

# Erste Schritte in die Objekterkennung

## Einführung

Die Objekterkennung ist eine Kernkomponente vieler moderner Computer-Vision-Anwendungen. Sie ermöglicht es, Objekte in Bildern oder Videos zu identifizieren und ihre Positionen (Bounding Boxes) zu bestimmen. Dieses Verfahren basiert auf der Kombination von **maschinellem Lernen**, insbesondere **Deep Learning**, und Bildverarbeitungstechniken.

## 1. Grundprinzipien der Objekterkennung

Die Objekterkennung umfasst zwei Hauptaufgaben:

- **Klassifikation:** Bestimmung, um welches Objekt es sich handelt.
- **Lokalisierung:** Identifikation der Position des Objekts im Bild.

In modernen Systemen wie **YOLOv5** wird dies gleichzeitig durchgeführt. Die Architektur zerlegt die Aufgabe in die folgenden Schritte:

1. **Feature Extraction:** Extraktion relevanter Merkmale aus dem Bild durch Convolutional Neural Networks (CNNs).
2. **Region Proposals:** (Nur bei manchen Modellen) Erstellung von Vorschlägen für mögliche Positionen von Objekten.
3. **Bounding Box Regression:** Präzise Bestimmung der Umrandungen der erkannten Objekte.
4. **Klassifikation:** Zuordnung einer Objektklasse zu jeder Bounding Box.

**YOLO** (You Only Look Once) verwendet ein End-to-End-System, bei dem alle diese Schritte in einem einzigen Durchgang durchgeführt werden. Das Modell unterteilt das Bild in ein Gitter und bestimmt in jedem Gitterfeld mögliche Objekte.

## 2. Verwendete Softwaresysteme

### a. YOLOv5

**YOLOv5** ist eines der führenden Frameworks für Objekterkennung. Es ist bekannt für seine Geschwindigkeit und Genauigkeit. Die Pipeline basiert auf:

- **PyTorch:** Eine Deep-Learning-Bibliothek für Training und Deployment des Modells.
- **Pretrained Models:** Vortrainierte Modelle wie COCO (Common Objects in Context) bieten eine breite Palette an erkennbaren Objekten.
- **NMS (Non-Maximum Suppression):** Ein Algorithmus zur Auswahl der relevantesten Bounding Boxes bei überlappenden Vorhersagen.

### b. NVIDIA Jetson Inference Toolkit

Das Toolkit optimiert Modelle für den Einsatz auf NVIDIA Jetson Geräten. Es enthält:

- Unterstützung für TensorRT: Eine NVIDIA-Technologie zur Beschleunigung von KI-Modellen.
- Werkzeuge zur Bildverarbeitung und Modellkonvertierung.

### c. TensorRT

**TensorRT** ist ein Framework für die Optimierung und Beschleunigung von Deep-Learning-Modellen auf NVIDIA-Hardware. Vorteile:

- Reduzierte Latenz und beschleunigte Inferenz.
- Unterstützung für INT8-Quantisierung (Reduktion der Präzision zur Steigerung der Effizienz).

### d. OpenCV

**OpenCV** ist eine Open-Source-Bibliothek für Bildverarbeitung und Computer Vision:

- Ermöglicht die Verarbeitung von Eingabebildern und Videos.
- Unterstützt die Visualisierung der Erkennungsergebnisse.

## 3. Schlüsseltechnologien im Hintergrund

### a. Deep Learning

Die Objekterkennung basiert auf neuronalen Netzen, speziell Convolutional Neural Networks (CNNs). Diese Netze lernen Merkmale wie Kanten, Formen und Texturen und können daraus komplexe Muster erkennen.

### b. GPU-Computing

Da die Berechnungen für Deep-Learning-Modelle sehr aufwendig sind, werden GPUs (Graphics Processing Units) verwendet. NVIDIA Jetson Geräte und CUDA-Unterstützung sind wesentliche Bestandteile für die schnelle Verarbeitung.

### c. Dataset und Training

- **COCO-Dataset:** Ein Standard-Datensatz, der häufig für das Training von Modellen wie YOLO verwendet wird.
- Modelle wie YOLOv5 nutzen Techniken wie Transfer Learning, um vortrainierte Netzwerke an spezifische Anwendungsfälle anzupassen.

## 4. Beispielhafter Workflow

1. **Installation:** YOLOv5 wird installiert und die erforderlichen Abhängigkeiten werden eingerichtet.
2. **Modellinitialisierung:** Ein vortrainiertes YOLOv5-Modell wird geladen.
3. **Datenverarbeitung:** Eingabebilder oder Video-Frames werden vorverarbeitet.
4. **Inference:** Das Modell verarbeitet die Eingabe und liefert Bounding Boxes, Klassen und Konfidenzwerte zurück.
5. **Visualisierung:** Die Ergebnisse werden mit OpenCV oder anderen Werkzeugen angezeigt.

## Zusammenfassung

Die Kombination aus leistungsstarken Frameworks (**PyTorch**, **TensorRT**), Hardware (**Jetson**), und modernen Algorithmen (**YOLO**) macht die Objekterkennung effizient und vielseitig einsetzbar – von Überwachungskameras bis hin zu autonomen Fahrzeugen.

# Software-Abhängigkeiten installieren

## 1. Jetson-Inferenz-Toolkit

NVIDIA bietet das **Jetson Inference**-Projekt an, das vortrainierte Modelle und Beispielanwendungen für Objekterkennung, Klassifikation und Segmentierung enthält.

```
1 git clone --recursive https://github.com/dusty-nv/jetson-inference
2 # Falls der obige Befehl nicht funktioniert:
3 git clone --recursive --depth 1 https://github.com/dusty-nv/jetson-inference
```

Wechseln Sie ins Verzeichnis:

```
1 cd jetson-inference
```

Überprüfen Sie den Status des Repositories:

```
1 git status
```

## 2. Build-Prozess

Erstellen Sie das Projektverzeichnis und bauen Sie das Projekt:

```
1 cd jetson-inference
2 mkdir build
3 cd build
4 cmake ../
5 make -j$(nproc)
6 sudo make install
```

## 3. Python-Bibliotheken installieren

Je nach Projekt können folgende Python-Bibliotheken installiert werden:

```
1 sudo apt-get update
2 sudo apt-get install python3-pip
3 pip3 install numpy opencv-python matplotlib
4 pip3 install torch torchvision --extra-index-url https://download.pytorch.org/whl/cu118
```

## 4. YOLOv5/YOLONano verwenden

Laden Sie YOLOv5 herunter und installieren Sie die Abhängigkeiten:

```
1 git clone https://github.com/ultralytics/yolov5
2 cd yolov5
3 pip3 install -r requirements.txt
4 python3 detect.py --source 0
```

Falls ein Fehler wie folgend auftritt: *ValueError: numpy.dtype size changed, may indicate binary incompatibility. Expected 96 from C header, got 88 from PyObject*

Nutze diesen Befehl zur Behebung:

```
1 pip3 install --upgrade --force-reinstall -r requirements.txt
```

Wiederholen Sie die vorherigen Schritte.

## 1 YOLOv5 startet

die Kamera wird geöffnet und der Modell fängt an Objekte zu erkennen. Dabei sieht der Terminal wie folgendes aus:

```
0: 480x640 1 cell phone, 325.5ms
0: 480x640 1 cell phone, 330.2ms
0: 480x640 1 cell phone, 323.9ms
0: 480x640 1 cell phone, 319.8ms
0: 480x640 1 mouse, 1 cell phone, 311.1ms
0: 480x640 1 cell phone, 312.9ms
0: 480x640 1 cell phone, 315.9ms
0: 480x640 1 cell phone, 339.3ms
0: 480x640 1 cell phone, 319.6ms
0: 480x640 1 cell phone, 313.4ms
0: 480x640 1 cell phone, 312.6ms
```

## 2 Erkennungsbeispiele:

Im folgenden sind ein paar Gegenstände, die mit YOLOv5 erkannt wurden.

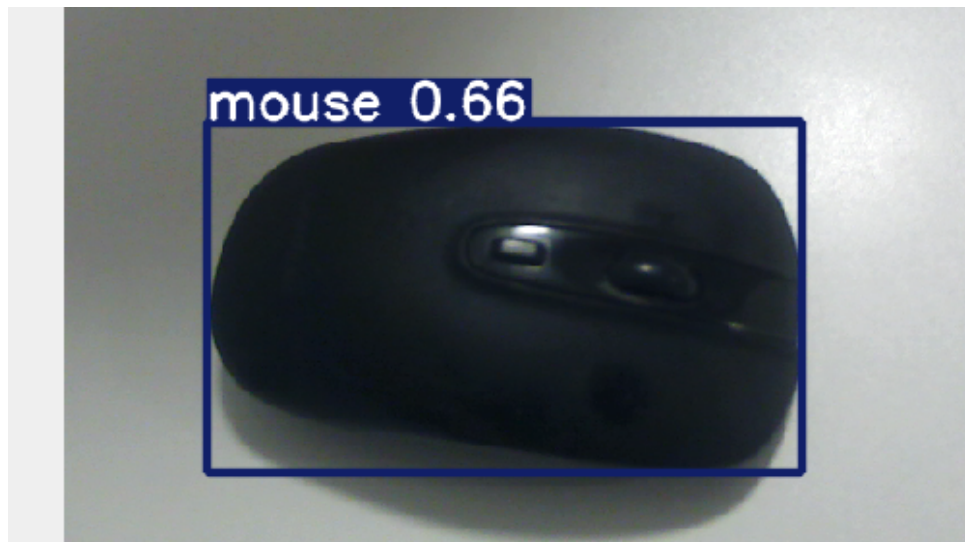
### 2.1 Handy

Erkennung von Handy:



### 2.2 Maus

Erkennung von Maus:



## 2.3 Tastatur

Erkennung von Tastatur:

