Jeffrey Bataille IMR3

Rendu de projet d’analyse et de traitement d’images

Introduction

L’objectif de ce projet était de mettre en pratique les cours vus dans le module d’analyse et de traitement d’images.
Ce projet consiste à réaliser une classification des images par catégorie (exemple: chien et chat). Dans une première partie nous verrons la classification par attributs des images puis dans une seconde partie la classification par réseau de neurones.

Introduction **1**

Prérequis pour le projet **2**
Environnement de développement 2
Base de données 2

Partie 1 : classification d’images par attributs **3**
Jeu de données 3
Projet initial 3
Adaptation du projet 4
Résultat de classification par la couleur 4

Partie 2 : classification d’images par réseau de neurones **5**
Jeu de données 5

Prérequis pour le projet

Environnement de développement

Le projet utilise Python3 en version 3.7.4.
L'installation des dépendances est réalisée par Anaconda, elles sont les suivantes :

- Numpy
- Scipy
- Scikit-learn
- Scikit-image
- Matplotlib
- Tensorflow
- Keras

L'IDE utilisé pour les développement est Spyder, qui a été conseillé par les enseignants pour réaliser le projet.

Base de données

Le jeu d'images utilisé provient de la base CorelDB, qui contient 10 800 images classées en catégories bien distinctes (avion, bateau, drapeau, …). Chaque catégorie contient plus de 100 images distinctes de petite taille (environ 100x100 px). La petite taille permet de réaliser l'analyse sur un ordinateur personnel dans un temps assez court.
Pour chacune des partie j'utilise 3 catégories d'images :

- fitness
- art_1
- obj_aviation

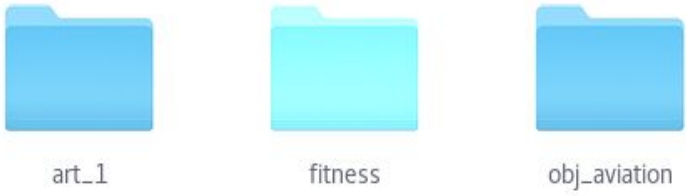
Partie 1 : classification d'images par attributs

Jeu de données

Les images issues des trois catégories sélectionnés sont séparées dans deux répertoires :

- train (60% des images)
- test (40% des images)

Pour séparer ses images j'ai réalisé un script qui copie 60% de chaque catégorie dans le dossier **train** et 40% dans le dossier **test**. Au sein du dossier **train** les images sont rangées dans un sous-dossier du nom de la catégorie.



Dossier Validation

Contrairement au dossier **train**, les images dans le dossier **test** ne sont pas rangées dans un sous-dossier.

Projet initial

Le code initial provient du projet CBIR disponible sur Github.
Il permet de comparer une image avec un jeu d'images catégories pour fournir la catégorie de cette image.
Le principe est d'attribuer un score pour chaque catégorie et de considérer la classe ayant le score le plus élevé comme la bonne classe.

Plusieurs fonctions sont disponibles pour calculer ce score sur des méthodes différentes :

- Color
- Daisy
- Edge
- Gabor
- HOG
- ResNet
- VGG

Adaptation du projet

L’objectif du était d’adapter le projet pour définir la catégorie de chaque image du dossier **test** en se basant sur les images disponibles dans le dossier **train**. J’ai réalisé cela en utilisant la fonction de classification par couleur.
Lorsque j’exécute le fichier **Color.py** les images provenant du dossier **test** sont catégorisées avec l’algorithme de couleurs pour écrire dans un dossier **res** les images avec leur catégorie.

Résultat de classification par la couleur

Les résultats obtenus par la classification de la couleur est moyenne, les résultats sont très bon sur la catégorie fitness car le fond de toutes ses images sont entre le blanc et le gris d’où le résultat élevés.
Pour les deux autres catégories les résultats sont nettement moins bon avec même des faux positif.

Catégorie	Vrai positif	Faux positif
fitness	80	0
art_1	38	8
obj_aviation	32	2

Partie 2 : classification d'images par réseau de neurones

Jeu de données

Les images issues des trois catégories sélectionnés sont séparées dans deux répertoires :

- train (50% des images)
- test (% des images)
- validation (20% des images)

Pour séparer ses images j'ai réalisé un script qui copie 50% de chaque catégorie dans le dossier **train**, 30% dans le dossier **test** et 20% dans le dossier validation.

Pour augmenter artificiellement le nombre d'images par catégorie il est possible de réaliser des duplications avec transformation (rotation, translation, zoom). Cependant cela augmente le nombre d'images et donc le temps de calcul pour le réseau de neurone.

Projet initial

Le code provient d'un blog pour réaliser une classification entre deux types d'images (chien et chat). Pour plus de documentation sur le code initial veuillez lire le blog.

Adaptation du code

Le projet initial permettait de faire une classification qu'entre deux catégories. Pour utiliser les trois catégories choisies au début il a fallu adapter le code.
Par défaut la classification était binaire, j'ai du remplacé par categorical

```
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

Le réseau de neurone est composé des 6 couches :

```
model = Sequential()
model.add(Conv2D(16, (5, 5), input_shape=input_shape, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))  # groupe 2x2 pixel

model.add(Conv2D(32, (5, 5)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (5, 5)))
```

```
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (5, 5)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())  # mise à plat des coeffs des neurones
model.add(Dense(128))  # dense = fully connected
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
```

Résultat

Pour la partie apprentissage les résultats sont les suivant:

```
Epoch 20/20
 - 8s - loss: 0.2545 - accuracy: 0.9185 - val_loss: 0.0720 - val_accuracy:
0.9216
```

Malheureusement je n'ai pas réussi à faire fonctionner la partie test.

Conclusion

La classification par attributs à montrer qu’elle n’était pas optimale, les résultats sont très basiques. La classification par réseau de neurones pourrait donner des bon résultats cependant je n’ai pas réussi à réaliser la partie test.

Ce projet a été difficile à réaliser. Le traitement d’image était complexe et ajouter du réseau de neurones était encore plus complexe. Le fait d’utiliser des codes “préfaits” permettait d’avoir certaines bases pour le projet heureusement.