Projet IOT : Détection d'objets en temps réel avec YOLOv4

Aperçu du projet

Ce projet illustre la mise en œuvre de la détection d'objets en utilisant le modèle YOLOv4. Le code utilise OpenCV, NumPy, et d'autres bibliothèques pertinentes pour accomplir la détection d'objets.
Pour lINstant, le code recupere une liste d'images depuis le web et les traite pour détecter des objets.
Les résultats de détection sont initialement affichés dans la console pour des tests et développements ultérieurs, ils pourront être envoyés à un broker MQTT et stockés dans une base de données.

Le code source est disponible dans mon dépôt GitHub ProjetIOT

Étapes à suivre

1. Importation des bibliothèques

Les bibliothèques nécessaires pour le projet incluent OpenCV pour le traitement d'images, NumPy pour la manipulation des données, et PIL pour la gestion des images. Matplotlib est utilisé pour l'affichage des images dans un environnement de notebook.

```
!pip install paho-mqtt
!pip install pymongo

from IPython.display import display, Javascript, Image
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

2. Clonage et préparation du modèle YOLOv4

Le modèle YOLOv4 est cloné depuis le dépôt officiel et préparé avec les configurations nécessaires pour l'utilisation du GPU et d'OpenCV.

```
!git clone https://github.com/AlexeyAB/darknet
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!make
```

3. Téléchargement et chargement des poids du modèle

Les poids pré-entraînés de YOLOv4 sont téléchargés et chargés dans le modèle configuré.

```
!wget https://pjreddie.com/media/files/yolov4.weights
```

4. Détection d'objets sur des images aléatoires

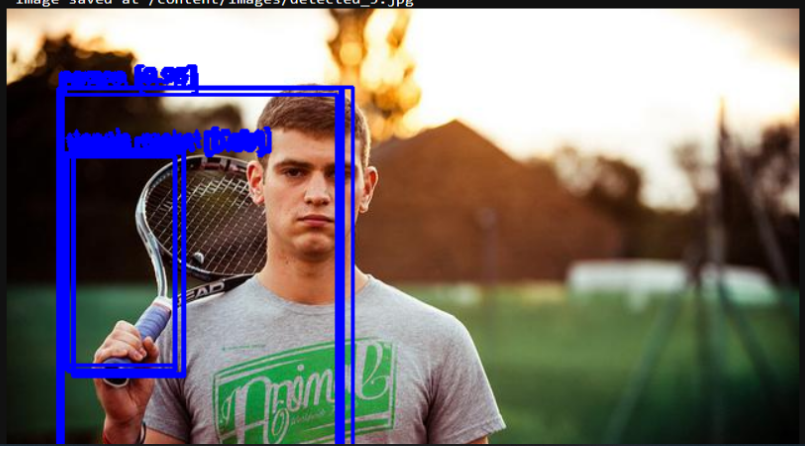
Les images sont récupérées aléatoirement du web et traitées pour détecter des objets. Chaque image traitée est ensuite affichée avec les boîtes de délimitation et les étiquettes des objets détectés.

```
images = [
    "https://example.com/image1.jpg",
    "https://example.com/image2.jpg",
    # Plus d'URLs d'images
]

for image_url in images:
    frame = get_image(image_url)
    detections = darknet_helper(frame, 416, 416)
    # Dessin des boîtes de délimitation et affichage des résultats
    display_image_with_boxes(frame, detections)
    cv2.imwrite('/content/detected_images/detected_image.jpg', frame)
```

Validation de la détection

```
Detections: [('tennis racket', 0.5754929, (48, 116, 135, 287)), ('tennis racket', 0.58198076, (47, 116, 135, 287)), ('tennis
racket', 0.8314192, (48, 119, 136, 291)), ('tennis racket', 0.83317477, (49, 118, 135, 292)), ('tennis racket', 0.64284366,
(54, 116, 142, 294)), ('tennis racket', 0.64439887, (53, 113, 143, 296)), ('person', 0.5744028, (45, 69, 266, 411)), ('perso
n', 0.9829408, (41, 64, 271, 413)), ('person', 0.9816127, (42, 63, 269, 414)), ('person', 0.98004913, (44, 63, 279, 411)),
('person', 0.979796, (44, 63, 279, 411))]
Image saved at /content/images/detected_5.jpg
```




```
Detections: [('bottle', 0.91730773, (184, 1, 322, 180)), ('bottle', 0.95619655, (189, -1, 327, 184)), ('bottle', 0.5829324, (19
1, 6, 326, 190)), ('bottle', 0.92682457, (160, -3, 336, 177)), ('bottle', 0.91915303, (159, -4, 336, 179)), ('bottle', 0.9681774
```

Vous pouvez trouver le code complet dans le fichier `main.ipynb`.

Emplacement du dépôt: Lien