7

Utilisation des modèles de deep learning

Dans ce chapitre, nous allons présenter les principales fonctions d'OpenCV disponibles dans le module dnn permettant d'utiliser des modèles d'apprentissage profond ou *deep learning*.

Le deep learning est une méthode qui a surpassé, dans de nombreux domaines, les méthodes traditionnelles utilisées dans le traitement de l'image, le traitement de la parole, la classification des données, le diagnostique médical, la traduction automatique et bien d'autres domaines encore. Avant de décrire l'utilisation du deep learning dans OpenCV, nous allons présenter ses principes.

Apprentissage profond (ou deep learning)

Le deep learning permet de résoudre un problème complexe en utilisant des données d'entraînement (voir chapitre Apprentissage automatique – Machine Learning). Le processus appliqué aux données pour obtenir le résultat souhaité est décrit dans une architecture. Cette architecture contient un ensemble d'opérations ordonnées décrit dans un graphe. L'étape permettant de calculer le résultat en utilisant l'architecture est nommée l'*inférence*. L'architecture peut contenir des filtres de convolutions, des extractions d'extremums, des opérations linéaires ou non linéaires, des réseaux de neurones. Cette liste est non exhaustive car l'imagination des spécialistes du deep learning est sans limite. L'architecture, constituée donc d'un ensemble d'opérations ordonnées, est complétée par l'ensemble des valeurs permettant l'étape d'inférence. Ces valeurs définissent par exemple les valeurs des filtres de convolutions, les limites des régions où les extremums sont à extraire, les constantes des opérations non linéaires ou bien les poids des nœuds des réseaux de neurones. Ces valeurs sont trouvées après une phase d'optimisation utilisant un jeu de données d'entraînement. L'étape permettant de trouver ces valeurs est nommée l'*apprentissage*.

De nombreux modèles sont disponibles sur le web. Ils sont issus de travaux de recherche. Les auteurs des publications donnent très souvent les liens vers leurs modèles, architecture et poids. Une recherche sur Google à partir du mot clé "deep

learning" et du domaine qui vous intéresse (comme "leaf", "gender", "cats", "face") permet bien souvent de trouver ces modèles.

Aucune fonction d'OpenCV ne permet d'optimiser ces valeurs pour une architecture de deep learning. L'optimisation et la sauvegarde du modèle (l'architecture et les données associées) doivent être faites à l'aide de librairies spécialisées en deep learning comme Caffe, Darknet, Keras, TensorFlow, Torch, etc. Les fonctions d'OpenCV permettent de charger le modèle et de calculer un résultat à partir de données (inférence).

Dans ce chapitre, nous allons présenter les fonctions d'OpenCV permettant d'utiliser les principales architectures de deep learning. Les modèles de deep learning sont des fichiers. La taille de ces fichiers peut être de 100 mégaoctets. Il faudra donc les télécharger et préciser leur localisation dans le code source avant de pouvoir exécuter les exemples qui suivent. Les liens de téléchargement sont fournis dans chaque exemple.

Les modèles sont souvent protégés par une licence open-source, mais ce n'est pas toujours le cas.

7.1. Principales fonctions du module *dnn*

Avant d'examiner quelques exemples d'usage de modèles de deep learning dans OpenCV, on va présenter les principales fonctions du module.

Pour utiliser un modèle de deep learning, il faut en premier lieu lire le ou les fichiers définissant le réseau, préparer les données et les associer à l'entrée du réseau et enfin calculer la sortie du réseau. Pour lire le réseau, il faut utiliser la fonction `cv.dnn.readNet` ; pour préparer les données, la fonction `cv.dnn.blobFromImage` ; pour fixer les entrées d'un réseau , la méthode `setInput` et enfin pour calculer (étape d'inférence) le résultat il faut utiliser la méthode `forward`.

Lecture du réseau

La fonction `cv.dnn.readNet` permet de lire le réseau. Elle utilise deux paramètres : le premier est le chemin complet du fichier des poids, le second le chemin complet du fichier contenant l'architecture. Il existe un paramètre supplémentaire que nous n'utiliserons pas. Il sert à préciser sur quelle librairie a été entraîné le modèle lorsqu'il y a ambiguïté sur le nom du modèle. Dans nos exemples, cela ne se produira pas.

Le résultat de la fonction est un objet de la classe `Net` représentant le réseau, architecture et poids, prêt à être utilisé pour la prédiction.

Préparation des données

La fonction `cv.dnn.blobFromImage` permet de convertir une image (tableau NumPy) en blob. Un blob est une manière d'organiser les données en mémoire et contient bien souvent un tenseur, c'est-à-dire un tableau multidimensionnel. Le blob est le format d'entrée des données pour le modèle de deep learning.

`cv.dnn.blobFromImage` utilise sept paramètres :

- le tableau NumPy contenant l'image ;
- le paramètre nommé `scaleFactor` : un réel ;
- le paramètre nommé `size` pour définir la taille de l'image dans le blob (par défaut, la taille de l'image est inchangée) ;
- le paramètre nommé `mean` : un tuple (`moyenne_p0`, `moyenne_vert`, `moyenne_p2`) qui est soustrait à l'image. `moyenne_p0` et `moyenne_p2` sont les moyennes du plan rouge et bleu si `swapRB` est `True` (voir point suivant). Cette valeur est nulle par défaut ;
- le paramètre nommé `swapRB`, qui est égal à `True` lorsque les plans 0 et 2 doivent être inversés (par défaut, les plans ne sont pas inversés) ;
- le paramètre nommé `crop`, qui est égal à `True` lorsque l'image doit être coupée (par défaut, l'image n'est pas coupée) ;
- le paramètre nommé `ddepth` : la profondeur du blob de sortie, réel (`np.float32`) ou entier (`np.uint8`) ; par défaut le blob est du type `np.float32`.

Le résultat de la fonction est un tableau NumPy représentant le blob.

Exemple 7.1 : *Code source dans le dossier `py_dnn/blob`*

Pour bien illustrer l'usage de chaque paramètre de la fonction `cv.dnn.blobFromImage`, nous allons appeler celle-ci avec différents paramètres. Le résultat de chaque appel est montré sur la Figure 7.1. Les appels successifs sont donnés dans le code suivant :

```
import numpy as np
import cv2 as cv

img = cv.imread('g:/lib/livreOpencv/Figure/OCV_Haribo.png')
cv.imshow("Original", img) ❶
cv.waitKey(10)
b1 = cv.dnn.blobFromImage(img, 1.0, (350, 100), swapRB=True,
 crop=False) ❷
imgb = cv.dnn.imagesFromBlob(b1)
```