Machine & Deep Learning for Multimedia Retrieval (I-ILIA-014)

TP 4 : Deep Learning pour la recherche et indexation d'images

• Introduction :

L'objectif de ce TP est faire l'extraction de caractéristiques d'images avec les techniques du Deep Learning. Ces caractéristiques seront utilisées pour développer un moteur de recherche. Durant ce TP, trois principales tâches seront donc réalisées :

a. **Indexation d'images :** nous utiliserons pour cela différentes architectures (modèles) de Deep Learning comme VGG16, VGG19, Inception, etc. L'idée est d'extraire les caractéristiques de l'image de la dernière couche du modèle, ou l'avant dernière couche par exemple.

b. **Moteur de recherche :** des mesures de similarités seront développées en vue d'établir le moteur de recherche. En effet, dans le domaine de recherche et indexation multimédia, la validation d'un moteur de recherche se fait en établissant des requêtes test avec 3 images requêtes (au minimum) appartenant à la base d'image.

c. **Evaluation :** consiste à tracer les courbes Rappel/Précision, pour les trois images requêtes choisies.

• Base de données et architectures :

Afin de comparer les résultats de ce TP avec ceux des deux premiers TPs, nous utiliserons la même base d'images Wang. Nous testerons différentes architectures pré-entrainées avec la base ImageNet. Notons que cette base fait partie de la base d'images ImageNet, vous n'êtes donc pas obligés d'entrainer de nouveau ces modèles. On pourra exploiter les caractéristiques déjà extraites pour développer le moteur de recherche. Dans le cas d'une autre base d'images, il faut utiliser la technique du ***fine tuning*** *(transfer learning)* pour entrainer avec la nouvelle base d'images en partant des poids du modèle pré-entrainé avec la base ImageNet par exemple.

Pour ce TP, nous vous proposons d'utiliser Google Colab comme environnement avec sélection du GPU comme ressource de calcul. Ceux qui ont un PC équipé d'une carte graphique GPU peuvent utiliser Anaconda pour travailler en local.

• Exercice 1 : Moteur de recherche utilisant l'architecture VGG16

1. **Téléchargement et décompression de la base d'images wang:** avec la commande wget depuis :

   https://cluster.ig.umons.ac.be/workshop_ia/image.orig.zip

2. Importation des librairies :

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications import vgg16
from keras.applications import vgg19
from keras.applications import resnet50
from keras.applications import inception_v3
from keras.applications import mobilenet
from keras.applications import xception
from matplotlib.pyplot import imread
import matplotlib.pyplot as plt
import numpy as np
import operator
import math
import os
import tensorflow as tf
```

3. Fonction de calcul de la distance euclidienne : (A compléter)

```
def euclidianDistance(l1,l2):
    distance = 0
    length = min(len(l1),len(l2))
    for i in range(length):
        distance += # ??????????????
    return math.sqrt(distance)
```

4. Fonction de calcul des K plus proches voisins (KNN) :

```
def getkVoisins(lfeatures, test, k) :
    ldistances = []
    for i in range(len(lfeatures)):
        dist = euclidianDistance(test[1], lfeatures[i][1])
        ldistances.append((lfeatures[i][0], lfeatures[i][1], dist))
    ldistances.sort(key=operator.itemgetter(2))
    lvoisins = []
    for i in range(k):
        lvoisins.append(ldistances[i])
    return lvoisins
```

5. Initialisation d'un modèle VGG entrainé avec la base ImageNet : (A compléter)

```
model1 = # ??????????????
```

6. Lecture de la base d'images :

```
files = "image.orig" #chemin vers la base d'images
features1 = [] #Stocker les caractérstiques
big_folder="Features_train/" #Dossier pour stocker les caractérstiques
if not os.path.exists(big_folder):
    os.makedirs(big_folder)
folder_model1="Features_train/VGG16/"
if not os.path.exists(folder_model1):
    os.makedirs(folder_model1)
```

7. Génération des caractéristiques de la base d'images :

```
pas =0
for j in os.listdir(files) :
    data = os.path.join(files, j)
    print (data)
    if not data.endswith(".jpg"):
        continue
    file_name = os.path.basename(data)
    # load an image from file
    image = load_img(data, target_size=(224, 224))
    # convert the image pixels to a numpy array
    image = img_to_array(image)
    # reshape data for the model
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
    # prepare the image for the VGG model
    image = preprocess_input(image)
    # predict the probability across all output classes
    feature = model1.predict(image)
    feature = np.array(feature[0])
    np.savetxt(folder_model1+"/"+os.path.splitext(file_name)[0]+".txt",feature)
    features1.append((data,feature))
    print (pas)
    pas = pas+1
with open("Features_train/VGG16.txt", "w") as output:
      output.write(str(features1))
```

8. Test du moteur de recherche avec une image requête :

```
import warnings
warnings.filterwarnings('ignore')
sortie=20
voisins = getkVoisins(features1, features1[0],sortie)

nom_image_plus_proches = []
nom_image_plus_proches_sans = []

for k in range(sortie):
    nom_image_plus_proches.append(voisins[k][0])
plt.figure(figsize=(5, 5))
plt.imshow(imread(features1[0][0]), cmap='gray', interpolation='none')
plt.title("Image requête")
nom_image_requete=os.path.splitext(os.path.basename(features1[0][0]))[0]
print(nom_image_requete)

plt.figure(figsize=(25, 25))
plt.subplots_adjust(hspace=0.2, wspace=0.2)

for j in range(sortie):
    plt.subplot(sortie/4,sortie/5,j+1)
    plt.imshow(imread(nom_image_plus_proches[j]), cmap='gray',
interpolation='none')
nom_image_plus_proches_sans.append(os.path.splitext(os.path.basename(nom_image_
plus_proches[j]))[0])
    title = "Image proche n°"+str(j)
    plt.title(title)
```

9. Calcul des courbes de rappel et précision :

```
      text_file = open("VGG_RP.txt", "w")
      rappel_precision=[]
      rp = []
      position1=int(nom_image_requete)//100
```

```
for j in range(sortie):
  position2=int(nom_image_plus_proches_sans[j])//100
  if position1==position2:
    rappel_precision.append("pertinant")
  else:
    rappel_precision.append("non pertinant")

for i in range(sortie):
  j=i
  val=0
  while j>=0:
    if rappel_precision[j]=="pertinant":
      val+=1
    j-=1
  rp.append(str((val/(i+1))*100)+" "+str((val/sortie)*100))

with open("VGG_RP.txt", 'w') as s:
  for a in rp:
    s.write(str(a) + '\n')
```

10. Tracé des courbes de rappel et précision :

```
import csv
x = []
y = []
fichier = "VGG_RP.txt"
with open(fichier) as csvfile:
    plots = csv.reader(csvfile, delimiter=' ')
    for row in plots:
        x.append(float(row[0]))
        y.append(float(row[1]))
        fig = plt.figure()
line, =plt.plot(y,x,'C1', label="VGG16" )
plt.xlabel('Rappel')
plt.ylabel('Précison')
plt.title("R/P")
plt.legend()
```

• Exercice 2 : amélioration du moteur de recherche

1. Calculer et tracez les courbes de R/P pour d'autres images : 140, 450, 610, 277 et 805
2. Refaire le même processus en utilisant les caractéristiques de l'avant dernière couche (à la place des caractéristiques de la dernière couche utilisées dans l'exercice 0). Notons que l'accès aux caractéristiques de la dernière et avant dernières couche se fait, respectivement, avec :

```
model0=vgg16.VGG16(weights='imagenet', include_top=True,pooling='avg')
model1 = Model(inputs=model0.input, outputs=model0.layers[-1].output)
```

3. Utiliser d'autres modèles pour l'extraction de caractéristiques (et la recherche) : VGG19, Inception, Xception, ResNet, etc.
4. Pour chaque modèle, tracez les courbes R/P