TP2 : DÉCOUVERTE DES RÉSEAUX DE NEURONES CONVOLUTIF (CNN)

Les réseaux de neurones convolutif (CNN) sont un type de réseau de neurone très puissant et très utilisé pour l'analyse d'images. Dans ce TP, nous allons travailler sur la base CIFAR10. Cette base de données est composée de 10 classes d'images (dimensions 32x32), avec 6000 samples par classe. Dans ce TP, nous n'allons travailler qu'avec deux classes, et pas tous les samples disponibles, pour rendre le problème plus facile. Il consistera donc en la construction d'un CNN capable de différencier des images de chien et de chat.

COMPTE-RENDU

Un compte rendu est à préparer pour le Vendredi 20 Novembre. Ce compte-rendu doit contenir la réponse aux questions posées au fur et à mesure du TP. N'hésitez pas à commenter et à ajouter des captures d'écran si nécessaire. La présentation sera prise en compte dans la notation.

QUESTIONS

1. **Complétez la fonction build_CNN(), pour construire un réseau de neurones avec la structure indiquée ci-dessous. A quoi sert le dropout ?**

N'oubliez pas de vous documenter sur les concepts que vous ne connaissez pas.

Couches
Conv2D, kernel 3x3, depth=32 => relu => MaxPooling2D, size=2x2, => Dropout, rate=0.1
Conv2D, kernel 3x3, depth=64 => relu => MaxPooling2D, size=2x2, => Dropout, rate=0.1
Conv2D, kernel 3x3, depth=64 => relu => MaxPooling2D, size=2x2, => Dropout, rate=0.1
Flatten => Dense, width=64 => relu, => Dropout, rate=0.3
Dense, width=output_dim => identity

Compilez le modèle avec categorical_crossentropy en loss, rmsprop en optimizer et accuracy en metric.

2. **Complétez les attributs demandés dans le code pour utiliser un générateur, afin d'entrainer le réseau de neurones. L'entrainez sur 20 itérations. Notez les résultats. Que pouvez-vous en dire ?**

Les générateurs sont utilisés pour éviter de devoir charger toute la base de donnés en mémoire. Keras propose une implémentation de générateur pour les images, permettant notamment de prétraiter les images très facilement.

Example de ImageDataGenerator :
https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator#flow_from_dataframe


Fonctions utiles : ImageDataGenerator(), flow_from_directory(), fit(), evaluate()

3. **Utilisez ImageDataGenerator pour faire de l'augmentation de données. Choisissez quel(s) type(s) de transformations vous semblent pertinent et expliquez pourquoi. Est-ce que ça améliore vos résultats ?**

4. **Préchargez le modèle entrainé par mes soins, sur les 10 classes (« pretrained_model.h5 »). Retirez la, ou les deux dernières couches. Pourquoi devez-vous faire ça ? Rajoutez ensuite les une (ou deux) couches adéquate(s), et fine-tunez le modèle. Réentrainez sur 20 epoch. Que constatez-vous ?**

Préentrainer un modèle est une technique souvent utilisée en deep learning. Cela permet de commencer l'entrainement avec un réseau qui marche à peu près, au lieu d'un réseau initialisé aléatoirement. Cela permet d'accélérer considérablement l'entrainement et éventuellement d'améliorer les résultats finaux.

Fonctions utiles : load(), pop()

5. **Utiliser TensorBoard pour monitorer l'entrainement. Comparer avec et sans prétraining, et comparer avec et sans dropout.**

TensorBoard est un outil propose par Tensorflow permettant, entre autre, de visualiser les modèles et de monitorer l'entrainement.

Tensorboard : https://www.tensorflow.org/tensorboard/get_started