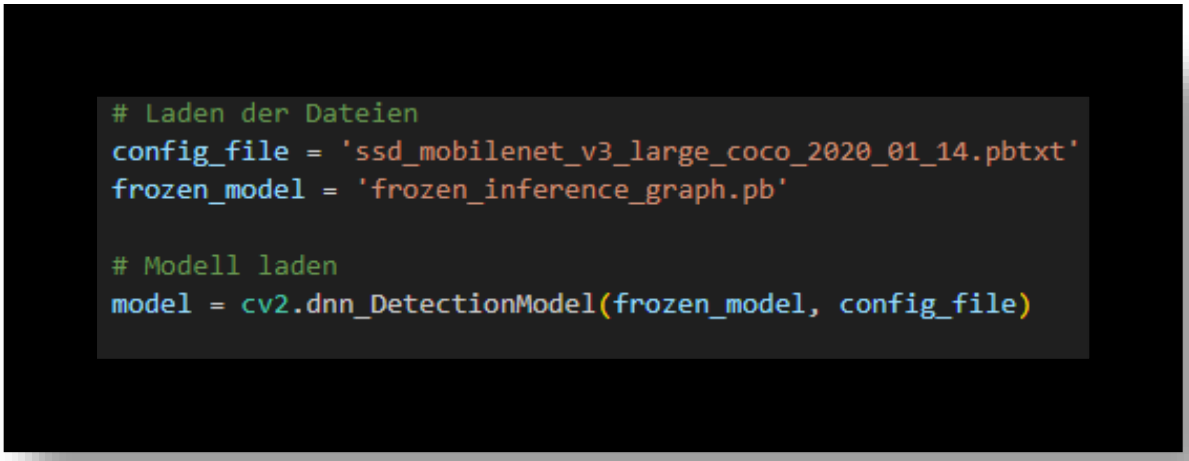
Dokumentation Add-on Programm Video-Objekterkennung

Einleitung
Diese Dokumentation beschreibt ein Python-Programm zur Objekterkennung unter Verwendung von OpenCV und einem vortrainierten SSD Mobilenet V3-Modell. Das Programm erkennt Objekte in Echtzeit auf Videodaten, sei es von einer Webcam oder einer Videodatei.

Abhängigkeiten
Um dieses Programm auszuführen, müssen Sie die folgenden Python-Bibliotheken installieren:

- `cv2` (OpenCV): Eine umfassende Bibliothek für Computer Vision.
- `matplotlib.pyplot`: Eine Bibliothek zum Anzeigen von Bildern.
- Die Dateien `ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt` und `frozen_inference_graph.pb`, die das vortrainierte Modell enthalten.
- Eine Textdatei `Labels.txt`, die die Namen der erkannten Klassen enthält.
- Eine Videodatei (z. B. "sample.mp4") oder eine funktionierende Webcam für die Videoquelle.

Laden der Modelldateien
Das Programm beginnt damit, die erforderlichen Modell- und Konfigurationsdateien zu laden. Die `frozen_inference_graph.pb`-Datei enthält das trainierte SSD Mobilenet V3-Modell, während `ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt` die Konfigurationsinformationen enthält.



```
# Laden der Dateien
config_file = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
frozen_model = 'frozen_inference_graph.pb'

# Modell laden
model = cv2.dnn_DetectionModel(frozen_model, config_file)
```

Laden der Klassenbezeichnungen
Die Klassenbezeichnungen, die das Modell erkennt, werden aus der Textdatei `Labels.txt` geladen und in einer Python-Liste `classLabels` gespeichert.

```python
# Leere Python-Liste
classLabels = []
# Laden der Klassenbezeichnungen
file_name = 'Labels.txt'
with open(file_name, 'rt') as fpt:
    classLabels = fpt.read().rstrip('\n').split('\n')
```

Konfiguration des Modells

Das Modell wird auf eine Eingabegröße von 320x320 Pixeln konfiguriert, da dieses spezielle Modell diese Größe erwartet. Weitere Konfigurationen umfassen die Skalierung der Eingabewerte, die Mittelwertanpassung und die Vertauschung der Farbkanäle.

```python
# Eingabegröße definieren (320x320), da unser Modell nur dieses Format unterstützt
model.setInputSize(320, 320)
model.setInputScale(1.0 / 127.5)
model.setInputMean((127.5, 127.5, 127.5))
model.setInputSwapRB(True)
```

Videoquelle öffnen

Das Programm versucht, eine Videodatei mit dem Namen "sample.mp4" zu öffnen. Wenn dies nicht gelingt, wird die Webcam als Videoquelle verwendet.

```python
cap = cv2.VideoCapture("sample.mp4")

if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Kann Video nicht öffnen")
```

Objekterkennungsschleife

In einer Endlosschleife wird kontinuierlich Video von der ausgewählten Quelle gelesen. Das Modell wird auf jedem Frame angewendet, und erkannte Objekte werden auf dem Frame markiert und beschriftet.

```
font_scale = 3
font = cv2.FONT_HERSHEY_PLAIN

while True:
    ret, frame = cap.read()
    ClassIndex, confidence, bbox = model.detect(frame, confThreshold=0.55)

    print(ClassIndex)
    if len(ClassIndex) != 0:
        for ClassInd, conf, boxes in zip(ClassIndex.flatten(), confidence.flatten(), bbox):
            if ClassInd <= 80:
                cv2.rectangle(frame, boxes, (255, 0, 0), 2)
                cv2.putText(frame, classLabels[ClassInd - 1], (boxes[0] + 10, boxes[1] + 40), font,
                            fontScale=font_scale, color=(0, 255, 0), thickness=3)
    cv2.imshow('Objekterkennungs-Tutorial', frame)
    if cv2.waitKey(2) & 0xFF == ord('q'):
        break
```

Beendigung des Programms

Das Programm kann beendet werden, indem Sie die Taste "q" drücken. Nach Beendigung wird die Videoquelle freigegeben, und alle geöffneten Fenster werden geschlossen.

```
cap.release()
cv2.destroyAllWindows()
```