En raison de l'évolution très rapide des framework, la dernière version de Python que vous devriez utiliser dans ce cours est Python 3.10.

Écrivez une IA pour identifier le panneau de signalisation qui apparaît sur une photographie.

```
$ python traffic.py gtsrb
Epoch 1/10
500/500 [==============================] - 5s 9ms/step - loss: 3.7139 - accuracy: 0.1545
Epoch 2/10
500/500 [==============================] - 6s 11ms/step - loss: 2.0086 - accuracy: 0.4082
Epoch 3/10
500/500 [==============================] - 6s 12ms/step - loss: 1.3055 - accuracy: 0.5917
Epoch 4/10
500/500 [==============================] - 5s 11ms/step - loss: 0.9181 - accuracy: 0.7171
Epoch 5/10
500/500 [==============================] - 7s 13ms/step - loss: 0.6560 - accuracy: 0.7974
Epoch 6/10
500/500 [==============================] - 9s 18ms/step - loss: 0.5078 - accuracy: 0.8470
Epoch 7/10
500/500 [==============================] - 9s 18ms/step - loss: 0.4216 - accuracy: 0.8754
Epoch 8/10
500/500 [==============================] - 10s 20ms/step - loss: 0.3526 - accuracy: 0.8946
Epoch 9/10
500/500 [==============================] - 10s 21ms/step - loss: 0.3016 - accuracy: 0.9086
Epoch 10/10
500/500 [==============================] - 10s 20ms/step - loss: 0.2497 - accuracy: 0.9256
333/333 - 5s - loss: 0.1616 - accuracy: 0.9535
```

Contexte

Alors que les recherches se poursuivent sur le développement des voitures autonomes, l'un des principaux défis est la vision par ordinateur, qui permet à ces voitures de comprendre leur environnement à partir d'images numériques. En particulier, cela implique la capacité de reconnaître et de distinguer les panneaux de signalisation - panneaux d'arrêt, panneaux de limitation de vitesse, panneaux de céder le passage, etc.

Dans ce projet, vous utiliserez TensorFlow pour construire un réseau neuronal permettant de classer les panneaux routiers à partir d'une image de ces panneaux. Pour ce faire, vous aurez besoin d'un ensemble de données étiquetées : une collection d'images qui ont déjà été classées en fonction du panneau routier qu'elles représentent.

Il existe plusieurs ensembles de données de ce type, mais pour ce projet, nous utiliserons l'ensemble de données German Traffic Sign Recognition Benchmark (GTSRB), qui contient des milliers d'images de 43 types de panneaux routiers différents.

Pour commencer

Téléchargez le code du projet à partir d'i campus
Téléchargez le jeu de données pour ce projet et décompressez-le. Déplacez le répertoire gtsrb résultant dans votre répertoire traffic.
Dans le répertoire traffic, exécutez pip3 install -r requirements.txt pour installer les dépendances de ce projet : opencv-python pour le traitement d'images, scikit-learn pour les fonctions liées au ML, et tensorflow pour les réseaux neuronaux.

Éléments pour la Compréhension

Tout d'abord, jetez un coup d'œil à l'ensemble de données en ouvrant le répertoire gtsrb. Vous remarquerez que cet ensemble de données comporte 43 sous-répertoires, numérotés de 0 à 42. Chaque sous-répertoire numéroté représente une catégorie différente (un type de panneau de signalisation différent). Dans le répertoire de chaque panneau de signalisation se trouve une collection d'images de ce type de panneau.

Jetons ensuite un coup d'œil au fichier traffic.py. Dans la fonction principale, nous acceptons comme arguments de ligne de commande un répertoire contenant les données et (optionnellement) un nom de fichier dans lequel enregistrer le modèle entraîné. Les données et les étiquettes correspondantes sont ensuite chargées à partir du répertoire de données (via la fonction load_data) et divisées en ensembles d'entraînement et de test. Ensuite, la fonction get_model est appelée pour obtenir un réseau neuronal compilé qui est ensuite ajusté sur les données d'entraînement. Le modèle est ensuite évalué sur les données de test. Enfin, si un nom de fichier de modèle a été fourni, le modèle formé est enregistré sur le disque.

Les fonctions load_data et get_model sont laissées à votre appréciation.

Spécification

Un outil automatisé aide le personnel à faire respecter les contraintes énoncées dans la spécification ci-dessous. Votre soumission échouera si l'une de ces contraintes n'est pas traitée correctement, si vous importez des modules autres que ceux explicitement autorisés, ou si vous modifiez des fonctions autrement que de manière autorisée.

Complétez l'implémentation de load_data et get_model dans traffic.py.

- La fonction load_data doit accepter comme argument data_dir, représentant le chemin vers un répertoire où les données sont stockées, et renvoyer des tableaux d'images et des étiquettes pour chaque image de l'ensemble de données.
    - Vous pouvez supposer que data_dir contiendra un répertoire nommé d'après chaque catégorie, numérotée de 0 à NUM_CATEGORIES - 1. Chaque répertoire de catégorie contiendra un certain nombre de fichiers d'images.

    - Utilisez le module OpenCV-Python (cv2) pour lire chaque image sous la forme d'un tableau multidimensionnel numpy.ndarray. Pour transmettre ces images à un réseau neuronal, les images devront avoir la même taille. Veillez donc à redimensionner chaque image pour qu'elle ait la largeur IMG_WIDTH et la hauteur IMG_HEIGHT.
    - La fonction doit retourner un tuple (images, labels).
      images doit être une liste de toutes les images de l'ensemble de données, où chaque image est représentée par un tableau numpy.ndarray de la taille appropriée.
      labels doit être une liste d'entiers, représentant le numéro de catégorie pour chacune des images correspondantes dans la liste d'images.
    - Votre fonction doit être indépendante de la plateforme : c'est-à-dire qu'elle doit fonctionner quel que soit le système d'exploitation. Notez que sous macOS, le caractère / est utilisé pour séparer les composants du chemin, alors que le caractère \ est utilisé sous Windows. Utilisez os.sep et os.path.join si nécessaire au lieu d'utiliser le caractère séparateur spécifique à votre plateforme.
- La fonction get_model doit renvoyer un modèle de réseau neuronal compilé.
    - Vous pouvez supposer que l'entrée du réseau neuronal aura la forme (IMG_WIDTH, IMG_HEIGHT, 3) (c'est-à-dire un tableau représentant une image de largeur IMG_WIDTH, de hauteur IMG_HEIGHT et de 3 valeurs pour chaque pixel de rouge, de vert et de bleu).
    - La couche de sortie du réseau neuronal doit comporter des unités NUM_CATEGORIES, une pour chacune des catégories de panneaux de signalisation.
    - Le nombre de couches et les types de couches que vous incluez entre les couches sont laissés à votre discrétion. Vous pouvez expérimenter avec
        - différents nombres de couches convolutives et de couches de mise en commun
        - différents nombres et tailles de filtres pour les couches convolutives
        - différentes tailles de pool pour les couches de mise en commun
        - différents nombres et tailles de couches cachées
        - l'abandon

Qu'avez-vous essayé ? Qu'est-ce qui a bien fonctionné ? Qu'est-ce qui n'a pas bien fonctionné ? Qu'avez-vous remarqué ?

En fin de compte, une grande partie de ce projet consiste à explorer la documentation et à étudier différentes options dans cv2 et tensorflow et à voir les résultats que vous obtenez lorsque vous les essayez !

Vous ne devez rien modifier d'autre dans traffic.py que les fonctions que la spécification vous demande d'implémenter, bien que vous puissiez écrire des fonctions supplémentaires et/ou importer d'autres modules de la bibliothèque standard de Python. Vous pouvez également importer numpy ou pandas, si vous les connaissez, mais vous ne devez pas utiliser d'autres modules Python tiers. Vous pouvez modifier les variables globales définies au début du fichier pour tester votre programme avec d'autres valeurs.

Conseils

Consultez la présentation officielle de Tensorflow Keras pour obtenir des indications sur la syntaxe de construction des couches de réseaux neuronaux. Le code source de la conférence peut également vous être utile. https://www.tensorflow.org/guide/keras?hl=fr

La documentation OpenCV-Python peut s'avérer utile pour lire les images en tant que tableaux et les redimensionner.
https://docs.opencv.org/4.5.2/d2/d96/tutorial_py_table_of_contents_imgproc.html

Une fois que vous avez redimensionné une image img, vous pouvez vérifier ses dimensions en imprimant la valeur de img.shape. Si vous avez redimensionné l'image correctement, sa forme devrait être (30, 30, 3) (en supposant que IMG_WIDTH et IMG_HEIGHT soient tous deux égaux à 30).

Si vous souhaitez vous entraîner avec un ensemble de données plus petit, vous pouvez utiliser l'ensemble gtrsb-small modifié qui ne contient que 3 types de panneaux routiers différents au lieu de 43.