Autoencoders (AE)

Jérémie Cabessa

Laboratoire DAVID, UVSQ

Introduction

- **But d'un autoencodeur:** projeter les data dans un espace de dimension plus petite – l'**espace latent (latent space)** – et être capable de les reconstruire ces data.
- Ceci revient à chercher un nombre restreint de degrés de libertés qui décrivent les data (cf. algorithme PCA).
- On est donc dans un contexte d'**apprentissage non-supervisé (unsupervised learning)**.
- **Applications:** compression, débruitage (denoising), etc.

Introduction

- **But d'un autoencodeur:** projeter les data dans un espace de dimension plus petite – l'**espace latent (latent space)** – et être capable de les reconstruire ces data.
- Ceci revient à chercher un nombre restreint de degrés de libertés qui décrivent les data (cf. algorithme PCA).
- On est donc dans un contexte d'**apprentissage non-supervisé (unsupervised learning)**.
- **Applications:** compression, débruitage (denoising), etc.

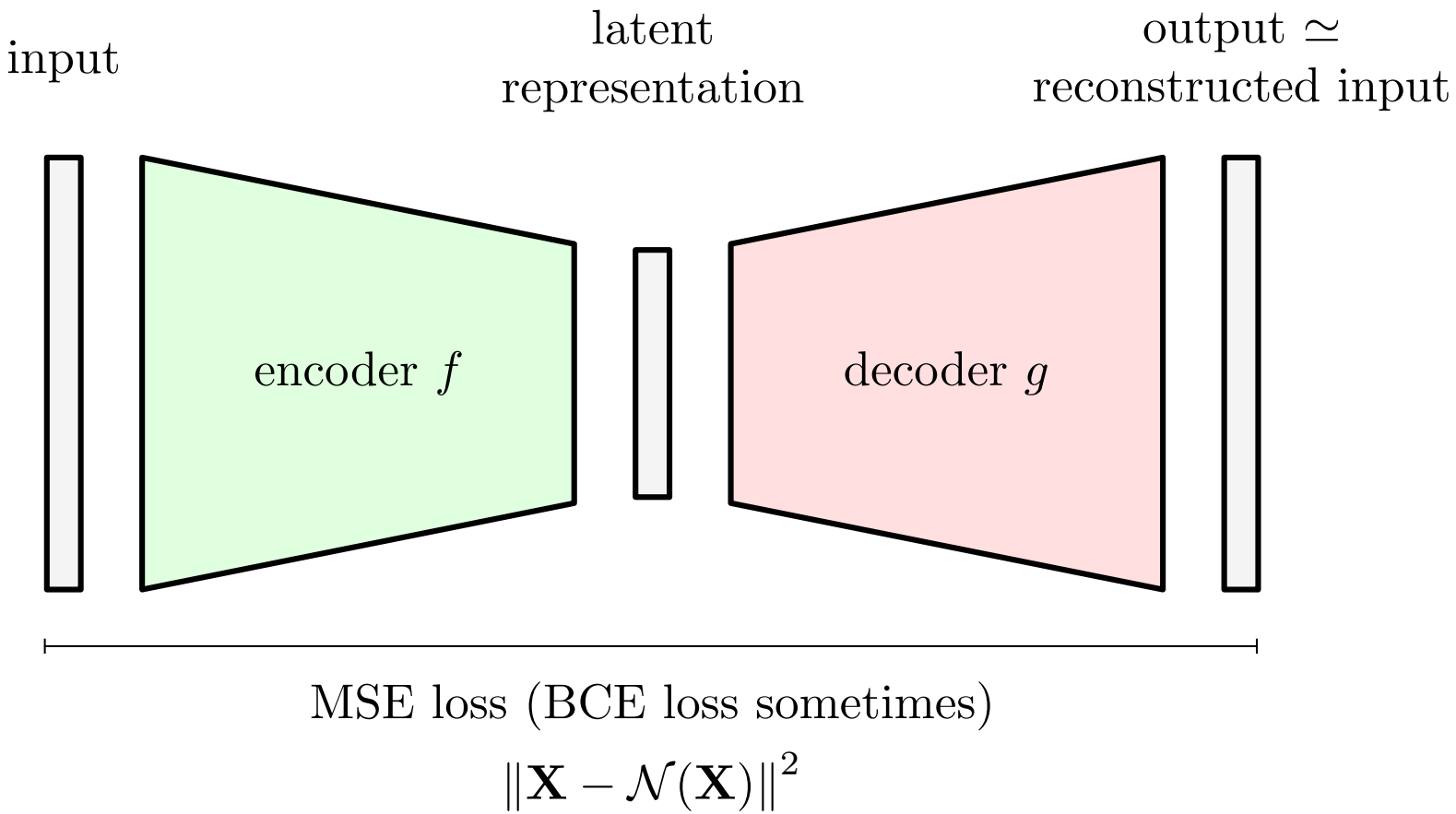
Introduction

- **But d'un autoencodeur:** projeter les data dans un espace de dimension plus petite – l'**espace latent (latent space)** – et être capable de les reconstruire ces data.
- Ceci revient à chercher un nombre restreint de degrés de libertés qui décrivent les data (cf. algorithme PCA).
- On est donc dans un contexte d'**apprentissage non-supervisé (unsupervised learning)**.
- **Applications:** compression, débruitage (denoising), etc.

Introduction

- **But d'un autoencodeur:** projeter les data dans un espace de dimension plus petite – l'**espace latent (latent space)** – et être capable de les reconstruire ces data.
- Ceci revient à chercher un nombre restreint de degrés de libertés qui décrivent les data (cf. algorithme PCA).
- On est donc dans un contexte d'**apprentissage non-supervisé (unsupervised learning)**.
- **Applications:** compression, débruitage (denoising), etc.

Architecture encodeur-décodeur



Architecture encodeur-décodeur

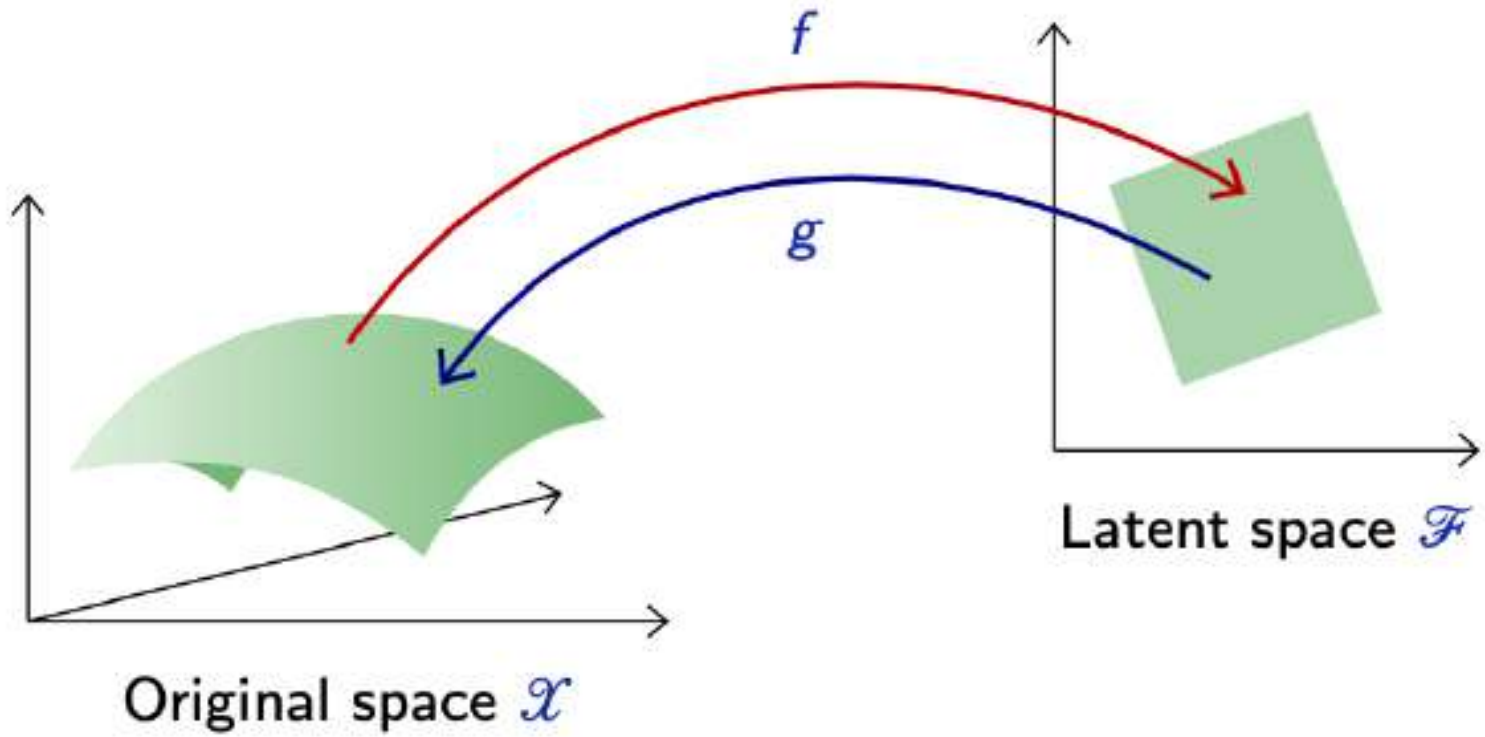


Figure taken from [Fleuret, 2022].

Autoencodeur (AE)

- L'**encodeur** est un deep neural network $f(\cdot; \mathbf{\Theta_f})$.
- Le **décodeur** est un deep neural network $g(\cdot; \mathbf{\Theta_g})$.
- La composition de $f(\cdot; \mathbf{\Theta_f})$ et $g(\cdot; \mathbf{\Theta_g})$ forme un réseau de neurones $\mathcal{N}(\cdot; \mathbf{\Theta_f}, \mathbf{\Theta_g})$.
- Le but de $\mathcal{N}$ est de reconstruire au mieux ses inputs $\mathbf{x}$:

$$\begin{aligned} \mathcal{N} &= g \circ f \simeq id \\ \mathcal{N}(\mathbf{x}; \mathbf{\Theta_f}, \mathbf{\Theta_g}) &= g\left(f(\mathbf{x}; \mathbf{\Theta_f}); \mathbf{\Theta_g}\right) \simeq \mathbf{x} \end{aligned}$$

Autoencodeur (AE)

- L'**encodeur** est un deep neural network $f(\cdot; \mathbf{\Theta_f})$.
- Le **décodeur** est un deep neural network $g(\cdot; \mathbf{\Theta_g})$.
- La composition de $f(\cdot; \mathbf{\Theta_f})$ et $g(\cdot; \mathbf{\Theta_g})$ forme un réseau de neurones $\mathcal{N}(\cdot; \mathbf{\Theta_f}, \mathbf{\Theta_g})$.
- Le but de $\mathcal{N}$ est de reconstruire au mieux ses inputs $\mathbf{x}$:

$$\begin{aligned} \mathcal{N} &= g \circ f \simeq id \\ \mathcal{N}(\mathbf{x}; \mathbf{\Theta_f}, \mathbf{\Theta_g}) &= g\left(f(\mathbf{x}; \mathbf{\Theta_f}); \mathbf{\Theta_g}\right) \simeq \mathbf{x} \end{aligned}$$

Autoencodeur (AE)

- L'**encodeur** est un deep neural network $f(\cdot; \mathbf{\Theta_f})$.
- Le **décodeur** est un deep neural network $g(\cdot; \mathbf{\Theta_g})$.
- La composition de $f(\cdot; \mathbf{\Theta_f})$ et $g(\cdot; \mathbf{\Theta_g})$ forme un réseau de neurones $\mathcal{N}(\cdot; \mathbf{\Theta_f}, \mathbf{\Theta_g})$.
- Le but de $\mathcal{N}$ est de reconstruire au mieux ses inputs $\mathbf{x}$:

$$\begin{aligned} \mathcal{N} &= g \circ f \simeq id \\ \mathcal{N}(\mathbf{x}; \mathbf{\Theta_f}, \mathbf{\Theta_g}) &= g\left(f(\mathbf{x}; \mathbf{\Theta_f}); \mathbf{\Theta_g}\right) \simeq \mathbf{x} \end{aligned}$$

Autoencodeur (AE)

- L'**encodeur** est un deep neural network $f(\cdot; \mathbf{\Theta_f})$.
- Le **décodeur** est un deep neural network $g(\cdot; \mathbf{\Theta_g})$.
- La composition de $f(\cdot; \mathbf{\Theta_f})$ et $g(\cdot; \mathbf{\Theta_g})$ forme un réseau de neurones $\mathcal{N}(\cdot; \mathbf{\Theta_f}, \mathbf{\Theta_g})$.
- Le but de $\mathcal{N}$ est de reconstruire au mieux ses inputs $\mathbf{x}$:

$$\begin{aligned} \mathcal{N} &= g \circ f \simeq id \\ \mathcal{N}(\mathbf{x}; \mathbf{\Theta_f}, \mathbf{\Theta_g}) &= g\left(f(\mathbf{x}; \mathbf{\Theta_f}); \mathbf{\Theta_g}\right) \simeq \mathbf{x} \end{aligned}$$

Autoencodeur (AE)

- Pour cela, on optimise la mean squared error (MSE):

$$\mathcal{L}(\mathbf{X}; \mathbf{\Theta_f}, \mathbf{\Theta_g}) = \frac{1}{N} \sum_{i=1}^{N} \| \mathbf{x_i} - g\left(f(\mathbf{x_i}; \mathbf{\Theta_f}); \mathbf{\Theta_g}\right) \|^2$$

où $\mathbf{X}$ est la concaténation des $\mathbf{x_i}$.

- L'entraînement du réseau consiste alors à trouver les poids $\hat{\mathbf{\Theta}}_\mathbf{f}$ et $\hat{\mathbf{\Theta}}_\mathbf{g}$ qui satisfont

$$\hat{\mathbf{\Theta}}_\mathbf{f}; \hat{\mathbf{\Theta}}_\mathbf{g} = \underset{\mathbf{\Theta_f}; \mathbf{\Theta_g}}{\arg\min}\, \mathcal{L}(\mathbf{X}; \mathbf{\Theta_f}, \mathbf{\Theta_g})$$

Autoencodeur (AE)

- Pour cela, on optimise la mean squared error (MSE):

$$\mathcal{L}(\mathbf{X}; \mathbf{\Theta_f}, \mathbf{\Theta_g}) = \frac{1}{N} \sum_{i=1}^{N} \| \mathbf{x_i} - g\left(f(\mathbf{x_i}; \mathbf{\Theta_f}); \mathbf{\Theta_g}\right) \|^2$$

  où $\mathbf{X}$ est la concaténation des $\mathbf{x_i}$.

- L'entraînement du réseau consiste alors à trouver les poids $\hat{\mathbf{\Theta}}_\mathbf{f}$ et $\hat{\mathbf{\Theta}}_\mathbf{g}$ qui satisfont

$$\hat{\mathbf{\Theta}}_\mathbf{f}; \hat{\mathbf{\Theta}}_\mathbf{g} = \underset{\mathbf{\Theta_f}; \mathbf{\Theta_g}}{\arg\min}\, \mathcal{L}(\mathbf{X}; \mathbf{\Theta_f}, \mathbf{\Theta_g})$$

Implémentation: AE linéaire

```python
class Autoencoder_Linear(nn.Module):
    """Implements an linear automencoder"""

    def __init__(self):
        """constructor"""

        super().__init__()

        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 12),
            nn.ReLU(),
            nn.Linear(12, 2) # -> N, 2 only!
        )

        self.decoder = nn.Sequential(
            nn.Linear(2, 12),
            nn.ReLU(),
            nn.Linear(12, 64),
            nn.ReLU(),
            nn.Linear(64, 128),
            nn.ReLU(),
            nn.Linear(128, 28 * 28),
            nn.Sigmoid()                # output neurons between 0 and 1
        )

    def forward(self, x):
        """forward pass"""

        encoded_data = self.encoder(x)
        decoded_data = self.decoder(encoded_data)

        return decoded_data
```

Implémentation: AE linéaire

```
self.encoder = nn.Sequential(
    nn.Linear(28 * 28, 128),
    nn.ReLU(),
    nn.Linear(128, 64),
    nn.ReLU(),
    nn.Linear(64, 12),
    nn.ReLU(),
    nn.Linear(12, 2) # -> N, 2 only!
)
```

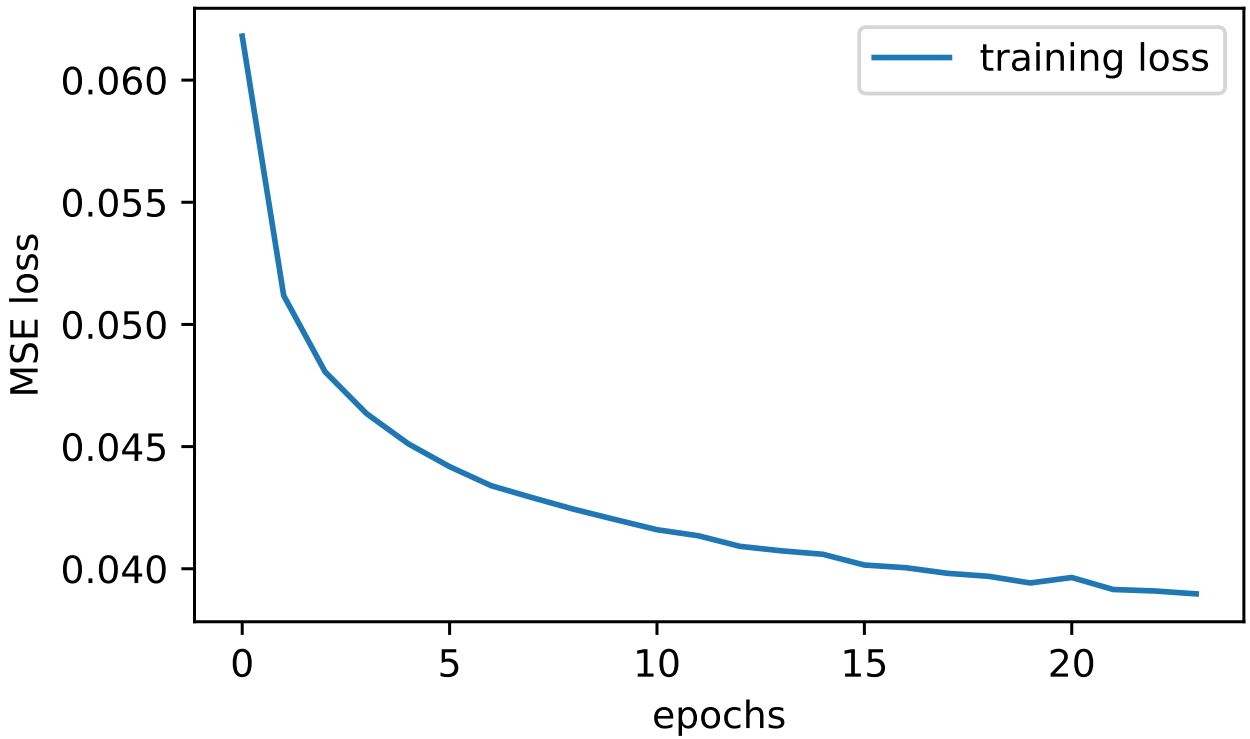
Implémentation: AE linéaire

```
self.decoder = nn.Sequential(
    nn.Linear(2, 12),
    nn.ReLU(),
    nn.Linear(12, 64),
    nn.ReLU(),
    nn.Linear(64, 128),
    nn.ReLU(),
    nn.Linear(128, 28 * 28),
    nn.Sigmoid()
)
```

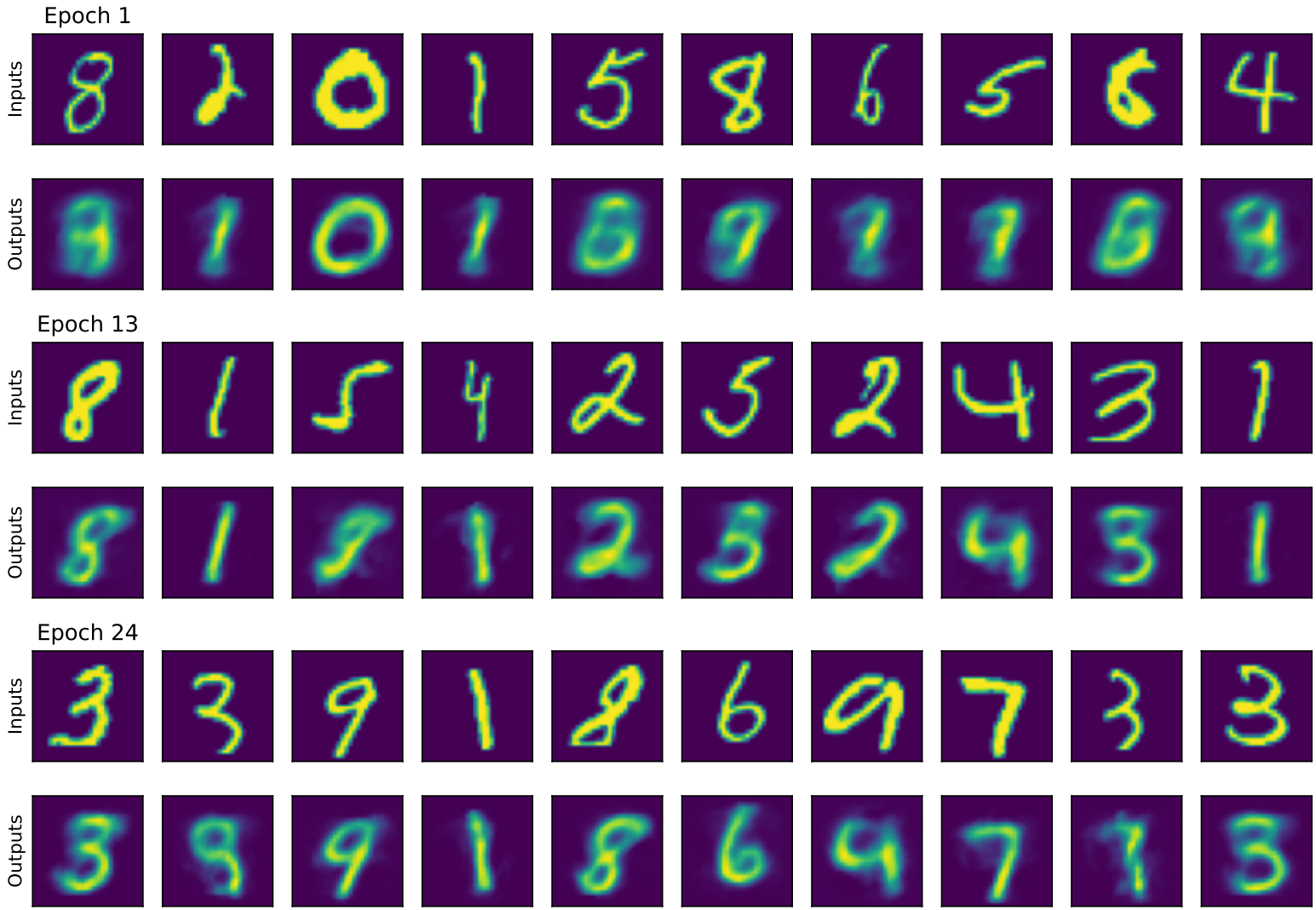
Implémentation: AE linéaire

```
model = Autoencoder_Linear()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(),
                             lr=1e-3,
                             weight_decay=1e-5)
```

Implémentation: AE linéaire



Implémentation: AE linéaire



Implémentation: AE convolutionnel

```python
class Autoencoder_CNN(nn.Module):
    """Implements a CNN autoencoder"""

    def __init__(self):
        """constructor"""

        super().__init__()

        # N, 1, 28, 28
        self.encoder = nn.Sequential(
            # -> N, 16, 14, 14
            nn.Conv2d(1, 16, 3, stride=2, padding=1),
            nn.ReLU(),
            # -> N, 32, 7, 7
            nn.Conv2d(16, 32, 3, stride=2, padding=1),
            nn.ReLU(),
            # -> N, 64, 1, 1
            nn.Conv2d(32, 64, 7)
            )

        # N , 64, 1, 1
        self.decoder = nn.Sequential(
            # -> N, 32, 7, 7
            nn.ConvTranspose2d(64, 32, 7),
            nn.ReLU(),
            # N, 16, 14, 14 (N, 16, 13, 13 without output_padding)
            nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            # N, 1, 28, 28 (N,1, 27, 27)
            nn.ConvTranspose2d(16, 1, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid()
            )

    def forward(self, x):
        """forward function"""

        encoded_data = self.encoder(x)
        decoded_data = self.decoder(encoded_data)

        return decoded_data
```

Implémentation: AE convolutionnel

```
# N, 1, 28, 28
self.encoder = nn.Sequential(
    # -> N, 16, 14, 14
    nn.Conv2d(1, 16, 3, stride=2, padding=1),
    nn.ReLU(),
    # -> N, 32, 7, 7
    nn.Conv2d(16, 32, 3, stride=2, padding=1),
    nn.ReLU(),
    # -> N, 64, 1, 1
    nn.Conv2d(32, 64, 7)
    )
```

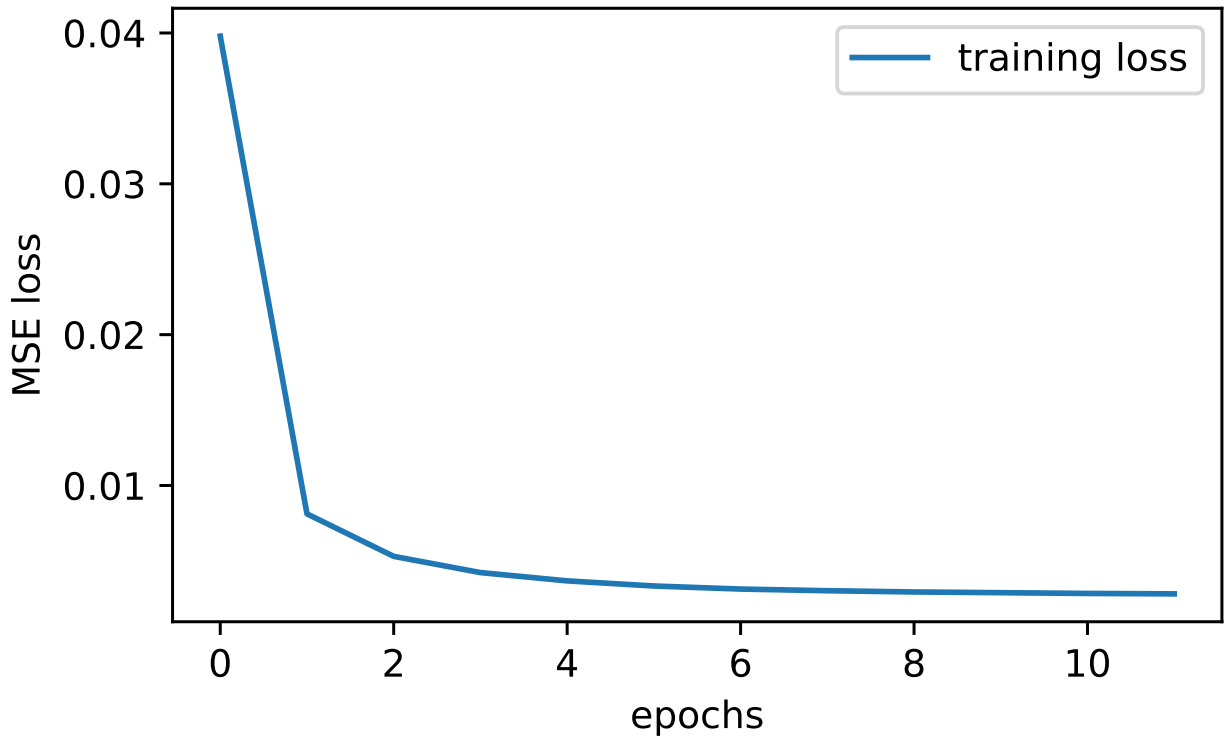
Implémentation: AE convolutionnel

```
# N , 64, 1, 1
self.decoder = nn.Sequential(
    # -> N, 32, 7, 7
    nn.ConvTranspose2d(64, 32, 7),
    nn.ReLU(),
    # N, 16, 14, 14 (N, 16, 13, 13 without output_padding)
    nn.ConvTranspose2d(32, 16, 3,
                       stride=2, padding=1, output_padding=1),
    nn.ReLU(),
    # N, 1, 28, 28 (N,1, 27, 27)
    nn.ConvTranspose2d(16, 1, 3,
                       stride=2, padding=1, output_padding=1),
    nn.Sigmoid()
    )
```

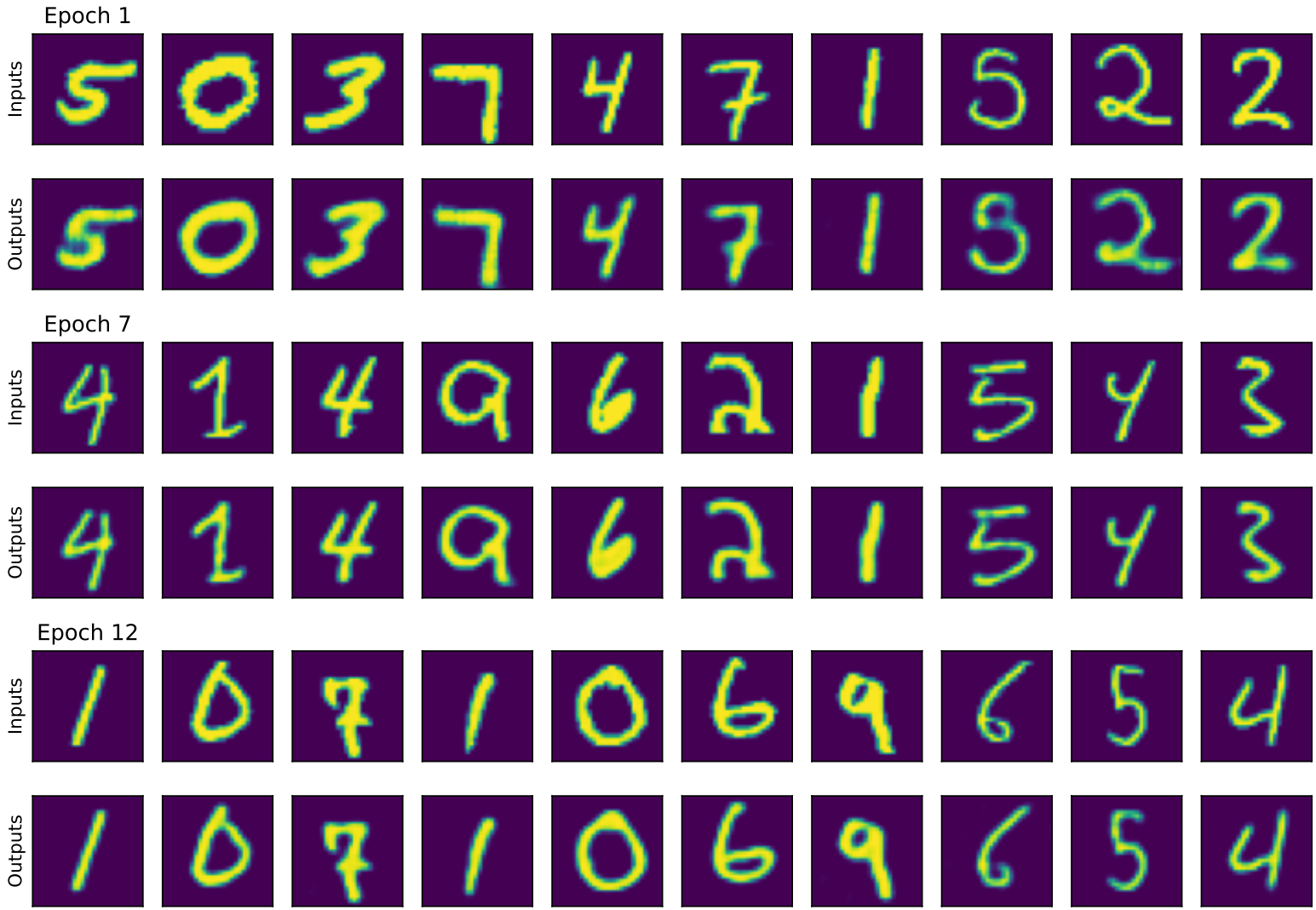
Implémentation: AE convolutionnel

```
model = Autoencoder_CNN()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(),
                             lr=1e-3,
                             weight_decay=1e-5)
```

Implémentation: AE convolutionnel

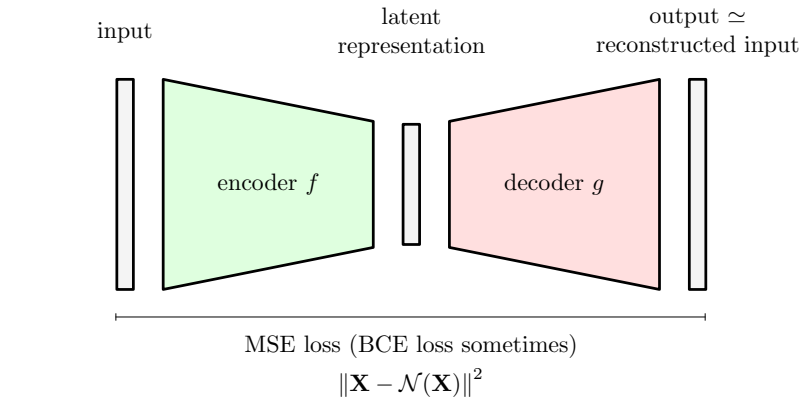


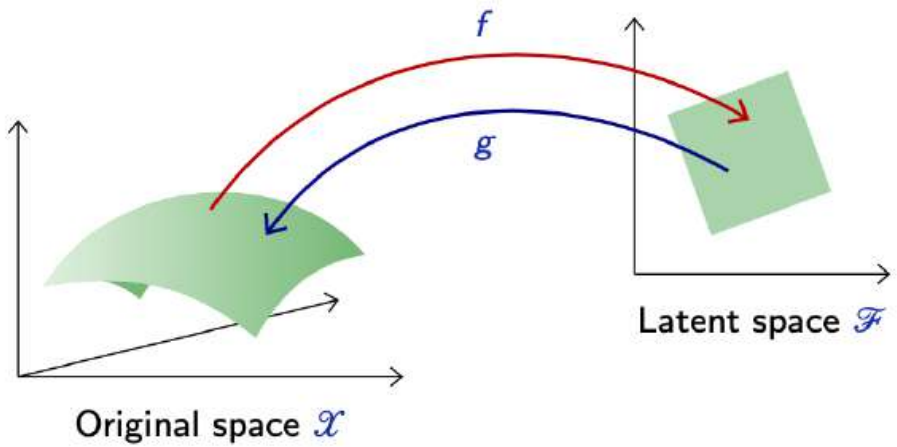
Implémentation: AE convolutionnel



Encodeur: compression de data

- Une fois l'autoencodeur entraîné, sa partie **encodeur** peut-être utilisée comme un **data compressor**.





Encodeur: compression de data

- Dans le cas de l'autoencodeur linéaire, la partie encodeur a projeté les data dans un **espace latent** de dimension 2.
- On peut donc visualiser ce plongement (cf. slide suivant).
- On s'aperçoit que les 10 différents types de data sont (plus ou moins) clusterisées dans différentes régions de l'espace latent.
- Il y a des "overlaps" entre les classes 3 et 8 ou 5 et 9 qui sont difficiles à séparer (car patterns similaires).

Encodeur: compression de data

- Dans le cas de l'autoencodeur linéaire, la partie encodeur a projeté les data dans un **espace latent** de dimension 2.
- On peut donc visualiser ce plongement (cf. slide suivant).
- On s'aperçoit que les 10 différents types de data sont (plus ou moins) clusterisées dans différentes régions de l'espace latent.
- Il y a des "overlaps" entre les classes 3 et 8 ou 5 et 9 qui sont difficiles à séparer (car patterns similaires).

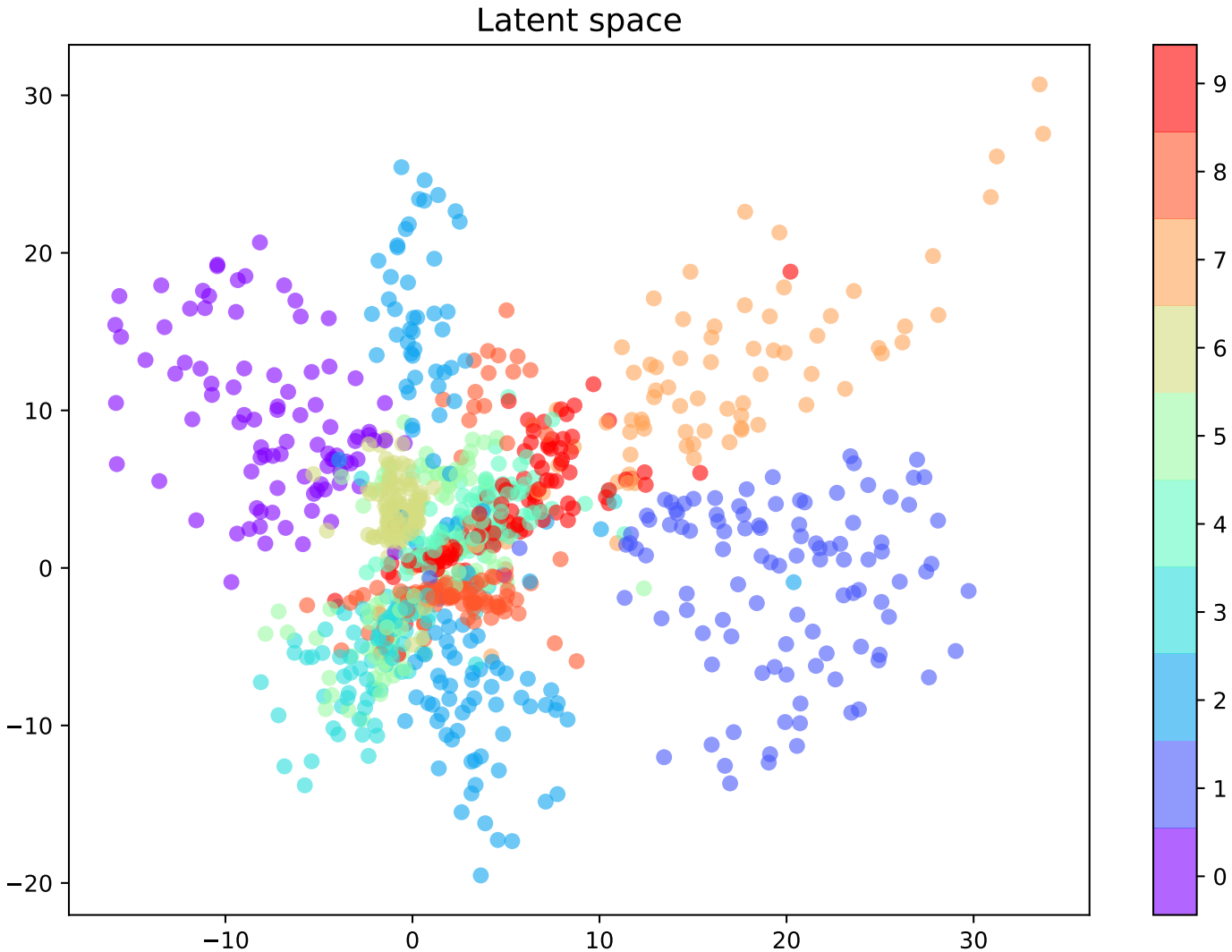
Encodeur: compression de data

- Dans le cas de l'autoencodeur linéaire, la partie encodeur a projeté les data dans un **espace latent** de dimension 2.
- On peut donc visualiser ce plongement (cf. slide suivant).
- On s'aperçoit que les 10 différents types de data sont (plus ou moins) clusterisées dans différentes régions de l'espace latent.
- Il y a des "overlaps" entre les classes 3 et 8 ou 5 et 9 qui sont difficiles à séparer (car patterns similaires).

Encodeur: compression de data

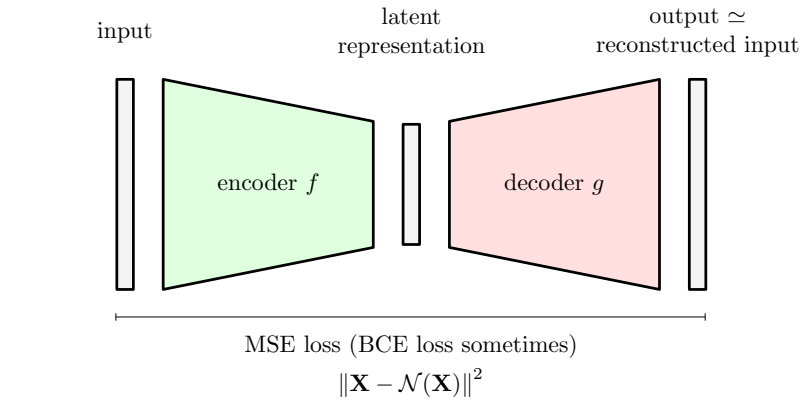
- Dans le cas de l'autoencodeur linéaire, la partie encodeur a projeté les data dans un **espace latent** de dimension 2.
- On peut donc visualiser ce plongement (cf. slide suivant).
- On s'aperçoit que les 10 différents types de data sont (plus ou moins) clusterisées dans différentes régions de l'espace latent.
- Il y a des "overlaps" entre les classes 3 et 8 ou 5 et 9 qui sont difficiles à séparer (car patterns similaires).

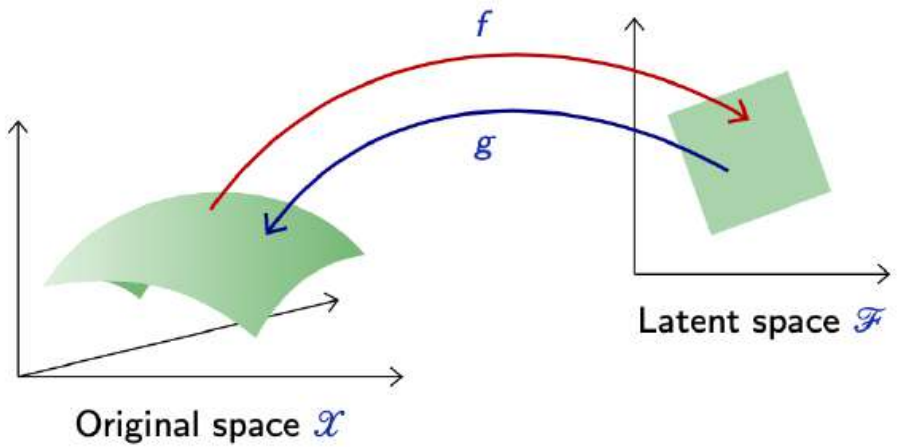
Encodeur: compression de data



Décodeur: génération de data

- Une fois l'autoencodeur entraîné, sa partie **décodeur** peut-être utilisée comme un **data generator**.





Décodeur: génération de data

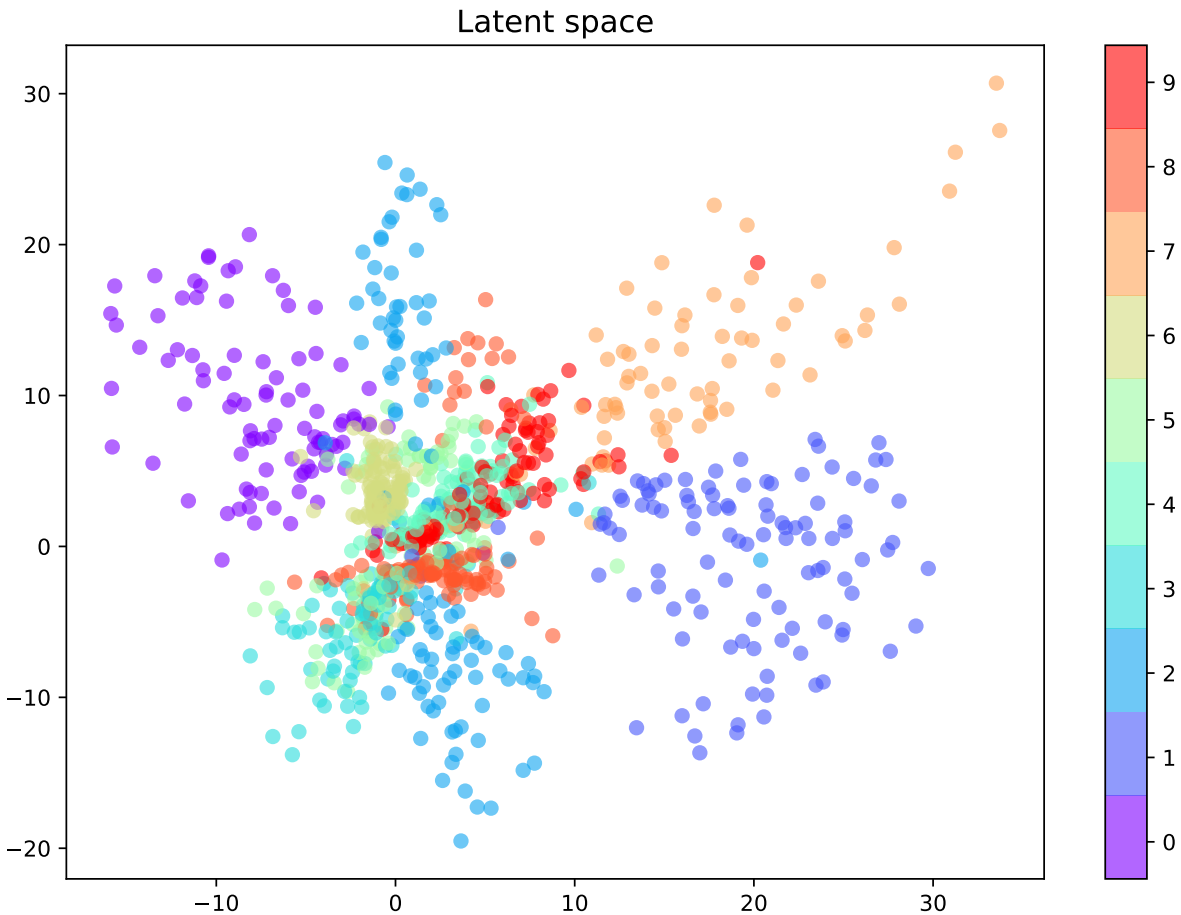
- Pour générer des data, il suffit de sampler des points aléatoirement dans l'espace latent puis de les décoder.
- **Question:** Comment sampler l'espace latent afin que les data générées soient de bonne qualité? Quelle distribution utiliser? Faut-il sampler des points de manière uniforme / normale / etc. dans l'espace latent?

Décodeur: génération de data

- Pour générer des data, il suffit de sampler des points aléatoirement dans l'espace latent puis de les décoder.
- **Question:** Comment sampler l'espace latent afin que les data générées soient de bonne qualité? Quelle distribution utiliser? Faut-il sampler des points de manière uniforme / normale / etc. dans l'espace latent?

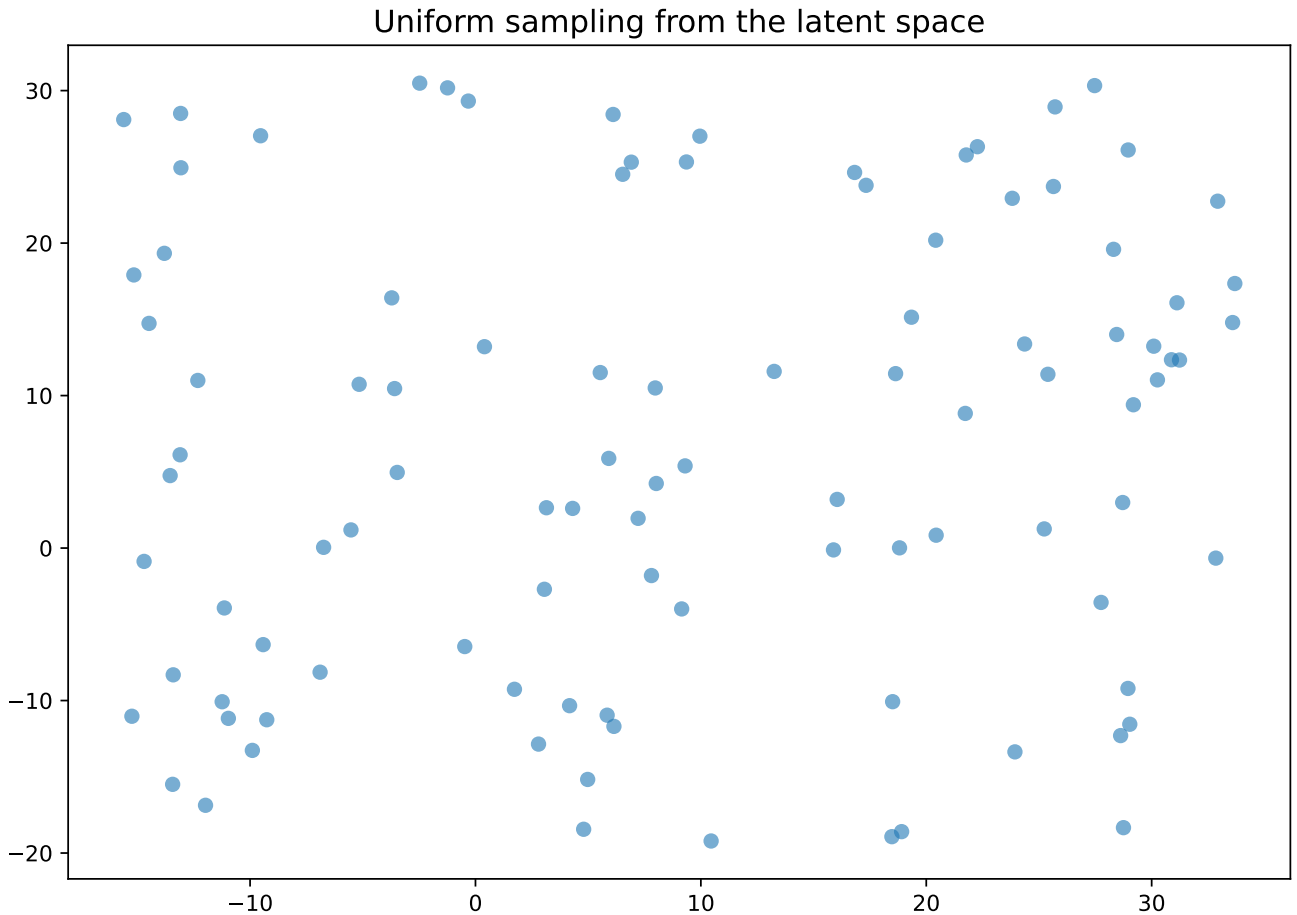
Décodeur: génération de data (uniforme)

- Sampling uniforme: les bornes de la distribution sont estimées à partir des data.



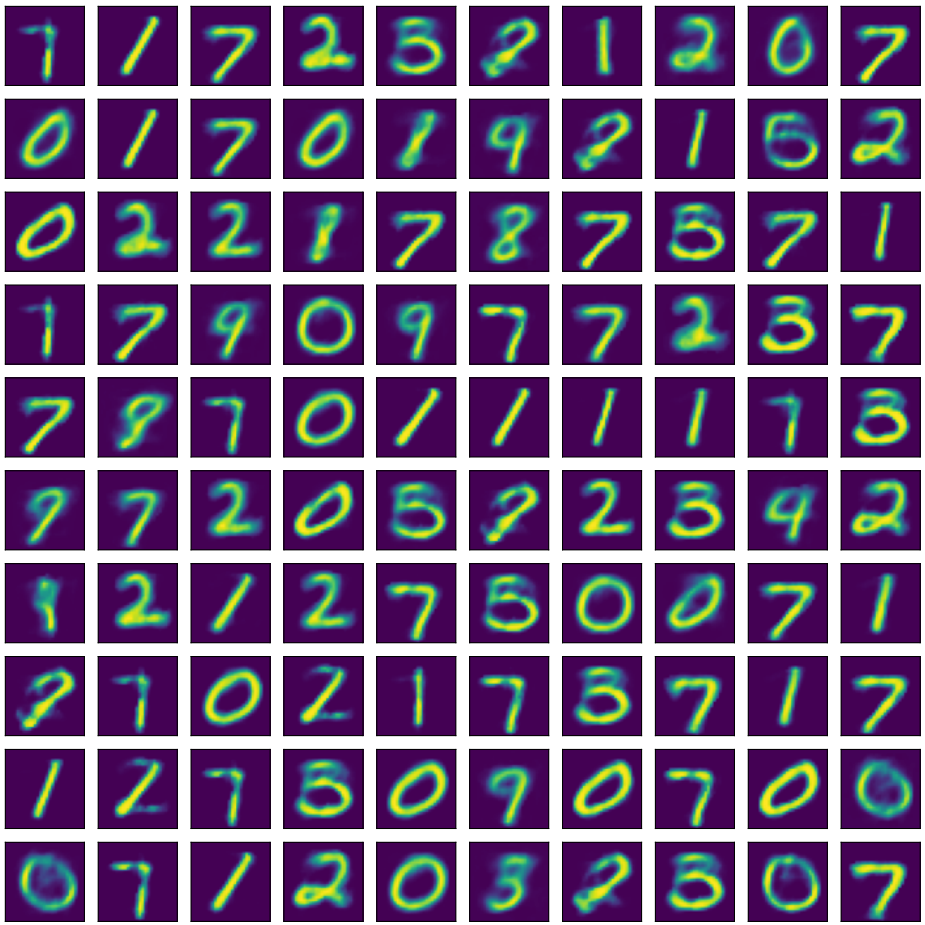
Décodeur: génération de data (uniforme)

- Sampling uniforme: les bornes de la distribution sont estimées à partir des data.



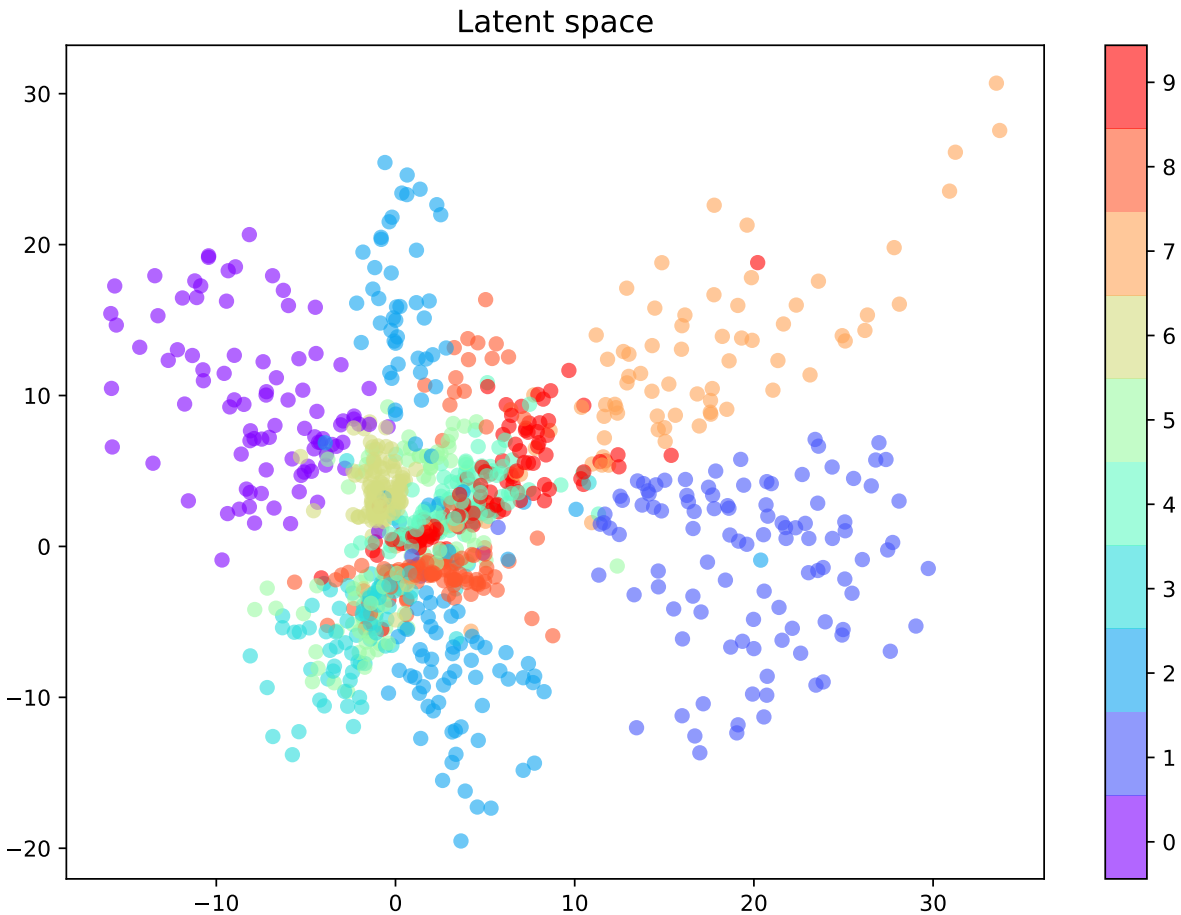
Décodeur: génération de data (uniforme)

- Sampling uniforme: les bornes de la distribution sont estimées à partir des data.



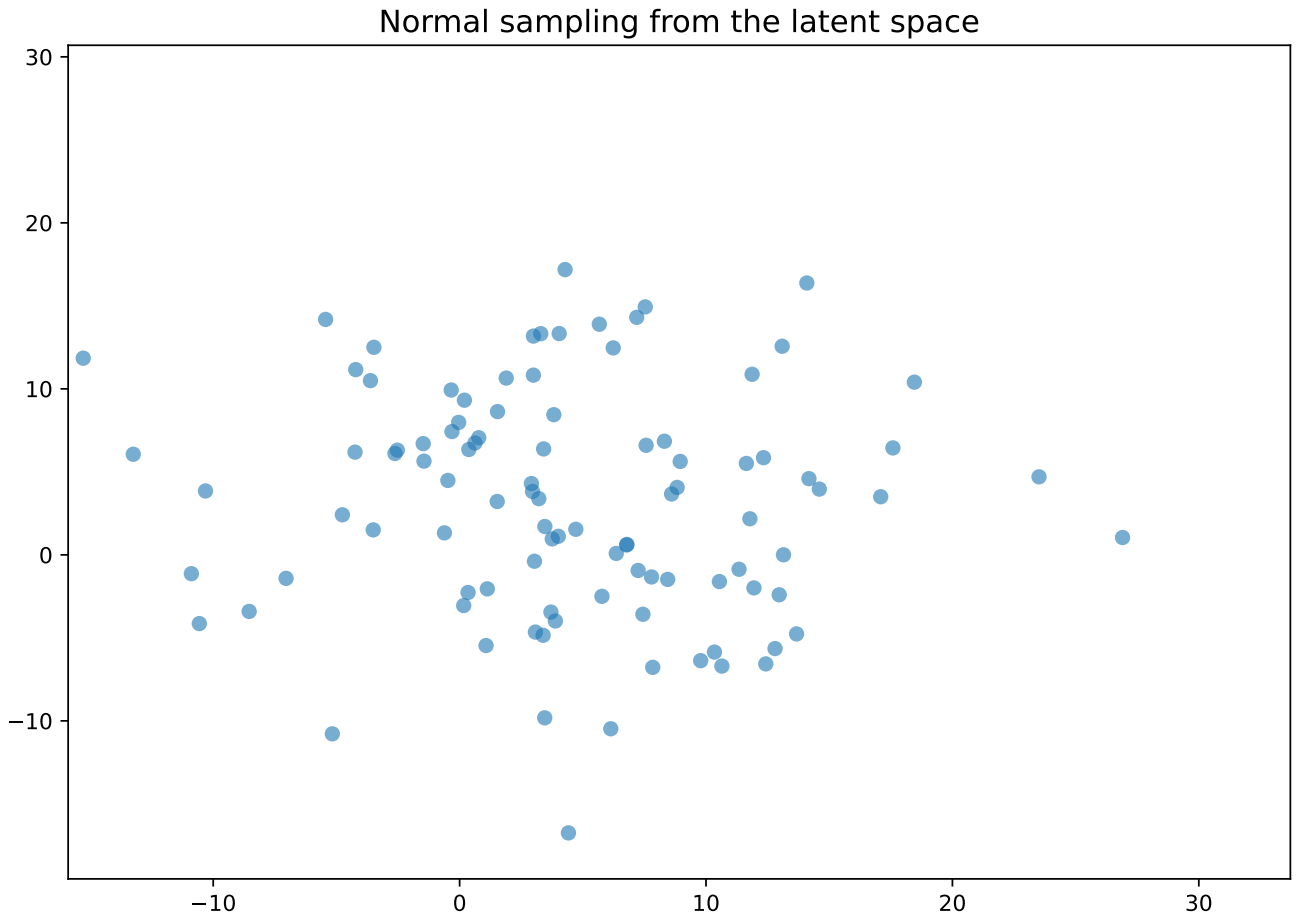
Décodeur: génération de data (normal)

- Sampling normal: la moyenne et la matrice de covariance de la distribution sont estimées à partir des data.



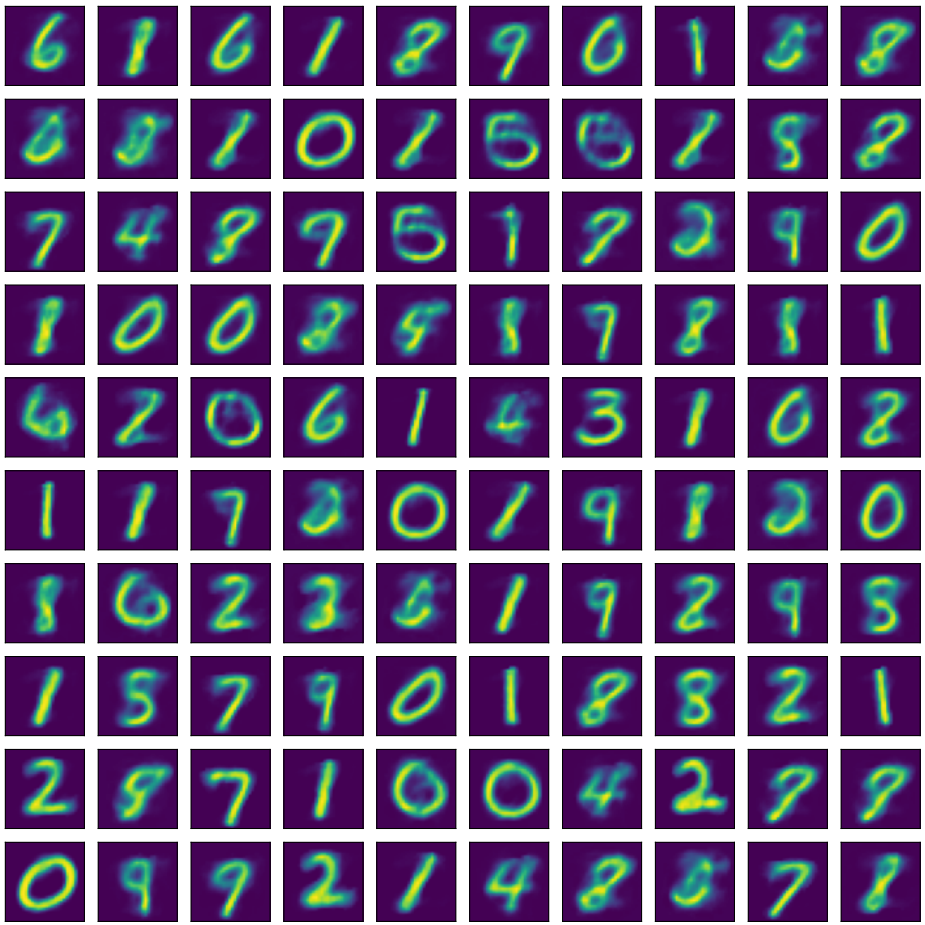
Décodeur: génération de data (normal)

- Sampling normal: la moyenne et la matrice de covariance de la distribution sont estimées à partir des data.



Décodeur: génération de data (normal)

- Sampling normal: la moyenne et la matrice de covariance de la distribution sont estimées à partir des data.



Décodeur: génération de data

- Le sampling uniforme semble donner de meilleurs résultats que le sampling normal dans ce cas (à l'oeil).
- Mais on voit que certaines data sont floues (blurred) et que certain chiffres sont sur-représentés.
- C'est normal: nous n'avons pas requis que les data soient distribuées d'une certaine manière dans l'espace latent.
- Il faudrait imposer que les data projetées dans l'espace latent suivent une certaine distribution.
- Autrement dit, il faudrait pouvoir imposer une distribution sur l'espace latent (cf. autoencodeur variationnel).

Décodeur: génération de data

- Le sampling uniforme semble donner de meilleurs résultats que le sampling normal dans ce cas (à l'oeil).
- Mais on voit que certaines data sont floues (blurred) et que certain chiffres sont sur-représentés.
- C'est normal: nous n'avons pas requis que les data soient distribuées d'une certaine manière dans l'espace latent.
- Il faudrait imposer que les data projetées dans l'espace latent suivent une certaine distribution.
- Autrement dit, il faudrait pouvoir imposer une distribution sur l'espace latent (cf. autoencodeur variationnel).

Décodeur: génération de data

- Le sampling uniforme semble donner de meilleurs résultats que le sampling normal dans ce cas (à l'oeil).
- Mais on voit que certaines data sont floues (blurred) et que certain chiffres sont sur-représentés.
- C'est normal: nous n'avons pas requis que les data soient distribuées d'une certaine manière dans l'espace latent.
- Il faudrait imposer que les data projetées dans l'espace latent suivent une certaine distribution.
- Autrement dit, il faudrait pouvoir imposer une distribution sur l'espace latent (cf. autoencodeur variationnel).

Décodeur: génération de data

- Le sampling uniforme semble donner de meilleurs résultats que le sampling normal dans ce cas (à l'oeil).
- Mais on voit que certaines data sont floues (blurred) et que certain chiffres sont sur-représentés.
- C'est normal: nous n'avons pas requis que les data soient distribuées d'une certaine manière dans l'espace latent.
- Il faudrait imposer que les data projetées dans l'espace latent suivent une certaine distribution.
- Autrement dit, il faudrait pouvoir imposer une distribution sur l'espace latent (cf. autoencodeur variationnel).

Décodeur: génération de data

- Le sampling uniforme semble donner de meilleurs résultats que le sampling normal dans ce cas (à l'oeil).
- Mais on voit que certaines data sont floues (blurred) et que certain chiffres sont sur-représentés.
- C'est normal: nous n'avons pas requis que les data soient distribuées d'une certaine manière dans l'espace latent.
- Il faudrait imposer que les data projetées dans l'espace latent suivent une certaine distribution.
- **Autrement dit, il faudrait pouvoir imposer une distribution sur l'espace latent (cf. autoencodeur variationnel).**

Bibliographie


Fleuret, F. (2022).
Deep Learning Course.