Cours 8 - Apprentissage de représentations et apprentissage génératif avec des autoencodeurs et des GAN

Neila Mezghani

1 mars 2022

COLLÈGE DE
BOIS-DE-BOULOGNE

Plan du cours

Représentation des données

Plan du cours

Inspirations (1/2)

- Parmi les séries de nombres suivantes, laquelle trouvez-vous la plus facile à mémoriser ?
  - 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
  - 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14

Inspirations (1/2)

- Parmi les séries de nombres suivantes, laquelle trouvez-vous la plus facile à mémoriser ?
  - 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
  - 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14
- Puisque la première série est plus courte, on pourrait penser qu'il est plus facile de s'en souvenir. Cependant, en examinant attentivement la seconde, on remarque qu'il s'agit simplement d'une liste de nombres pairs allant de 50 à 14.

  $\Longrightarrow$ La deuxième est évidemment plus simple

Inspirations (2/2)

- Constat : les grands joueurs d'échecs sont capables de mémoriser la position de toutes les pièces en regardant l'échiquier pendant cinq secondes seulement (un défi irréalisable pour nous).
- Cependant, ce n'est le cas que si les pièces se trouvaient dans des configurations réalistes (des parties réelles) et non lorsqu'elles étaient placées aléatoirement.

  $\Longrightarrow$ Les joueurs d'échecs professionnels n'ont pas une meilleure mémoire que nous, ils reconnaissent simplement des motifs de placement plus facilement en raison de leur expérience de ce jeu. Ils sont ainsi capables de stocker les informations plus efficacement.

Plan du cours

Inspirations... vers l'apprentissage de représentations (3/3)

- À l'instar des joueurs d'échecs dans cette expérience sur la mémoire, un autoencodeur examine les entrées, les convertit en une représentation latente efficace et produit en sortie quelque chose qui, idéalement, ressemble énormément à l'entrée

  $\Longrightarrow$ **Apprentissage de représentations** : Capacité d'un réseau de neurones profond d'apprendre des caractéristiques ou des attributs de complexité hiérarchiquement croissante à partir d'un ensemble de données brutes.

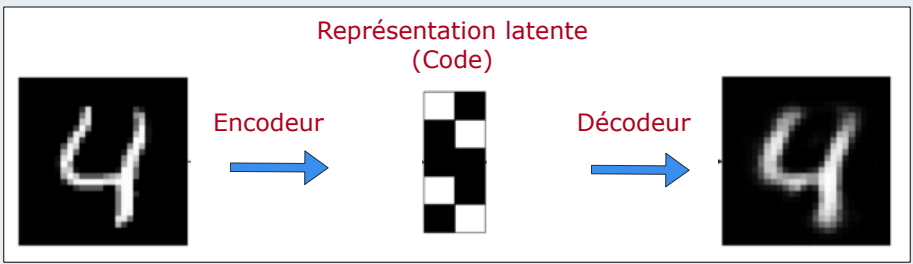
Les autoencodeurs

Plan du cours

Architecture des autoencodeurs (1/5)

Un autoencodeur est toujours constitué de deux parties :

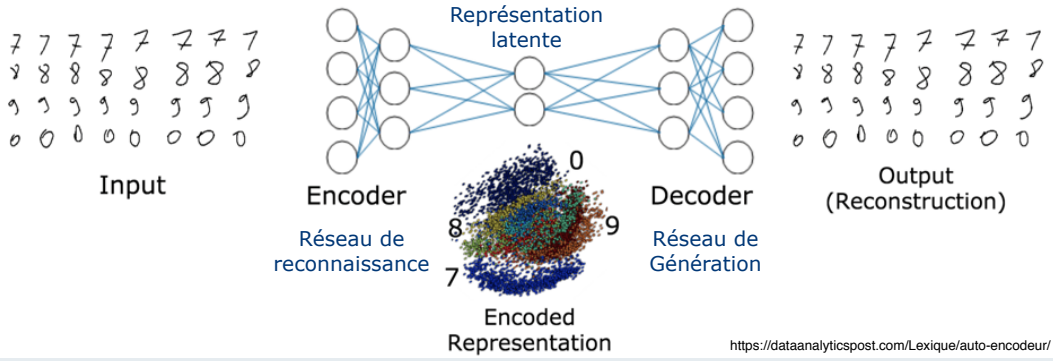
- Un encodeur (ou réseau de reconnaissance) qui convertit les entrées en une représentation latente
- Un décodeur (ou réseau de génération) qui convertit la représentation interne en sorties



Architecture des autoencodeurs (2/5)

- Un auto-encodeur est un réseau de neurones non récurrents qui se propage vers l'avant
- Il possède généralement la même architecture qu'un PMC ayant une couche d'entrée, une couche de sortie ainsi qu'une ou plusieurs couches cachées les reliant
- Les sorties sont souvent appelées reconstructions car l'autoencodeur tente de reconstruire les entrées

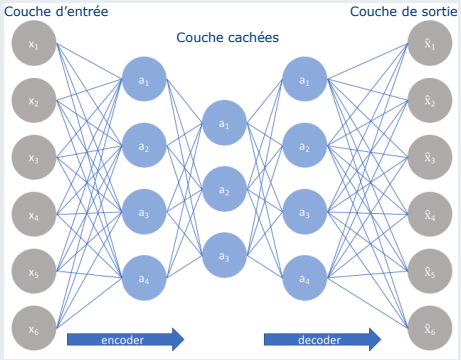
Architecture des autoencodeurs (3/5)



https://dataanalyticspost.com/Lexique/auto-encodeur/

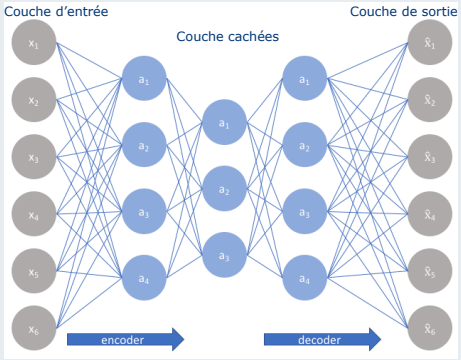
Architecture des autoencodeurs (4/5)

- La couche de sortie (du décodeur) possède le même nombre de nœuds que la couche d'entrée (de l'encodeur)



Architecture des autoencodeurs (5/5)

- L'apprentissage est non-supervisé car fonction de coût consiste à minimiser le coût de reconstruction entre la sortie et l'entrée.
- Les données n'ont donc pas à être étiquetées, car elles sont leurs propres étiquettes, ce qui fait donc de ce modèle un modèle non supervisé.



Plan du cours

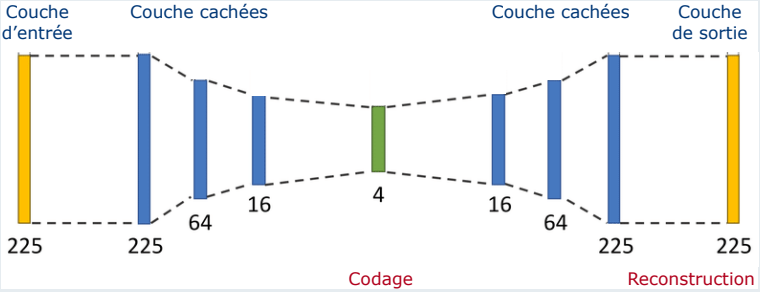
Les types d'autoencodeurs

Il existe différents types d'encodeurs dont :

- Autoencodeur profond (empilés)
- Autoencodeur convolutif
- Autoencodeur débruiteur
- Autoencodeur épars
- Autoencodeur variationnel

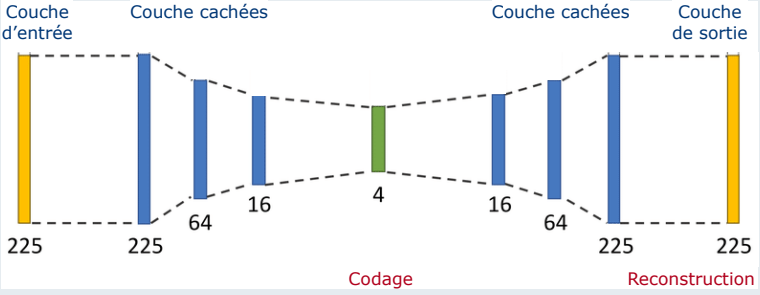
Autoencodeurs profonds (1/2)

- Les autoencodeurs peuvent comprendre plusieurs couches cachées.
- Dans ce cas, il s'agit d'autoencodeurs empilés ou autoencodeurs profonds.



Autoencodeurs profonds (2/2)

- L'architecture d'un autoencodeur empilé est le plus souvent symétrique par rapport à la couche cachée de codage.
- En ajoutant des couches, l'autoencodeur devient capable d'apprendre des codages plus complexes.



Exemple : Autoencodeurs profonds (1/2)

Le modèle de l'autoencodeur est divisé en deux sous-modèles : l'encodeur et le décodeur.

```
stacked_encoder = keras.models.Sequential([
        keras.layers.Flatten(input_shape=[28, 28]),
        keras.layers.Dense(100, activation="selu"),
        keras.layers.Dense(30, activation="selu")])
stacked_decoder = keras.models.Sequential([
        keras.layers.Dense(100, activation="selu",
        input_shape=[30]),
        keras.layers.Dense(28 * 28, activation="sigmoid"),
        keras.layers.Reshape([28, 28])])
```

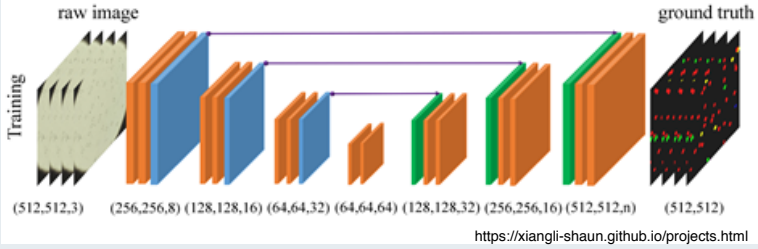
Exemple : Autoencodeurs profonds (2/2)

Pour la compilation de l'autoencodeur empilé, on utilise l'entropie croisée binaire. On considère la tâche de reconstruction comme un problème de classification binaire multiétiquette : chaque intensité de pixels représente la probabilité que le pixel doit être noir.

```
stacked_ae = keras.models.Sequential([stacked_encoder,
        stacked_decoder])
stacked_ae.compile(loss="binary_crossentropy",
optimizer=keras.optimizers.SGD(lr=1.5))
history = stacked_ae.fit(X_train, X_train, epochs=10,
validation_data=[X_valid, X_valid])
```

Autoencodeurs convolutifs (1/2)

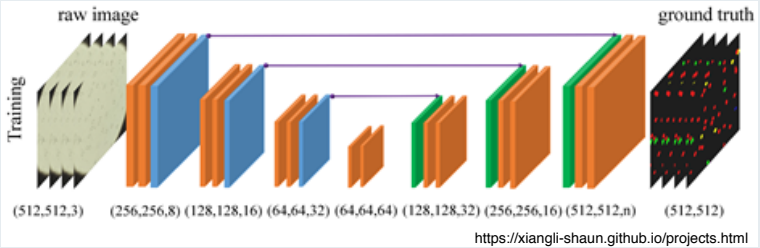
- Les réseaux de neurones convolutifs conviennent mieux au traitement des images de grandes dimensions que les réseaux denses $\Longrightarrow$ On utilise des Autoencodeurs convolutifs pour le codage et la reconstruction d'image
- L'encodeur est un CNN constitué de couches de convolution et de couches de pooling.
- Il permet de réduire la dimensionnalité spatiale des entrées



https://xiangli-shaun.github.io/projects.html

Autoencodeurs convolutifs (2/2)

- Le décodeur doit réaliser l'opération inverse, c-à-d, augmenter la taille de l'image et réduire sa profondeur aux dimensions d'origine
- On utilise des couches de convolution transposées pour le suréchantillonnage des données.



Exemple : Autoencodeurs convolutif (1/2)

L'encodeur pour Fashion MNIST est donné par :

```
conv_encoder = keras.models.Sequential([
keras.layers.Reshape([28, 28, 1], input_shape=[28, 28]),
keras.layers.Conv2D(16, kernel_size=3, padding="same",
        activation="selu"),
keras.layers.MaxPool2D(pool_size=2),
keras.layers.Conv2D(32, kernel_size=3, padding="same",
        activation="selu"),
keras.layers.MaxPool2D(pool_size=2),
keras.layers.Conv2D(64, kernel_size=3, padding="same",
        activation="selu"),
keras.layers.MaxPool2D(pool_size=2)])
```

Exemple : Autoencodeurs convolutif (2/2)

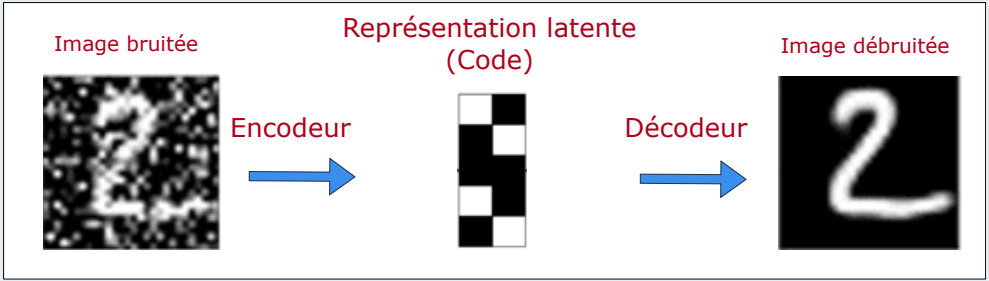
et le décodage par :

```
conv_decoder = keras.models.Sequential([
keras.layers.Conv2DTranspose(32, kernel_size=3, strides=2,
        padding="valid", activation="selu",
        input_shape=[3, 3, 64]),
keras.layers.Conv2DTranspose(16, kernel_size=3, strides=2,
        padding="same", activation="selu"),
keras.layers.Conv2DTranspose(1, kernel_size=3, strides=2,
        padding="same", activation="sigmoid"),
keras.layers.Reshape([28, 28])
])
conv_ae = keras.models.Sequential([conv_encoder, conv_decoder])
```

Autoencodeurs débruiteurs (1/2)

- Pour obliger l'autoencodeur à apprendre des caractéristiques utiles, une autre approche consiste à rajouter du bruit sur ses entrées et à l'entraîner pour qu'il retrouve les entrées d'origine, sans le bruit.
- Le bruit peut être un bruit blanc gaussien ajouté aux entrées ou un blocage aléatoire des entrées.

Autoencodeurs débruiteurs (2/2)



Exemple : Autoencodeurs débruiteurs

Un autoencodeur débruiteur basé sur l'ajout d'une couche `dropout`

```
dropout_encoder = keras.models.Sequential([
        keras.layers.Flatten(input_shape=[28, 28]),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(100, activation="selu"),
        keras.layers.Dense(30, activation="selu")])
dropout_decoder = keras.models.Sequential([
        keras.layers.Dense(100, activation="selu",
        input_shape=[30]),
        keras.layers.Dense(28 * 28, activation="sigmoid"),
        keras.layers.Reshape([28, 28])])
dropout_ae = keras.models.Sequential([dropout_encoder,
        dropout_decoder])
```

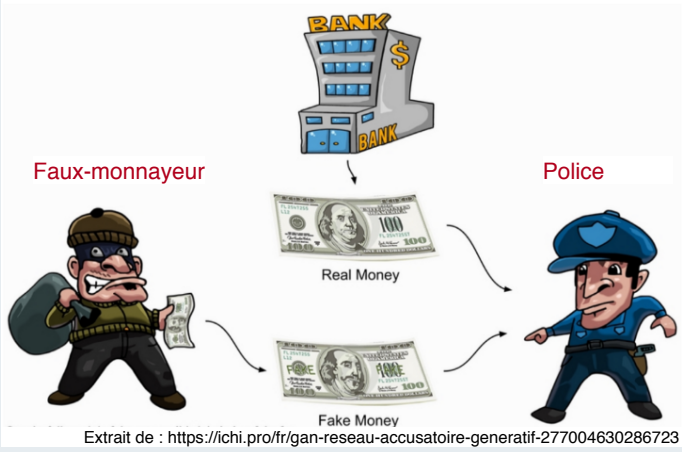
Les réseaux antagonistes génératifs (GAN)

Plan du cours

Les réseaux antagonistes génératifs (1/4)

- Les réseaux adverses génératifs (en anglais generative adversarial networks ou GANs) se basent sur l'apprentissage non supervisé.
- Les GAN étaient introduits par Goodfellow et al. 2014.
- Ils permettent de générer des images avec un fort degré de réalisme.

Les réseaux antagonistes génératifs (2/4)



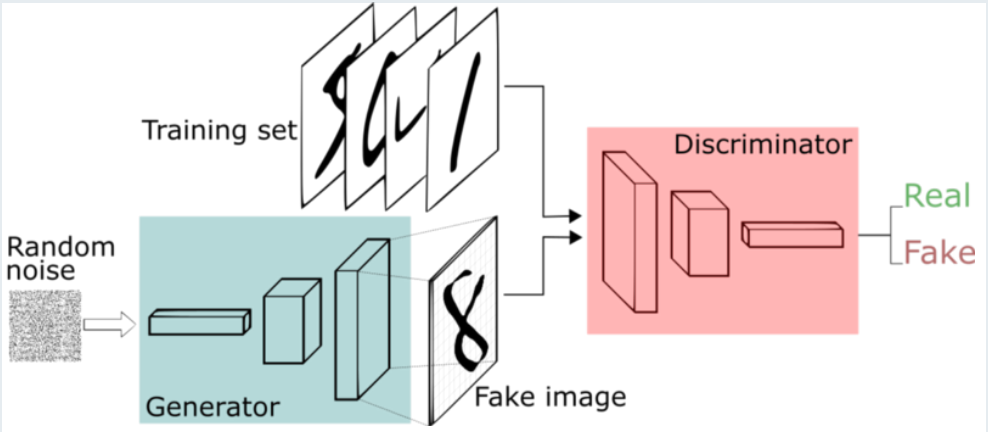
Extrait de : https://ichi.pro/fr/gan-reseau-accusatoire-generatif-277004630286723

Les réseaux antagonistes génératifs (3/4)

Un GAN est constitué de deux réseaux de neurones :

- Un générateur : Il prend en entrée une distribution aléatoire (typiquement gaussienne) et produit en sortie des données – généralement une image.
- Un discriminateur : Il prend en entrée soit une image factice provenant du générateur, soit une image réelle provenant du jeu d'entraînement, et doit deviner si cette image d'entrée est fausse ou réelle.

Les réseaux antagonistes génératifs (4/4)



https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

Plan du cours

Entrainement d'un GAN (1/5)

Pendant l'entraînement, le générateur et le discriminateur ont des objectifs opposés :

- Le discriminateur tente de distinguer les images factices des images réelles.
- Le générateur tente de produire des images suffisamment réelles pour tromper le discriminateur.

Entrainement d'un GAN (2/5)

Le GAN est constitué de deux réseaux aux objectifs différents $\Longrightarrow$ il ne peut pas être entraîné à la manière d'un réseau de neurones normal.
Chaque itération d'entraînement comprend deux phases :

- Au cours de la première phase, le discriminateur est entrainé.
- Au cours de la deuxième phase, le générateur est entrainé.

Entrainement d'un GAN (3/5)

L'entrainement du discriminateur :

- Un lot d'images réelles est choisi à partir du jeu entraînement, auquel est ajouté un nombre égal d'images factices produites par le générateur.
- Les étiquettes sont fixées à 0 pour les images factices et à 1 pour les images réelles.
- Le discriminateur est entraîné sur ce lot étiqueté pendant une étape, en utilisant une perte d'entropie croisée binaire.
- Au cours de cette étape, la rétropropagation optimise uniquement les poids du discriminateur.

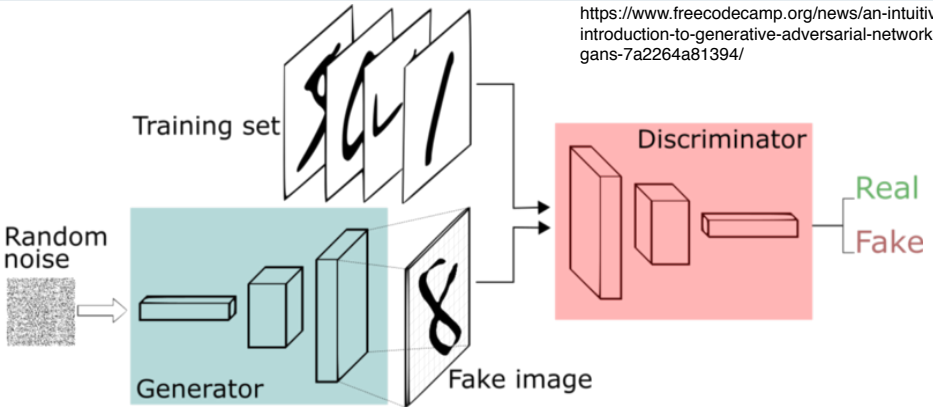
Entrainement d'un GAN (4/5)

L'entrainement du générateur :

- Un lot d'images factices est produits par le générateur.
- Le discriminateur doit dire si les images sont fausses ou réelles.
- Le générateur est forcé à produire des images que le discriminateur considérera (à tort) réelles
- La rétropropagation affecte uniquement les poids du générateur.

Entrainement d'un GAN (5/5)

https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/



Le générateur ne voit pas les images réelles
Il apprend progressivement à produire de fausses images convaincantes en se basant sur les gradients qui reviennent du discriminateur.

Exemple : Les GAN (1/2)

Modèle du générateur :

```
codings_size = 30
generator = keras.models.Sequential([
keras.layers.Dense(100, activation="selu",
        input_shape=[codings_size]),
keras.layers.Dense(150, activation="selu"),
keras.layers.Dense(28 * 28, activation="sigmoid"),
keras.layers.Reshape([28, 28])
])
```

Exemple : Les GAN (2/2)

Modèle du discriminateur :

```
discriminator = keras.models.Sequential([
keras.layers.Flatten(input_shape=[28, 28]),
keras.layers.Dense(150, activation="selu"),
keras.layers.Dense(100, activation="selu"),
keras.layers.Dense(1, activation="sigmoid")
])
gan = keras.models.Sequential([generator, discriminator])
```