Notes de cours
Semaine 18

Cours Turing

1 Colorisation d'images

La colorisation d'images a pour objectif de coloriser une image en noir et blanc.

Figure 1: Exemple de colorisation d'images

Les applications de la colorisation d'images sont nombreuses, elles permettent de redonner vie à des images anciennes, de les rendre plus réalistes, ou de les utiliser pour l'entrainement de modèles de reconnaissance d'objets.

De plus, nous avons discuté de l'importance des données pour l'entrainement des réseaux de neurones. La colorisation d'images est un problème pour lequel il est extrêmement facile de trouver des données d'entrainement. En effet, chaque image en couleur peut être convertie en image en noir et blanc, ce qui permet de générer des données d'entrainement facilement.

2 Paramétrisation du problème

Afin que l'entrainement des modèles de colorisation soit efficace, il est nécessaire de paramétriser le problème. Cela signifie qu'il est nécessaire de définir la représentation des données d'entrée et de sortie.

Image en gris - formule NTSC

Nous utiliserons la formule NTSC pour convertir les images en couleur en images en noir et blanc. Pour rappel, la formule NTSC est la suivante : $Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$.

Image en couleur - Espace de couleurs Lab

L'espace de couleurs Lab est un espace de couleurs tridimensionnel.

Il est composé de trois composantes : L, a et b. La composante L représente la luminance de l'image, tandis que les composantes a et b représentent les couleurs.
Cette représentation est généralement utilisée pour la colorisation d'images, car elle permet de séparer la luminance des couleurs, ce qui permet, à priori, un entrainement plus efficace des modèles de colorisation.

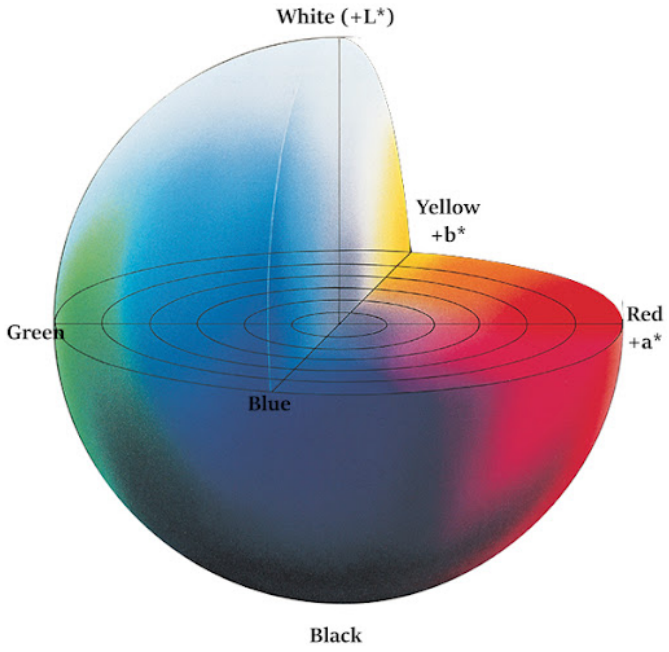

Figure 2: Espace de couleurs Lab

! Il est possible d'utiliser n'importe quel espace de couleur pour la colorisation d'images.
En fonction de la paramétrisation, l'entrainement du modèle sera plus ou moins efficace et la qualité des résultats sera plus ou moins bonne. Il est impossible de savoir à l'avance ce qui fonctionnera le mieux.

Astuces

Il existe de nombreuses astuces permettant d'entrainer un réseau plus facilement.

1. Utiliser des données centrées et réduites. Les entrées sont généralement dans l'intervalle $[0, 1]$ ou $[-1, 1]$. Cela permet de faciliter l'entrainement des modèles.
2. La sortie d'un réseau n'est généralement pas bornée. S'il est possible de borner la sortie, cela facilitera l'entrainement du modèle.

Dans l'exercice de la semaine, les données d'entrées, des images en noir et blanc, seront dans l'intervalle $[0, 1]$. Les données de sortie, des images en couleur, sont dans l'intervalle $[-1, 1]$.

Pour borner les données de sortie, nous utiliserons la fonction sigmoïde donnée par la formule :

$$f(x) = \frac{1}{1 + e^{-x}}$$

L'intervalle de sortie de cette fonction est $[0, 1]$. Pour obtenir un intervalle $[-1, 1]$, il suffit de multiplier la sortie par 2 et de soustraire 1.

La fonction suivante sera utilisée à la sortie du modèle pour borner les données de sortie :

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

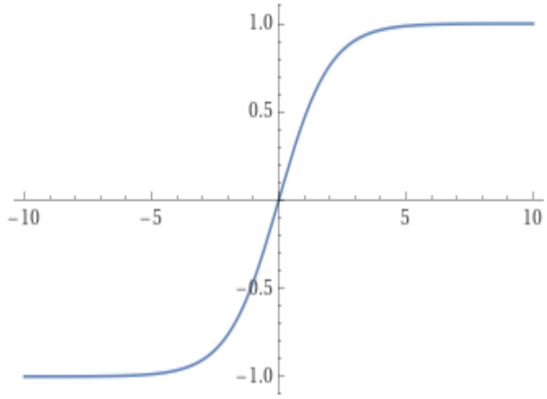

Figure 3: La fonction sigmoïde ajustée dans l'intervalle $[-1, 1]$

! Le but ici est de ne pas permettre au réseau de produire des valeurs extrêmes afin d'éviter la divergence lors de l'entrainement.

Résumé de la paramétrisation

- Données d'entrée : images en niveaux de gris (formule NTSC) dans l'intervalle $[0, 1]$. Le tenseur d'entrée est de taille $Hauteur \times Largeur \times 1$.
- Données de sortie : images en couleur (espace de couleurs Lab) dans l'intervalle $[-1, 1]$. Le tenseur de sortie est de taille $Hauteur \times Largeur \times 3$.

Il y aura donc un pré-traitement des données d'entrée et un post-traitement des données de sortie. Si ces étapes ne sont pas effectuées, les résultats seront imprévisibles.

3 U-Net

Le modèle U-Net est une forme d'auto-encodeur, il s'agit d'un modèle très populaire qui a été prévu initialement pour la segmentation sémantique d'images.

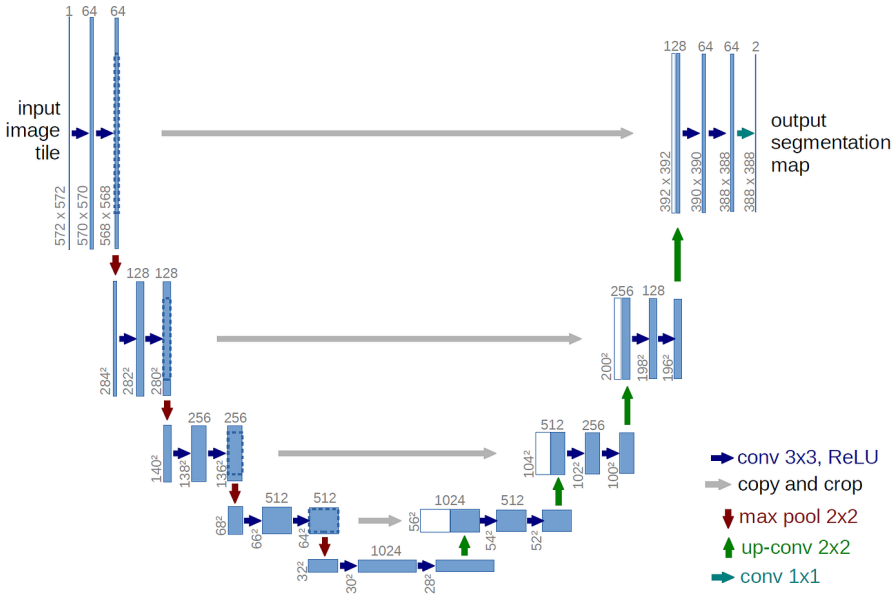

Figure 4: L'architecture du modèle U-Net

Il est composé de deux parties : un encodeur et un décodeur. L'encodeur est composé de couches de convolution et de sous-échantillonnage, tandis que le décodeur est composé de couches de convolution et de **sur-échantillonnage**.

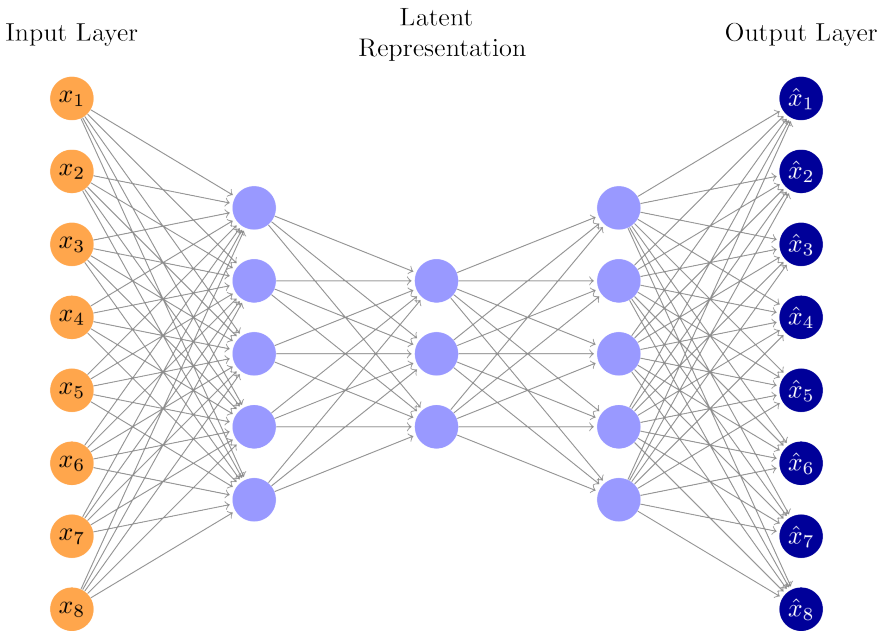

Figure 5: Structure d'un Auto-encodeur

! Dans les exemples vus en semaine 19, les réseaux de neurones n'étaient consitués que de la partie "encodeur", le nombre de neurones diminuait à chaque couche jusqu'à atteindre la taille de sortie de 10 pour classifier les chiffres dans le dataset MNIST. Les auto-encodeurs produisent une sortie de

taille similaire à l'entrée en utilisant les couches de sur-échantillonnage qui ont pour but d'inverser les opérations de sous-échantillonnage (MaxPooling).

Pourquoi les auto-encodeurs sont intéressants ?

L'espace latent est un espace de représentation des données appris par le modèle. Il est généralement de dimension inférieure à l'espace d'entrée et permet de représenter les données de manière très compacte.
En modifiant l'espace latent, il est possible de générer de nouvelles données. Cela signifie que les auto-encodeurs sont capables de générer des données similaires à celles d'entrée.
Exemples :

1. This Person Does Not Exist
2. Parcours et visualisation de l'espace latent
3. ...

! Les auto-encodeurs sont la base des modèles génératifs que nous connaissons aujourd'hui.
Il existe de nombreuses variations des auto-encodeurs, mais le principe reste le même.

Les couches de sur-échantillonnage

Les couches de sur-échantillonnage sont des couches qui permettent d'augmenter la taille des données. Si vous jouez à des jeux vidéo sur PC, il est probable que vous utilisiez une technologie de sur-échantillonnage basée sur des réseaux de neurones convolutifs sans même le savoir.

DLSS (Deep Learning Super Sampling) est la technologie de sur-échantillonnage développée par NVIDIA.
FSR (FidelityFX Super Resolution) est la technologie de sur-échantillonnage développée par AMD.

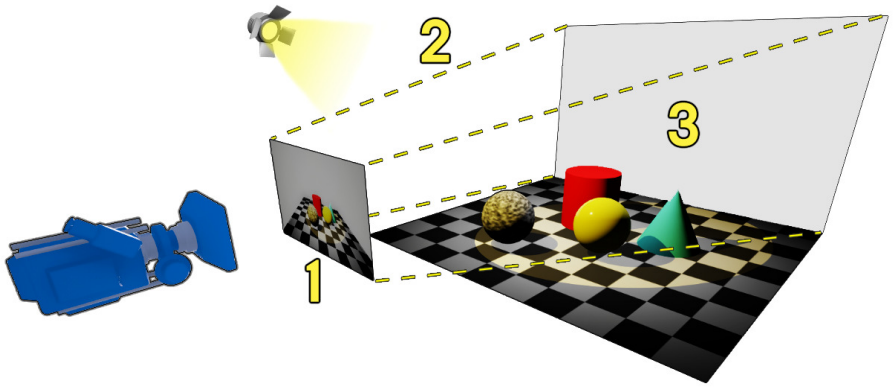

Figure 6: Comment les images sont générées dans les jeux vidéo ou les films d'animation

Les réglages graphiques dans les jeux vidéo modifient la fidélité des images en utilisant des modèles de lumière plus ou moins compliqués. La technique du ray tracing est une technique de rendu qui permet de simuler le comportement de la lumière de manière très réaliste, mais qui est très coûteuse en ressources.

L'idée derrière les technologies de sur-échantillonnage est de générer des images de résolution inférieure et de les sur-échantillonner, avec des réseaux de neurones convolutifs, pour obtenir des images de résolution supérieure.
Cette approche permet de réduire le coût de rendu des images ce qui permet d'augmenter le nombre d'images par seconde dans les jeux vidéo ou de réduire le coût de rendu des films d'animation.

Sur ce lien, vous pouvez voir une vidéo de présentation de la technologie DLSS de NVIDIA. Le gain d'images par seconde (IPS / FPS) est important; de plus, les images sur-échantillonnées semblent plus nettes.