Object Detection

1. Nutzen der EspCam:
   Das Format der durch aufnahme() in Python aufgenommenen Bilder hängt von den im Browser angewendeten Einstellungen ab. Ist dort eine Auflösung von 240x320 eingesetllt, so haben die aufgenommenen Bilder ebenfalls dieses Format. Bei der Aufnahme mehrerer Bilder hintereinander ist daher zu beachten, dass die Änderung der Einstellungen im Browser sofortige Auswirkungen auf die Aufnahmen hat.
   Camera Model: CAMERA_MODEL_AI_THINKER

2. Bilder in Roboflow annotieren:
   - Neues Object Detection Projekt anlegen
   - Bilder/Ordner mit Bildern hochladen
   - Bilder annotieren in dem Bounding Boxen eingezeichnet werden
   - Neue Version des Datensatzes erstellen: Train-/Val-/Test-Daten balancen (Für jede Kategorie mindestens ein Bild!), Resize, Augmentation
   - Falls man bereits annotierte Bilder aus einem Roboflow Projekt in ein anderes übertragen will: Neuen Datensatz erstellen (am besten ohne Augmentation), Bilder als zip Downloaden (bspw. in v8-Format), einen Ordner erstellen und in diesen data.yaml, die Bilder und die Labels kopieren, den neuen Ordner normal in Roboflow hochladen
3. Annotierten Datensatz laden:
   - In Roboflow: Versioons -> Download Dataset -> Show Download Code
   - Kopieren Des Codes der Form:

```
- from roboflow import Roboflow
- rf = Roboflow(api_key="<key>")
- project = rf.workspace("karlsruher-institut-fr-technologie-7bdnc").project("hksd-gsu3r")
- version = project.version(2)
- dataset = version.download("yolov8")
```

   - In den Python Code einfügen ´

Funktion: aufnahme()

Zum erstellen eines Datensatzes: nimmt Bilder mit der EspCam auf und speichert diese.

Argumente:

- url = <IP-Adresse der EspCam>
- interval = <Abstand zwischen zwei Bildern in Sekunden>
- klasse = <Name des Unterordners, für Object detection nicht relevant, muss aber angegeben werden>
- ordner = <Name der Ordners in dem die Klassen und Bilder gespeichert werden (Muss nicht bereits existieren)>

Beispiel:

```
aufnahme(url = 'http://192.168.0.235',
    interval = 1,
```

klasse = 'nan',
ordner = 'daten')

Funktion: training_detection()

Trainiert ein neuronales Netz zur Object detection.

Argumente:

- dataset = <dataset (Aus dem von Roboflow kopierten Code)>
- epochen = <Anzahl Epochen>
- img_size = < Form: (width, height), Alle Bilder werden auf angegebene Form skaliert. Default: (640,640)>

Beispiel:

```
training_detection(dataset,
        epochen=15,
        img_size=(320,320))
```

Anmerkungen:

Das genutzte Modell ist auf einen Datensatz des Formats 640x640 vortrainiert. Dieses Format muss jedoch für das eigene Training nicht beibehalten werden.

Die Bilder werden unter Anwendung von Padding in das gewünschte Format gebracht.

Die Zusammenfassung des Trainingsergebnisses wird automatisch gespeichert. Der Pfad dazu wird automatisch auf der Konsole ausgegeben. Bspw.: runs\detect\train152. Es werden sowohl verschiedene Auswertungsmetriken, als auch das Ergebnis des Testens auf den Testbildern angezeigt.

Das Modell wird ebenfalls in einem Unterordner ‚weights' gespeichert. Bspw.: Results saved to runs\detect\train15.

Funktion: testen_detection()

Nimmt ein Bild mit der EspCam auf und führt basierend auf einem angegebenen Modell Object detection durch.

Argumente:

- url = <URL/IP der EspCam>
- model = <Pfad zum gewünschten Modell, übliche Form: 'runs/detect/train/weights/best.pt'>
- conf_thresh = <Sicherheit ab der ein Objekt detektiert werden soll, zwischen 0 und 1>
- img_size = <optinale Angabe, Form: (width, height) mit default=(640,640), falls das Bild zuvor in ein bestimmtes Format gebracht werden soll>

Beispiel:

```
testen_detection(url='http://192.168.0.235/capture',
        model='runs/detect/train12/weights/best.pt',
        conf_thresh=0.3,
        img_size=None)
```

Anmerkungen:

Das Modell kann jegliche Bild-Eingabegröße verarbeiten, skaliert diese Standardmäßig jedoch immer auf 640x640 für die Verarbeitung. Zum Schluss wird es jedoch automatisch wieder zurück in seine originale Größe skaliert. Ist dies nicht gewünscht, so kann eine explizite Angabe für img_size gemacht werden.

Das aufgenommene Bild wird als latest_capture.pic gespeichert und das annotierte Bild mit vorhergesagten bounding Boxen unter latest_capture_annotated.pic. Auf der Konsole werden die vorhergesagten Labels und die Koordinaten der bounding Boxen ausgegeben.

Funktion: neural_network_detection()

Stellt eine Verbindung zum Arduino sowie der EspCam her. Erhält der Code vom Arduino ein Signal, wird automatisch ein Bild mit der EspCam aufgenommen und Object detection durchgeführt. Das Ergebnis wird zurück an den Arduino gesendet.

Argumente:

- url = <URL/IP der EspCam>
- arduino_ip = <IP des Arduinos die beim Laden des Beispielprogramms ausgegeben wird>
- port = <wird im Arduino-Code festgelegt, default: port=12345>
- model = <Pfad zum gewünschten Modell, übliche Form: 'runs/detect/train/weights/best.pt'>
- conf_thresh = <Sicherheit ab der ein Objekt detektiert werden soll, zwischen 0 und 1>

Beispiel:

```
neural_network_detection(url='http://192.168.1.100',
          arduino_ip='192.168.1.102',
          port=12345,
          model='runs/detect/train16/weights/best.pt',
          conf_thresh=0.4)
```