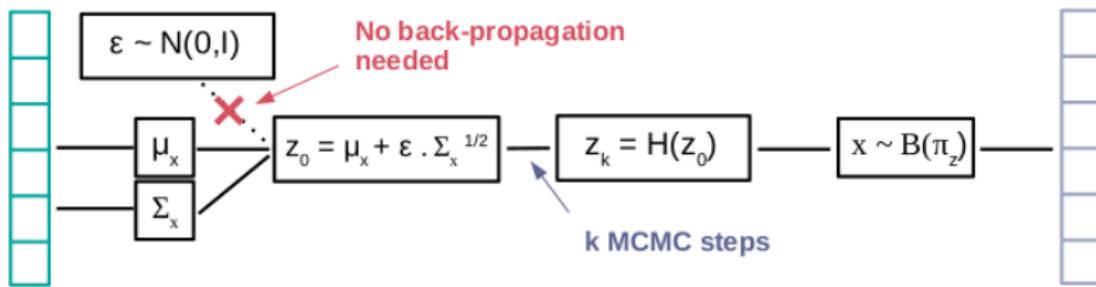


Figure: Hamiltonian VAE

Hamiltonian VAE

- The graphical scheme [CDS18]

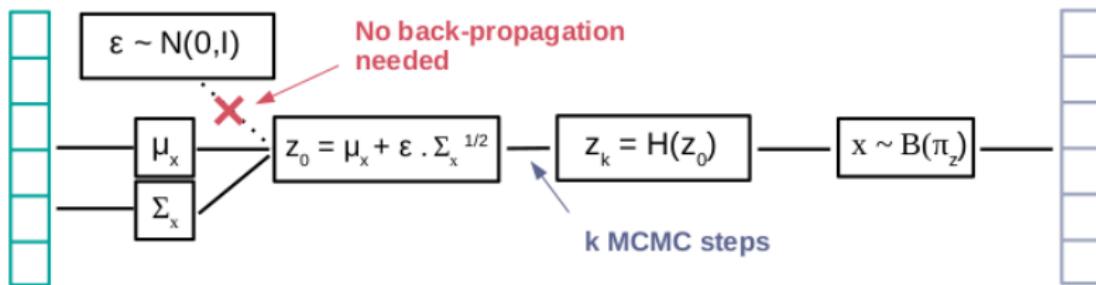


Figure: Hamiltonian VAE

Issue: Perform poorly when trained on small data set and so we need to define a new framework

Hamiltonian VAE

- The graphical scheme [CDS18]

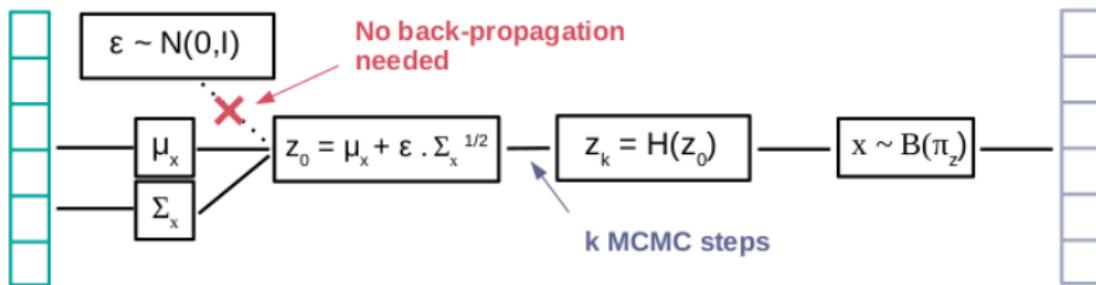


Figure: Hamiltonian VAE

Issue: Perform poorly when trained on small data set and so we need to define a new framework

Hamiltonian VAE

- The graphical scheme [CDS18]

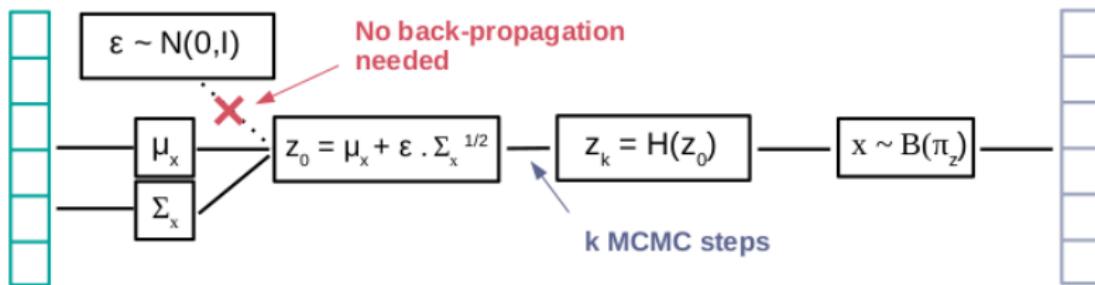


Figure: Hamiltonian VAE

Issue: Perform poorly when trained on small data set and so we need to define a new framework

What about geometry?

Benchmark of many of these methods

For an extensive comparison of the VAE and VAE like models:

https://github.com/clementchadebec/benchmark_VAE

- Benchmark of the learning strategies
- Benchmark of the sampling strategies

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} .
- It was shown that exploiting the geometrical aspect of probability distributions can lead to far more efficient sampling [GCC09, GC11]

Defining a New Framework

Assumptions:

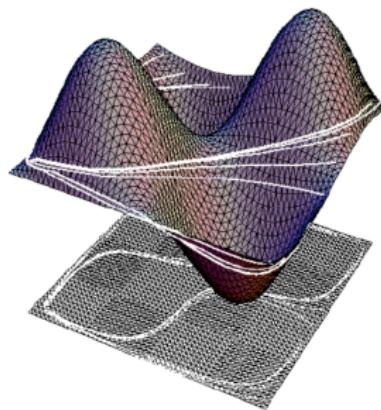
- As of now the latent space structure was supposed to be Euclidean (i.e. $\mathcal{Z} = \mathbb{R}^d$)
- Let us now relax this hypothesis and assume that \mathcal{Z} is a Riemannian manifold endowed with a metric \mathbf{G} .
- It was shown that exploiting the geometrical aspect of probability distributions can lead to far more efficient sampling [GCC09, GC11]

Our ideas:

- ① Exploit the manifold structure of the latent space to improve the posterior sampling
- ② Learn the metric defined in the latent space
- ③ Use the learned geometry to generate instead of the prior [CTSBA21]

Riemannian geometry principles

- Riemannian manifold: (reduced to our model) \mathbb{R}^d endowed with a metric \mathbf{G} :
 $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$.
⇒ \mathbb{R}^d not flat anymore, curved space (as mountains)



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

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

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 $\gamma : [0, 1] \rightarrow \mathcal{M}$ from z_1 to z_2 living in a Riemannian manifold \mathcal{M}

$$L(\gamma) = \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)}} dt \quad \gamma(0) = z_1, \gamma(1) = z_2. \quad (3)$$

- **Geodesic paths** = curve γ minimizing Eq. (3)
- or equivalently **minimizing the curve energy**

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

1) Improve Posterior Sampling - Riemannian HMC

- Rely on the **Riemannian** Hamiltonian Monte Carlo Sampler [GC11]:
 - Introduce a **Position-specific** random momentum $\rho \sim \mathcal{N}(0, \mathbf{G}(z))$
 - Simulates the evolution $(z(t), \rho(t))$ of a particle whose motion is governed by Hamiltonian dynamics **on the manifold**
- The Hamiltonian writes

$$H_x^{Riem}(z, \rho) = \log p_{\text{target}}(z) + \frac{1}{2} \log((2\pi)^D \det \mathbf{G}(z)) + \frac{1}{2} \rho^\top \mathbf{G}(z)^{-1} \rho.$$

1) Improve Posterior Sampling - Riemannian HMC

- Rely on the **Riemannian** Hamiltonian Monte Carlo Sampler [GC11]:
 - Introduce a **Position-specific** random momentum $\rho \sim \mathcal{N}(0, \mathbf{G}(z))$
 - Simulates the evolution $(z(t), \rho(t))$ of a particle whose motion is governed by Hamiltonian dynamics **on the manifold**
- The Hamiltonian writes

$$H_x^{Riem}(z, \rho) = \log p_{\text{target}}(z) + \frac{1}{2} \log((2\pi)^D \det \mathbf{G}(z)) + \frac{1}{2} \rho^\top \mathbf{G}(z)^{-1} \rho.$$

- Use of the “Generalized” Leapfrog integrator to sample from p_{target}

1) Improve Posterior Sampling - Riemannian HMC

- Rely on the **Riemannian** Hamiltonian Monte Carlo Sampler [GC11]:
 - Introduce a **Position-specific** random momentum $\rho \sim \mathcal{N}(0, \mathbf{G}(z))$
 - Simulates the evolution $(z(t), \rho(t))$ of a particle whose motion is governed by Hamiltonian dynamics **on the manifold**
- The Hamiltonian writes

$$H_x^{Riem}(z, \rho) = \log p_{\text{target}}(z) + \frac{1}{2} \log((2\pi)^D \det \mathbf{G}(z)) + \frac{1}{2} \rho^\top \mathbf{G}(z)^{-1} \rho.$$

- Use of the “Generalized” Leapfrog integrator to sample from p_{target}

Pros:

- Use the underlying geometry of the data to improve sampling

Cons:

- The metric is unknown

2) Learn the Metric - The Choice of the Metric

- Parametric metric: [Lou19]:

$$\mathbf{G}^{-1}(z) = \sum_{i=1}^N L_{\psi_i} L_{\psi_i}^\top \exp\left(-\frac{\|z - c_i\|_2^2}{T^2}\right) + \lambda I_d,$$

- L_{ψ_i} lower triangular matrices parametrized using neural networks
- T temperature to smooth the metric
- c_i centroids
- λ regularization factor

2) Learn the Metric - The Choice of the Metric

- Parametric metric: [Lou19]:

$$\mathbf{G}^{-1}(z) = \sum_{i=1}^N L_{\psi_i} L_{\psi_i}^\top \exp\left(-\frac{\|z - c_i\|_2^2}{T^2}\right) + \lambda I_d,$$

- L_{ψ_i} lower triangular matrices parametrized using neural networks
- T temperature to smooth the metric
- c_i centroids
- λ regularization factor

Pros:

- Closed-form expression of the inverse metric \implies useful for geodesic computation
- Geodesics travel through most populated areas.

The Model - Riemannian Hamiltonian VAE

- The graphical scheme

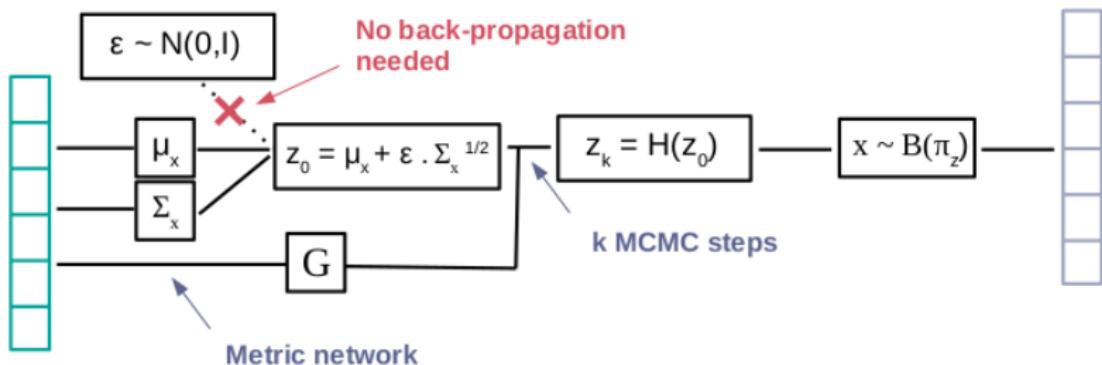
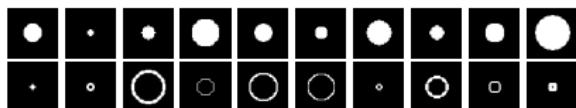


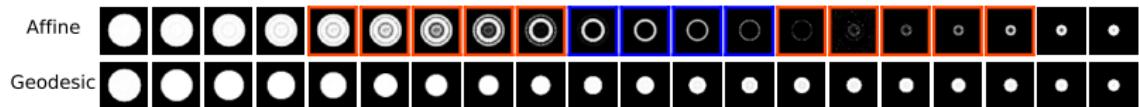
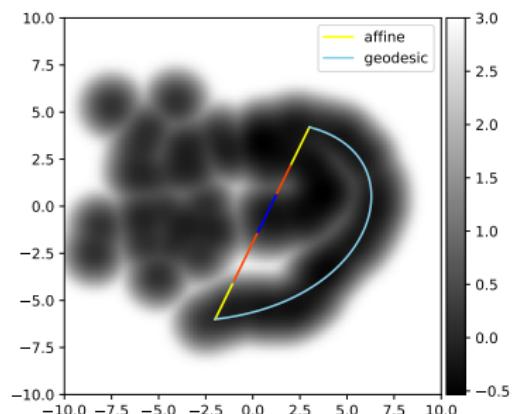
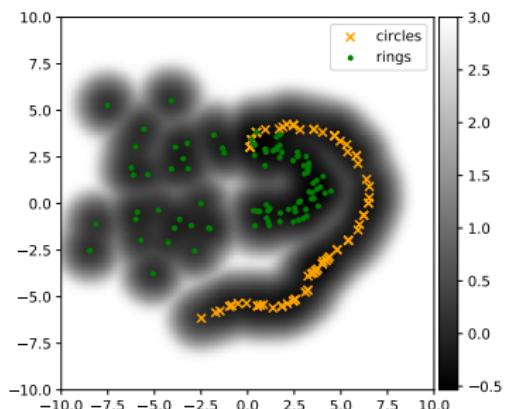
Figure: Riemannian Hamiltonian VAE.

The Learned Latent Space examples

Training samples:



Latent space and interpolations:

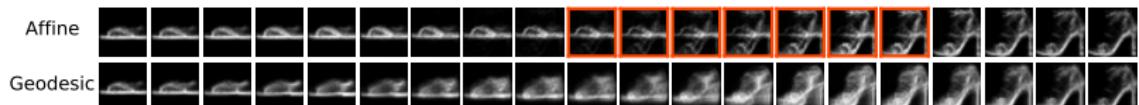
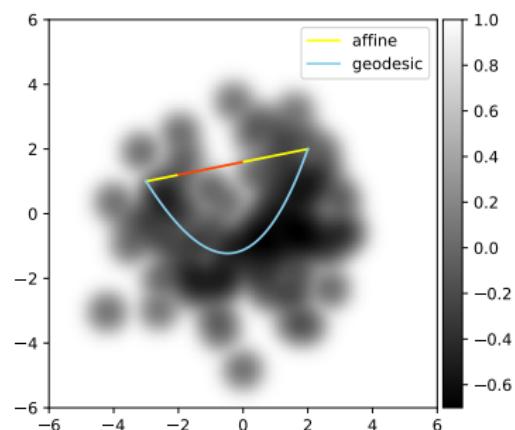
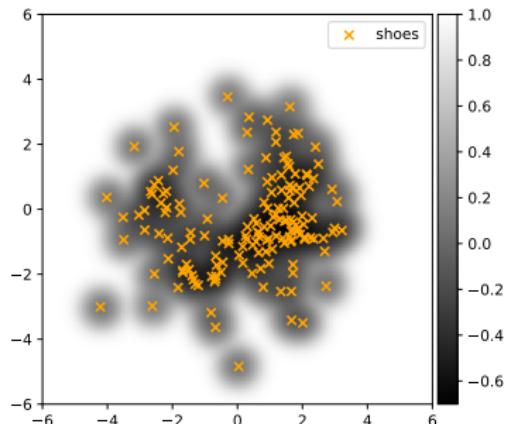


The Learned Latent Space examples

Training samples:



Latent space and interpolations:



3) Improve Data Generation - Sample With the Metric

Idea:

- Use a geometry-based sampling procedure: pdf driven by the metric

$$p(z) = \frac{\mathbb{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}}{\int\limits_{\mathbb{R}^d} \mathbb{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)} dz},$$

where S is a compact set and $\mathbb{1}_S(z) = 1$ if $z \in S$, 0 otherwise.

- Use of classic MCMC sampler (e.g. Hamiltonian Monte Carlo)

3) Improve Data Generation - Sample With the Metric

Idea:

- Use a geometry-based sampling procedure: pdf driven by the metric

$$p(z) = \frac{\mathbb{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}}{\int\limits_{\mathbb{R}^d} \mathbb{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)} dz},$$

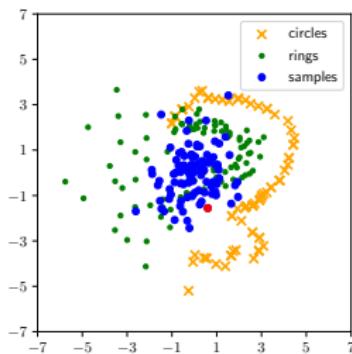
where S is a compact set and $\mathbb{1}_S(z) = 1$ if $z \in S$, 0 otherwise.

- Use of classic MCMC sampler (e.g. Hamiltonian Monte Carlo)

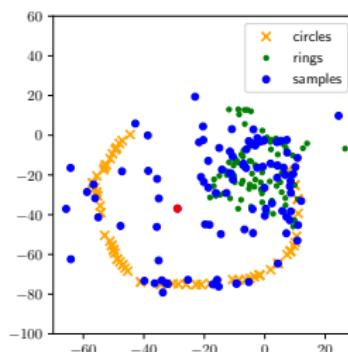
Pros:

- \mathbf{G}^{-1} easily computable
- Samples “close” to the data

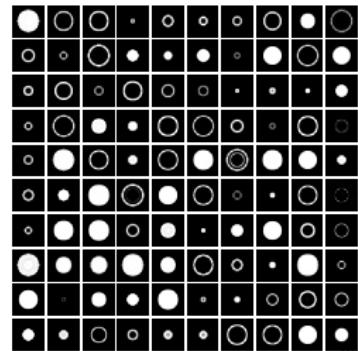
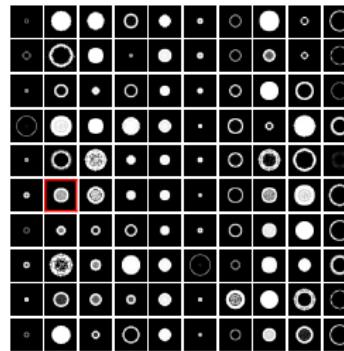
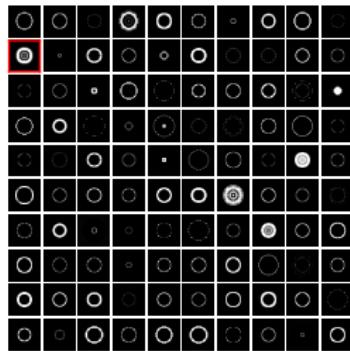
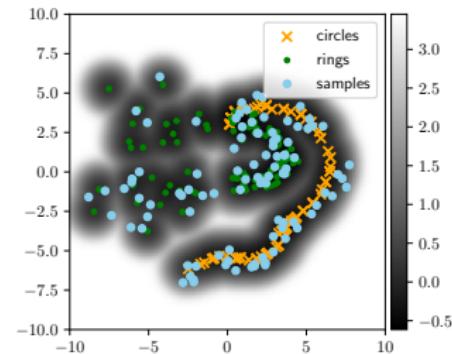
Sampling Comparison

(a) VAE - $\mathcal{N}(0, I)$ 

(b) VAE - VAMP (multimodal conditional prior)



(c) Ours



Sampling Comparison - Higher Dimension

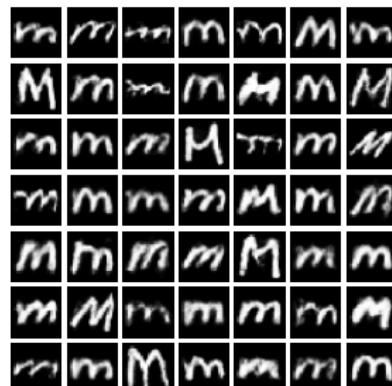
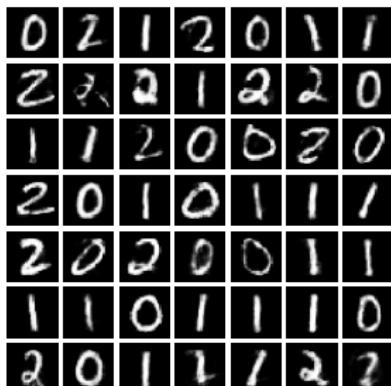
(a) reduced MNIST (120)



(b) reduced EMNIST (120)



(c) reduced Fashion (120)



Data Augmentation

1. Framework
2. Toy Data
3. Medical Imaging

Data Augmentation - Framework

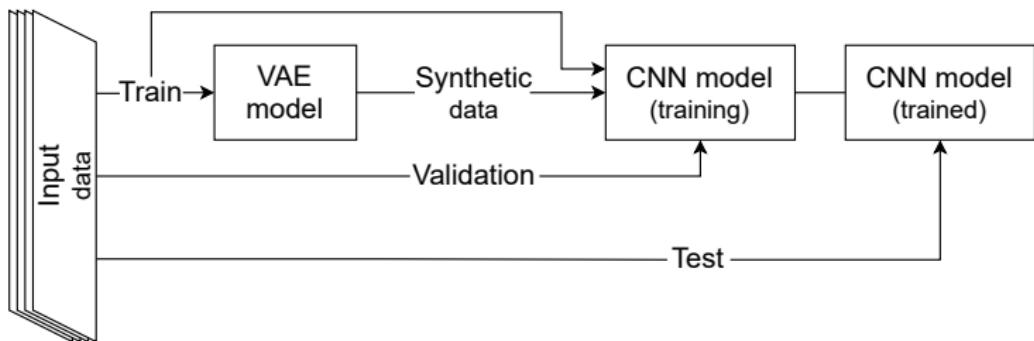


Figure: Data Augmentation pipeline

Performances are estimated using cross-validation.

Data Augmentation

1. Framework
2. Toy Data
3. Medical Imaging

Robustness Across Data Sets

Table: Classification results on *reduced* data sets (~ 50 samples per class)

	MNIST	MNIST (unbal.)	EMNIST (unbal.)	FASHION
Baseline	89.9 ± 0.6	81.5 ± 0.7	82.6 ± 1.4	76.0 ± 1.5
	Baseline + Synthetic			
Basic Augmentation (X5)	92.8 ± 0.4	86.5 ± 0.9	85.6 ± 1.3	77.5 ± 2.0
Basic Augmentation (X10)	88.2 ± 2.2	82.0 ± 2.4	85.7 ± 0.3	79.2 ± 0.6
Basic Augmentation (X15)	92.8 ± 0.7	85.8 ± 3.4	86.6 ± 0.8	80.0 ± 0.5
VAE - 200*	88.5 ± 0.9	84.0 ± 2.0	81.7 ± 3.0	78.6 ± 0.4
VAE - 2k*	92.2 ± 1.6	88.0 ± 2.2	86.0 ± 0.2	79.3 ± 1.1
Ours-200	91.0 ± 1.0	84.1 ± 2.0	85.1 ± 1.1	77.0 ± 0.8
Ours-500	92.3 ± 1.1	87.7 ± 0.9	85.1 ± 1.1	78.5 ± 0.9
Ours-1k	93.2 ± 0.8	89.7 ± 0.8	87.0 ± 1.0	80.2 ± 0.8
Ours-2k	94.3 ± 0.8	89.1 ± 1.9	87.6 ± 0.8	78.1 ± 1.8

* Using a standard normal prior to generate

- Classic DA is data set dependent
- Vanilla VAE performs as well as classic DA

Robustness Across Data Sets

Table: Classification results on *reduced* data sets (~ 50 samples per class) **on synthetic samples only**

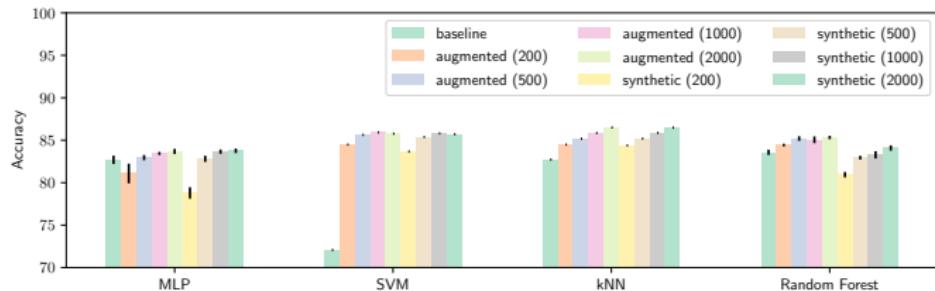
	MNIST	MNIST (unbal.)	EMNIST (unbal.)	FASHION
Baseline	89.9 ± 0.6	81.5 ± 0.7	82.6 ± 1.4	76.0 ± 1.5
Synthetic Only				
VAE - 200*	69.9 ± 1.5	64.6 ± 1.8	65.7 ± 2.6	73.9 ± 3.0
VAE - 2k*	86.5 ± 2.2	79.6 ± 3.8	78.8 ± 3.0	76.7 ± 1.6
Ours-200	87.2 ± 1.1	79.5 ± 1.6	77.0 ± 1.6	77.0 ± 0.8
Ours-500	89.1 ± 1.3	80.4 ± 2.1	80.2 ± 2.0	78.5 ± 0.8
Ours-1k	90.1 ± 1.4	86.2 ± 1.8	82.6 ± 1.3	79.3 ± 0.6
Ours-2k	92.6 ± 1.1	87.5 ± 1.3	86.0 ± 1.0	78.3 ± 0.9

* Using a standard normal prior to generate

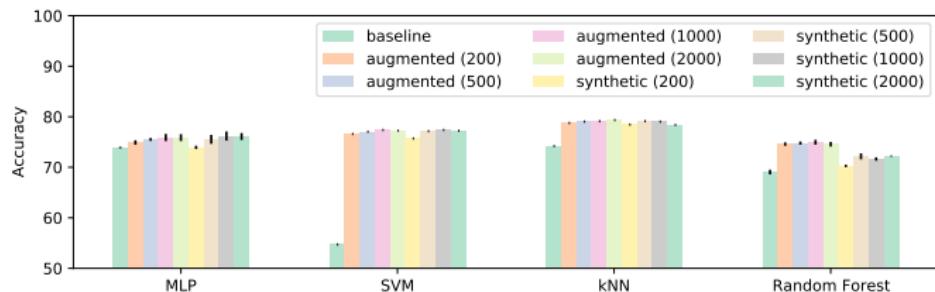
- The proposed model seems to create diverse samples relevant to the classifier

Robustness Across Classifiers

(a) *reduced MNIST balanced*



(b) *reduced MNIST unbalanced*



A Note on the Method Scalability

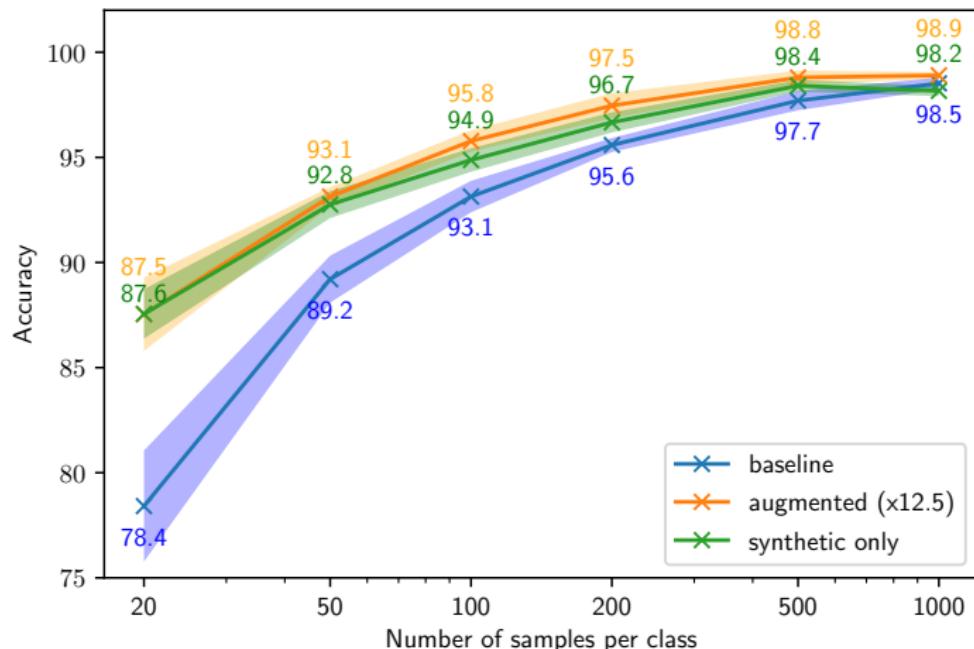


Figure: Benchmark classifier accuracy according to the number of samples in the training set on MNIST.

Data Augmentation

1. Framework
2. Toy Data
3. Medical Imaging

Datasets and classification task

Classification task: Alzheimer's disease patients (**AD**) vs Cognitively Normal participants (**CN**) using T1-weighted MR images.

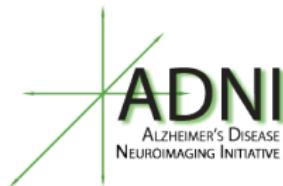


Table: Summary of participant demographics, mini-mental state examination (MMSE) and global clinical dementia rating (CDR) scores at baseline.

Data set	Label	Obs.	Age	Sex M/F	MMSE	CDR
ADNI	CN	403	73.3 ± 6.0	185/218	29.1 ± 1.1	0: 403
	AD	362	74.9 ± 7.9	202/160	23.1 ± 2.1	0.5: 169, 1: 192, 2: 1
AIBL	CN	429	73.0 ± 6.2	183/246	28.8 ± 1.2	0: 406, 0.5: 22, 1: 1
	AD	76	74.4 ± 8.0	33/43	20.6 ± 5.5	0.5: 31, 1: 36, 2: 7, 3: 2

MRI preprocessing

Bias field correction (N4ITK) + linear registration (ANTS) + cropping

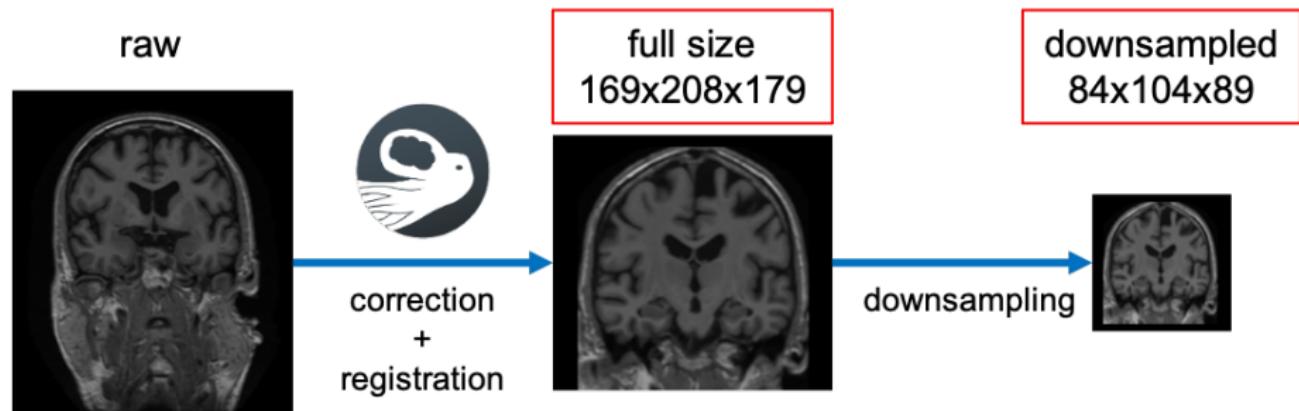


Figure: Preprocessed MRI used in the study

Find wonderful data at:

[/network/lustre/dtlake01/aramis/datasets/adni/caps/caps_v2021](http://network/lustre/dtlake01/aramis/datasets/adni/caps/caps_v2021)

Synthesized images

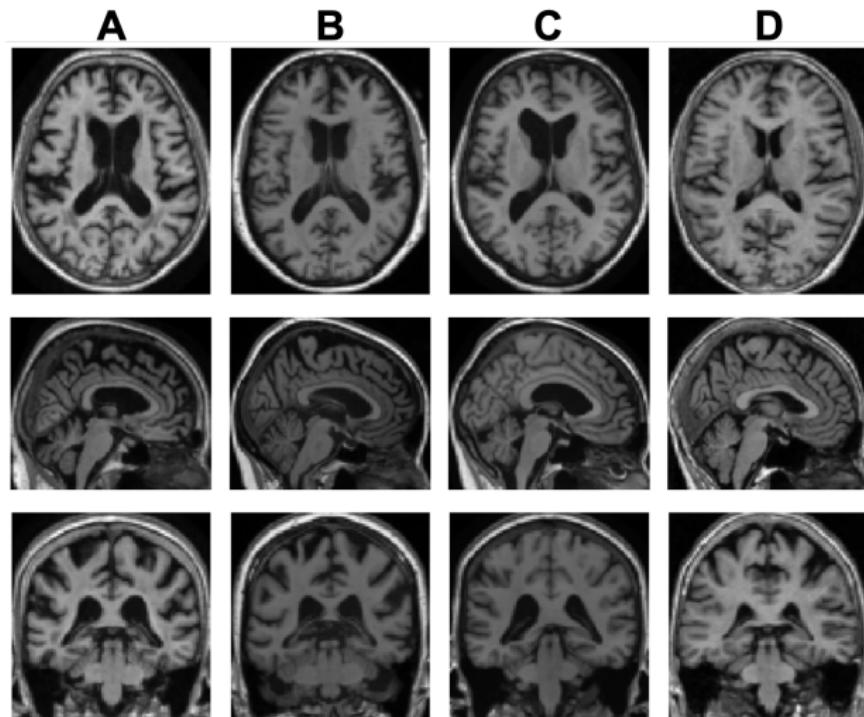


Figure: Example of two *true* patients compared to two generated by our method. Can you find the intruders ?

Synthesized images

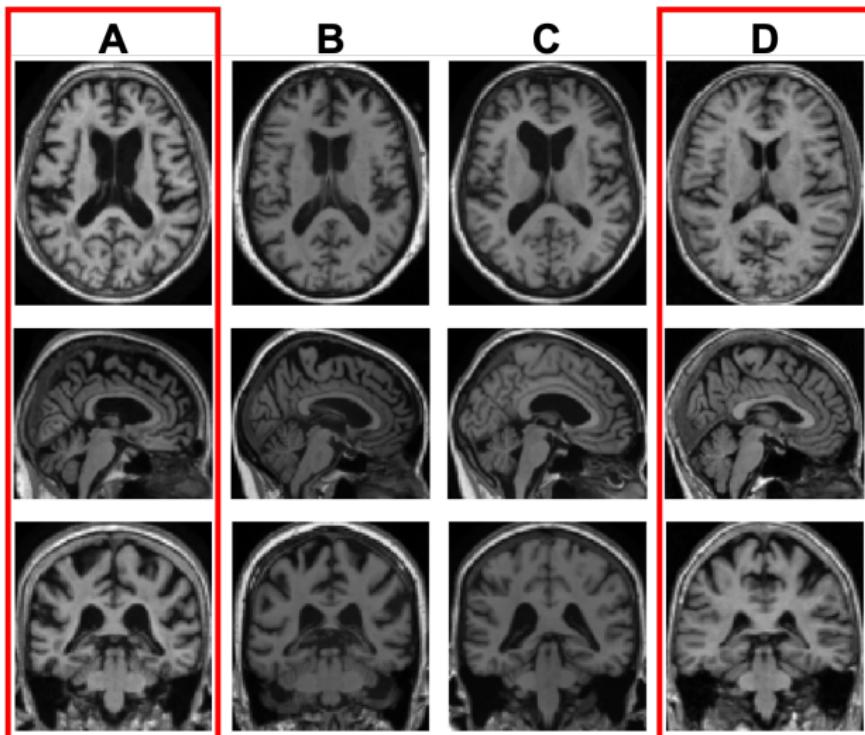
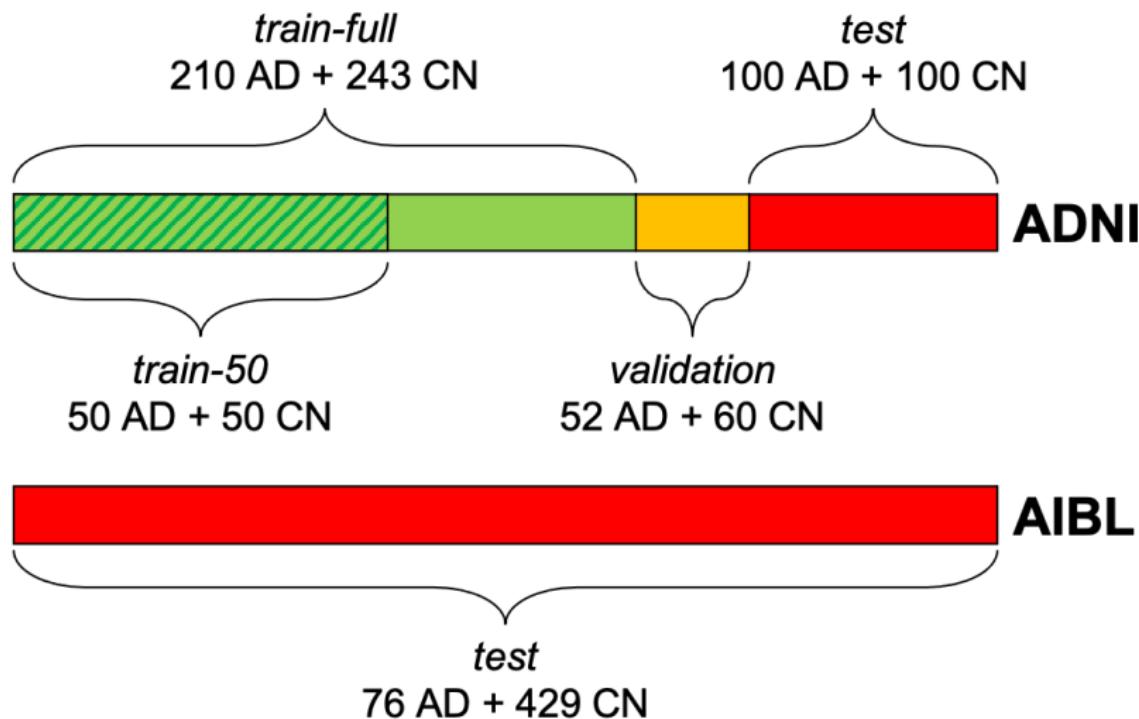


Figure: Example of two *true* patients compared to two generated by our method. Can you find the intruders ?

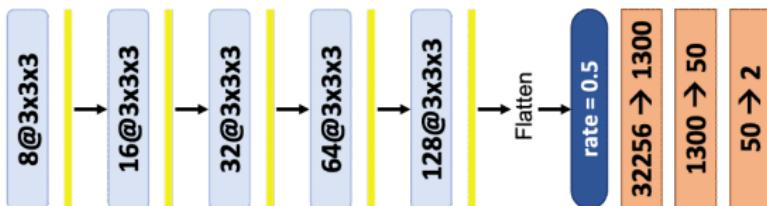
Evaluation procedure



CNN architectures for classifier

Baseline architectures provided by a previous study [WTSDM⁺20]

1. Full size image



2. Downsampled image



3D Convolution (stride=1, padding=1) + Batch normalization + LeakyReLU

— MaxPooling (kernel=2, stride=2)

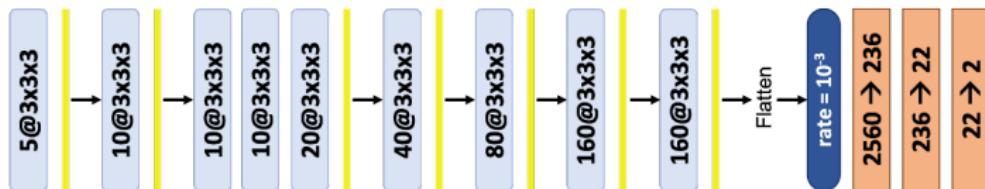
Dropout

Fully-connected layer (+ LeakyReLU except last layer)

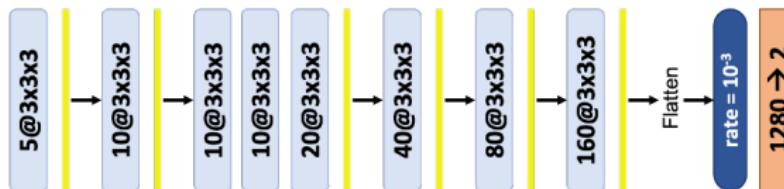
CNN architectures for classifier

Optimized architectures optimized with random search procedure for this training set (ClinicaDL)

1. Full size image



2. Downsampled image



3D Convolution (stride=1, padding=1) + Batch normalization + LeakyReLU

— MaxPooling (kernel=2, stride=2)

● Dropout

Fully-connected layer (+ LeakyReLU except last layer)

Experiments

Four series of experiments:

- **baseline** architecture on *train-50*
- **baseline** architecture on *train-full*
- **optimized** architecture on *train-50*
- **optimized** architecture on *train-full*

For each experiment 20 CNNs are run and the performance is the mean value of the 20 performance values.

Results on train-50 with baseline CNN

Table: Mean test performance of each series of 20 runs trained with the **baseline** hyperparameters on *train-50* set.

data set	ADNI balanced accuracy	AIBL balanced accuracy
real	66.3 ± 2.4	67.2 ± 4.1
real (high-resolution)	67.9 ± 2.3	66.5 ± 3.0
500 synthetic + real	69.4 ± 1.6	68.5 ± 2.5
1000 synthetic + real	70.5 ± 2.1	70.6 ± 3.1
2000 synthetic + real	71.2 ± 1.6	72.8 ± 2.2
3000 synthetic + real	72.6 ± 1.6	73.6 ± 3.0
5000 synthetic + real	74.1 ± 2.2	76.1 ± 3.6
10000 synthetic + real	74.0 ± 2.7	74.9 ± 3.2

Increase of balanced accuracy of 6.2 points on ADNI and 8.9 points on AIBL

Results on train-full with baseline CNN

Table: Mean test performance of each series of 20 runs trained with the **baseline** hyperparameters on *train-full* set.

data set	ADNI balanced accuracy	AIBL balanced accuracy
real	77.7 ± 2.5	78.4 ± 2.4
real (high-resolution)	80.6 ± 1.1	80.4 ± 2.6
500 synthetic + real	82.2 ± 2.4	82.9 ± 2.5
1000 synthetic + real	84.4 ± 1.8	83.7 ± 2.3
2000 synthetic + real	85.9 ± 1.6	83.8 ± 2.2
3000 synthetic + real	85.8 ± 1.7	84.4 ± 1.8
5000 synthetic + real	85.7 ± 2.1	84.2 ± 2.2
10000 synthetic + real	86.3 ± 1.8	85.1 ± 1.9

Increase of balanced accuracy of 5.7 points on ADNI and 4.7 on AIBL

Results on train-50 with optimized CNN

Table: Mean test performance of each series of 20 runs trained with the **optimized** hyperparameters on *train-50* set.

data set	ADNI balanced accuracy	AIBL balanced accuracy
real	75.5 ± 2.7	75.6 ± 4.1
real (high-resolution)	72.1 ± 3.1	71.2 ± 5.1
500 synthetic + real	75.6 ± 2.5	76.0 ± 4.2
1000 synthetic + real	77.8 ± 2.3	80.9 ± 3.2
2000 synthetic + real	76.9 ± 2.4	80.0 ± 3.6
3000 synthetic + real	77.8 ± 1.9	81.2 ± 3.7
5000 synthetic + real	76.9 ± 2.5	80.9 ± 2.7
10000 synthetic + real	78.0 ± 2.1	81.9 ± 2.2

Increase of balanced accuracy of 2.5 points on ADNI and 6.3 points on AIBL

Results on train-full with optimized CNN

Table: Mean test performance of each series of 20 runs trained with the **optimized** hyperparameters on *train-full* set.

data set	ADNI balanced accuracy	AIBL balanced accuracy
real	85.5 ± 2.4	81.9 ± 3.2
real (high-resolution)	85.7 ± 2.5	84.4 ± 1.7
500 synthetic + real	86.0 ± 1.8	83.2 ± 2.4
1000 synthetic + real	86.5 ± 1.9	83.7 ± 2.0
2000 synthetic + real	87.2 ± 1.7	84.0 ± 2.0
3000 synthetic + real	85.8 ± 2.6	83.6 ± 3.2
5000 synthetic + real	86.4 ± 1.3	83.5 ± 2.2
10000 synthetic + real	86.7 ± 1.8	84.3 ± 1.8

Increase of balanced accuracy of 1.5 point on ADNI and -0.1 point on AIBL

Conclusion

We have proposed

- a new geometry aware VAE-based data augmentation framework relevant for representing and classifying data in the HDLSS setting.
- Validated on classification tasks on *toy* and *real-life* data sets in particular in the *High dimension low sample size setting*.

Conclusion

We have proposed

- a new geometry aware VAE-based data augmentation framework relevant for representing and classifying data in the HDLSS setting.
- Validated on classification tasks on *toy* and *real-life* data sets in particular in the *High dimension low sample size setting*.

Strengths:

- Independent on the nature of the data set: from 2D images (MNIST, EMNIST, FASHION) to 3D medical images (ADNI and AIBL),
- Relevant synthetic data: classifiers achieved a similar or better classification performance when trained only on synthetic data than on the *real* train set.
- Classifier independence: MLP, random forest, k-NN and SVM (on toy data sets) ; baseline and optimized parameters (on medical images).

Conclusion

We have proposed

- a new geometry aware VAE-based data augmentation framework relevant for representing and classifying data in the HDLSS setting.
- Validated on classification tasks on *toy* and *real-life* data sets in particular in the *High dimension low sample size setting*.

Limitations - what could be improved:

- No extensive search on VAE architecture.
- Would it benefit from the use of longitudinal data?
- *train-50* is still large compared to some medical data sets...

Implementation available

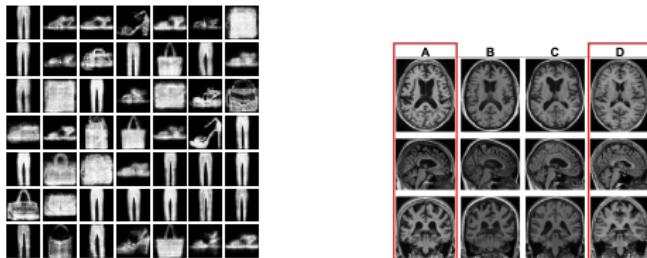
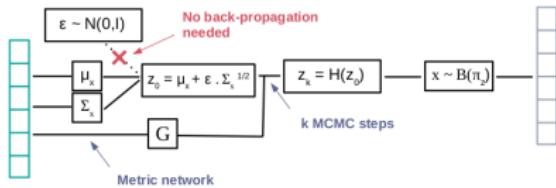
<https://clementchadebec.github.io/projects/>



AND Extensive comparison of data generation based on VAEs

Models	Training example	Paper	Official Implementation	Models	MNIST	CELEBA
Autoencoder (AE)	Raug Code			3	8 6 9 6	
Variational Autoencoder (VAE)	Raug Code	link		4	5 3 8 4	
Beta Variational Autoencoder (BetaVAE)	Raug Code	link		5	2 3 8 4	
Disentangled Beta Variational Autoencoder (DisentangledBetaVAE)	Raug Code	link		8	1 5 0 5	
Disentangling Factoring Factor(VAE)	Raug Code	link		9	7 4 1 0	
Beta TC-VAE (BetaTCVAE)	Raug Code	link	link	3	8 6 9 6	
Importance Weighted Autoencoder (IWADE)	Raug Code	link	link	4	5 3 8 4	
VAE with perceptual metric similarity (MSSSIM_VAE)	Raug Code	link		5	2 3 8 4	
Wasserstein Autoencoder (WAE)	Raug Code	link	link	8	1 5 0 5	
Info Variational Autoencoder (INFORVAE_JMD)	Raug Code	link		9	7 4 1 0	
VAEP Autoencoder (VAEP)	Raug Code	link	link	3	8 6 9 6	
Hyperspherical VAE (SVAE)	Raug Code	link	link	4	5 3 8 4	
Adversarial Autoencoder (Adversarial_AE)	Raug Code	link		5	2 3 8 4	
Variational Autoencoder GAN (MEGAN)	Raug Code	link	link	8	1 5 0 5	
Vector Quantized VAE (VQVAE)	Raug Code	link	link	9	7 4 1 0	
Hamiltonian VAE (HVAE)	Raug Code	link	link	3	8 6 9 6	
Regulated AE with L2 decoder param (RAE_L2)	Raug Code	link		4	5 3 8 4	
Required AE with gradient penalty (RAE_GP)	Raug Code	link	link	5	2 3 8 4	
Reparameter Hamiltonian VAE (RHVAE)	Raug Code	link	link	8	1 5 0 5	
				9	7 4 1 0	

Thank you!



<https://clementchadebec.github.io/projects/>

Contacts:

clement.chadebec@inria.fr
 stephanie.allassonniere@inria.fr

References I

-  Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic, *Hamiltonian variational auto-encoder*, Advances in Neural Information Processing Systems, 2018, pp. 8167–8177.
-  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*, arXiv preprint arXiv:2105.00026 (2021).
-  Alain Durmus, Eric Moulines, and Eero Saksman, *On the convergence of hamiltonian monte carlo*, arXiv preprint arXiv:1705.00166 (2017).
-  Mark Girolami and Ben Calderhead, *Riemann manifold langevin and hamiltonian monte carlo methods*, Journal of the Royal Statistical Society: Series B (Statistical Methodology) **73** (2011), no. 2, 123–214.
-  Mark Girolami, Ben Calderhead, and Siu A Chin, *Riemannian manifold hamiltonian monte carlo*, arXiv preprint arXiv:0907.1100 (2009).

References II

-  Diederik P. Kingma and Max Welling, *Auto-encoding variational bayes*, arXiv:1312.6114 [cs, stat] (2014).
-  Samuel Livingstone, Michael Betancourt, Simon Byrne, Mark Girolami, and others, *On the geometric ergodicity of hamiltonian monte carlo*, Bernoulli **25** (2019), no. 4, 3109–3138.
-  Maxime Louis, *Computational and statistical methods for trajectory analysis in a Riemannian geometry setting*, PhD Thesis, Sorbonnes universités, 2019.
-  Radford M Neal and others, *MCMC using hamiltonian dynamics*, Handbook of Markov Chain Monte Carlo **2** (2011), no. 11, 2.
-  Danilo Rezende and Shakir Mohamed, *Variational inference with normalizing flows*, International Conference on Machine Learning, PMLR, 2015, pp. 1530–1538.

References III

-  Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra, *Stochastic backpropagation and approximate inference in deep generative models*, International conference on machine learning, PMLR, 2014, pp. 1278–1286.
-  Junhao Wen, Elina Thibeau-Sutre, Mauricio Diaz-Melo, Jorge Samper-González, Alexandre Routier, Simona Bottani, Didier Dormont, Stanley Durrleman, Ninon Burgos, and Olivier Colliot, *Convolutional neural networks for classification of Alzheimer's disease: Overview and reproducible evaluation*, Medical Image Analysis **63** (2020), 101694.

Clustering

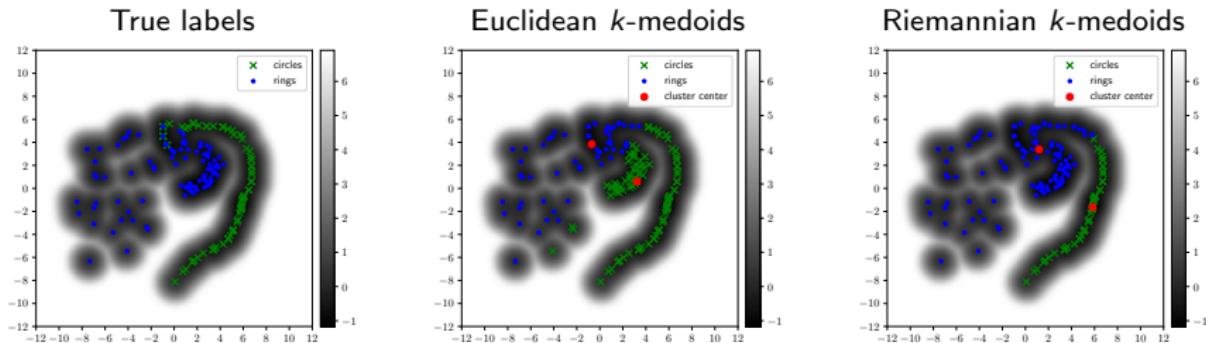


Figure: Euclidean and Riemannian k -medoids clustering.

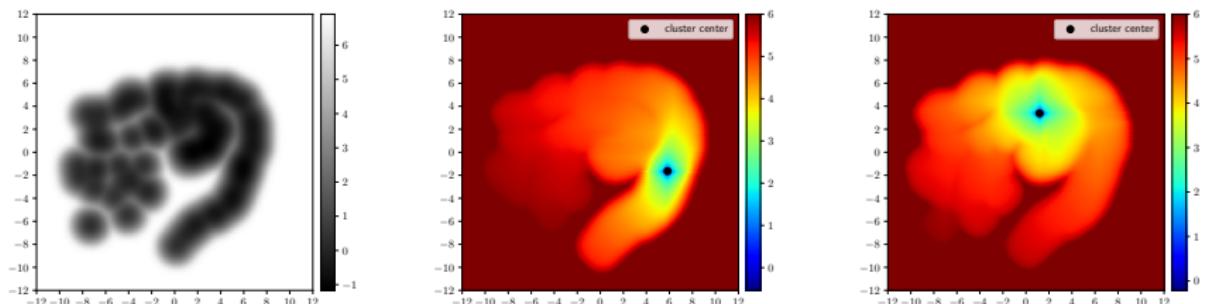


Figure: Distance maps.

Results - Clustering

Data set	Model	Subset 1	Subset 2	Subset 3	Mean
Synthetic data	linear	53.88	62.52	71.63	62.68
	geodesic	71.41	81.39	79.49	77.43
MNIST 1	linear	89.73	93.11	91.80	91.55
	geodesic	91.68	94.51	95.63	93.94
MNIST 2	linear	68.24	69.22	79.05	71.17
	geodesic	70.35	71.34	79.64	73.78
MNIST 3	linear	75.55	75.76	81.70	77.67
	geodesic	76.08	77.94	81.96	78.66
FashionMNIST 1	linear	90.47	91.63	86.78	89.63
	geodesic	91.44	92.55	87.46	90.48
FashionMNIST 2	linear	92.20	91.26	93.30	92.25
	geodesic	93.56	91.80	94.12	93.16
FashionMNIST 3	linear	72.46	79.58	83.16	78.40
	geodesic	74.89	81.88	84.83	80.53

Table: F1-Scores.