SY32 – Analyse et synthèse d'images
TD 06 : Classification d'images par réseaux de neurones profonds

Dans ce TD, nous allons utiliser la bibliothèque `Keras` pour utiliser des algorithmes de réseaux de neurones profonds.

Installation

Si l'ordinateur dispose d'une carte graphique NVIDIA, il est possible d'installer `keras-gpu` à la place de `keras` (ne pas installer les deux). Dans ce cas, il faut également prendre le soin de mettre à jour les pilotes de la carte graphique.

Installation à partir de l'interface Anaconda

— Lancer l'application `Anaconda Navigator`.

— Aller dans l'onglet `Environments` dans le menu de gauche.

— Dans le menu déroulant à gauche du bouton `Channels` sélectionner `All`.

— Dans la barre de recherche taper : `keras`

— Cocher la case correspondant à `keras` et cliquer sur le bouton `Apply`.

Installation avec `conda`

Depuis un terminal, lancer la commande suivante :

```
conda install -c conda-forge keras
```

— Sous Windows, lancer cette commande depuis l'application `Anaconda Prompt`.

— Sous Mac, l'exécutable `conda` est normalement installé dans le dossier : `/Users/[username]/opt/anaconda3/condabin/`

Installation avec `PyPI`

Depuis un terminal, lancer la commande suivante :

```
pip install keras
```

Classification d'images

On souhaite utiliser le réseau de neurones convolutifs VGG16 du Vision Geometry Group de l'université d'Oxford.

1. Charger le modèle VGG16 à l'aide des commandes ci-dessous :

```
from keras.applications.vgg16 import VGG16
model = VGG16(weights='imagenet')
```

Note : la première fois que le modèle est chargé, il doit être téléchargé sur la machine ($\approx$ 530 Mo).

2. Utiliser la commande `print(model.summary())` pour afficher l'architecture du réseau de neurones.

3. Charger l'image `cougar.jpg` du TD précédent à l'aide des commandes ci-dessous :

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
import numpy as np
img = load_img('crab.jpg', target_size=(224, 224))
img = img_to_array(img)
img = np.expand_dims(img, axis=0)
img = preprocess_input(img)
```

4. La commande `y = model.predict(img)` prédit la classe de l'image `img` sous la forme d'un vecteur de score sur un ensemble 1 000 classes. La fonction `decode_predictions` permet de retrouver les noms des classes ayant les scores les plus élevés.

```
from keras.applications.vgg16 import decode_predictions
label = decode_predictions(y)
```

5. Utiliser VGG16 pour prédire les classes des images `crab.jpg` et `kangaroo.jpg`.

6. La dernière couche du réseau de neurones sert à prédire la classe de l'image. Les couches intermédiaires font parti du processus d'extraction de caractéristiques. Pour récupérer la représentation de l'image sur l'avant dernière couche `fc2`, on peut construire le modèle suivant :

```
from keras.models import Model
model_feat = Model(inputs=model.input,
                   outputs=model.get_layer('fc2').output)
```

7. La commande `x = model_feat.predict(img)` permet alors de récupérer un vecteur de dimension 4096 représentant l'image. Reprendre les images du TD précédent et utiliser le réseau VGG16 pour trouver, pour chacune des images `cougar.jpg`, `crab.jpg` et `kangaroo.jpg`, les images les plus proches parmi les 300 images de la base.

8. Reprendre l'exercice en utilisant le réseau `ResNet50` à la place de `VGG16`.

```
from keras.applications.resnet import ResNet50
model = ResNet50(weights='imagenet')
```