

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.

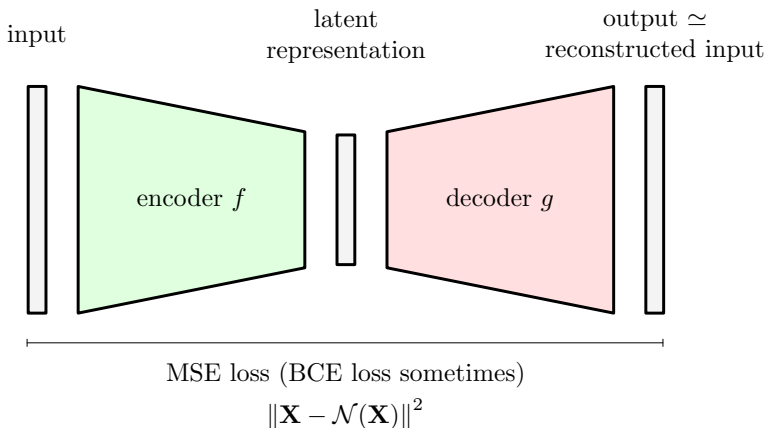
INTRODUCTION

- ▶ **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

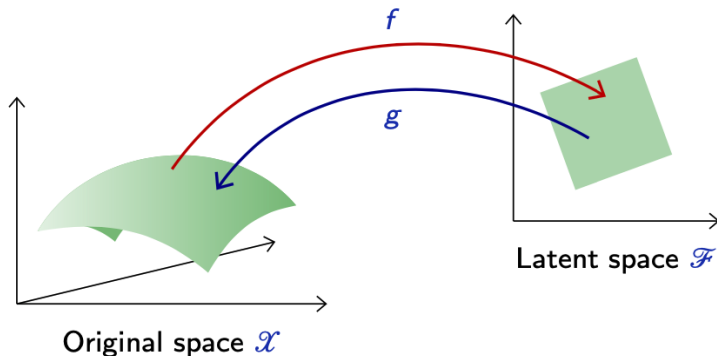


Figure taken from [Fleuret, 2022].

ARCHITECTURE ENCODEUR-DÉCODEUR

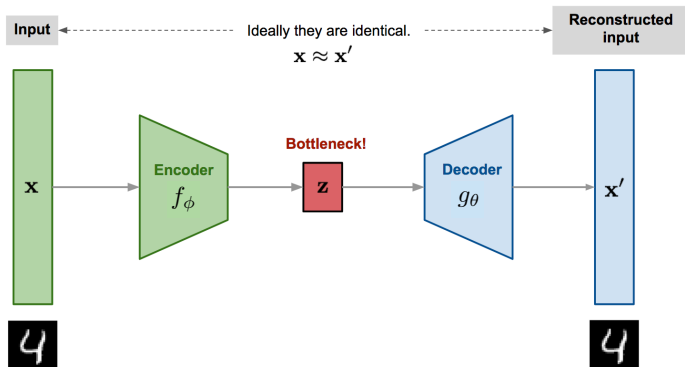


Figure taken from [Weng, 2018].

AUTOENCODEUR (AE)

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

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

AUTOENCODER (AE)

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

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

AUTOENCODEUR (AE)

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

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

AUTOENCODEUR (AE)

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

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

AUTOENCODEUR (AE)

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

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

où \mathbf{X} est la concaténation des \mathbf{x}_i .

- L'entraînement du réseau consiste alors à trouver les poids $\hat{\Theta}_f$ et $\hat{\Theta}_g$ qui satisfont

$$\hat{\Theta}_f; \hat{\Theta}_g = \arg \min_{\Theta_f; \Theta_g} \mathcal{L}(\mathbf{X}; \Theta_f, \Theta_g)$$

AUTOENCODEUR (AE)

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

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

où \mathbf{X} est la concaténation des \mathbf{x}_i .

- L'entraînement du réseau consiste alors à trouver les poids $\hat{\Theta}_f$ et $\hat{\Theta}_g$ qui satisfont

$$\hat{\Theta}_f; \hat{\Theta}_g = \arg \min_{\Theta_f; \Theta_g} \mathcal{L}(\mathbf{X}; \Theta_f, \Theta_g)$$

IMPLÉMENTATION: AE LINÉAIRE

```
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!  
)
```

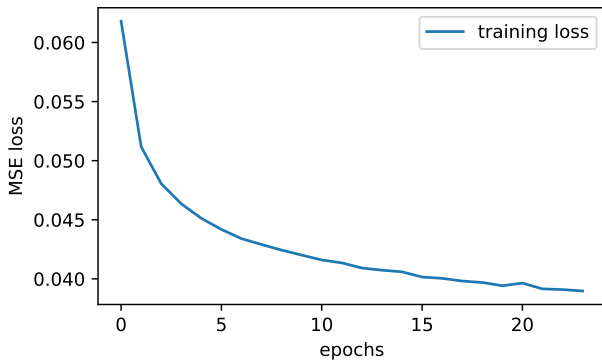

IMPLÉMENTATION: AE LINÉAIRE

```
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()  
)
```

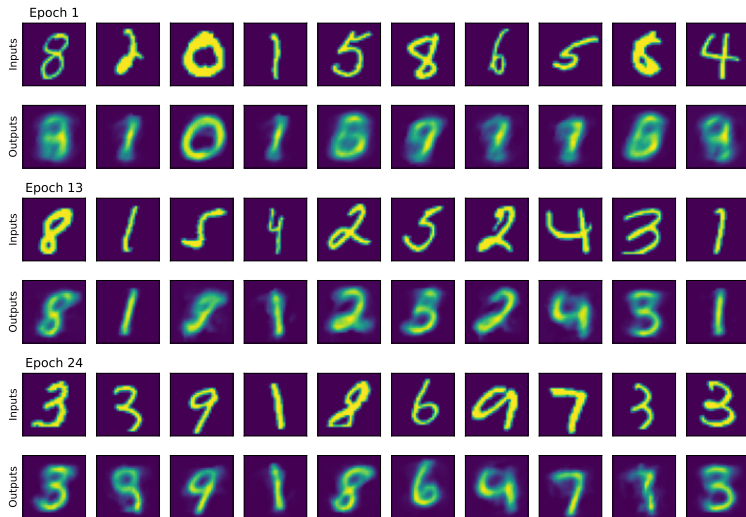
IMPLÉMENTATION: AE LINÉAIRE

```
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

```
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)
)
```

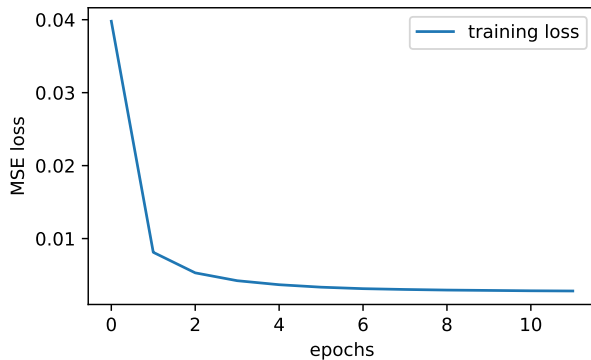
IMPLÉMENTATION: AE CONVOLUTIONNEL

```
# 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()
)
```

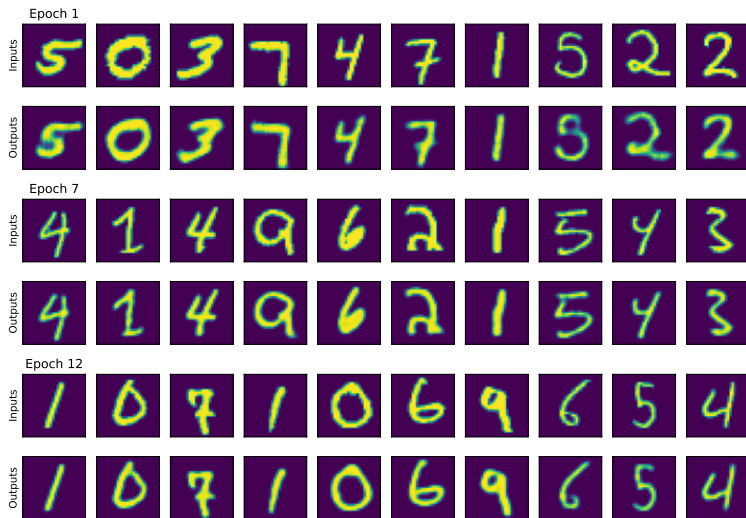
IMPLÉMENTATION: AE CONVOLUTIONNEL

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


IMPLÉMENTATION: AE CONVOLUTIONNEL

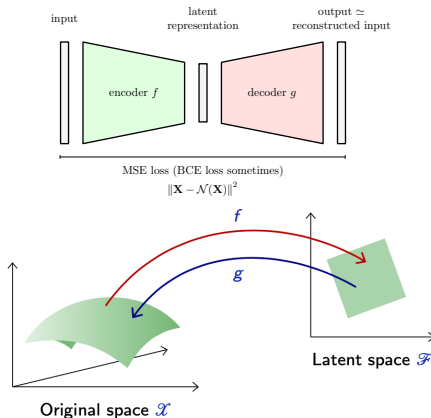


IMPLÉMENTATION: AE CONVOLUTIONNEL



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).

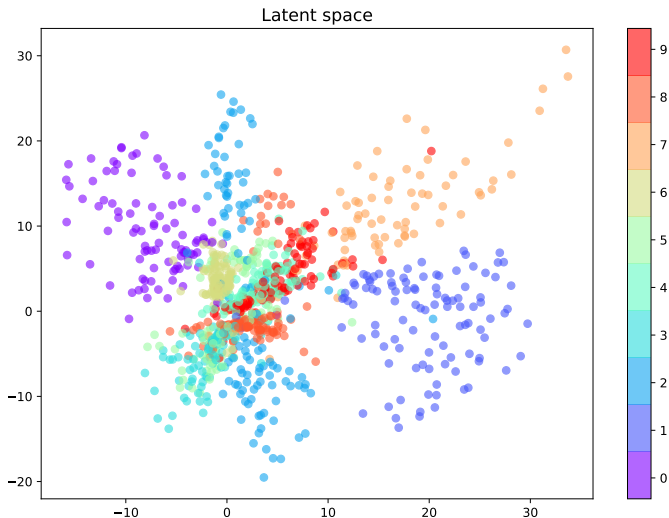
ENCODEUR: COMPRESSION DE DATA

- ▶ 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

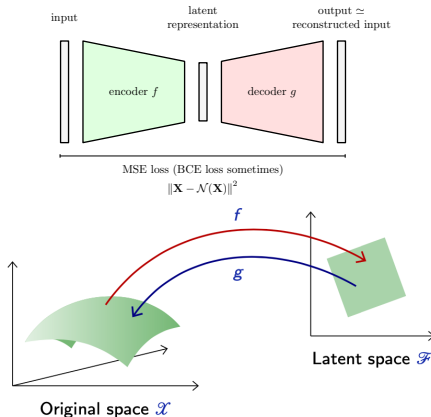
- ▶ Dans le cas de l'autoencodeur linéaire, la partie encodeur a projeté les data dans un **espace latent** de dimension 2.
- ▶ On peut donc visualiser ce plongement (cf. slide suivant).
- ▶ On s'aperçoit que les 10 différents types de data sont (plus ou moins) clusterisées dans différentes régions de 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

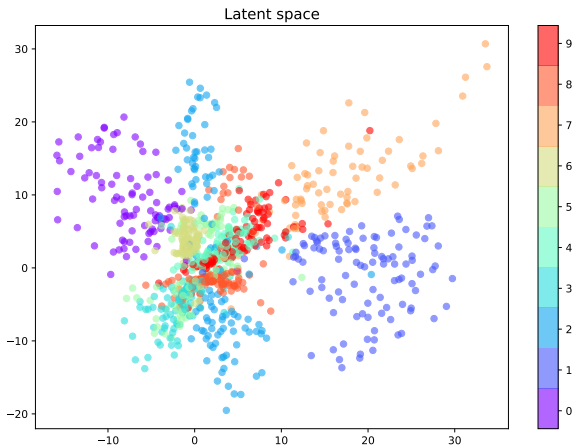
- ▶ Pour générer des data, il suffit de sampler des points aléatoirement dans l'espace latent puis de les décoder.
- ▶ **Question:** Comment sampler l'espace latent afin que les data générées soient de bonne qualité? Quelle distribution utiliser? Faut-il sampler des points de manière uniforme / normale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

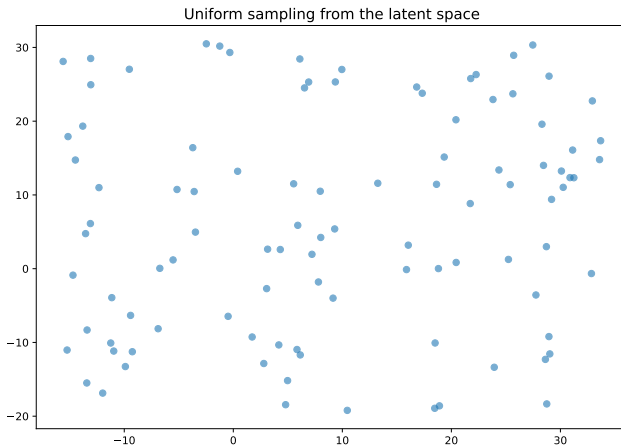
DÉCODEUR: GÉNÉRATION DE DATA (UNIFORME)

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



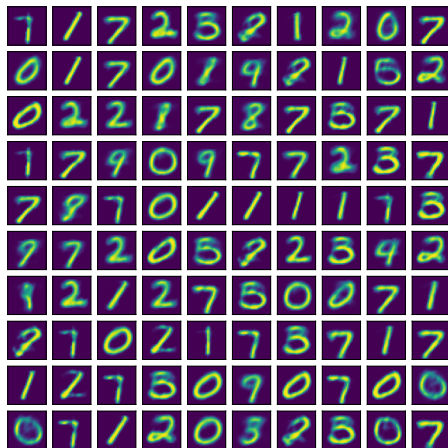
DÉCODEUR: GÉNÉRATION DE DATA (UNIFORME)

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



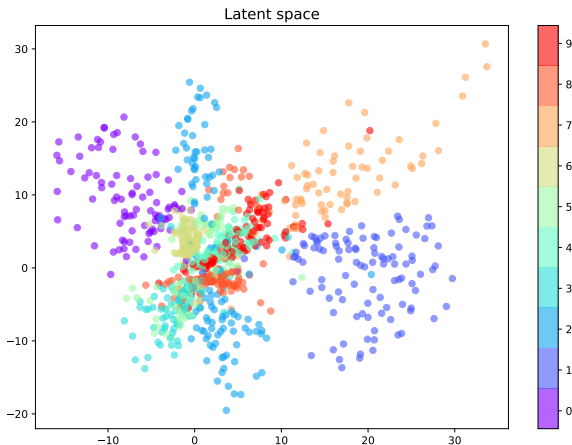
DÉCODEUR: GÉNÉRATION DE DATA (UNIFORME)

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



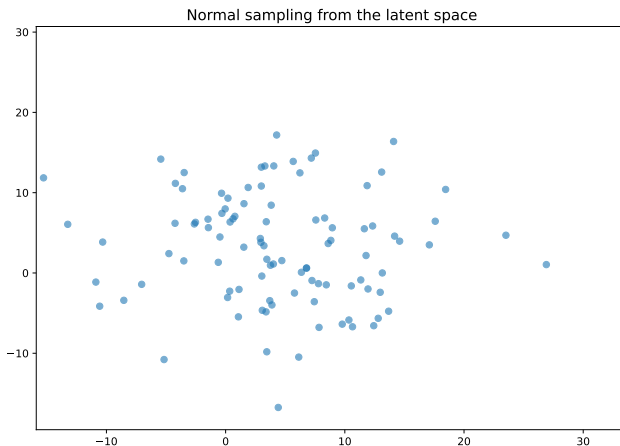
DÉCODEUR: GÉNÉRATION DE DATA (NORMAL)

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



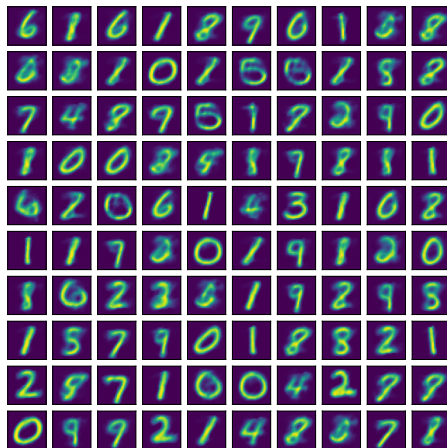
DÉCODEUR: GÉNÉRATION DE DATA (NORMAL)

- 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 certains chiffres sont sur-représentés.
- ▶ C'est normal: nous n'avons pas requis que les data soient distribuées "de manière propre" 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 certains chiffres sont sur-représentés.
- ▶ C'est normal: nous n'avons pas requis que les data soient distribuées "de manière propre" 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 certains chiffres sont sur-représentés.
- ▶ C'est normal: nous n'avons pas requis que les data soient distribuées "de manière propre" 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 certains chiffres sont sur-représentés.
- ▶ C'est normal: nous n'avons pas requis que les data soient distribuées "de manière propre" 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 certains chiffres sont sur-représentés.
- ▶ C'est normal: nous n'avons pas requis que les data soient distribuées "de manière propre" 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.



Weng, L. (2018).
From autoencoder to beta-vae.
lilianweng.github.io.