2024 l'odyssée de l'IA

détection d'objets

Brian Tristam Williams (Elektor)

En explorant la détection d'objets sur notre Raspberry Pi *headless*, nous détaillons la configuration de la caméra et le réglage de TensorFlow Lite pour les applications en temps réel.

Dans dans l'épisode précédent [1], j'ai installé Tensorflow Lite sur le Raspberry Pi, qui exécute une version de Raspberry Pi OS sans interface utilisateur graphique (GUI), aussi connue sous le nom d'installation « headless » (sans tête). Cette fois, je vais détailler comment j'ai réussi à faire fonctionner la caméra et à essayer de détecter des objets.

Faire fonctionner la caméra

Je voulais d'abord installer quelques paquets de caméras pour tester la caméra. J'utilise une version ***headless*** du Raspberry Pi OS, donc je ne peux pas simplement lancer une application de caméra. Cette fois, je n'utilise même pas de clavier, car j'utilise l'application PuTTY [2] sous Windows pour se connecter en SSH au Raspberry Pi (**figure 1**). Il suffit d'entrer le nom d'hôte de l'appareil - dans mon cas *raspberrypi* - dans la fenêtre de PuTTY et de cliquer sur *Open*. Vous pouvez également utiliser une adresse IP.

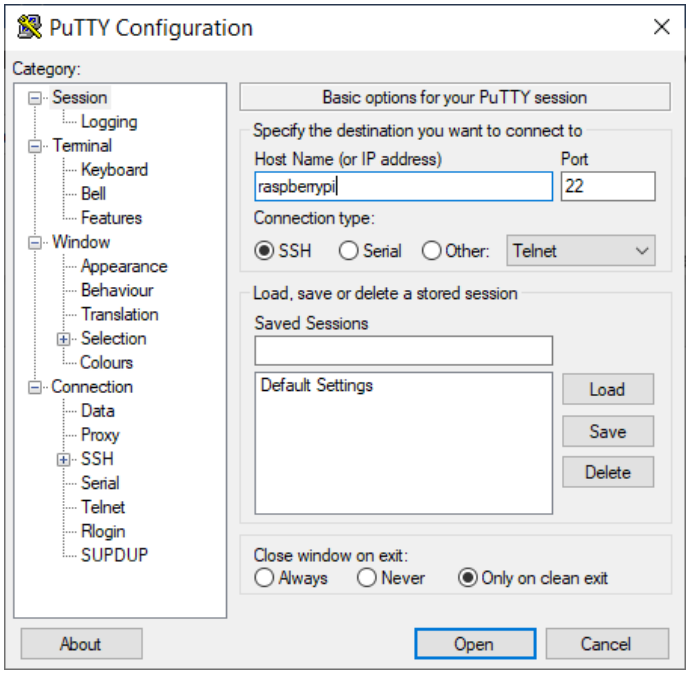

Figure 1. Interface utilisateur PuTTY.

Une fois la connexion établie, vous devez saisir votre nom d'utilisateur et votre mot de passe pour le Raspberry Pi. Il est possible d'enregistrer ces informations d'identification, et c'est recommandé, car vous aurez à le faire souvent.

Auparavant, nous devions exécuter `sudo raspi-config` et activer *Camera* sous *Interface Options*, mais avec les récentes distributions du Raspberry Pi OS, la caméra est détectée automatiquement.

Pour tester rapidement la caméra sur le système ***headless***, j'ai installé quelques paquets :

```
sudo apt-get install -y libraspberrypi-bin
sudo apt-get install libcamera-apps
```

Après s'être reconnecté, j'ai testé la caméra en transmettant sa sortie en direct à mon moniteur, en utilisant

```
libcamera-hello
```

Cela a fonctionné, et c'est formidable que je puisse voir la sortie en direct de la caméra embarquée sur le moniteur (**figure 2**), mais c'est là que se termine le succès du matériel, et nous y reviendrons plus tard.

Comme j'ai rebooté, j'ai dû accéder à nouveau au sous-répertoire *tflite1* et lancer l'environnement virtuel. Après de multiples débogages, essais et erreurs, j'ai fini par créer un nouvel environnement virtuel appelé `new-tflite-env`, et j'y ai accédé avec

```
source new-tflite-env/bin/activate
```

Comme il s'agit d'une installation *headless*, j'ai dû installer *opencv-python-headless* dans cet environnement, en saisissant

```
pip3 install opencv-python-headless
```

Ensuite, j'ai dû me passer des scripts recommandés dans le guide que je consultais sur GitHub, car les méthodes Python utilisées étaient destinées à un système d'exploitation à interface graphique, ce qui n'est pas le cas de notre système. Nous avons besoin du paquet *TensorFlow Lite*, que nous avons déjà installé, ainsi que de *numpy*. À ce stade, j'avais déjà installé une longue liste de paquets Python, prêts à prendre

Figure 2. `libcamera-hello` *nous indique que notre caméra fonctionne.*

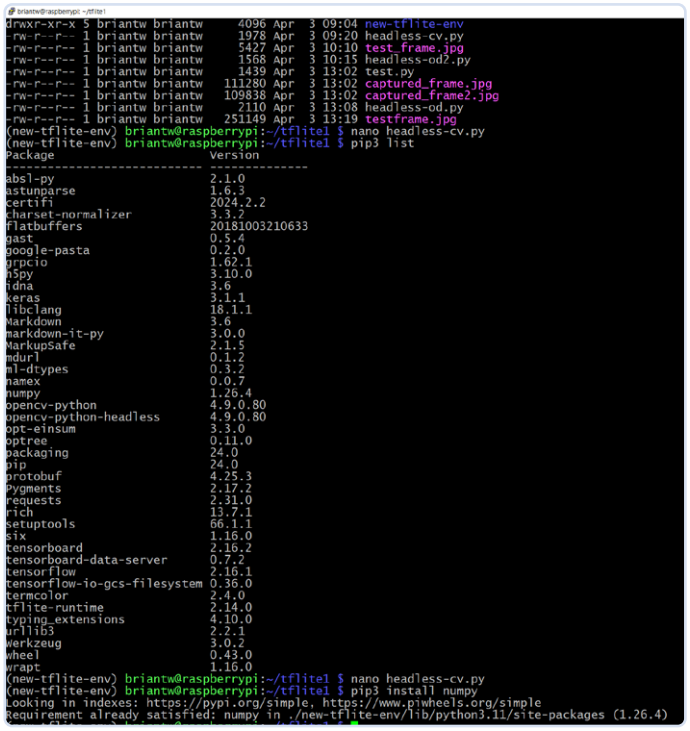

Figure 3. Paquets installés dans mon environnement virtuel.

en charge le travail. Pour voir quels sont les paquets installés dans votre environnement, saisissez *pip3 list* dans l'invite de commande. Ma liste ressemble à celle de la **figure 3**. Si vous n'avez pas encore installé *numpy*, faites-le avec la même démarche que pour *opencv-python-headless* ci-dessus.

Script de test

Passons maintenant au script de test (**listage 1**). Notre petit script Python utilise l'interpréteur TensorFlow Lite pour charger le modèle et effectuer la détection d'objets sur les images entrées. Il capture en continu des images vidéo depuis une caméra, traite chaque image pour qu'elle corresponde aux exigences d'entrée du modèle TensorFlow Lite pré-entraîné (*detect.tflite*), puis utilise le modèle pour détecter des objets dans ces images. Les objets détectés dont le score de confiance est supérieur à 0,5 sont affichés sur la console avec leur étiquette, leur score de confiance et les coordonnées de leur boîte englobante. Le script utilise OpenCV pour la capture vidéo et le prétraitement des images, et l'API Python de TensorFlow Lite pour l'inférence du modèle. Principaux éléments du script :

- Initialisation de l'interprète TensorFlow Lite : Charge le modèle *detect.tflite* et le prépare pour l'inférence.
- Configuration de la capture vidéo : Initialise la capture vidéo de la webcam en utilisant OpenCV (`cv2.VideoCapture(0)`).
- Prétraitement des images : Convertit les images capturées en RGB, les redimensionne pour correspondre aux dimensions d'entrée du modèle, et les encapsule dans un format de lot accepté par TensorFlow Lite.
- Détection d'objets : Introduit les images prétraitées dans le modèle TensorFlow Lite et récupère les résultats de la détection.
- Traitement des résultats : Itère sur les résultats de détection, filtre par un seuil de confiance de 0,5, et imprime l'étiquette, le score de confiance et la boîte englobante pour chaque objet détecté.

Le script fait appel à un traitement en temps réel adapté aux applications nécessitant un retour immédiat de détection d'objets à partir du flux vidéo, affichant les objets identifiés et leurs emplacements dans l'image, avec des boîtes englobantes. Le nombre de détections possibles par seconde dépend de la puissance de votre Raspberry Pi, j'attendrais donc plus d'un Raspberry Pi 5 par rapport au 4 que j'utilise.

Pour exécuter le script avec l'invite de commande, il suffit de saisir

```
python3 objdet.py
```

Défis

En théorie, tout est parfait, mais lorsque j'ai exécuté ce script, il ne détectait rien. Finalement, j'ai essayé de réduire le seuil de confiance à 0,2 au lieu de 0,5, et tout ce qu'il détectait était « ??? » avec des niveaux de confiance variables. Pour déboguer, j'ai ajouté une ligne pour enregistrer l'image capturée dans un fichier :

```
cv2.imwrite('test_frame.jpg'), frame
```

Je l'ai placé juste après l'instruction `cap.read()`. En exécutant et en vérifiant la sortie, j'ai découvert qu'il sauvegardait un fichier de 5 427 octets qui ressemblait à ceci, d'après la commande *file* :

```
test_frame.jpg: JPEG image data, JFIF standard 1.01,
aspect ratio, density 1x1, segment length 16, baseline,
precision 8, 640x480, components 3
```

Tout semble correspondre à ce que l'on pourrait attendre, même s'il s'agit d'un fichier peu volumineux, et je n'ai donc rien soupçonné d'anormal jusqu'à ce que je jette un coup d'œil au fichier sur mon PC. J'ai utilisé SCP dans l'invite de commande Windows pour transférer le fichier :

```
C:\Users\Brian>scp briantw@raspberrypi:/home/briantw/
tflite1/test_frame.jpg .
```

Après avoir saisi votre mot de passe, il récupère le fichier du Raspberry Pi et l'écrit dans le répertoire courant (`.`) de Windows - dans mon cas, le répertoire par défaut de l'utilisateur. Lorsque j'ai ouvert le fichier, à ma grande surprise, il n'avait capturé qu'une image noire. J'ai essayé d'y remédier avec toutes sortes de moyens, des recherches Google à l'utilisation de ChatGPT, sans parvenir à faire en sorte que la caméra fournisse à mon script Python autre chose qu'une image noire.

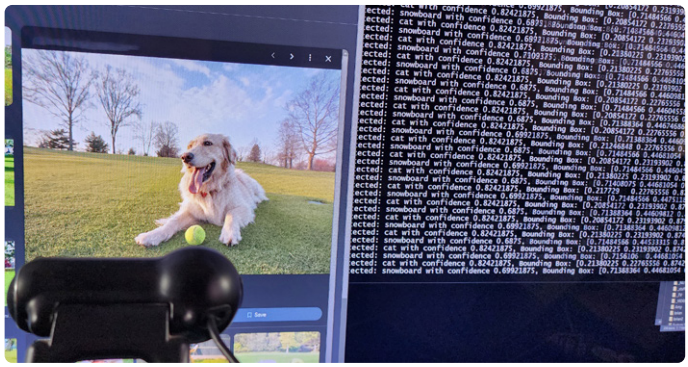

Figure 4. Un chat avec un snowboard.

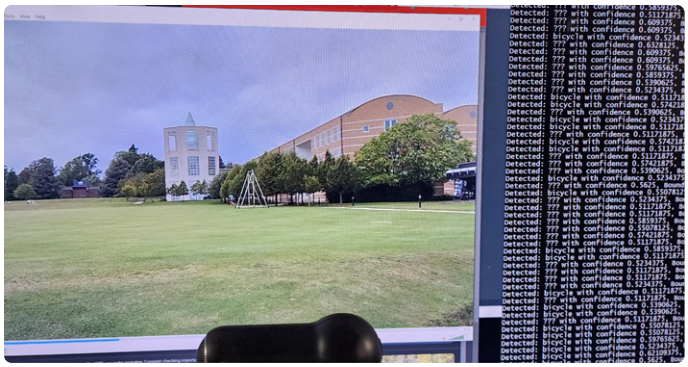

Figure 5. Pas une seule bicyclette en vue.

J'ai même essayé d'activer des délais de quelques secondes dans le code pour donner à la caméra le temps de « tourner » et de s'initialiser, en vain. J'ai également fait un `update` et puis un `upgrade`, et j'ai même mis à jour le micrologiciel en utilisant `sudo rpi-update`, puis j'ai effectué un `sudo reboot`. Curieusement, la caméra fonctionne, comme `libcamera-hello` continue de le prouver, c'est donc un mystère logiciel pour moi.
Finalement, j'ai pris une webcam Logitech et je l'ai branchée sur le Raspberry Pi. Après avoir redémarré le système et réintégré l'environnement virtuel, j'ai relancé le script, qui a fonctionné. Dans le script, vous pouvez sélectionner la caméra utilisée, et, comme vous le voyez dans le listage 1, il est écrit :

```
# Initialize video capture from the camera
cap = cv2.VideoCapture(0)
```

Cela indique à `cv2.VideoCapture()` d'utiliser la caméra 0. Une fois que j'ai branché la webcam USB, elle est devenue camera 0, et le Raspberry Pi Camera Module 3 embarqué est devenu camera 2. Donc, la webcam (`0`) fonctionne, et le Raspberry Pi Camera Module 3 embarqué (`2`) se contente de sortir une image noire.

Résultats de la détection

Maintenant que la détection d'objets fonctionne, le système affiche ce qu'il voit. J'ai pointé ma webcam sur toutes sortes d'objets, des numéros du magazine Elektor aux films sur mon écran en passant par les objets de mon bureau, et je peux dire qu'elle est très sensible aux bicyclettes. Elle a bien détecté la « TV », «télécommande» et « plante en pot », avec quelques faux positifs, comme « gâteau » pour la lumière de ma caméra, et elle a détecté une « cravate » là où il n'y en avait pas. Dans la **figure 4**, vous voyez qu'un chien avec une balle est détecté comme une balle et détecter un chat avec un snowboard. Lorsque les scènes sont animées, elle a souvent recours à « bicyclette ».

Encore du travail à faire

Il est évident qu'il y a beaucoup de modifications à apporter et de modèles différents à essayer, et qu'il est même possible de former les nôtres. Quant aux applications : un de mes amis a eu beaucoup de succès dans la détection de personnes sur son système de sécurité domestique en utilisant OpenCV, tandis que les ingénieurs d'Elektor ont eu beaucoup d'objectifs pour la détection d'objets dans la vidéo en direct, par exemple en étudiant le modèle de détection d'objets YOLO plutôt qu'OpenCV [3]. Pour ma part, je m'intéresse à la classification d'objets dans des vidéos et des photographies archivées.
Comme je l'ai mentionné dans le dernier article, j'ai des centaines d'heures de vidéo enregistrées à partir de journaux télévisés anciens, etc., que j'aimerais classer, mais je n'ai certainement pas le temps de les regarder toutes en prenant soigneusement des notes. J'aimerais les numériser et confier à l'IA toutes les tâches de classification, de la détection d'objets à la transcription de la parole en texte, et ajouter le tout à une base de métadonnées en ligne consultable.
Bien que la précision de la détection laisse actuellement à désirer. Ce qui est génial, c'est que vous pouvez changer la source de détection de la caméra en direct au fichier vidéo en changeant juste une ligne dans le script - celle avec l'appel `VideoCapture()`. J'ai essayé un exemple de fichier vidéo, pris au Churchill College à Cambridge l'année dernière, et je l'ai téléchargé sur le Raspberry Pi en utilisant à nouveau SCP :

```
C:\Users\Brian>scp 20230919_174323.mp4 briantw@
raspberrypi:/home/briantw/tflite1/20230919_174323.mp4
```

et j'ai ensuite changé la ligne dans le listage 1 en :

```
cap = cv2.VideoCapture('20230919_174323.mp4')
```

et c'est parti pour la détection d'objets dans la vidéo. La bonne nouvelle, c'est qu'il était si facile de changer de source. La mauvaise nouvelle est qu'il n'y avait pas de vélo dans la vidéo (**figure 5**).

Prochaine étape

Maintenant que le système fonctionne et qu'il est capable de détecter des objets, je vais m'efforcer d'améliorer la précision et d'explorer différents modèles et leur entraînement. Si j'ai de la chance, je parviendrai à ce que le système produise des résultats agréables qui serviront à des fins productives. Aurai-je de la chance ? Nous verrons bien.

230181-F-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (brian.williams@elektor.com).

À propos de l'auteur

Brian Tristam Williams est fasciné par les ordinateurs et l'électronique depuis qu'il a eu son premier «micro-ordinateur» à l'âge de 10 ans. Son aventure avec Elektor Magazine a commencé lorsqu'il a acheté son premier numéro à 16 ans, et depuis lors, il suit le monde de l'électronique et de l'informatique, explorant et apprenant sans cesse. Il a commencé à travailler pour Elektor en 2010 et, aujourd'hui, il s'attache à suivre les dernières tendances technologiques, en se concentrant notamment sur l'IA et les ordinateurs monocartes tels que le Raspberry Pi.

Listage 1. Test de la caméra et détection d'objets

```
import cv2
import numpy as np
from tflite_runtime.interpreter import Interpreter

# Initialize the TensorFlow Lite interpreter
model_path = 'Sample_TFLite_model/detect.tflite'
interpreter = Interpreter(model_path=model_path)
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
input_shape = input_details[0]['shape']

# Load labels
with open('Sample_TFLite_model/labelmap.txt', 'r') as file:
    labels = [line.strip() for line in file.readlines()]

# Initialize video capture from the camera
cap = cv2.VideoCapture(0)
while True:
     ret, frame = cap.read()
     if not ret:
         break

    # Preprocess the frame
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (input_shape[1], input_shape[2]))
    input_data = np.expand_dims(frame_resized, axis=0)

    # Perform detection
     interpreter.set_tensor(input_details[0]['index'], input_data)
     interpreter.invoke()

    # Retrieve detection results
    # Bounding box coordinates of detected objects
    boxes = interpreter.get_tensor(output_details[0]['index'])[0]
    classes = interpreter.get_tensor(output_details[1]['index'])[0]  # Class index of detected objects
    scores = interpreter.get_tensor(output_details[2]['index'])[0]  # Confidence of detected objects
    count = int(interpreter.get_tensor(output_details[3]['index'])[0])  # Total number of detected objects

    # Loop over all detections and print detection info
    for i in range(count):
        class_id = int(classes[i])
        score = scores[i]
        bbox = boxes[i]
        # Filter out weak detections by ensuring the confidence is greater than a minimum threshold
        if score > 0.5:
            label = labels[class_id]
            print(f"Detected: {label} with confidence {score}, Bounding Box: {bbox}")

cap.release()
```

LIENS

[1] Brian Tristam Williams, « premiers essais avec TensorFlow » Elektor 3/2024 : https://www.elektormagazine.fr/magazine/elektor-336/62796

[2] PuTTY — Terminal program for remote access to your headless Pi : https://putty.org

[3] Saad Imtiaz, « CaptureCount » Elektor 3/2024 : https://www.elektormagazine.fr/magazine/elektor-336/62793