
بسم الله الرحمن الرحيم

Université Constantine 2 Abdelhamid Mehri
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département de l'Informatique Fondamentale et ses Applications

Module

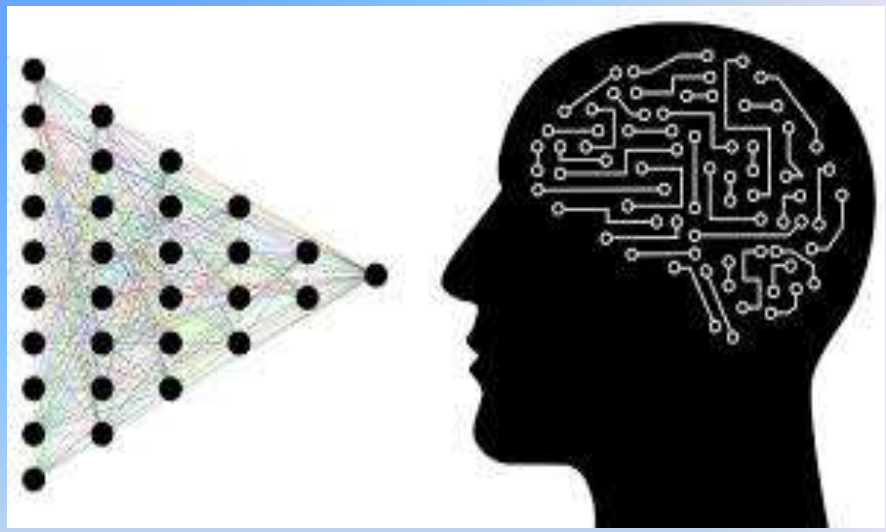
Machine Learning and Computational Intelligence

MLCI

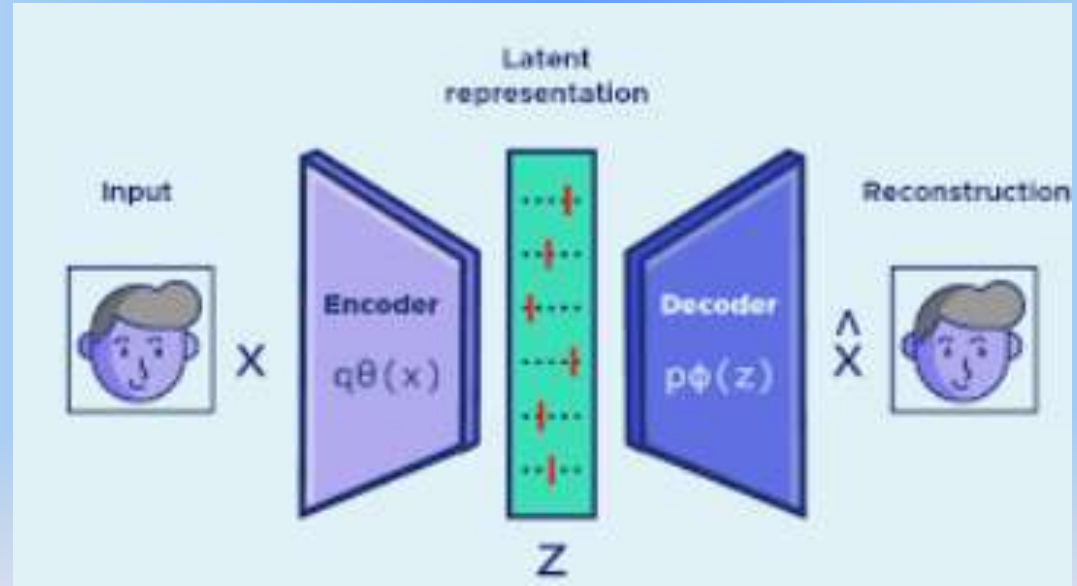
Unité d'enseignement: UEF3
Crédit: 3
Coefficient: 3
Cours:1H30/semaine
TP: 1H30/semaine

Dr. Fergani
Baha.fergani@univ-constantine2.dz

Apprentissage profond
(Deep Learning)



Auto-encodeurs



Auto-encodeurs

- Une technique du Deep Learning.
- Les auto-encodeurs sont des réseaux de neurones un peu particuliers.
- Ils possèdent **exactement** le même **nombre** de neurones sur leurs **couche d'entrée** et leurs **couche de sortie**.

Auto-encodeurs

Le **but** d'un auto-encodeur est:

d’avoir **une sortie la plus proche possible de l’entrée**.

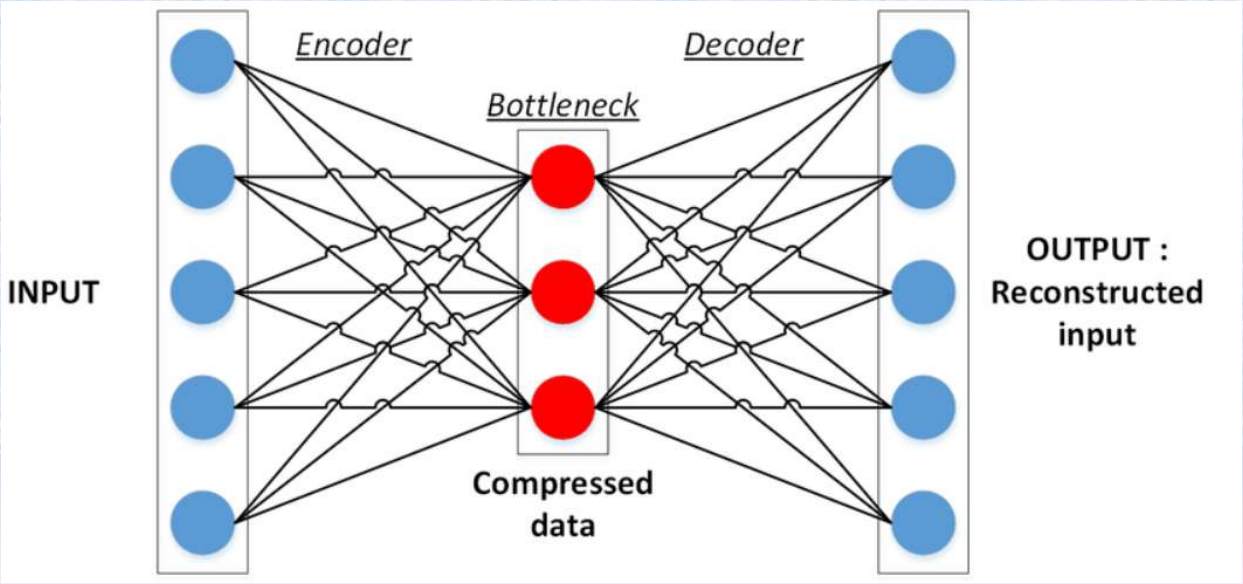
- L’apprentissage est **auto-supervisé** car la perte à minimiser est le coût de reconstruction entre la sortie et l’entrée.

- Les données n’ont ainsi pas à être labellisées, parce qu'elles sont leurs propres labels, ce qui fait alors de ce modèle un modèle **non supervisé**.

Architecture d'auto-encodeur

On peut décomposer un auto-encodeur en trois parties:

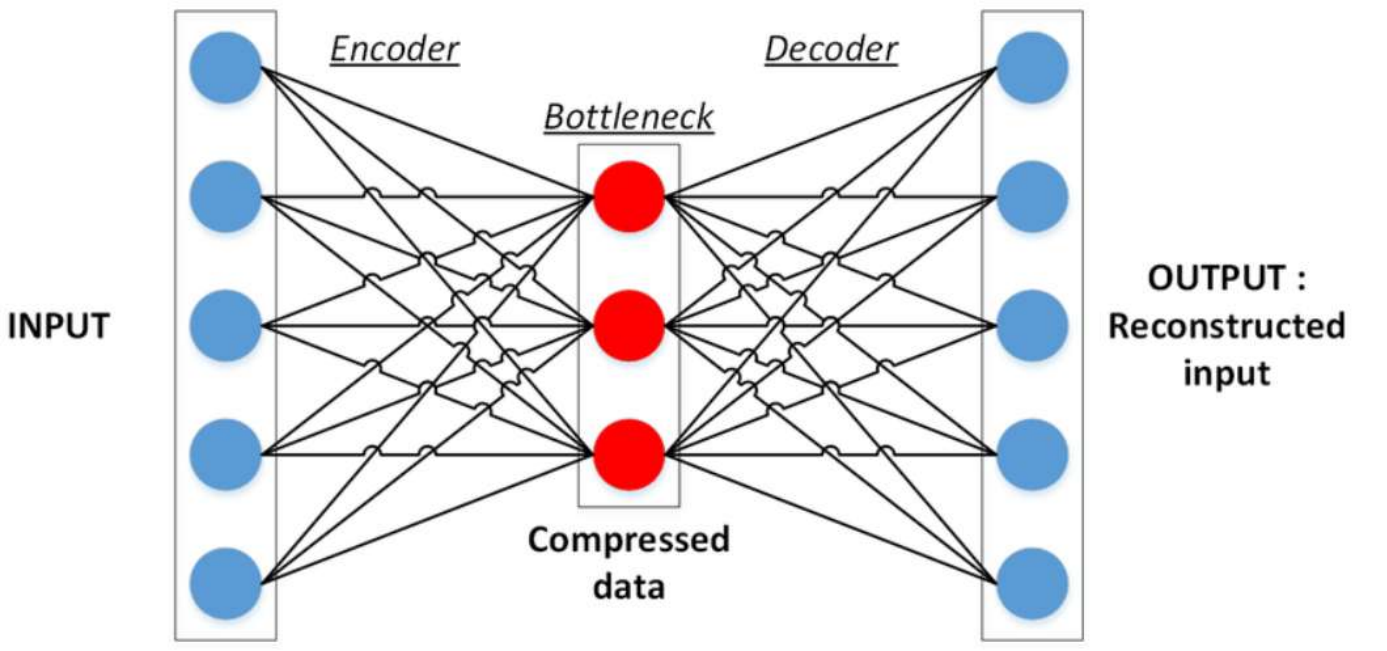
- **Encodeur :** L'encodeur transforme l'entrée en une représentation dans un espace de dimension plus faible appelé **espace latent**. L'encodeur compresse donc l'entrée dans une représentation moins coûteuse.



Architecture d'auto-encodeur

On peut décomposer un auto-encodeur en trois parties

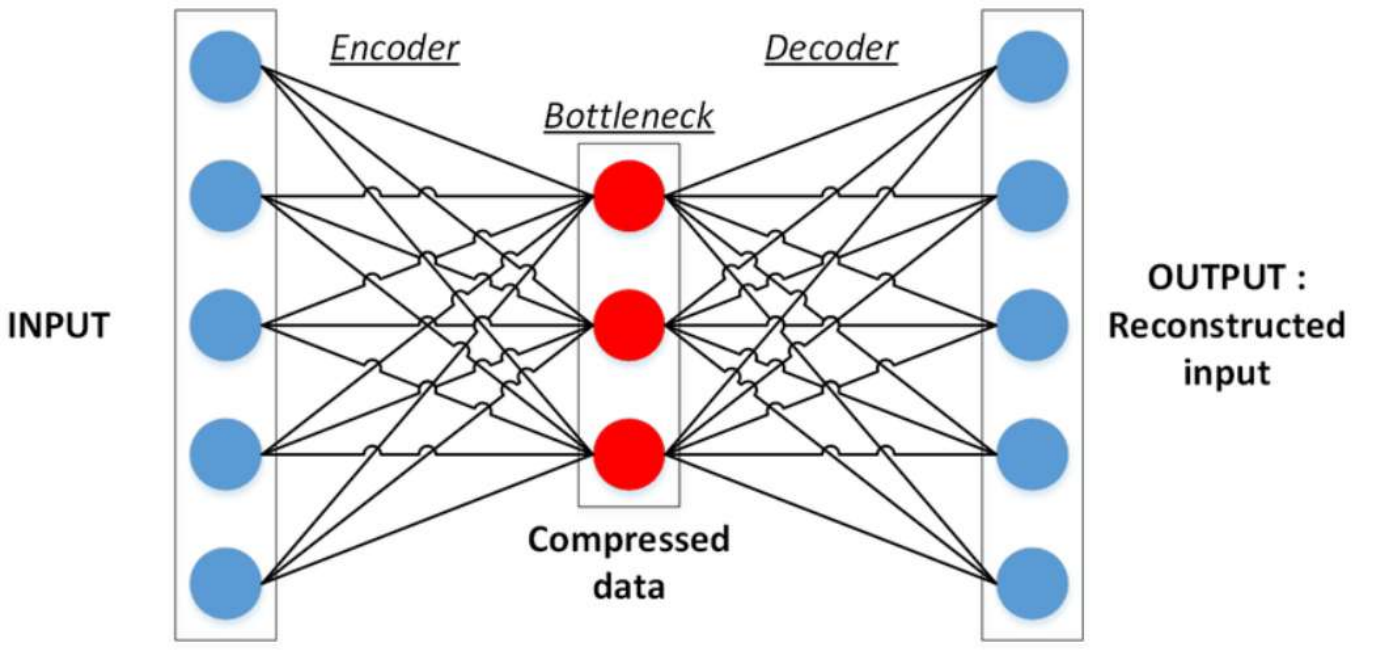
- **Goulot d'étranglement (ou *bottleneck ou code*) :** Cette partie du réseau contient la représentation compressée de l'entrée qui sera introduite dans le décodeur.



Architecture d'auto-encodeur

On peut décomposer un auto-encodeur en trois parties

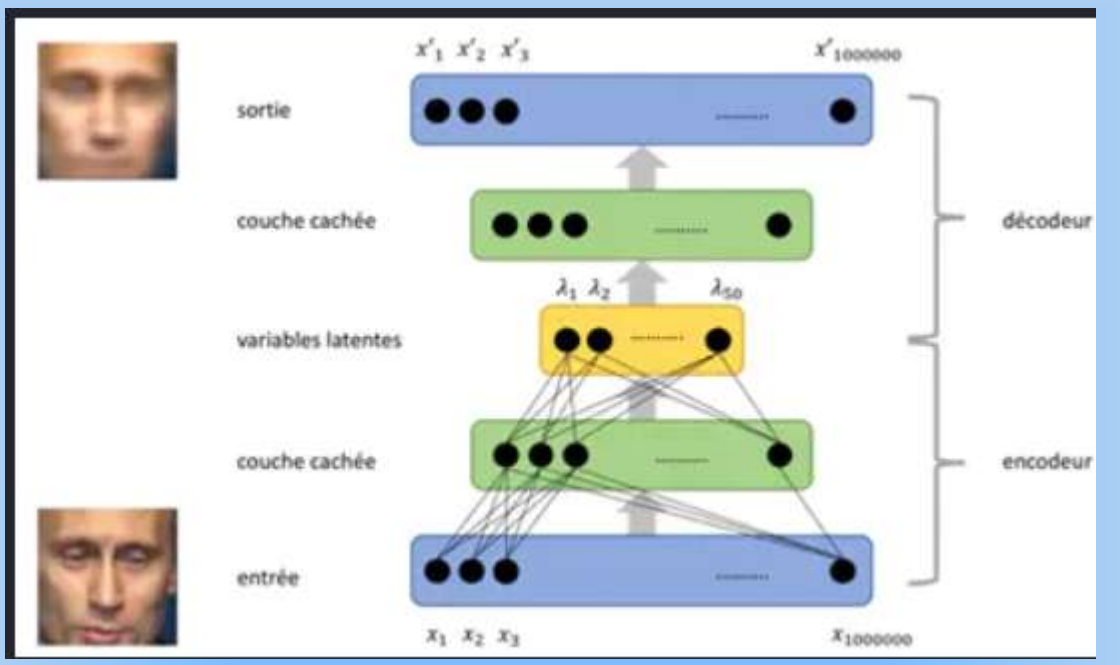
- **Décodeur :** cette partie doit construire l'output à l'aide de la représentation latente de l'entrée.

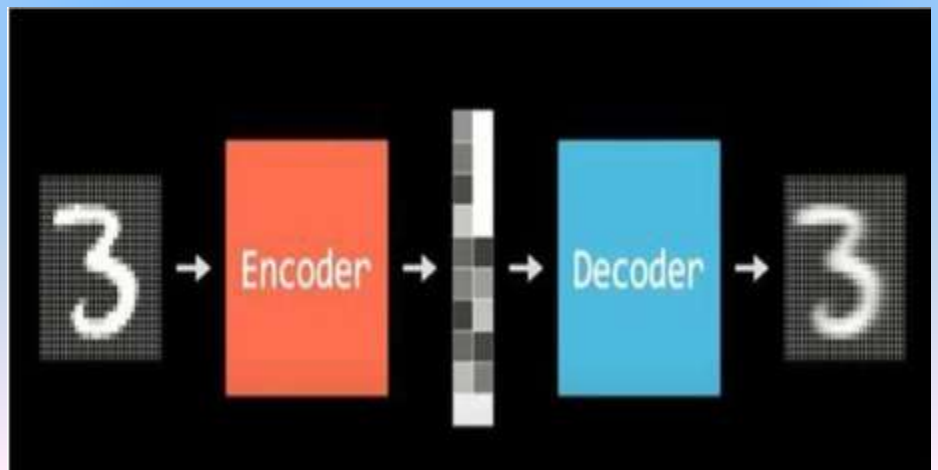


Exemple

- Soit une **collection d'images** de visages humains d'une résolution **1000x1000** pixels.
- Nous avons donc un espace de dimension 1 000 000.
- L'algorithme va donc prendre en entrée les 1 000 000 pixels qui sont stockés dans les variables x1,x2,x3,…,x1000000.
- Toutes ces variables sont en fait les paramètres qui sont fournis à l'entrée de l'algorithme du réseau de neurones.

Exemple


x'_1 x'_2 x'_3
$x'_{1000000}$
sortie
couche cachée
décodeur
λ_1 λ_2
λ_{50}
variables latentes
couche cachée
encodeur
entrée
x_1 x_2 x_3
$x_{1000000}$


Encoder
Decoder

Exemple

- Seulement un certain nombre de pixels peut servir à décrire les visage humains de notre exemple. Ainsi une réduction de dimension peut être envisagée.

- On peut par exemple émettre l'hypothèse que **seulement 50** variables correspondent à des visages humains. Ce sont ces 50 variables que l'on appelle: les variables latente λ1,λ2,λ3,…λ50.

- Ces dernières correspondent finalement à une forme compressée de l'in visuelle. Cela aboutit à la création d'un espace de plus faible dimension constitué de différents points et chaque point de cet espace correspond à un visage. Les coordonnées sont les variables latentes.

Auto-encodeurs

Couche cachée sous/sur-complète

Lorsque: $d < n$

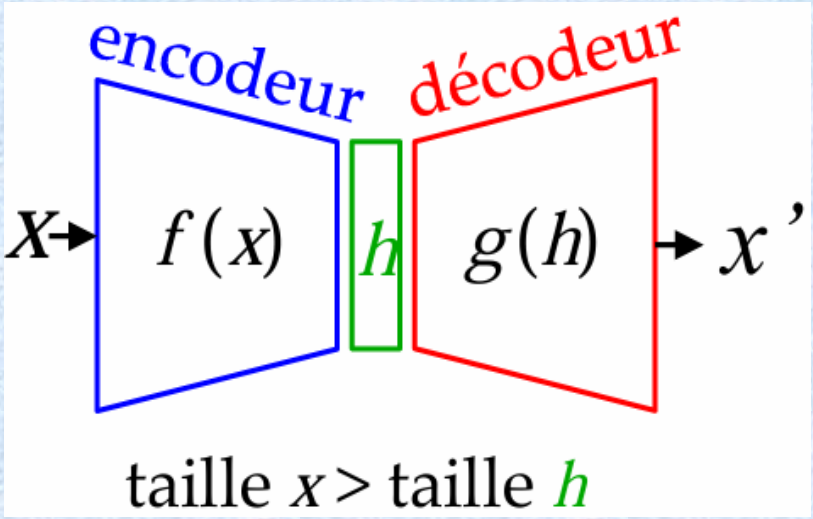
(la dimensionnalité de la couche cachée)<(la dimensionnalité de l'entrée).

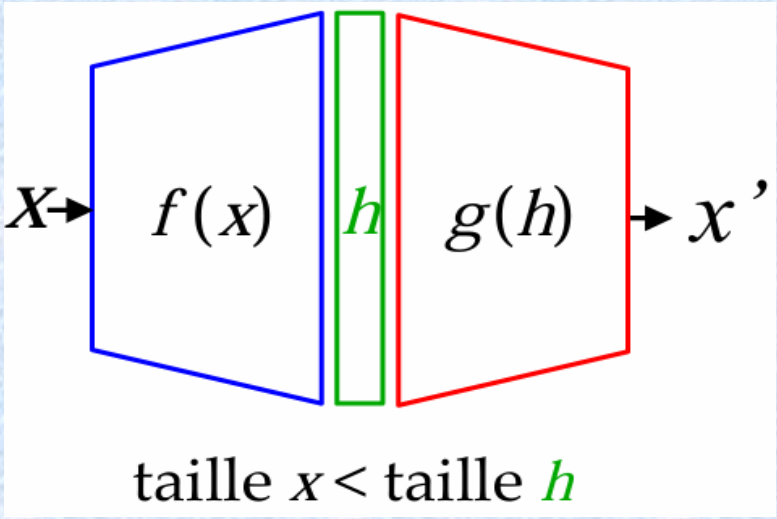
on dit que la couche cachée est **sous-complète**.

- Lorsque **d>*n***, nous disons qu'il s'agit d'une couche cachée **sur-complète**.

Auto-encodeurs

Couche cachée sous/sur-complète

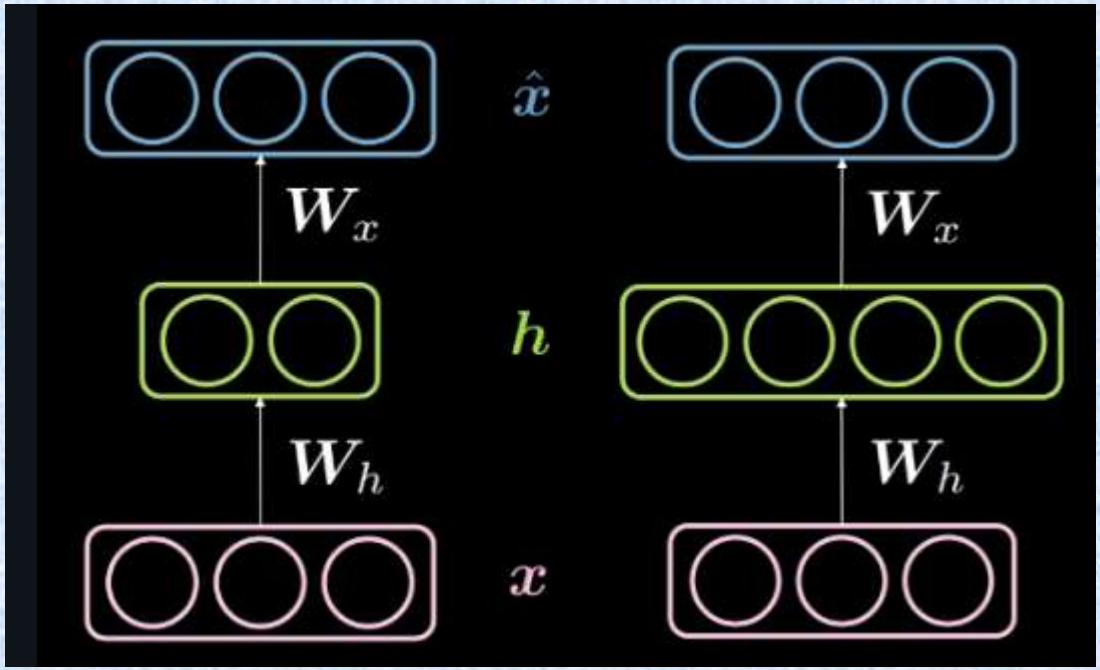




Une couche cachée sous-complète vs une couche cachée sur-complète

Auto-encodeurs

Couche cachée sous/sur-complète



Une couche cachée sous-complète vs une couche cachée sur-complète

Auto-encodeurs

Couche cachée sous/sur-complète

- Une couche **cachée sous-complète** peut être utilisée pour la compression car nous encodons les informations provenant de l'entrée en moins de dimensions.

- Dans une **couche sur-complète**, nous utilisons un codage de dimension plus élevée que l'entrée. Cela facilite l'optimisation.

Exemples des auto-encodeurs

Il existe plusieurs variantes de ce réseau. Des versions améliorées ou bien adaptées à des cas bien précis:

Auto-encodeurs variationnels (VAEs) :

- Un VAE est un type d'auto-encodeur probabiliste.

- Un VAE ne se contente pas de mapper des données à une représentation compressée, mais tente également de modéliser la distribution des données.

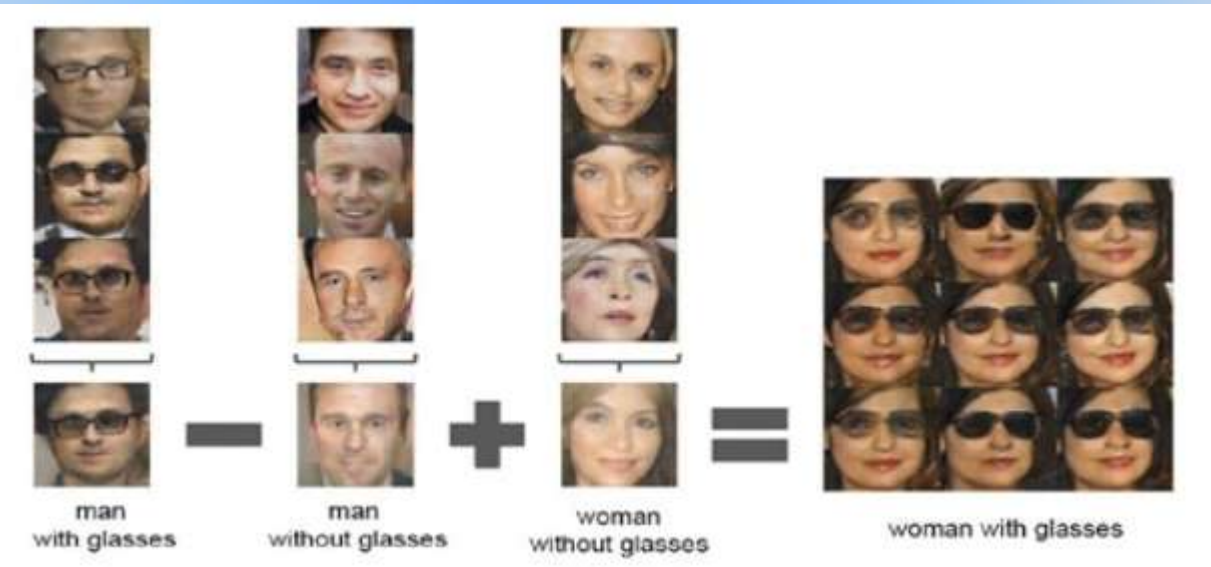
- Il peut être utilisé pour générer de nouvelles données en décodant une distribution gaussienne aléatoire.

Exemples des auto-encodeurs

Il existe plusieurs variantes de ce réseau.
Des versions améliorées ou bien adaptées à des cas bien précis:

Exemples des auto-encodeurs

Auto-encodeurs variationnels (VAEs) :



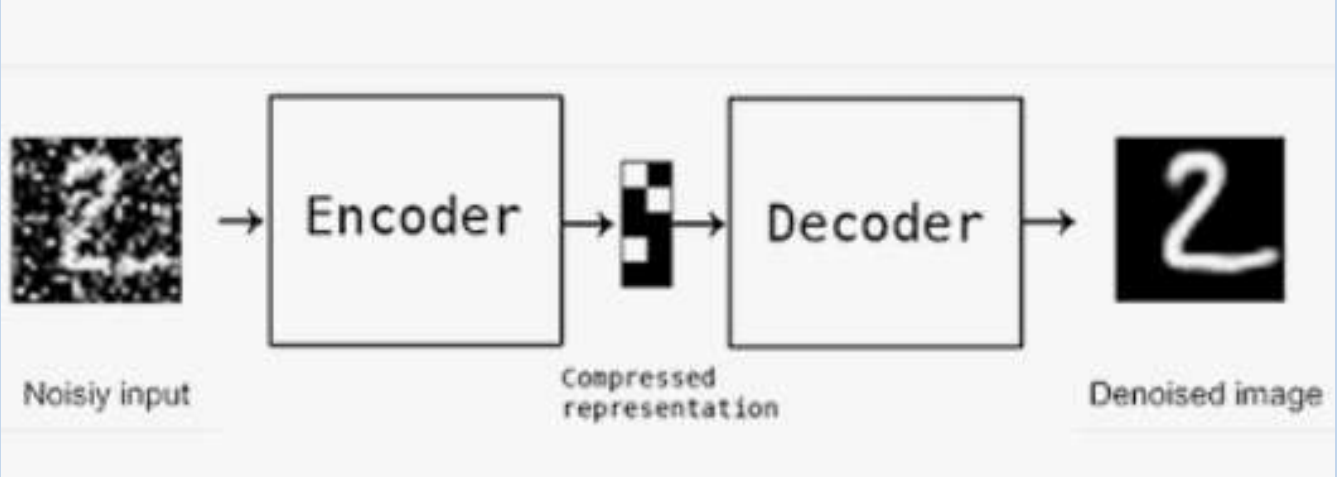
Exemples des auto-encodeurs

Les auto-encodeurs de débruitage :

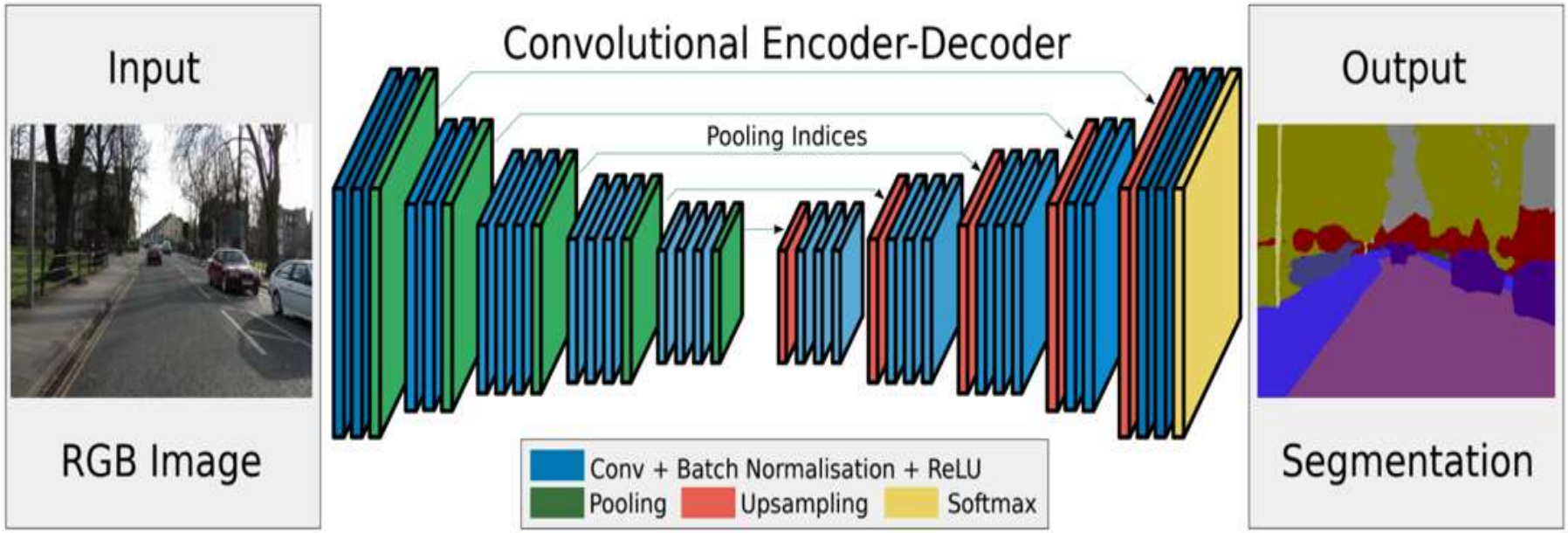
- Ce type ajoute du **bruit** à l'image d'entrée et apprend à le supprimer.
- Éviter ainsi de copier l'entrée vers la sortie sans en apprendre davantage sur les données.
- Ces auto-encodeurs prennent une entrée partiellement corrompue pendant la formation pour récupérer l'entrée originale non faussée.
- Ce type est utilisé pour améliorer la qualité des données bruitées ou pour restaurer des images dégradées.

Exemples des auto-encodeurs

Les auto-encodeurs de débruitage :

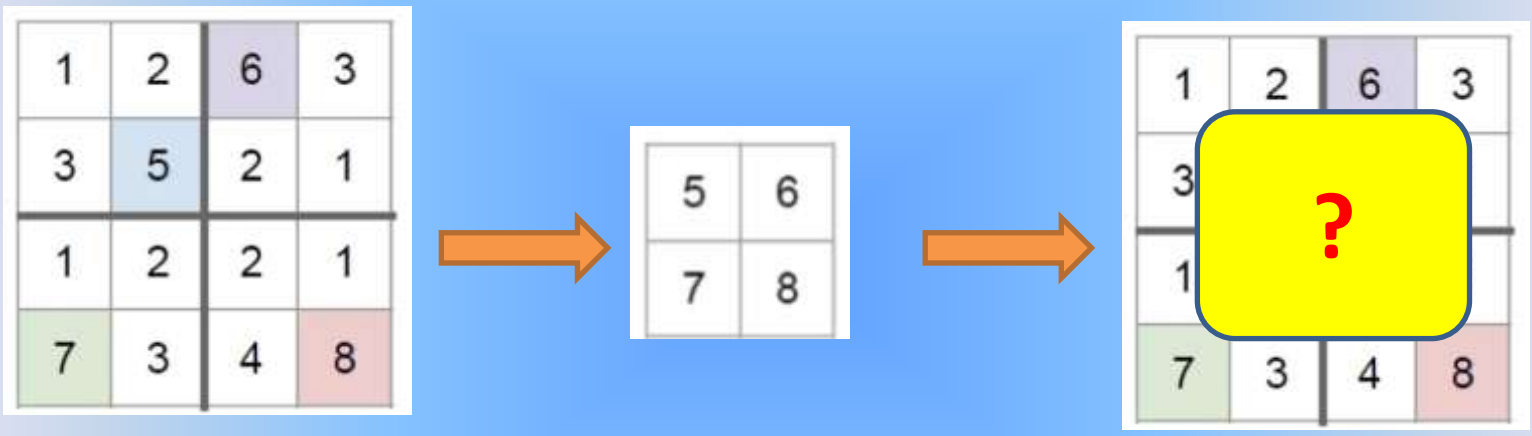


Exemple: SegNet



Downsampling: *sous-échantillonnage* dans l'encodeur (pooling).
Upsampling: sur-*échantillonnage* dans le décodeur.

Dégroupage

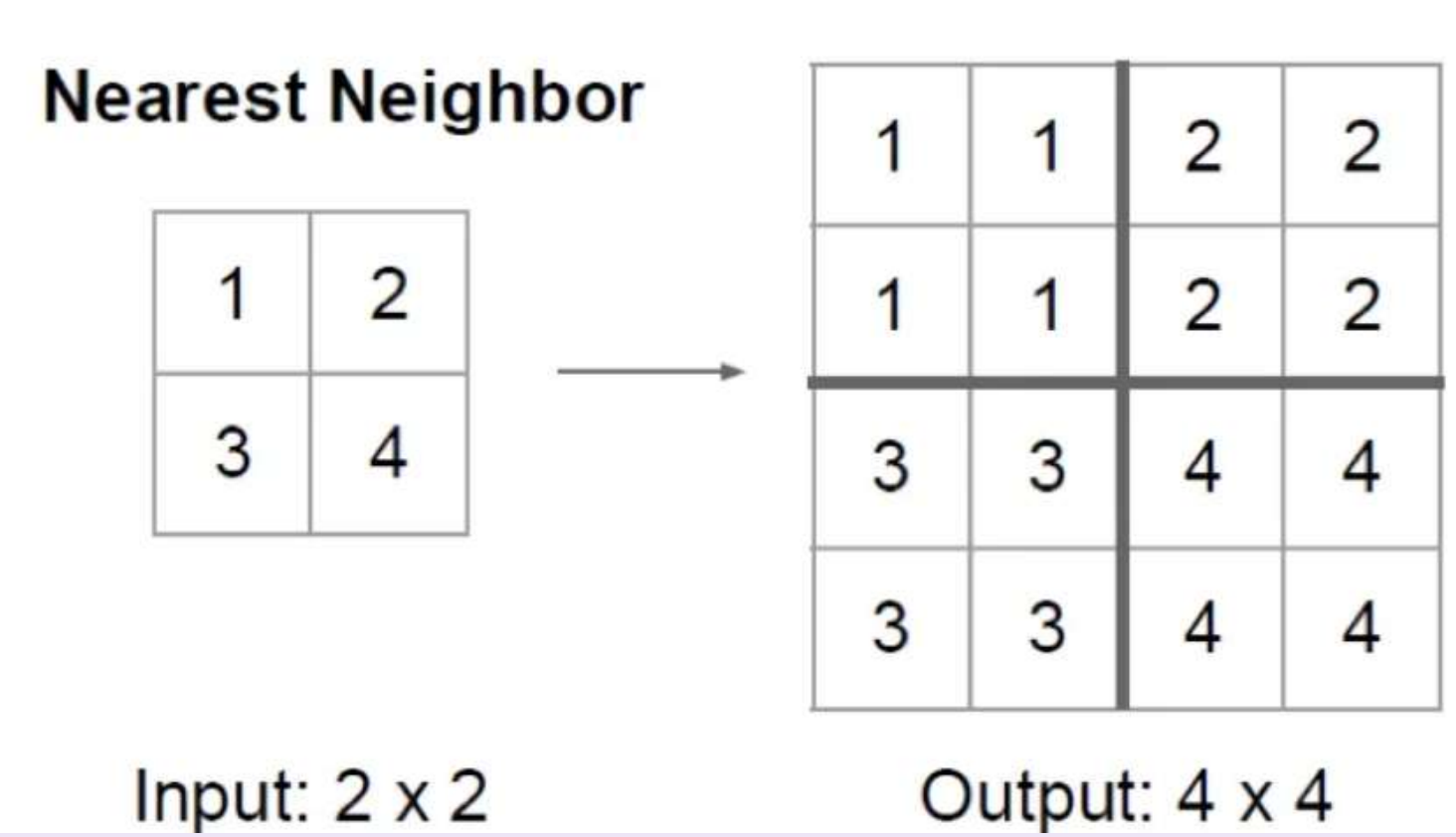


Groupage: Maxpooling

Dégroupage

Dégroupage

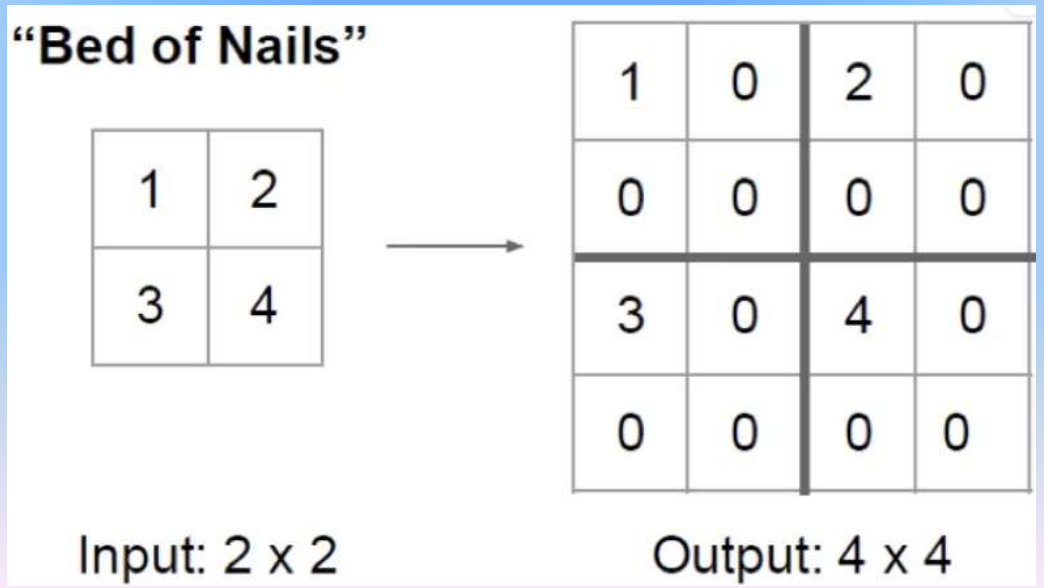
Voisin le plus proche



Dégroupage

Bed-of-Nails:

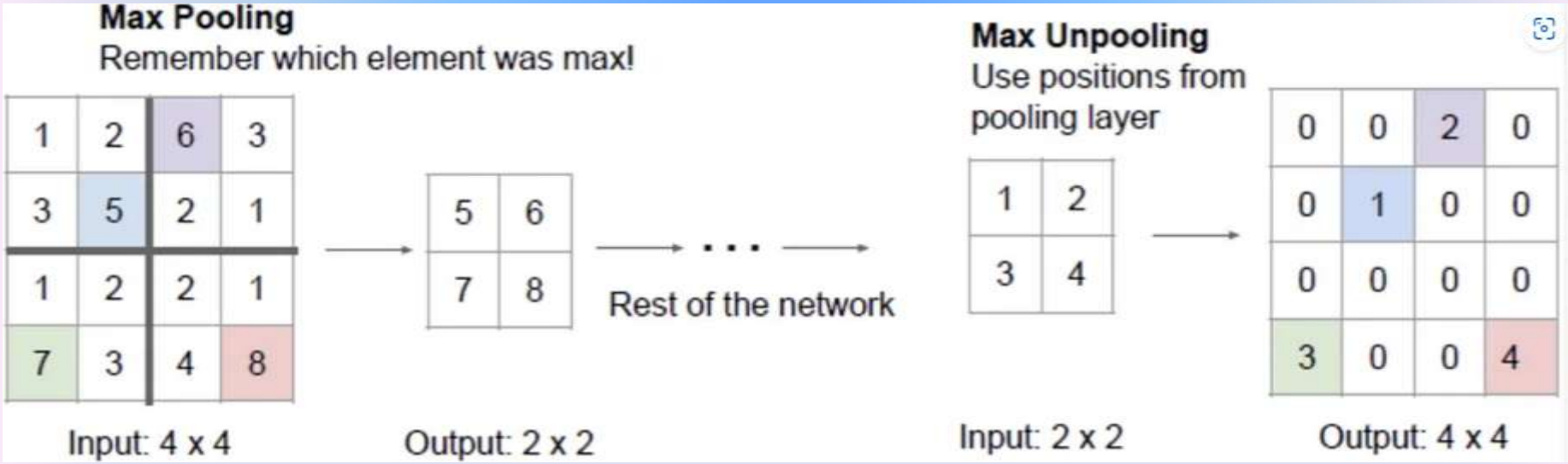
On place un élément d'entrée dans le coin supérieur gauche de la sous-région correspondante de la sortie de dépooling, et définit tous les autres éléments de la sous-région à zéro.



Dégroupage

Max *Unpooling*:

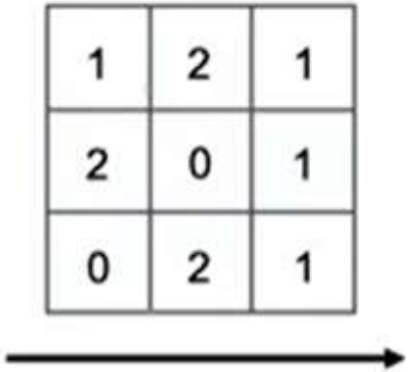
Il se souvient des **indices** d'où proviennent les éléments les plus importants avant le regroupement maximal.



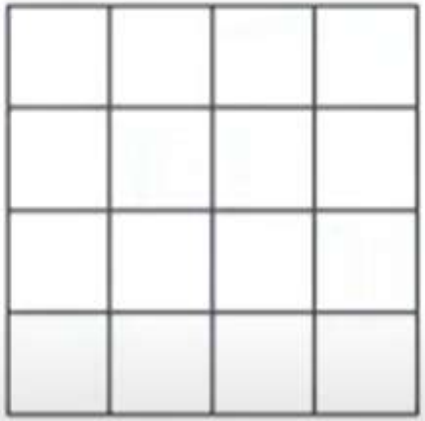
Convolution transposée

Exemple

Entrée 2*2



Filtre 3*3

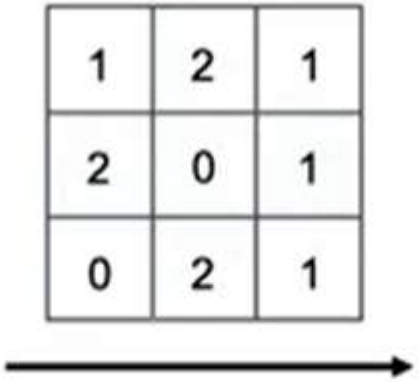


Sortie 4*4

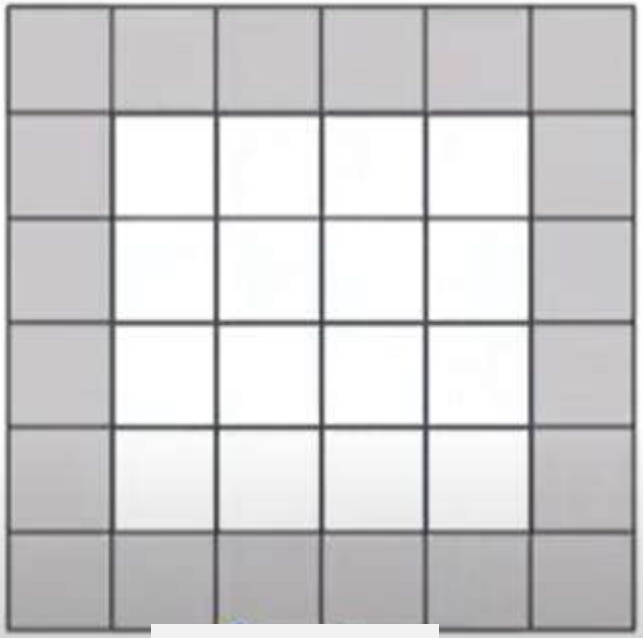
Padding =1 **Stride=2**

Convolution transposée

Entrée 2*2

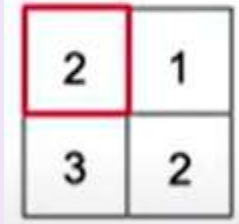


Filtre 3*3

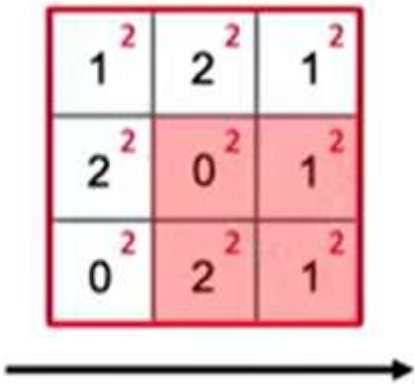


Padding=1

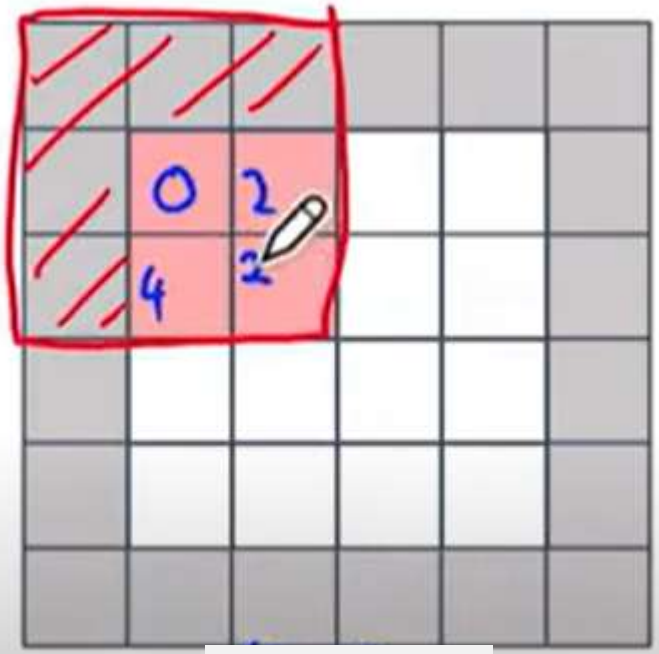
Convolution transposée



Entrée 2*2



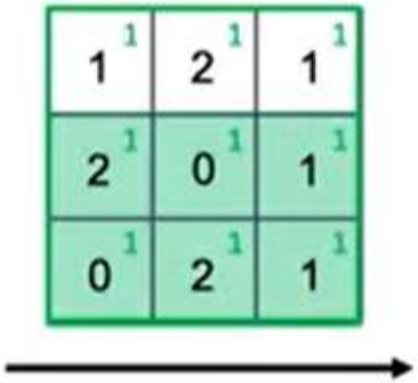
Filtre 3*3



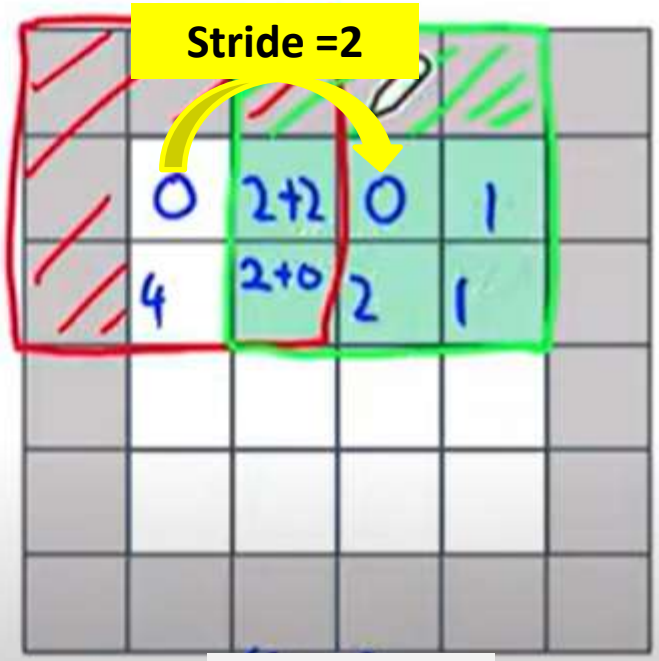
Sortie 4*4

Convolution transposée

Entrée 2*2

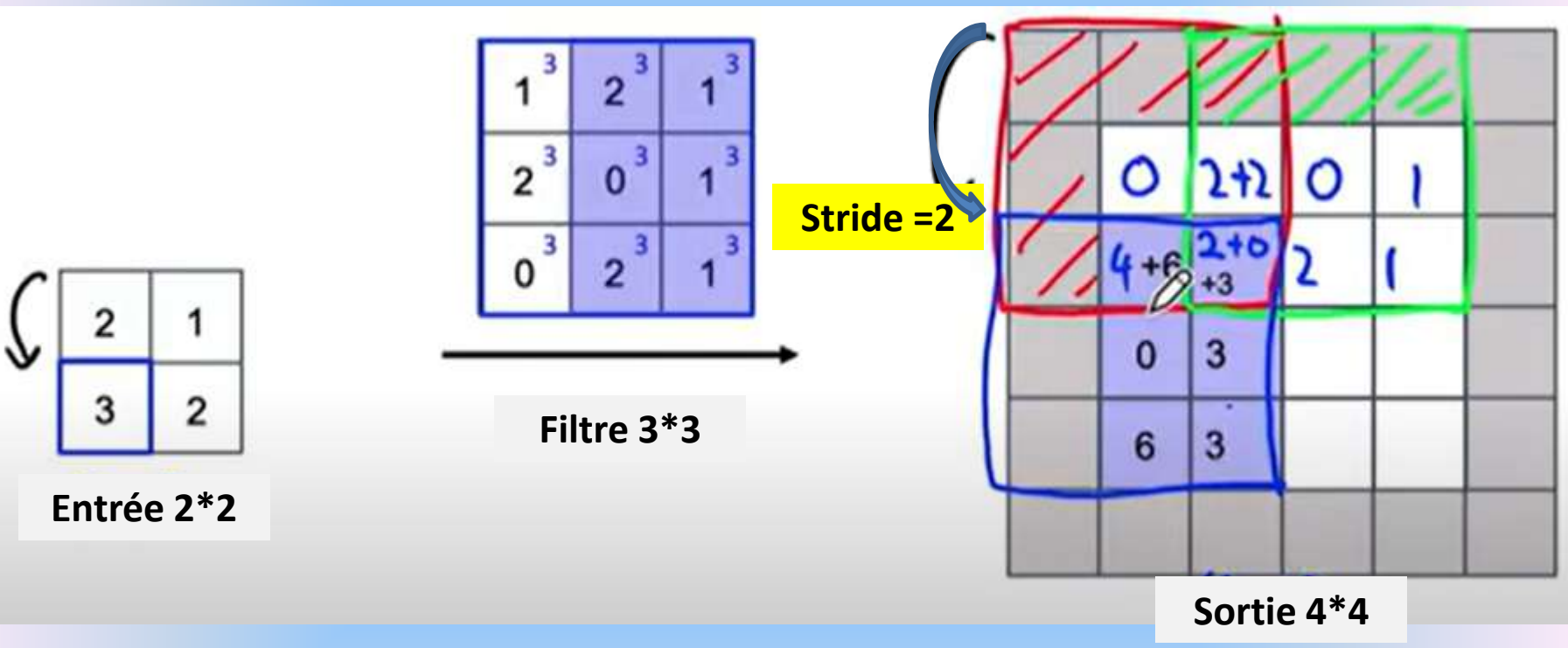


Filtre 3*3

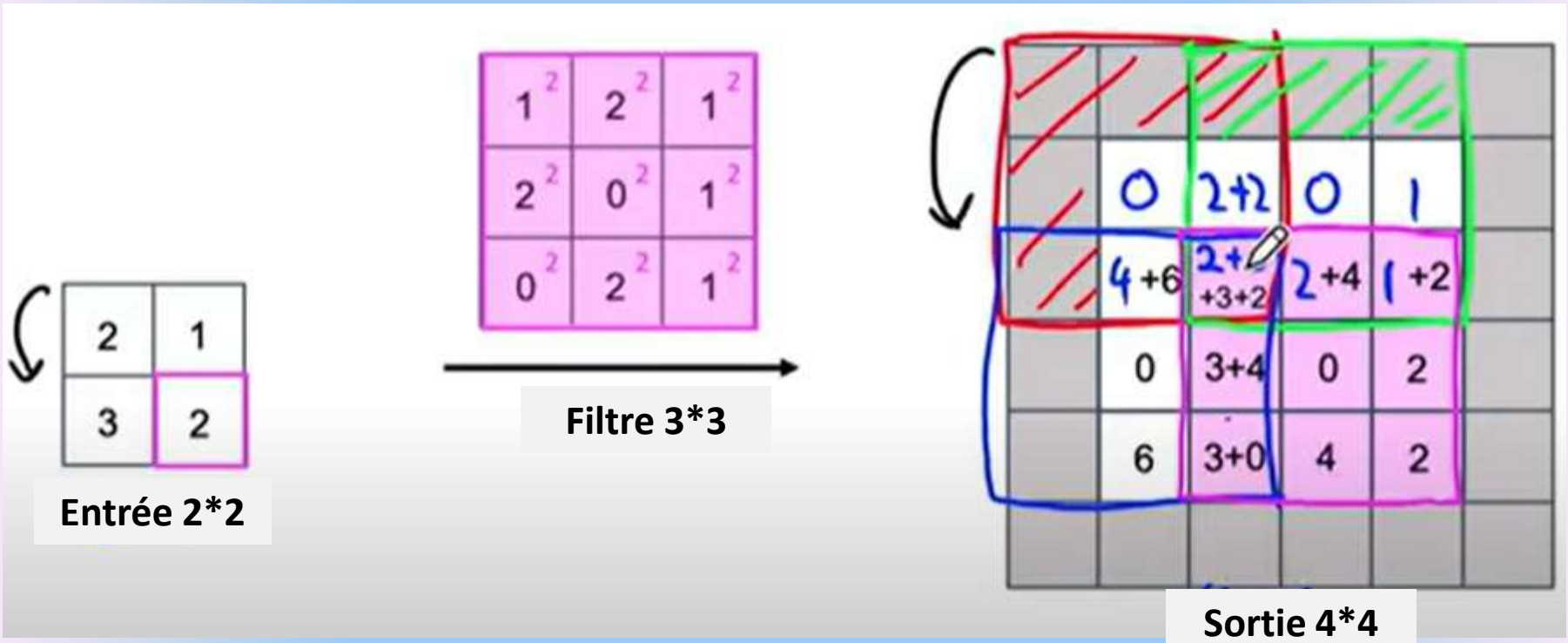


Sortie 4*4

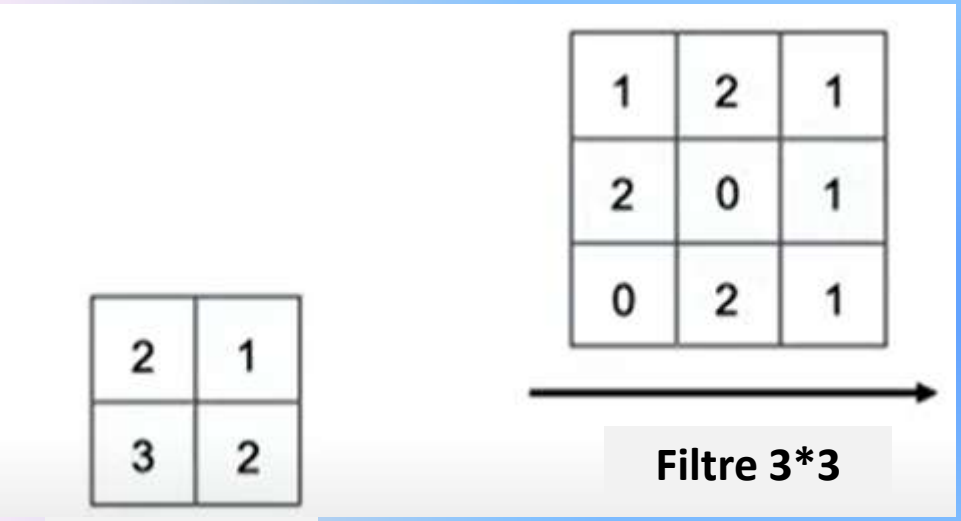
Convolution transposée



Convolution transposée



Convolution transposée



Entrée 2*2

Filtre 3*3

	0	4	0	1	
	10	7	6	3	
	0	7	0	2	
	6	3	4	2	

Sortie 4*4

Exemple

Etape 1: importer les bibliothèques nécessaires

Etape 2: charger le jeu de données

(X_train, _), (X_test, _) = mnist.load_data() X_train = X_train.reshape(X_train.shape[0], 28, 28, 1) X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)

Etape 3: prétraitement des images en entrée

X_train = X_train.astype("float32")/255. X_test = X_test.astype("float32")/255. print('X_train shape:', X_train.shape) print(X_train.shape[0], 'train samples') print(X_test.shape[0], 'test samples')

Etape 4: ***aplatir les matrices des images, afin de réduire leur dimension.***

X_train = X_train.reshape((len(X_train), np.prod(X_train.shape[1:]))) X_test = X_test.reshape((len(X_test), np.prod(X_test.shape[1:])))

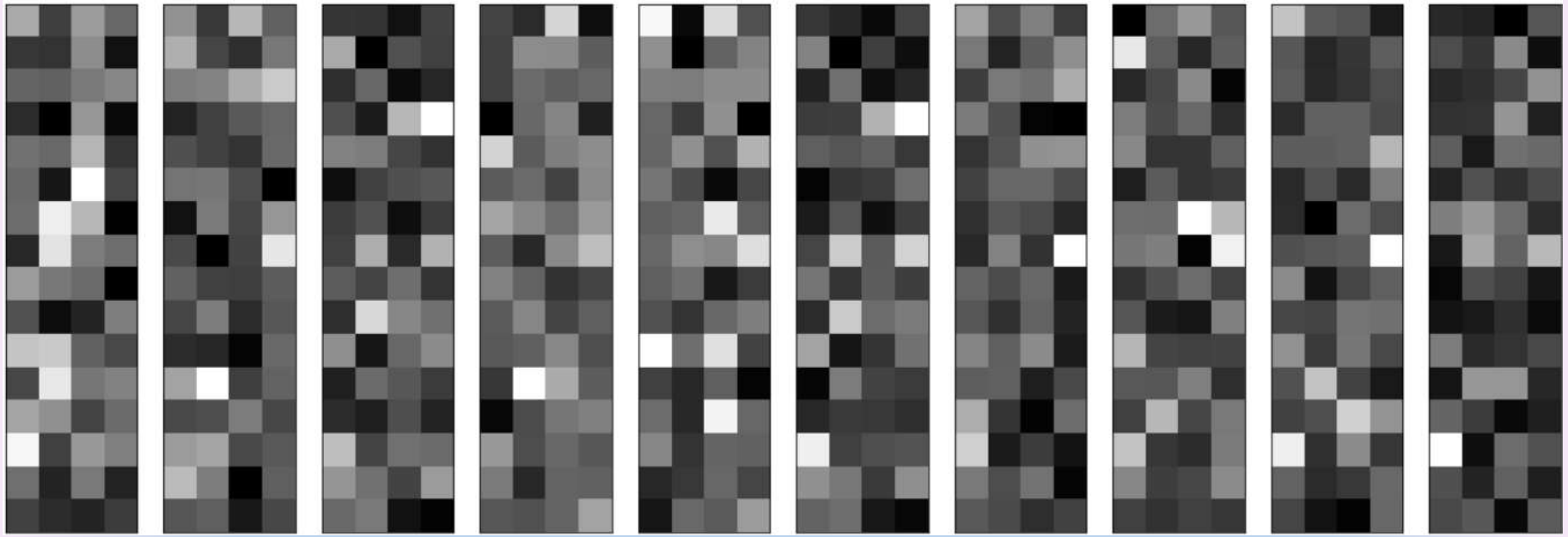
Exemple

Etape 5: Construire le modèle: une couche encodeur et une couche décodeur.

```
input_size = 784
hidden_size = 64
output_size = 784

x = Input(shape=(input_size,))
h = Dense(hidden_size, activation='relu')(x)
r = Dense(output_size, activation='sigmoid')(h)

autoencoder = Model(inputs=x, outputs=r)
autoencoder.compile(optimizer='adam', loss='mse')

epochs = 5
batch_size = 128
history = autoencoder.fit(X_train, X_train, batch_size=batch_size, epochs=epochs,
verbose=1, validation_data=(X_test, X_test))
```

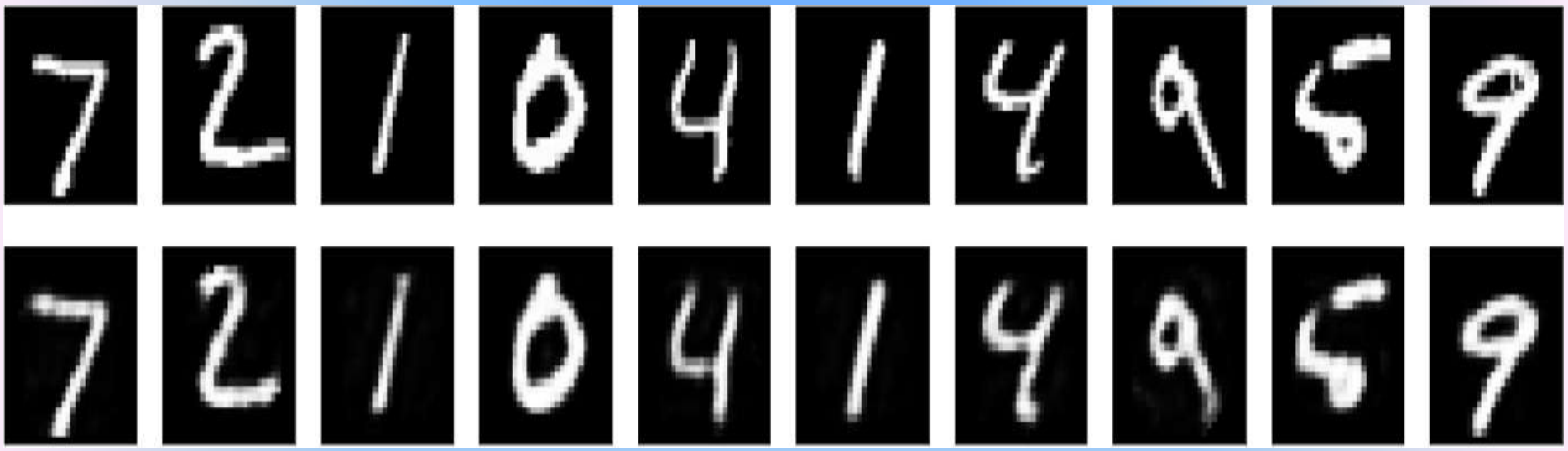
Exemple

Une représentation d'une image du dataset MNIST dans l'espace latent.
C'est-à-dire une représentation de l'image encodée.



Exemple

Une représentation d'une image du dataset MNIST dans l'espace latent.
C'est-à-dire une représentation de l'image encodée.



Applications des auto-encodeurs

Applications des auto-encodeurs

Image en super-résolution

Ce modèle vise à améliorer les images et à reconstruire les visages originaux.

Le visage reconstruit en bas à gauche est bizarre en raison de l'absence d'images sous cet angle dans les données d'entraînement.

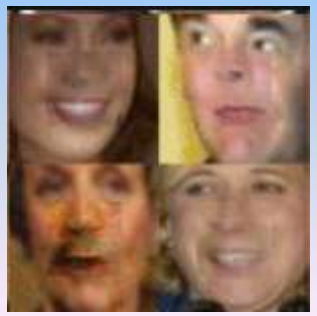
De gauche à droite, la première colonne est l'image d'entrée 16x16
la deuxième est le résultat d'une interpolation bicubique standard,
la troisième est la sortie générée par le réseau neuronal.
à droite est la véritable image.

Applications des auto-encodeurs

Image incomplète (inpainting)

En plaçant une tache grise sur le visage on éloigne l'image de la variété d'entraînement.

La reconstruction du visage est réalisée en trouvant l'échantillon d'image le plus proche sur la variété d'entraînement via la minimisation de la fonction énergie.



Applications des auto-encodeurs

Génération d'images d'après une légende

La traduction en image de la description textuelle de la figure 12 est réalisée en extrayant les représentations des caractéristiques textuelles associées à des informations visuelles importantes, puis en les décodant en images.

Applications des auto-encodeurs

Compression d'images:

Si nous avons une dimension intermédiaire *d* inférieure à la dimension d'entrée *n*, alors l'encodeur peut être utilisé comme un compresseur.

Hyper-paramètres des auto-encodeurs

La conception d'un auto-encodeur implique plusieurs hyperparamètres :

- **Taille du code :** la taille du goulot d'étranglement détermine la quantité de données à compresser

- **Nombre de couches :** la profondeur de l'auto-encodeur est mesurée par le ***nombre de couches*** de l'encodeur et du décodeur.

  Une plus grande profondeur permet des traitements plus complexes, alors qu'une profondeur plus petite offre une plus grande vitesse de traitement.

Hyper-paramètres des auto-encodeurs

La conception d'un auto-encodeur implique plusieurs hyperparamètres :

- **Nombre de nœuds par couche :** généralement, le nombre de nœuds (ou « neurones ») diminue à chaque couche d'encodage, il atteint son minimum au niveau du goulot d'étranglement et il augmente à chaque couche de décodage.

- **Fonction de perte :** mesure la perte de reconstruction entre la sortie et l'entrée.

Auto-encodeurs et encodeurs-décodeurs

Auto-encodeurs et encodeurs-décodeurs

- Bien que tous les modèles d'auto-encodeurs incluent à la fois un encodeur et un décodeur.

- Tous les modèles *d'encodeurs-décodeurs* ne sont pas des *auto-encodeurs.*

Auto-encodeurs et encodeurs-décodeurs

Dans les frameworks **encodeurs-décodeurs:**

- Un réseau d'**encodage** extrait les principales ***caractéristiques*** des données d'entrée.
- Un réseau de **décodage** *utilise* les caractéristiques extraites comme entrée.
- Ces frameworks interviennent dans divers modèles d'apprentissage profond : par exemple dans les architectures de CNN, ou dans les architectures de réseaux de neurones récurrents (RNN), utilisées dans les tâches de séquence à séquence (seq2seq).

Merci pour votre attention