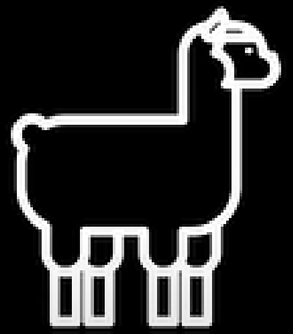
VARIATIONAL AUTOENCODER

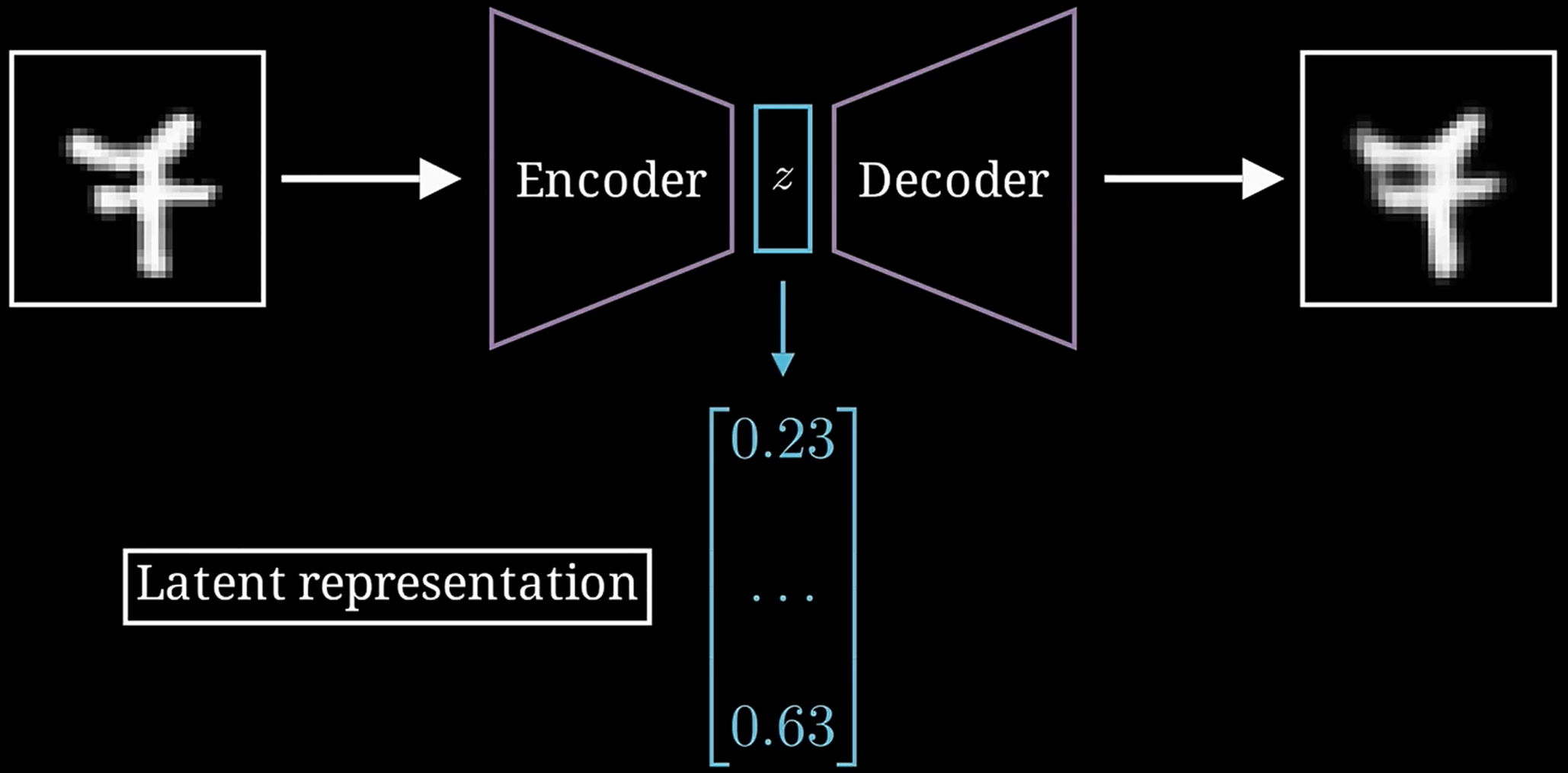
Quoi ? Pourquoi ? Comment ?

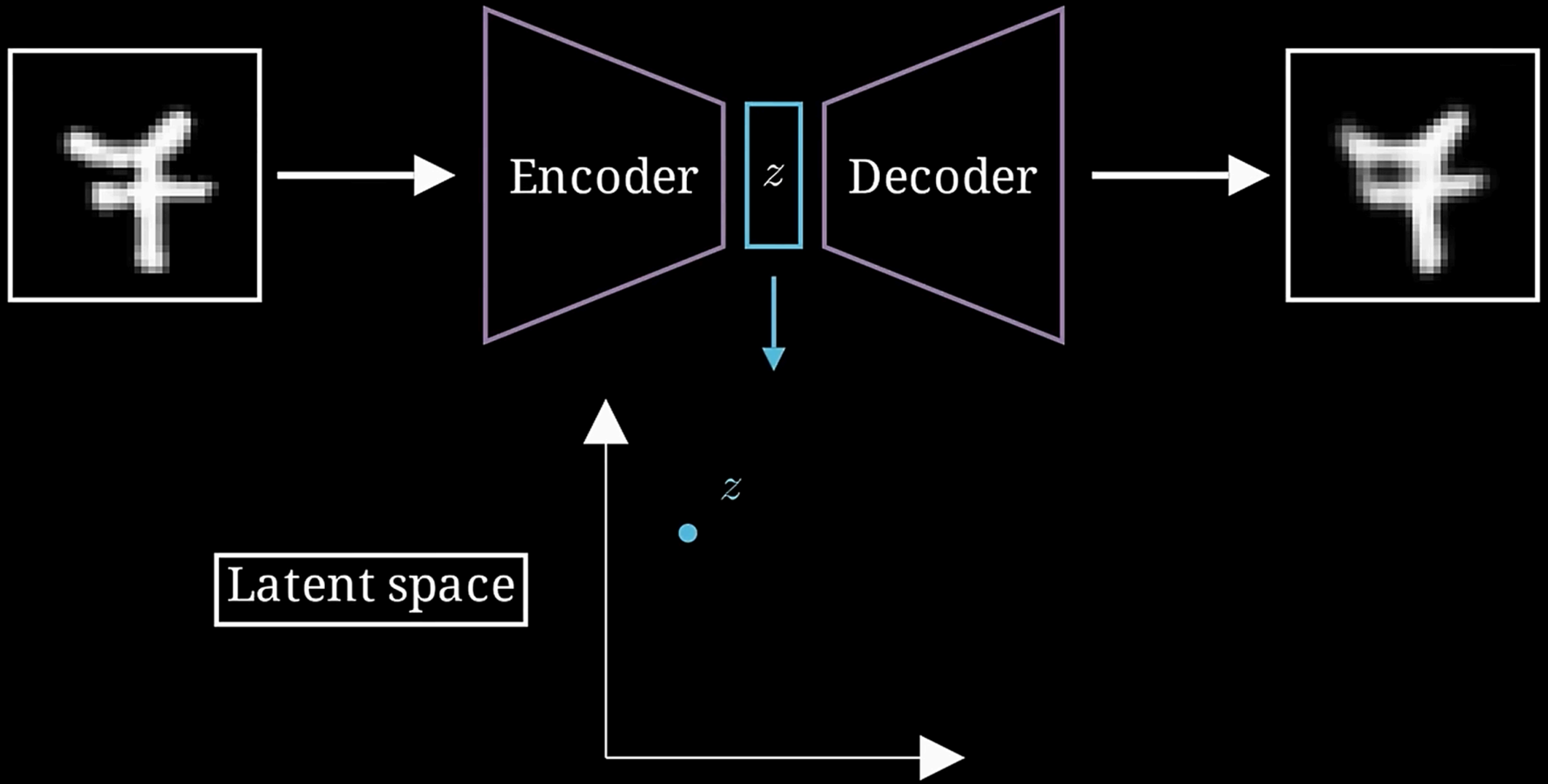
CYRIELLE MEGEMONT, ADRIEN LACROIX, ELIOT GALLAIS, DAMIEN GODELLE

2025

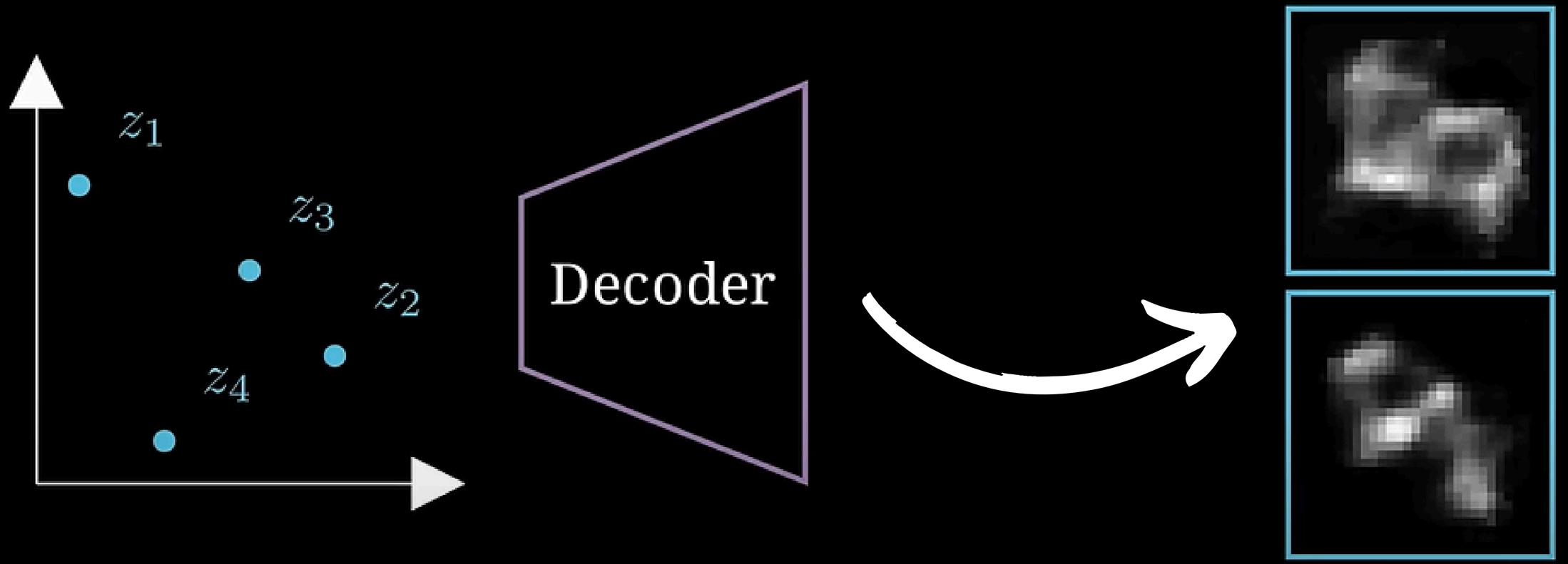


IA GÉNÉRATIVE

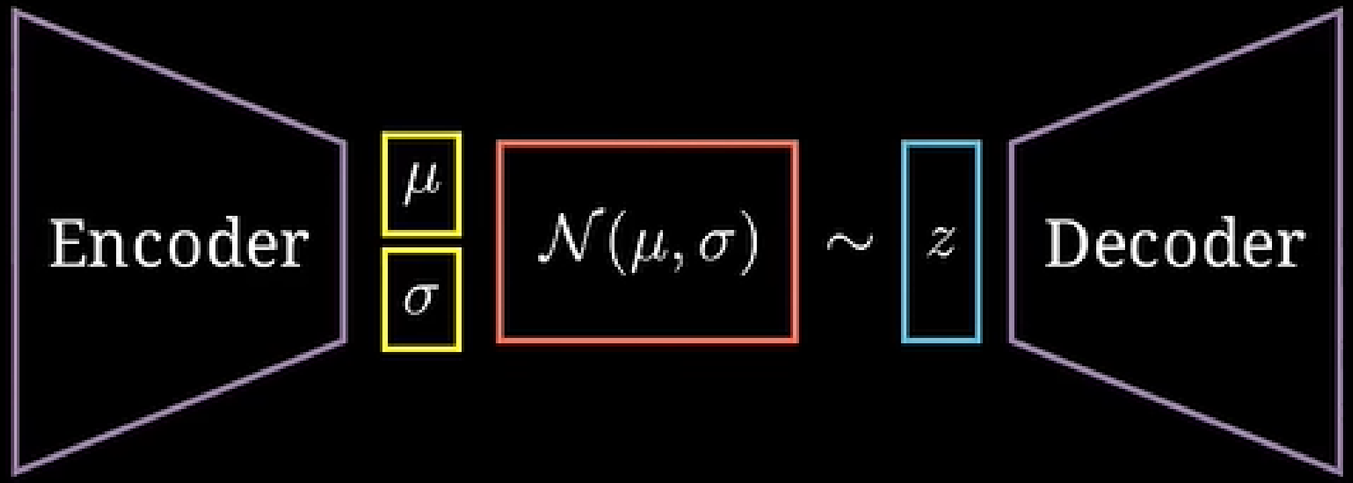

Encoder
z
Decoder
$\begin{bmatrix} 0.23 \\ \dots \\ 0.63 \end{bmatrix}$
Latent representation


Encoder
z
Decoder
z
Latent space

AUTOENCODER CLASSIQUE

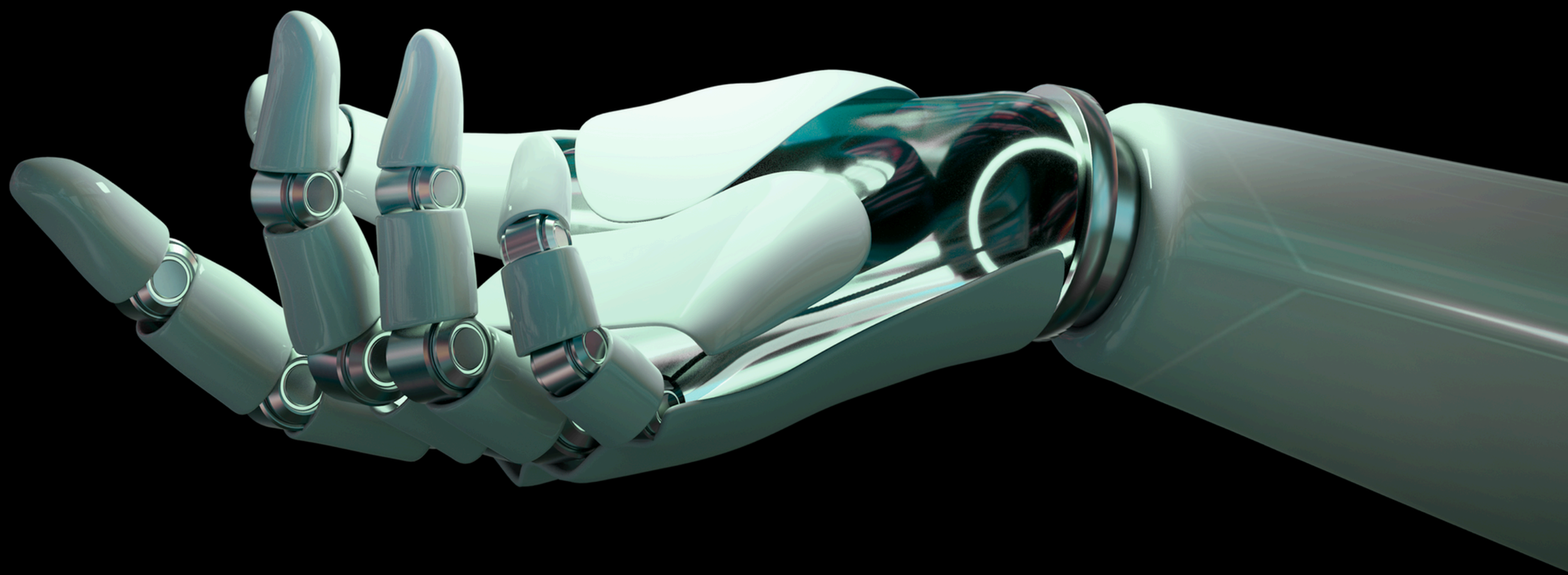


VARIATIONAL AUTOENCODER



Diederik P. Kingma

1 – Théorie : Comment fonctionne un VAE ?



AUTOENCODER

QU'EST CE QUE C'EST ?

Réseau de neurones non supervisé qui reconstruit des données d'origines après une compression.

Construit en deux parties : Encoder et Decoder

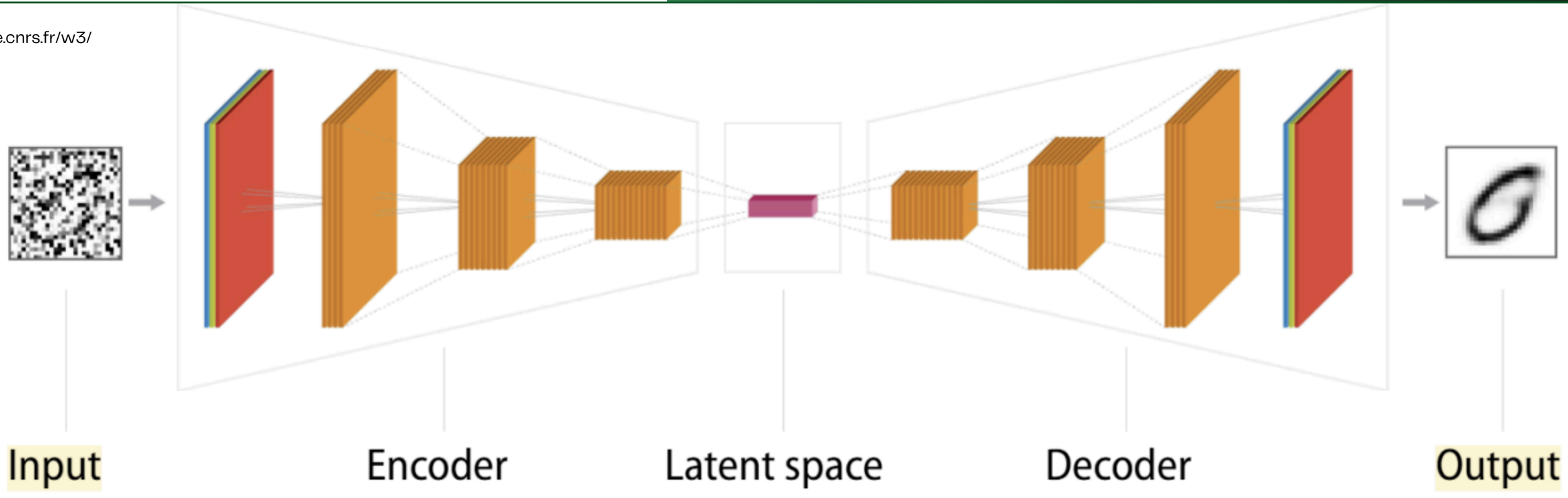
COMMENT ÇA MARCHE ?

- Encodeur : Compresse les données
- Espace latent : Contient les données compressées
- Décodeur : Reconstruit les données

À QUOI ÇA SERT ?

- Réduction dimension / Compression
- Détection d'anomalies
- Réduction du bruit

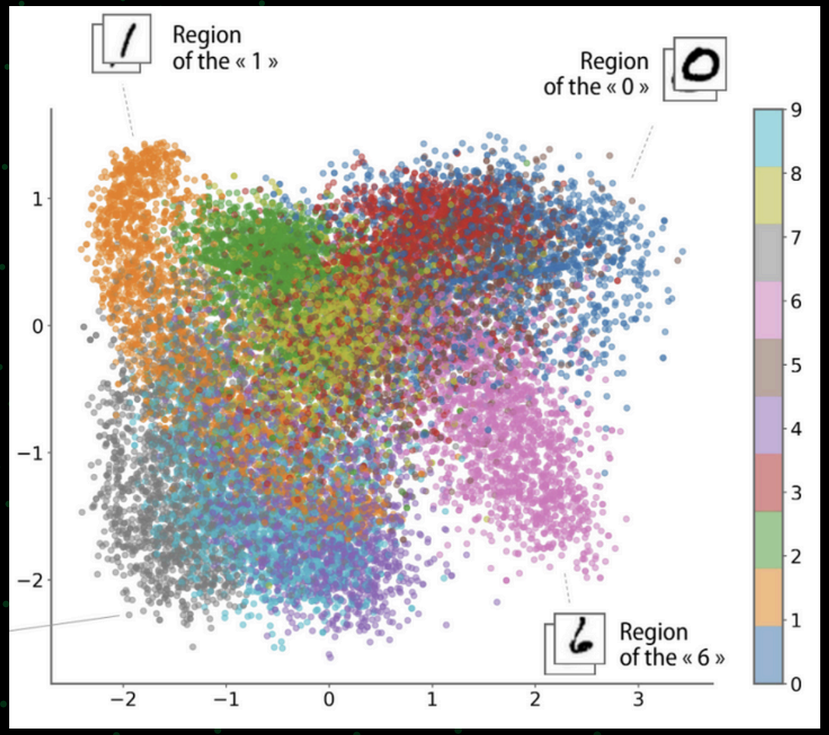
https://fidle.cnrs.fr/w3/



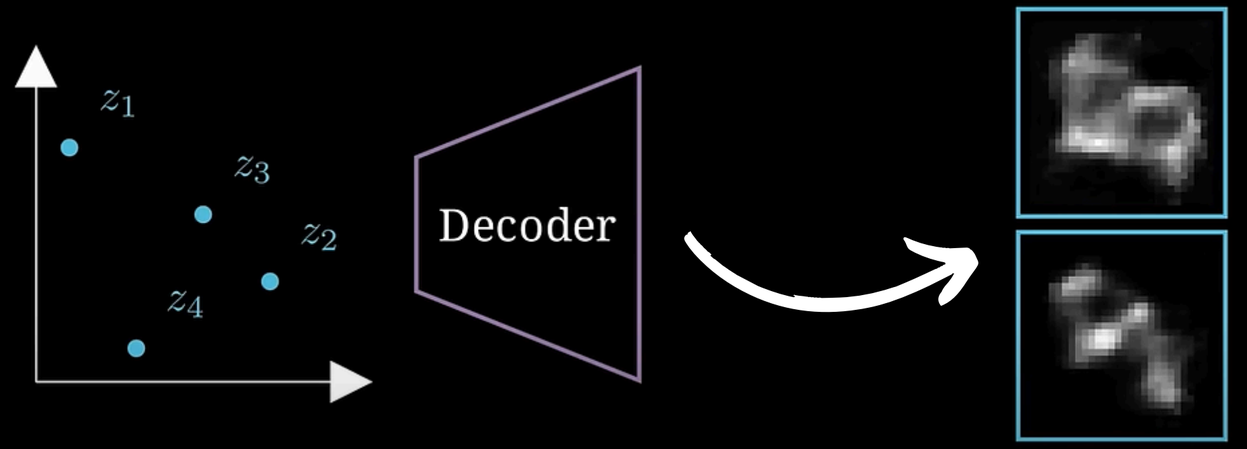
Principe du VAE

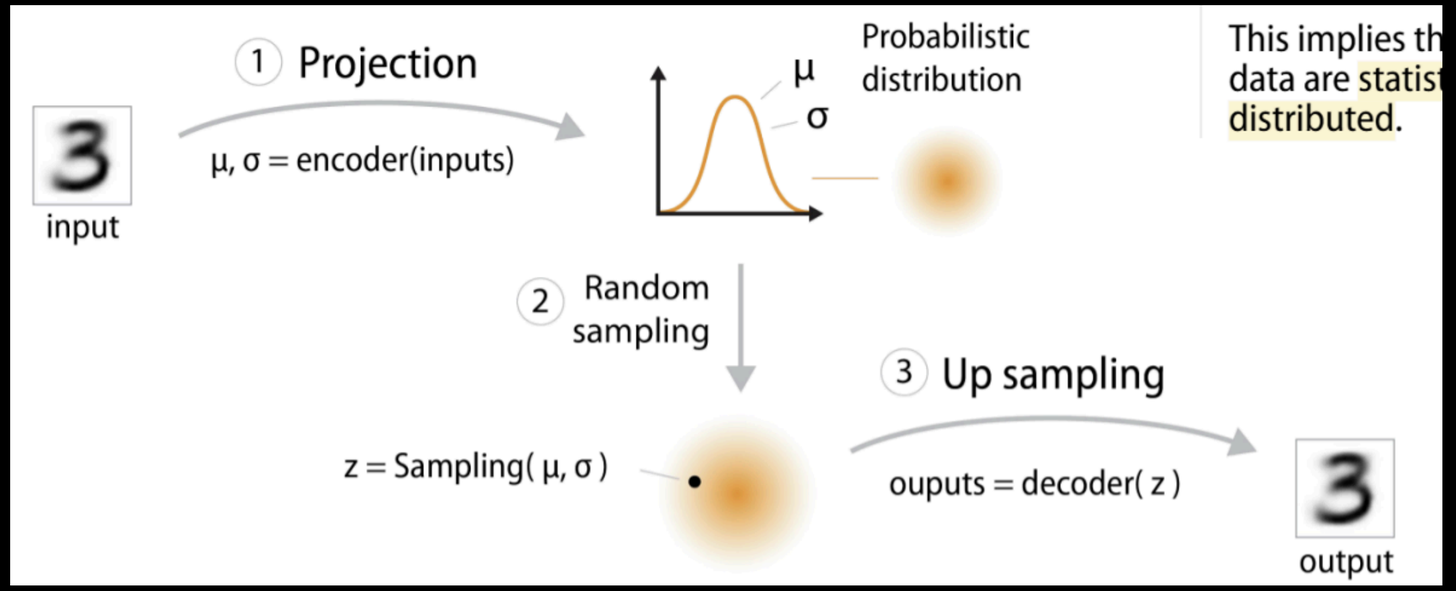
On a pas exploité l'espace latent !

Différence VAE/Autoencoder : Organiser le latent space afin de pouvoir générer des images sur demandes



AUTOENCODER CLASSIQUE

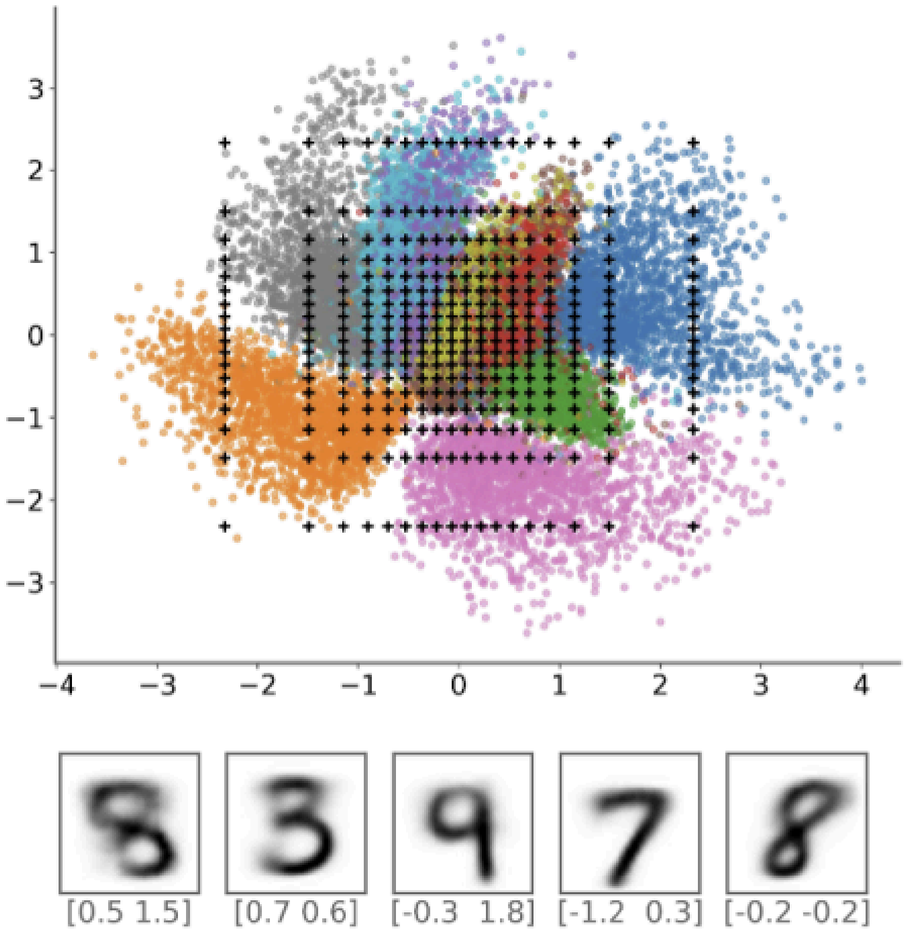


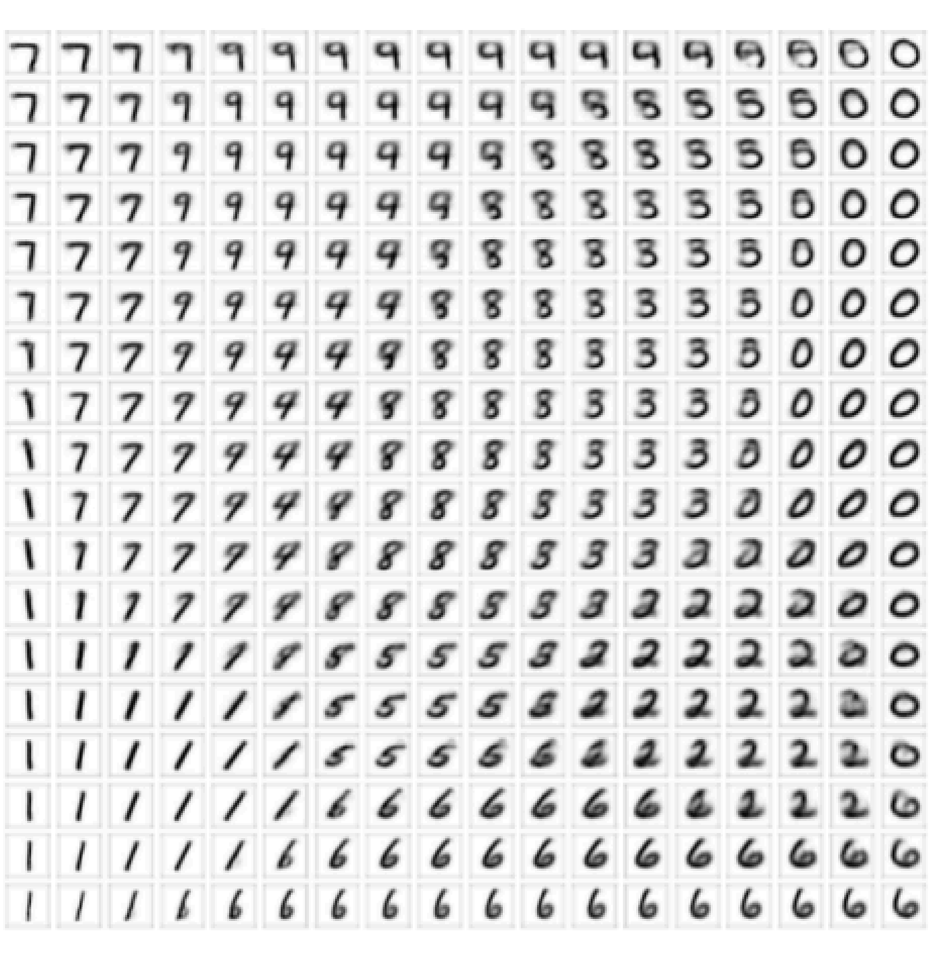


https://fidle.cnrs.fr/w3/

https://www.youtube.com/watch?v=qJeaCHQ1k2w

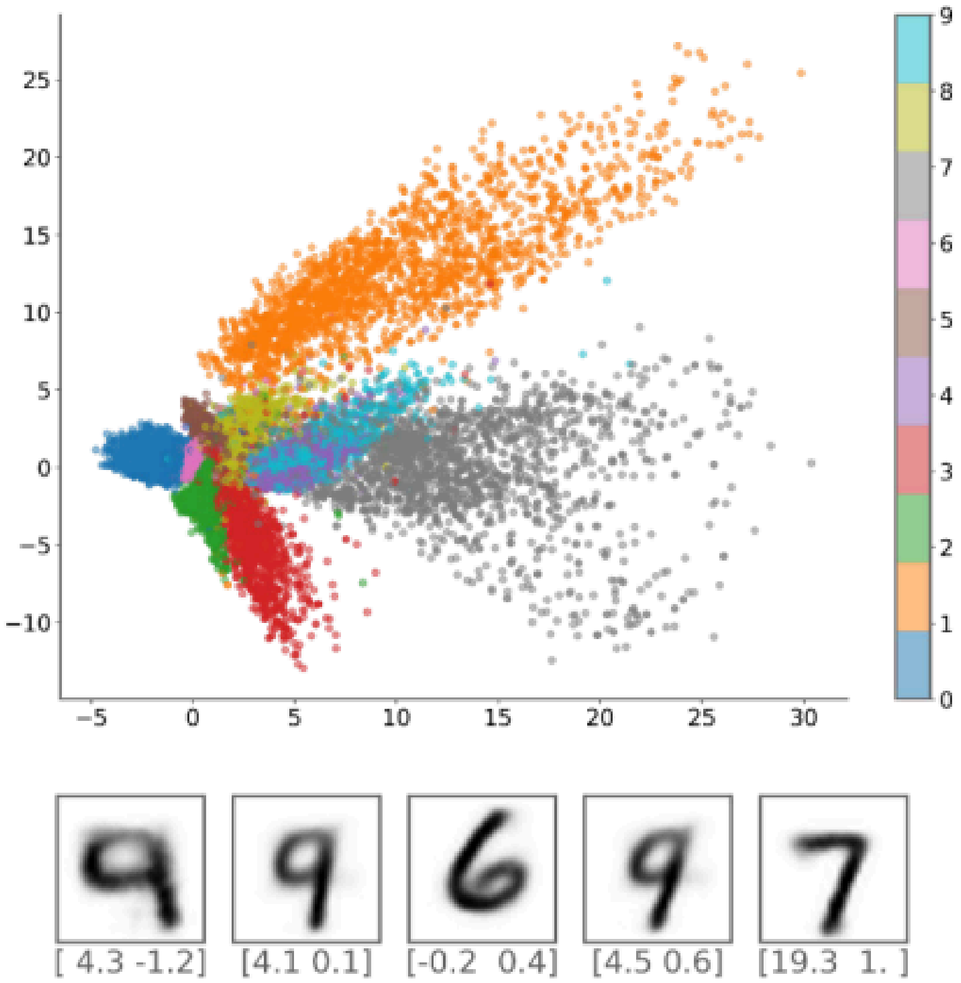
Principe du VAE

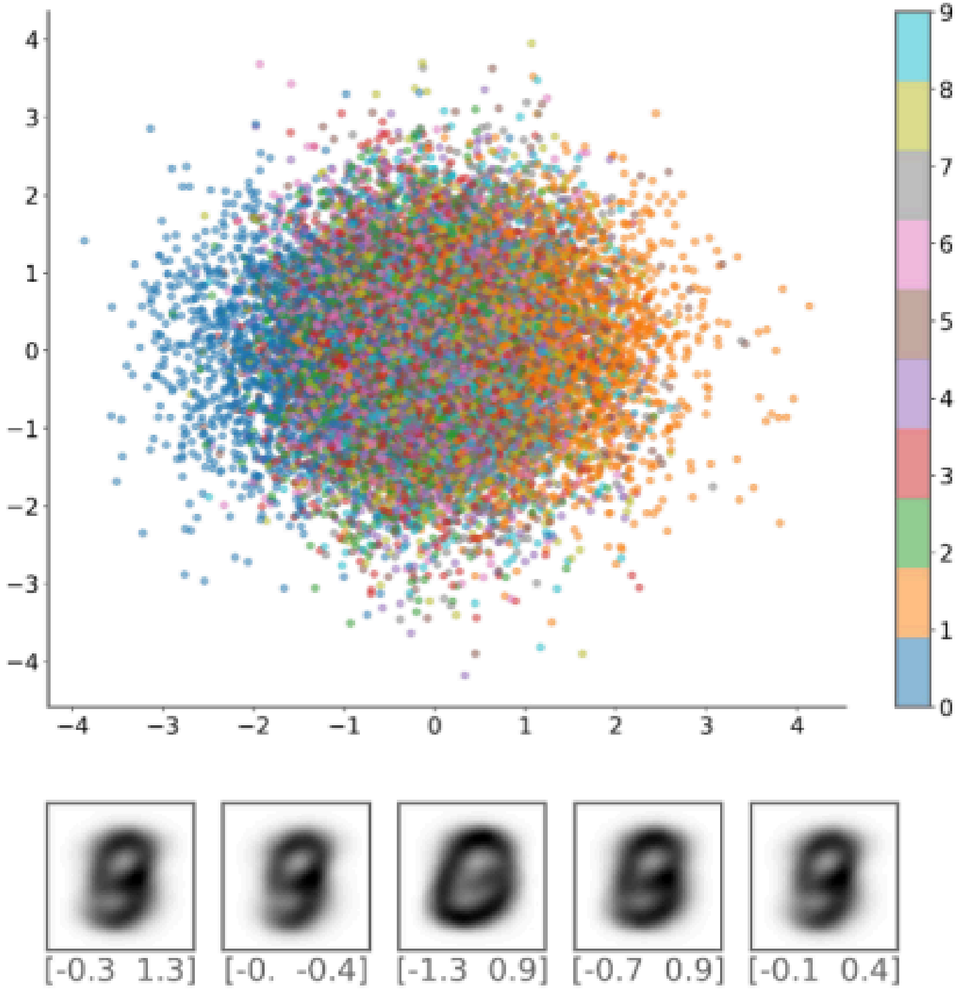




https://fidle.cnrs.fr/w3/

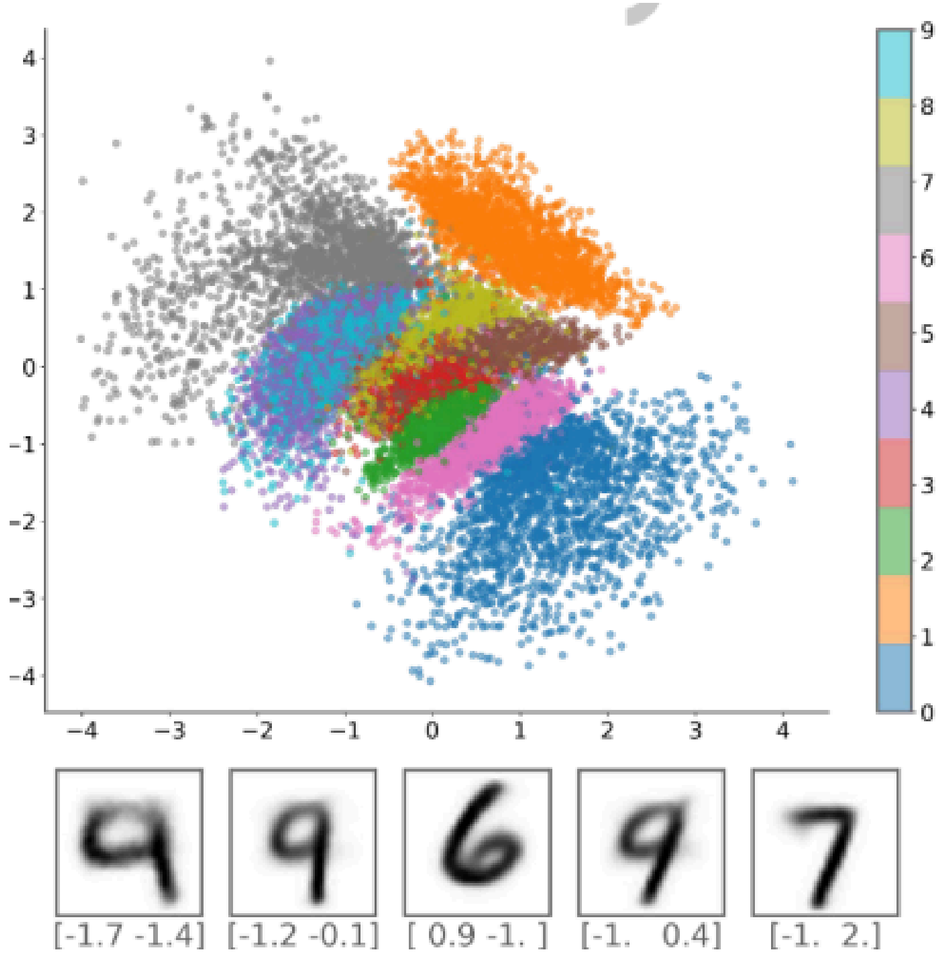
Perte utilisée – Quel objectif ?

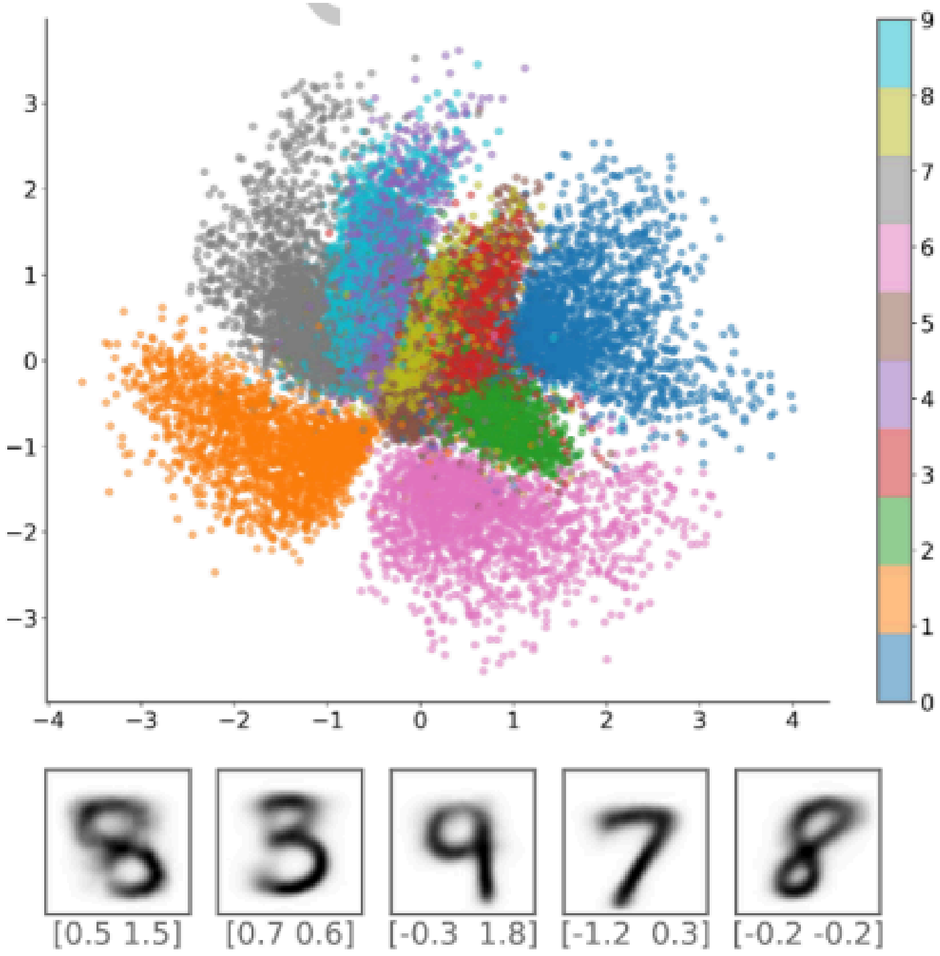




https://fidle.cnrs.fr/w3/

Perte utilisée – Quel objectif ?

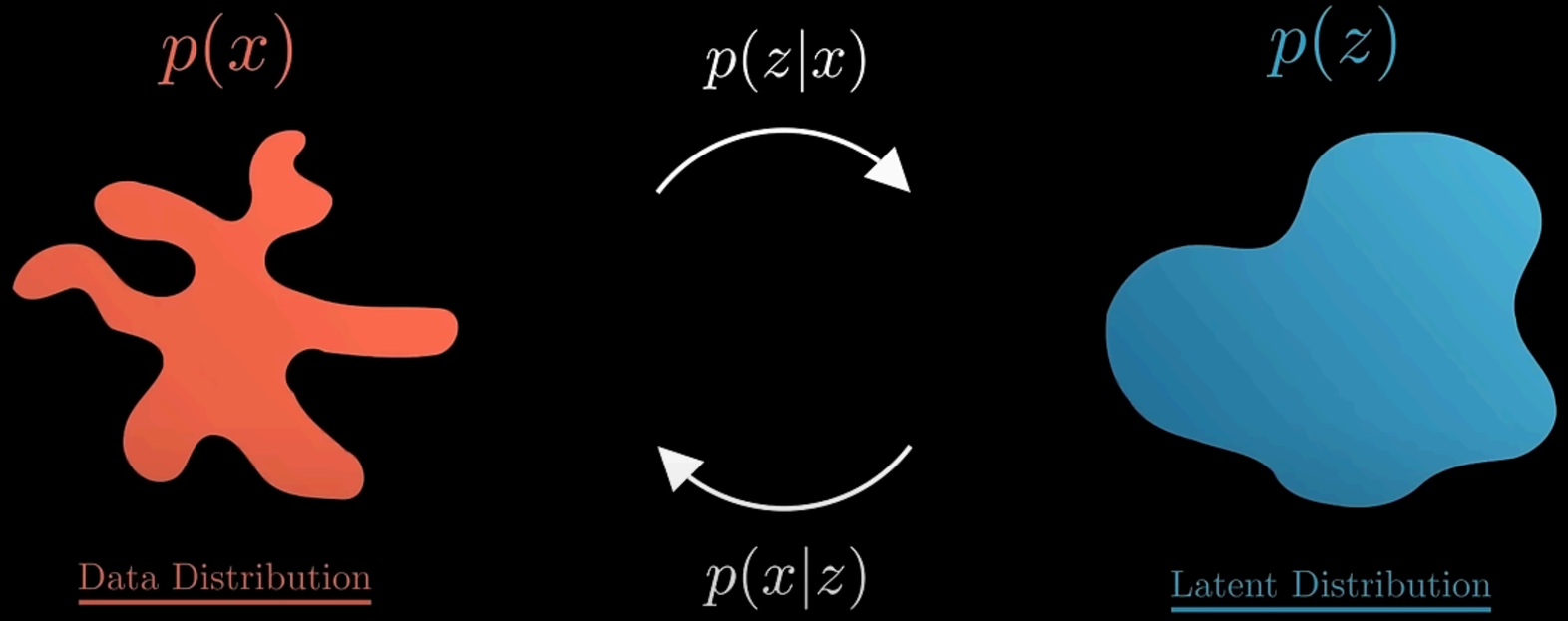




Nécessité d'un compromis

https://fidle.cnrs.fr/w3/

Perte utilisée

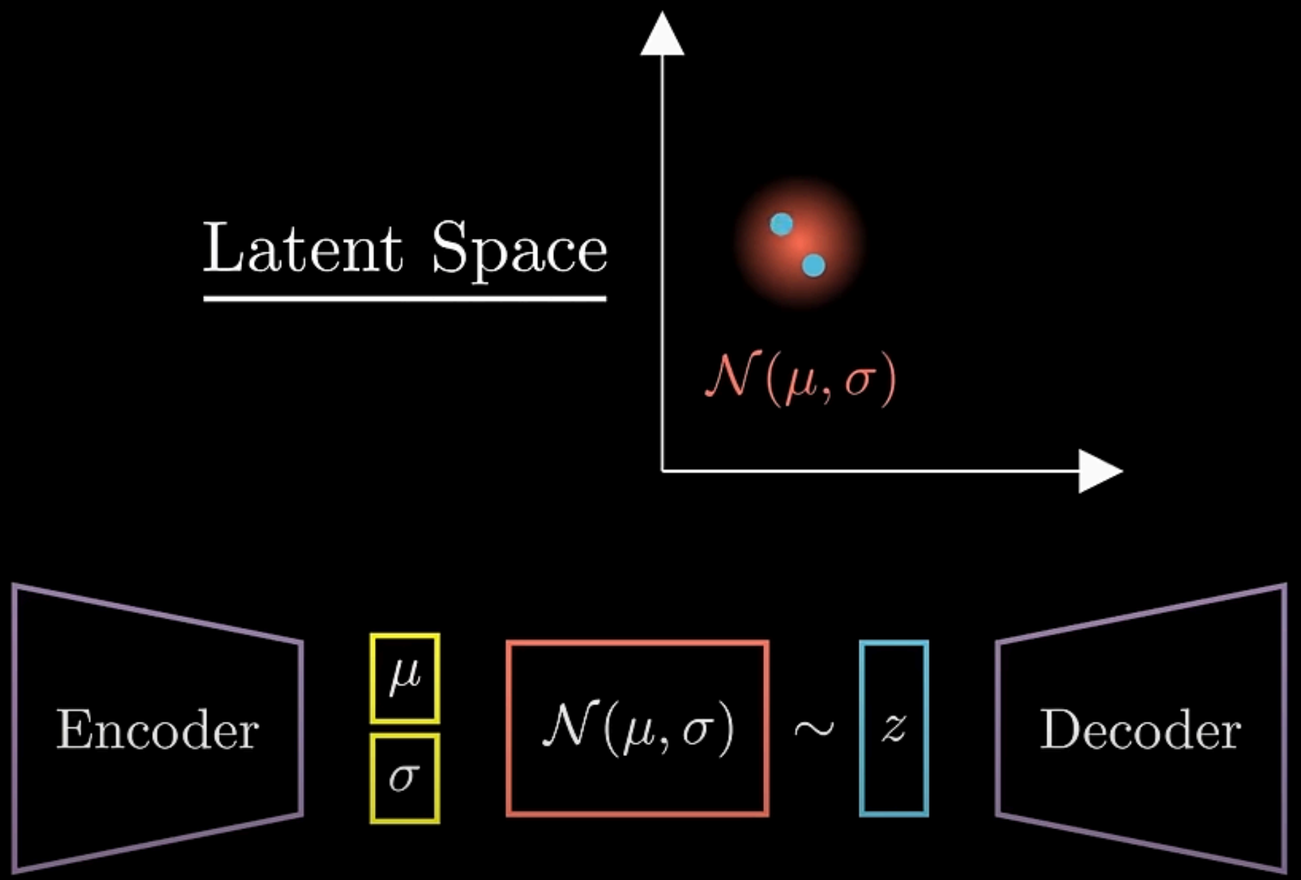


VARIATIONAL BAYES :

$$p(z|x) \approx q_{\mu,\sigma}(z|x) \sim \mathcal{N}(\mu(x), \sigma(x))$$

https://www.youtube.com/watch?v=qJeaCHQ1k2w

Perte utilisée



https://www.youtube.com/watch?v=qJeaCHQ1k2w

Perte utilisée

$$\mathcal{L}(x) = \underbrace{\mathbb{E}_{q(z|x)}\left[\log p(x|z)\right]}_{\text{Data consistency}} - \overbrace{\mathrm{KL}(q(z|x) \parallel p(z))}^{\text{Regularization}}$$

Avec :

- $D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$ La divergence de Kullback–Leibler
- $q(z|x) \sim \mathcal{N}(\mu(x), \sigma(x))$

Dérivable ?

Astuce de la Reparamétrisation

Soit une variable epsilon extérieure au réseau:

$$\epsilon \sim \mathcal{N}(0, 1)$$

on définit alors :

$$z = \mu + \sigma \times \epsilon$$

Take, for example, the univariate Gaussian case: let $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$. In this case, a valid reparameterization is $z = \mu + \sigma\epsilon$, where ϵ is an auxiliary noise variable $\epsilon \sim \mathcal{N}(0, 1)$. Therefore, $\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}[f(z)] = \mathbb{E}_{\mathcal{N}(\epsilon;0,1)}[f(\mu + \sigma\epsilon)] \simeq \frac{1}{L}\sum_{l=1}^{L} f(\mu + \sigma\epsilon^{(l)})$ where $\epsilon^{(l)} \sim \mathcal{N}(0, 1)$.

Avec $f = log(p(x|\cdot))$

Auto-Encoding Variational Bayes
DP Kingma, M Welling

Astuce de la Reparamétrisation

J la dimension de l'espace latent

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} | \mathbf{z}^{(i,l)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \qquad (10)$$

Auto-Encoding Variational Bayes
DP Kingma, M Welling

Exemple avec PyTorch

Structure globale du projet :

```
project/
├── data/                    # Contient les données MNIST
├── notebooks/               # Contient les notebooks Jupyter
│   ├── main_notebook.ipynb
├── scripts/                 # Contient le fichier data_loader.py
│   ├── data_loader.py
├── demo/
│   ├── Lower_Bound_Demonstration.pdf
├── pres/
│   ├── presentation.pdf
```

Exemple avec PyTorch

Structure de l’architecture du modèle :

- **Encodeur :**
  - Prend une image 28x28 en entrée
  - Réduis les dimensions à un espace latent (moyenne et variance)
- **Reparamétrisation** :
  - Echantillone z à partir de la moyenne et de la variance
- **Décodeur :**
  - Reconstruit l’image originale à partir de z

Exemple avec PyTorch

Calcul étape par étape :

- **Encodeur :**
  - Transformation des images en vecteurs latents
  - Utilisation de couches Linear et ReLU pour encoder les données.

- **Décodeur** :
  - Transformation des vecteurs z en images reconstruites.
  - Utilisation de couches Linear et Sigmoid.

```python
# Définition du modèle
class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()
        # Encoder
        self.encoder = nn.Sequential(
            nn.Flatten(),
            nn.Linear(28 * 28, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.ReLU(),
        )
        self.fc_mean = nn.Linear(256, latent_dim)
        self.fc_log_var = nn.Linear(256, latent_dim)

        # Decoder
        self.decoder_input = nn.Linear(latent_dim, 256)
        self.decoder = nn.Sequential(
            nn.ReLU(),
            nn.Linear(256, 512),
            nn.ReLU(),
            nn.Linear(512, 28 * 28),
            nn.Sigmoid()
        )
```

Exemple avec PyTorch

- **Reparamétrisation :**
  - Transformation des images en vecteurs latents.

```python
def reparameterize(self, z_mean, z_log_var):
    std = torch.exp(0.5 * z_log_var)
    epsilon = torch.randn_like(std)
    return z_mean + epsilon * std
```

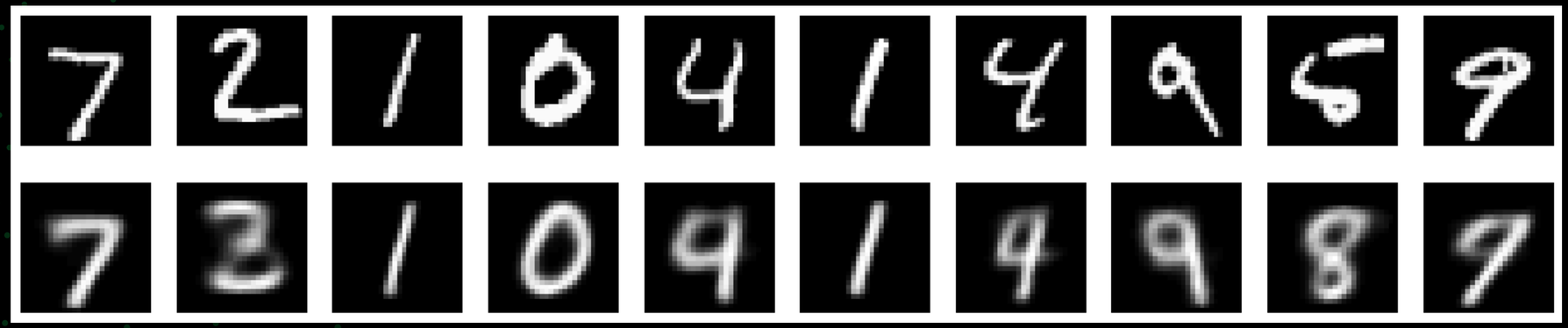
- **Calcul de la perte :**
  - Reconstruction : Erreur binaire croisée
  - Régularisation : Divergence KL

```python
# Perte pour le VAE
def vae_loss(reconstructed, original, z_mean, z_log_var):
    reconstruction_loss = nn.functional.binary_cross_entropy(reconstructed, original, reduction="sum")
    kl_divergence = -0.5 * torch.sum(1 + z_log_var - z_mean.pow(2) - z_log_var.exp())
    return reconstruction_loss + kl_divergence
```

Exemple avec PyTorch

Applications :

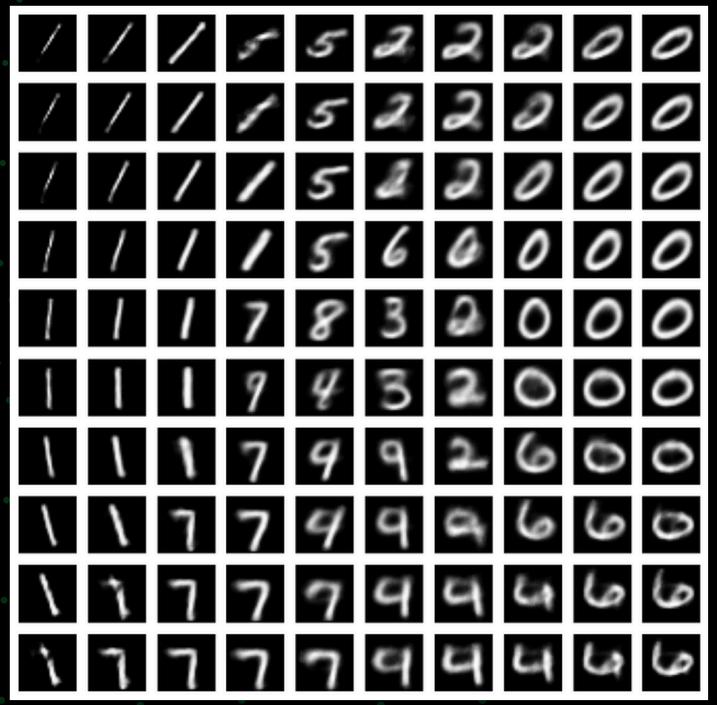
- **Reconstruction d’images :**
  - Les images MNIST sont passées dans l'encoder et le decoder.
  - Images reconstruites fidèles avec une légère perte de détails.

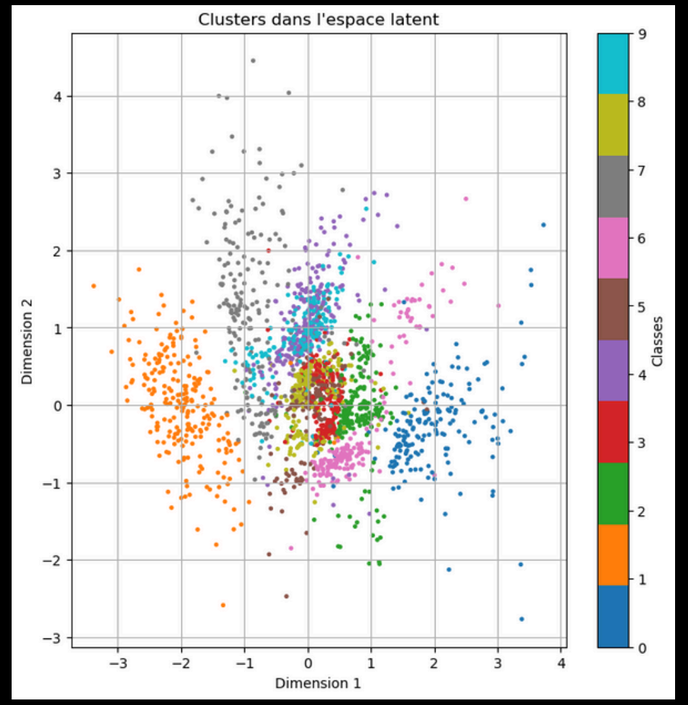


Exemple avec PyTorch

Applications :

- **Génération d'un dataset :**
  - Génération d'une grille uniforme de points z dans l'espace latent.
  - Points sont décodés pour produire des images synthétiques.

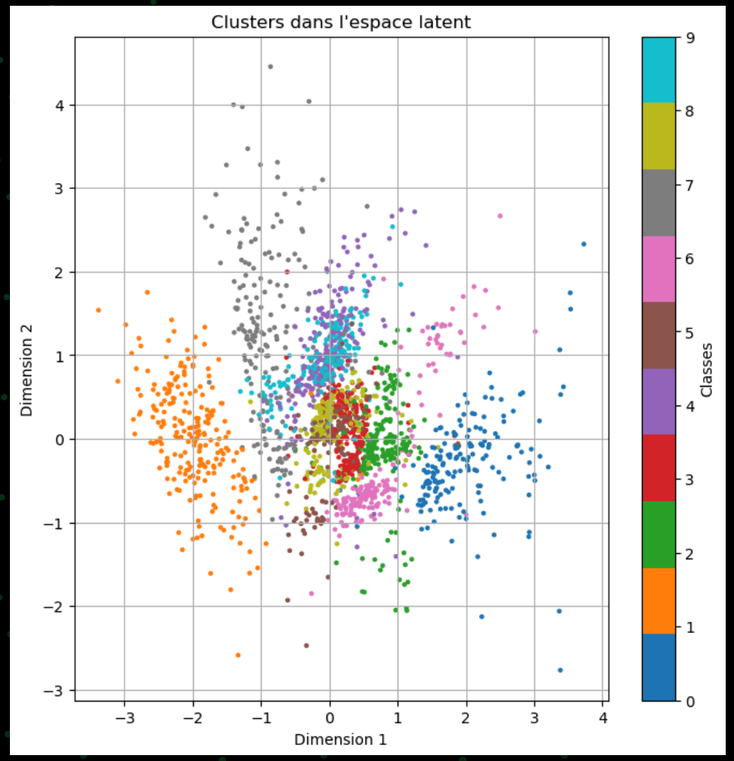


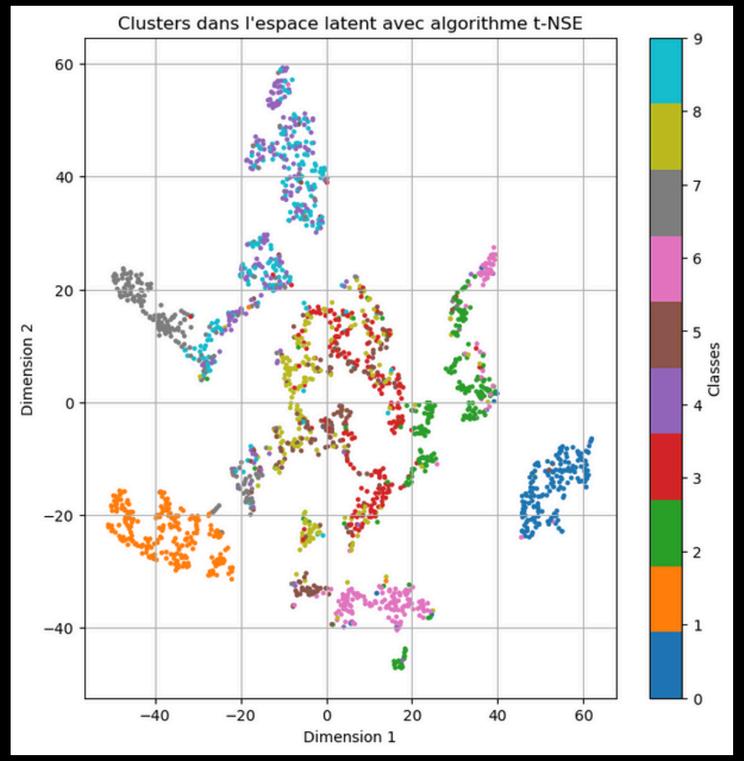


Exemple avec PyTorch

Applications :

- En utilisant l'algorithme t-SNE :
  - Possibilité de faire apparaître les clusters de façon plus nette





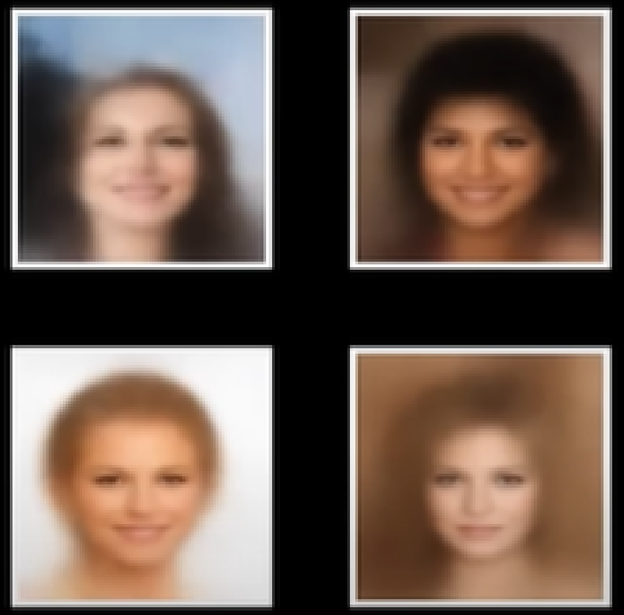
https://cedric.cnam.fr/vertigo/Cours/ml/docs/6coursReductionDimension.pdf

Limites

CelebA dataset

Reconstructions

VAE Samples



GAN Samples

Diffusion samples

Des améliorations :

- **CVAE :**
  - Ajout d’une condition c (comme une classe par exemple)
- **Béta-VAE** :
  - Intriduction d’un Hyperparamètre dans la fonction de perte
- **VQ-VAE :**
  - Permet de travailler dans un espace latent discret

CONCLUSIONS

Thank You!