Huber-energy VAE

Grégoire Mourre, David Premachandra

March 2024

Introduction

The goal of the project was to write a Tensorflow v2 implementation of a Variational AutoEncoder (VAE) using the Huber-energy statistical distances.
VAEs have emerged as one of the most popular approaches to unsupervised learning of complicated distributions.
An important aspect of it is the way to make the encoded data follow a certain standard distribution. For this, we have to chose a way to measure the distance between distributions. In this project, the Huber-energy distances are chosen.

Presentation outline

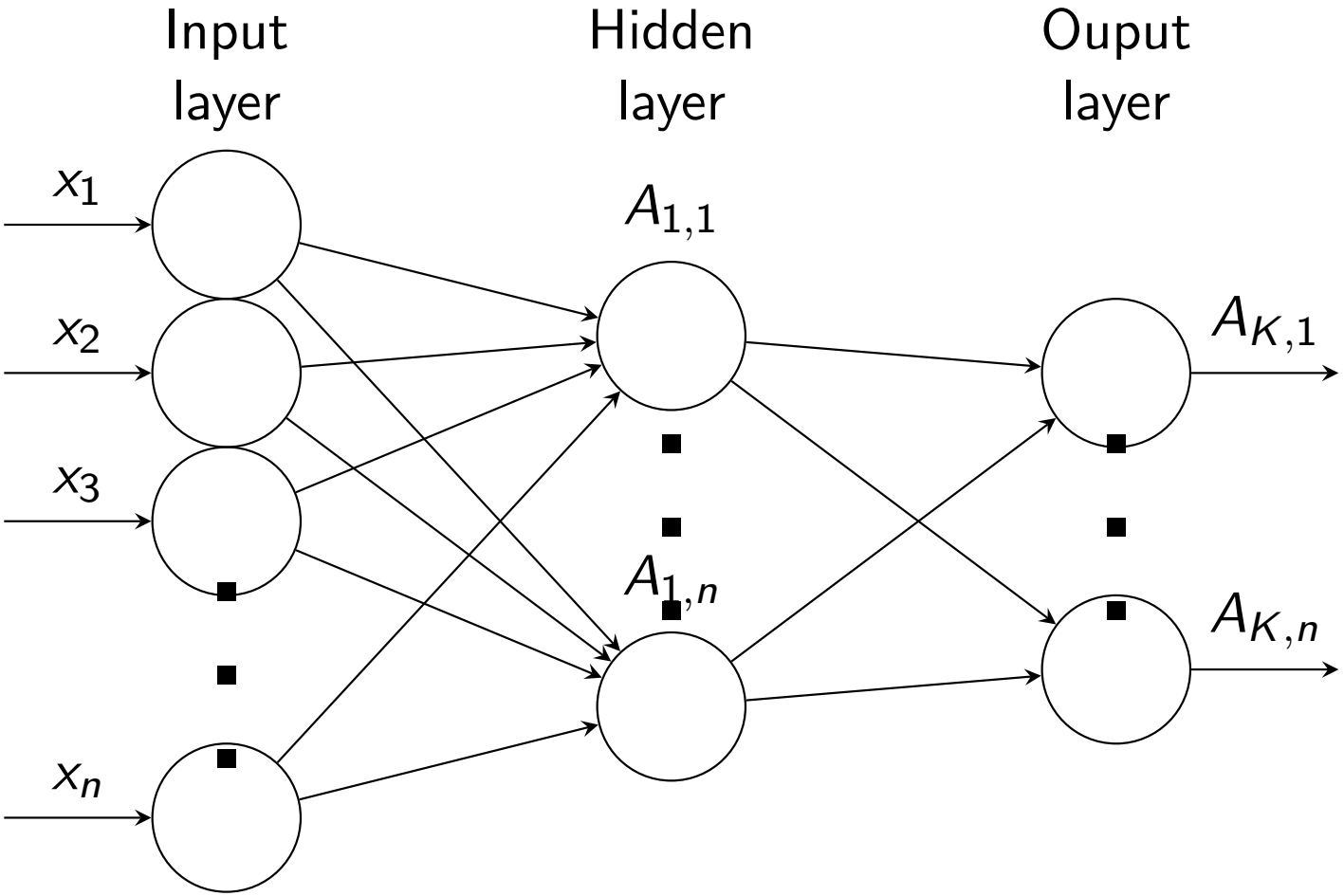
1. Deep learning
2. Variational Autoencoder
3. Huber energy distances and Huber energy VAE
4. Results

Deep learning

Definition (Wikipedia)

Deep learning is the subset of machine learning methods based on artificial neural networks (ANNs) with representation learning. The adjective "deep" refers to the use of multiple layers in the network. Methods used can be either supervised, semi-supervised or unsupervised.

Deep learning - Neural network



$$\Phi(\cdot|\theta) := A_K \circ \sigma_{K-1} \circ A_{K-1} \circ \sigma_{K-2} \circ \cdots \circ A_2 \circ \sigma_1 \circ A_1.$$

Variational Autoencoder - Generative AI

Figure: AI generated pictures

Suppose that we want to generate new human face pictures. One could consider that a human face is a realisation of some real distribution $\mathbb{P}^{real}$ on $\mathbb{R}^d$ with d being the dimension of the image.

Variational Autoencoder

Goal: Generate according to $\mathbb{P}^{real}$

Variational Autoencoder

Goal: Generate according to $\mathbb{P}^{real}$

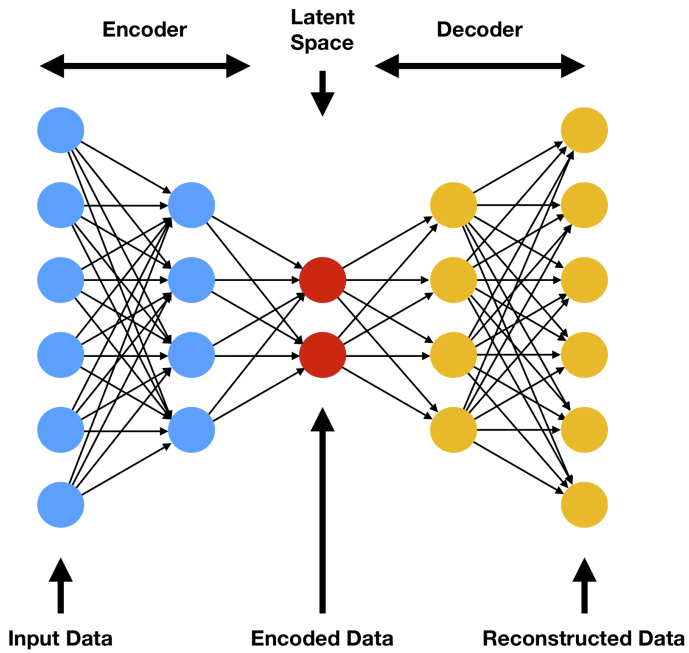


Figure: Varational Autoencoder structure

A VAE has two components : an encoder E_{θ_e} and a decoder D_{θ_d} where θ_e (resp. θ_d) are the parameters of the encoder (resp. decoder) neural network.

Variational Autoencoded

We will have two objectives:

1. With an input $X \sim \mathbb{P}^{real}$, we want $D_{\theta_d}(E_{\theta_e}(X)) \approx X$, so that the VAE is able to reconstruct X after being encoded in a lower dimension. It can be viewed as a non linear dimension reduction method.
2. We want $E_{\theta_e}(X)$ to follow the distribution μ, *ie* $(E_{\theta_e})_{\#}\mathbb{P}^{real} \approx \mu$.

If those are achieved, by generating some $Z \sim \mu$, then we have $D_{\theta_d}(Z) \sim \mathbb{P}^{real}$. We are then able to generate according to $\mathbb{P}^{real}$.

Variational Autoencoded - Loss function

One term per objective:

$$\hat{L}_{rec}(\theta_e, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} \left|X_i - D_{\theta_d}(E_{\theta_e}(X_i))\right|^2,$$

$$\hat{L}_{lat}(\theta_e) = d\left(\mu, \frac{1}{n}\sum_{i=1}^{n} \delta_{E_{\theta_e}(X_i)}\right)^2.$$

with d being a well chosen distance for distributions. The total loss is then:

$$\hat{L}_{tot}(\theta_e, \theta_d) = \hat{L}_{rec}(\theta_e, \theta_d) + \lambda \hat{L}_{lat}(\theta_e)$$

where $\lambda > 0$ is a parameter used for giving a weight to each term of the loss.

Desirable properties for the distance

1. Convexity: Let ν_0, ν_1 and μ three mesures. Let $t \in [0,1] \mapsto \nu_t$ a geodesic between ν_0 and ν_1. We would want the function $t \in [0,1] \mapsto d(\mu, \nu_t)$ to be convex.
2. Easy computability: The distance between a mean of dirac measure and some choosen distribution (for instance a $\mathcal{N}(0_k, I_k)$) has to be easily computable.

First idea: The norm of some dual of a Hilbert space H, with H being included in the space of continuous functions.
Issue: If we choose a Hilbert space $H^s(\mathbb{R}^k)$, for this space to be contained in $\mathcal{C}_b(\mathbb{R}^k, \mathbb{R})$, s needs to be at least $k/2$. This leads to a hardly computable distance.

Radon-Sobolev / Huber energy distances

Definition

Let H be a Hilbert space included in the set of bounded continuous functions $\mathcal{C}_b(\mathbb{R}, \mathbb{R})$, so that its topological dual space H' contains the set of probabilities on $\mathbb{R}$. Let $k \in \mathbb{N}$. Let $\mathcal{P}(\mathbb{R}^k)$ be the set of probability measures on $\mathbb{R}^k$. The Radon-Sobolev distance on $\mathcal{P}(\mathbb{R}^k)$ corresponding to H is defined by

$$d_H(\mu, \nu)^2 := \frac{1}{area(\mathbb{S}^{k-1})} \int_{\mathbb{S}^{k-1}} \|\theta_{\#}\mu - \theta_{\#}\nu\|_{H'} d\theta, \ \forall \mu, \nu \in \mathcal{P}(\mathbb{R}^k)$$

where $\theta_{\#}\mu$ is the one-dimensional projection of μ on $Vect(\theta)$. $\theta_{\#}\mu$ is a probability measure on $\mathbb{R}$ and then an element of H'.

Radon-Sobolev / Huber energy distances

Lemma (The convexity properties are satisfied!)

The distance d_H has the following properties :

1. *Any line $\nu_t = (1-t)\nu_0 + t\nu_1$ is a geodesic for d_H and d_H is convex on ν_t.*
2. *For any $\mu \in \mathcal{P}(\mathbb{R}^k)$ and any geodesic $(\nu_t)_{t\in[0,1]}$, we have*

$$d_H(\mu,\nu_t)^2 = (1-t)d_H(\mu,\nu_0)^2 + td_H(\mu,\nu_1)^2 - t(1-t)d_H(\nu_0,\nu_1)^2.$$

Radon-Sobolev / Huber energy distances

Lemma (Easy computability!)

By taking μ as the standard normal distribution on $\mathbb{R}^k$ and H being the Sobolev space $\dot{H}^1(\mathbb{R})$, we have the following approximation:

$$d_H\big(\mathcal{N}(0, I_k), \frac{1}{n}\sum_{i=1}^{n}\delta_{z_i}\big)^2$$
$$= c_{k_0} + \frac{1}{n}\sum_{i=1}^{n}\sqrt{|z_i|^2 + c_{k_1}} - \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}|z_i - z_j| + O(\sum_{i=1}^{n}|z_i|^4)$$

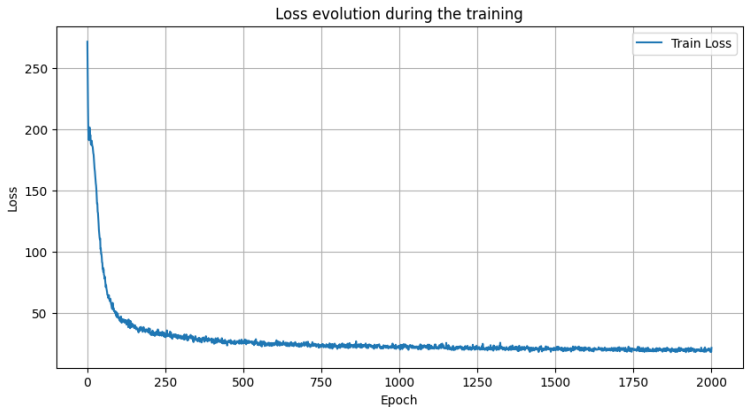
for some constants c_{k_0} and c_{k_1} detailled in the notebook.

We are then able to compute the latent loss term of our Huber-energy VAE. We have everything we need for its training.

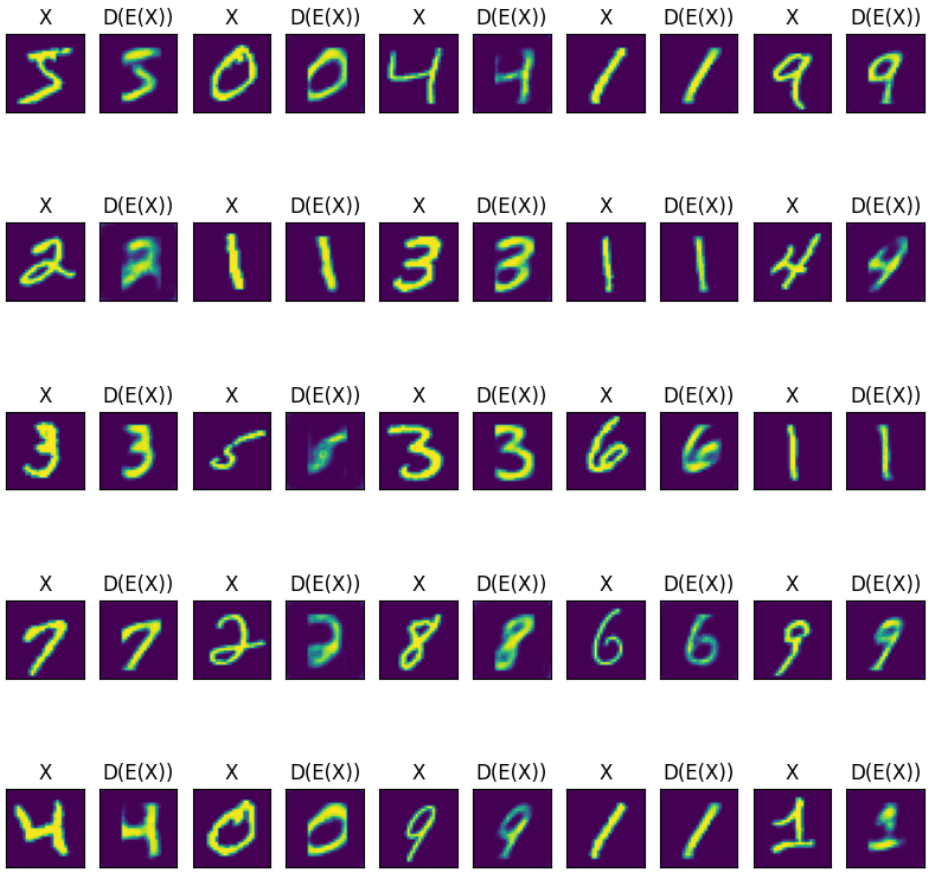
Programming the Huber-energy VAE

We made a Tensorflow v2 (eager mode) notebook that implements the model and trains it. The implementation does not depend on the precise choice of neural networks chosen as the encoder and decoder. It is also very easy to change the database on which it is trained.

Results - MNIST



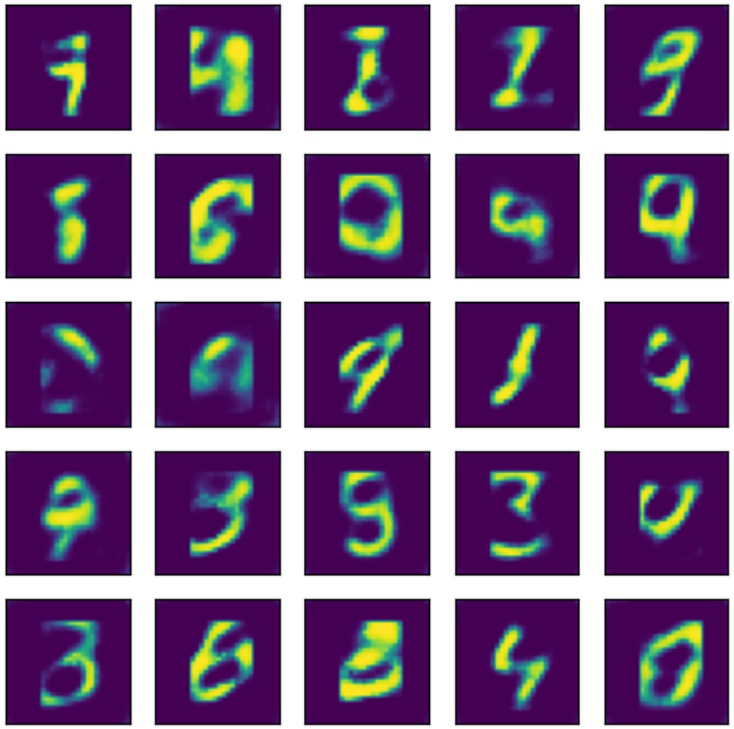
(a) Loss during the training



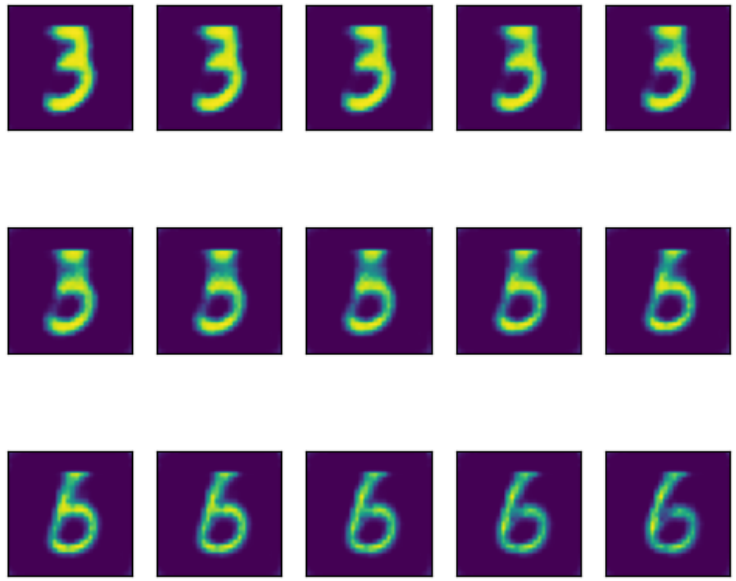
(b) Images and their reconstructed version

Figure: Test of the architecture and the parameters presented above for the MNIST dataset

Results - MNIST



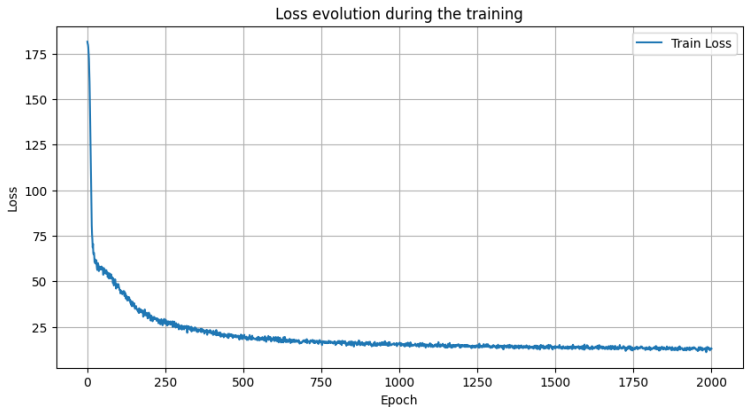
(a) Generated images



(b) Smooth transition between two images

Figure: Test of the architecture and the parameters presented above for the MNIST dataset

Results - MNIST (no latent loss)

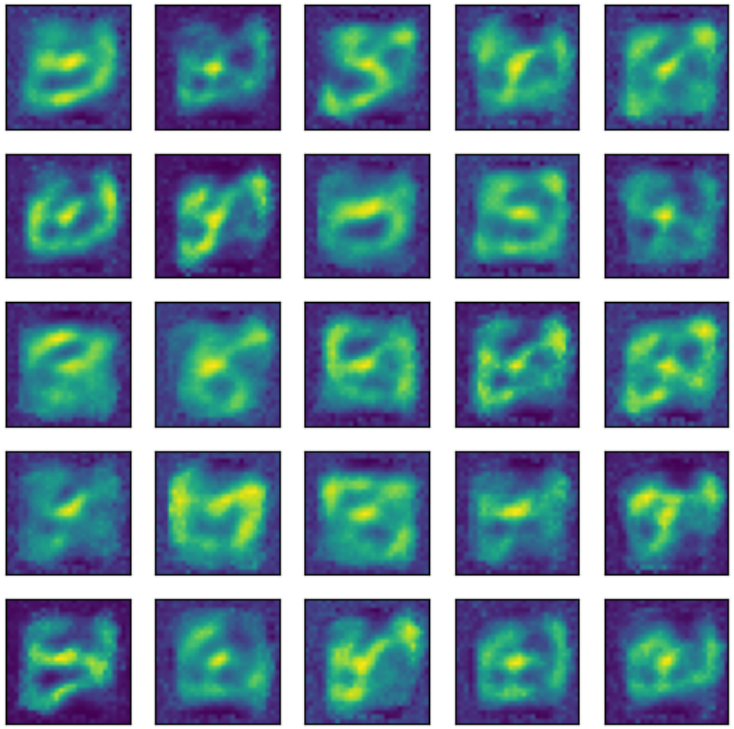


(a) Loss during the training

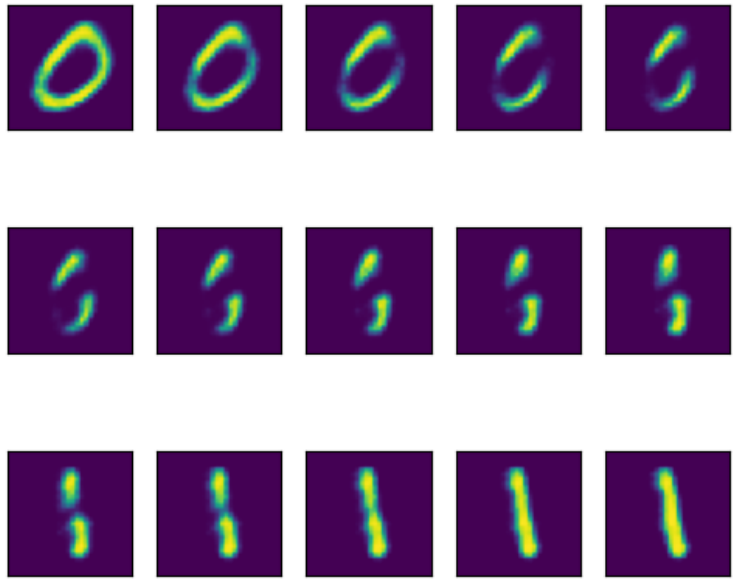
(b) Images and their reconstructed version

Figure: Test of the architecture and the parameters presented above for the MNIST dataset

Results - MNIST (no latent loss)



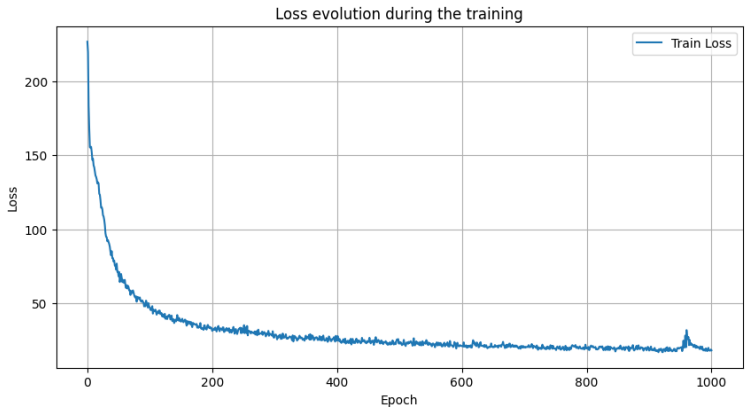
(a) Generated images



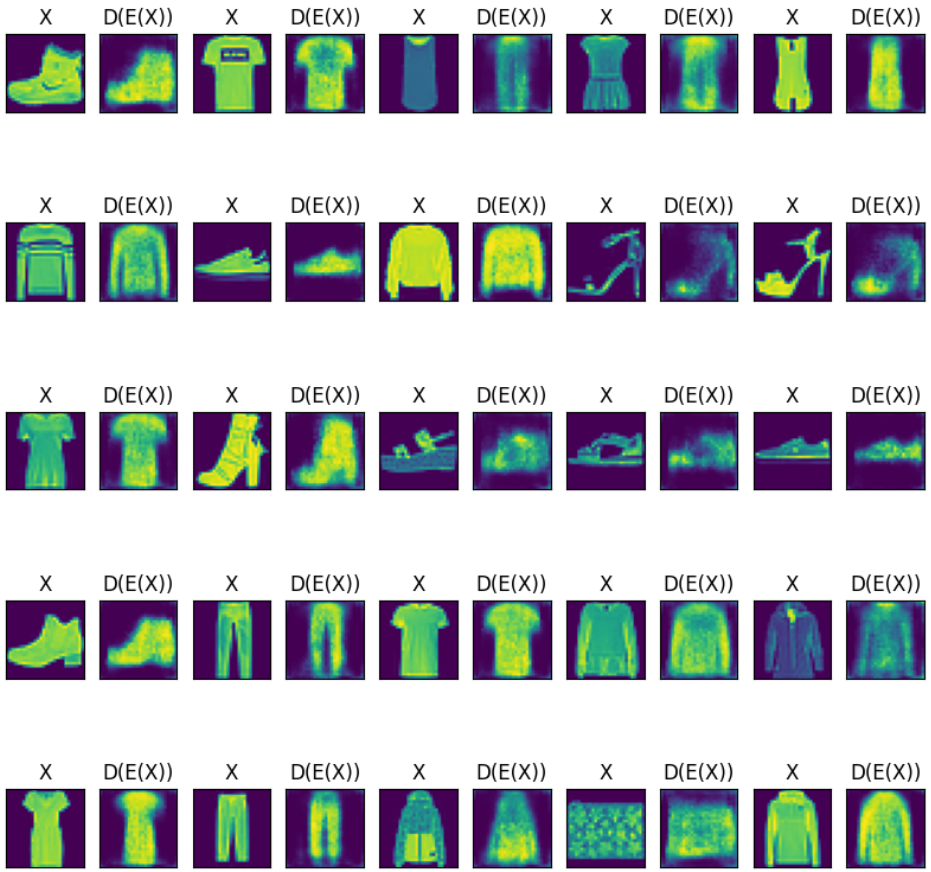
(b) Smooth transition between two images

Figure: Test of the architecture and the parameters presented above for the MNIST dataset

Results - Fashion-MNIST



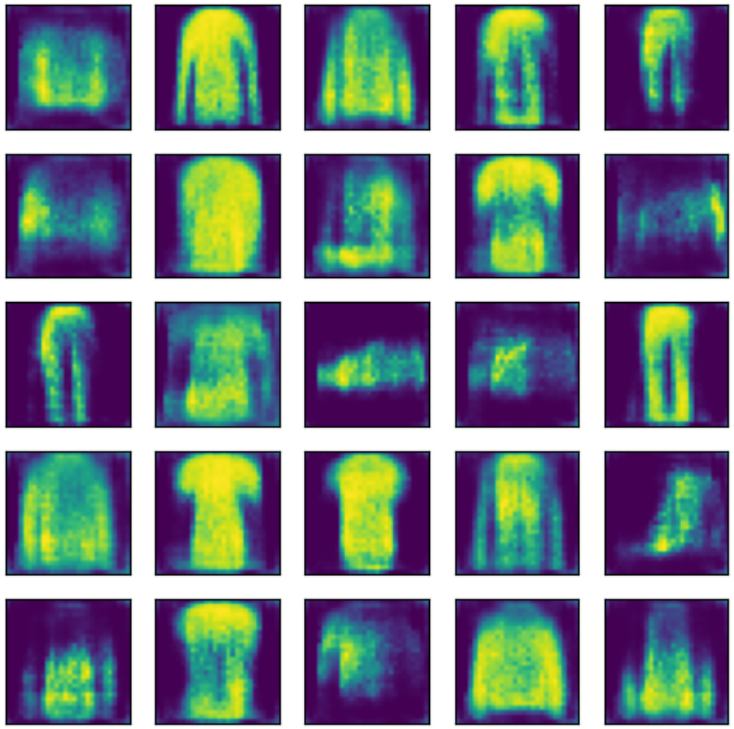
(a) Loss during the training



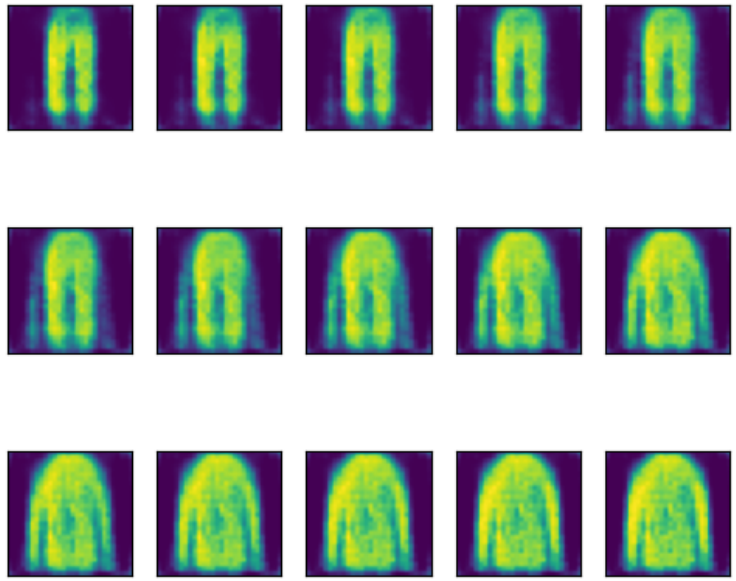
(b) Images and their reconstructed version

Figure: Test of the architecture and the parameters presented above for the Fashion-MNIST dataset

Results - Fashion-MNIST



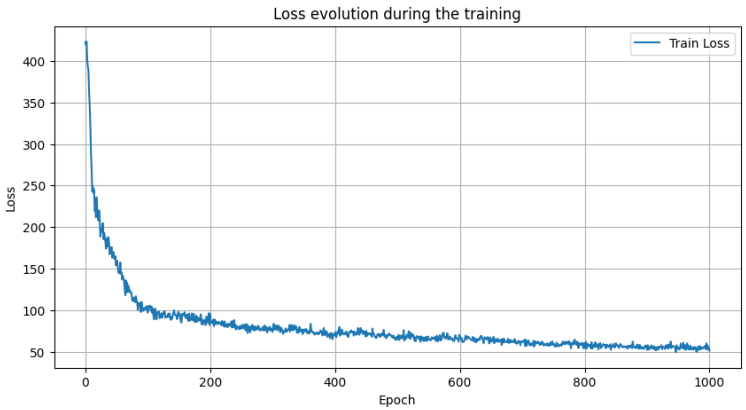
(a) Generated images



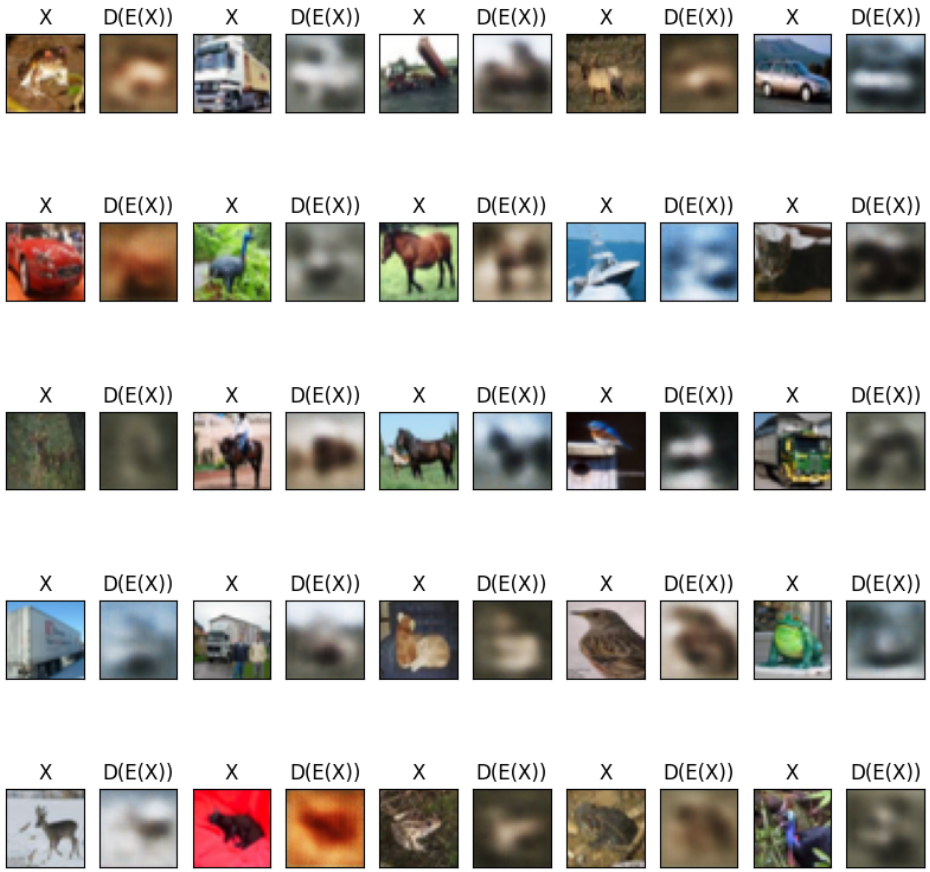
(b) Smooth transition between two images

Figure: Test of the architecture and the parameters presented above for the Fashion-MNIST dataset

Results - CIFAR-10



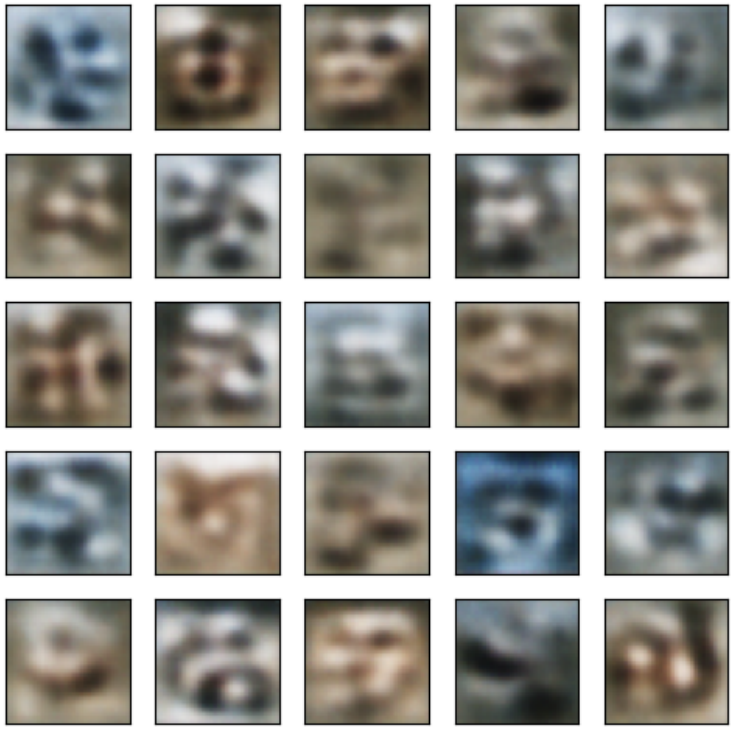
(a) Loss during the training



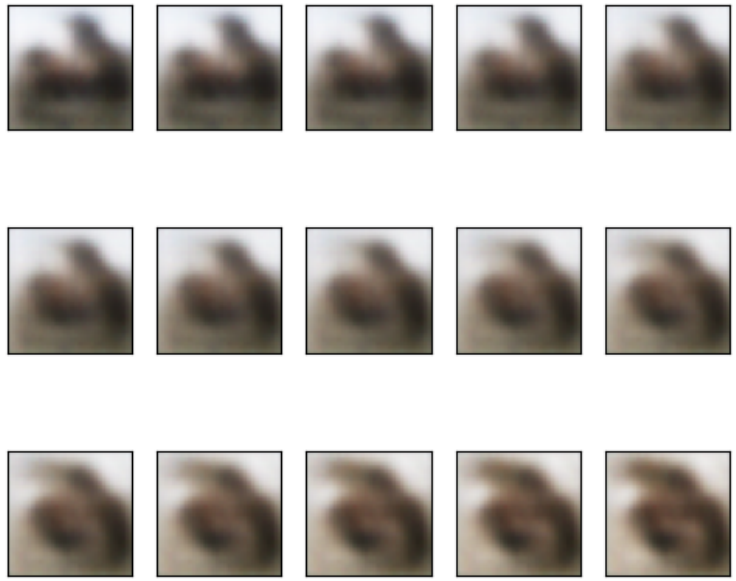
(b) Images and their reconstructed version

Figure: Test of the architecture and the parameters presented above for the CIFAR-10 dataset

Results - CIFAR-10



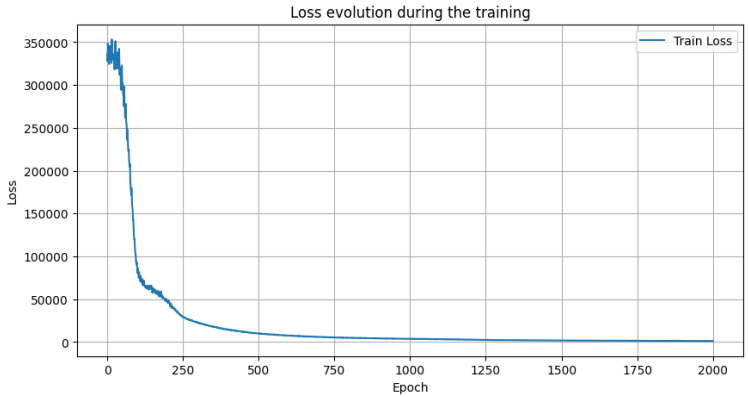
(a) Generated images



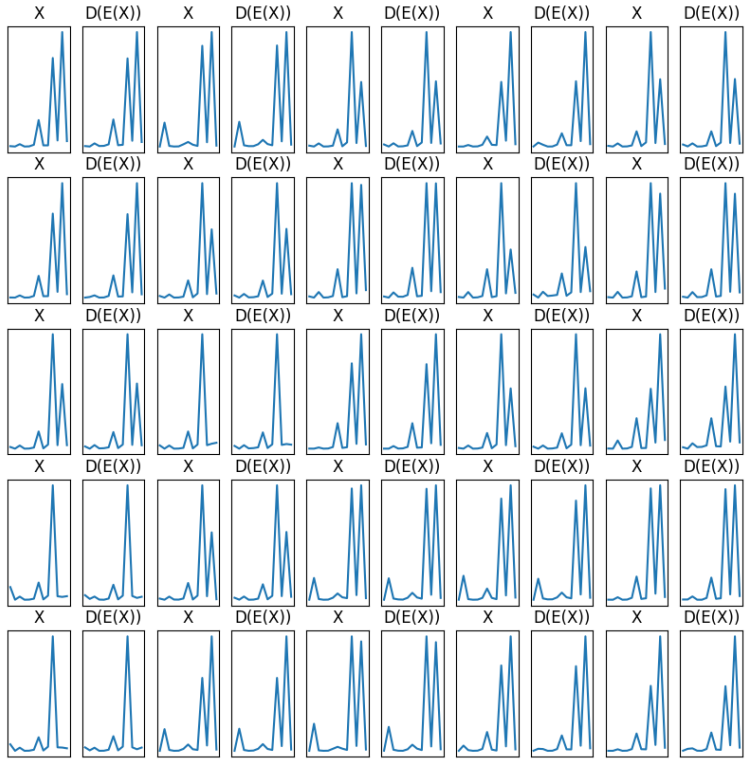
(b) Smooth transition between two images

Figure: Test of the architecture and the parameters presented above for the CIFAR-10 dataset

Results - Boston Housing



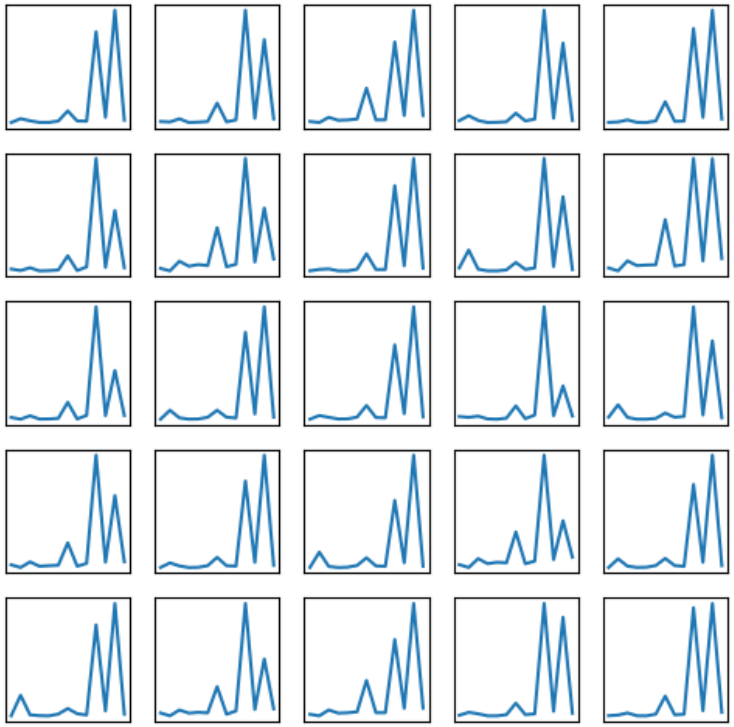
(a) Loss during the training



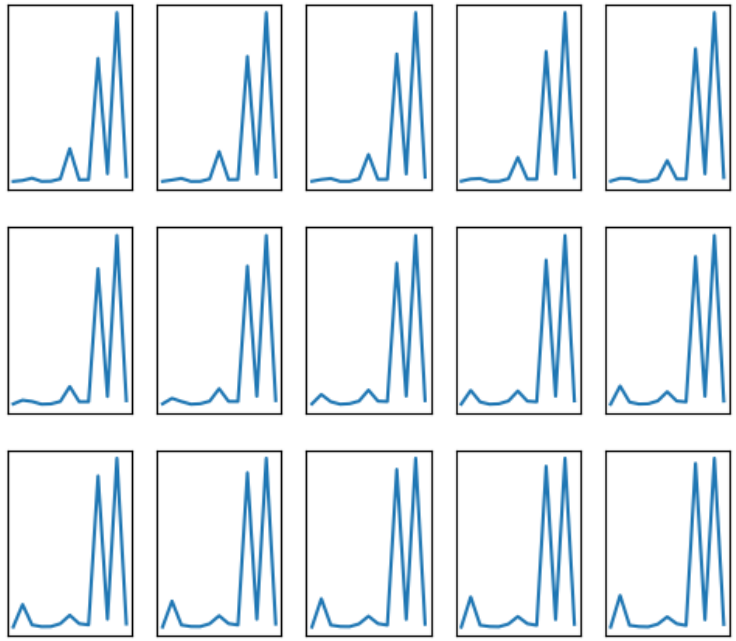
(b) Data and their reconstructed version

Figure: Test of the architecture and the parameters presented above for the Boston Housing dataset

Results - Boston Housing



(a) Generated data



(b) Smooth transition between two data

Figure: Test of the architecture and the parameters presented above for the Boston Housing dataset

Conclusion

We were pleased to discover a new way to learn complicated distributions and implement it. This follows in a certain way our last year's thesis which was about the approximation of Lipschitz functions using NNs (used for Wasserstein GANs).
We were able to realize that the choice of mathematical objects used in deep learning (in this case, the distance between distributions) is very important from a practical point of view. The search for optimal architectures is fascinating as it combines theoretical mathematical results and the experimental aspect of science.

References

Turinici, G. (2019).
X-ray sobolev variational auto-encoders.
CoRR, abs/1911.13135.

Turinici, G. (2024).
Diversity in Deep Generative Models and Generative AI, page 84–93.
Springer Nature Switzerland.