Wo ist Waldo?

Objekterkennung mit YOLOv10

Data Science 07.01.2025

https://img.clipart-library.com/2/clip-waldo-background/clip-waldo-background-18.png

Janina Dörflinger (211923)
Marvin Hill (211926)

Gliederung

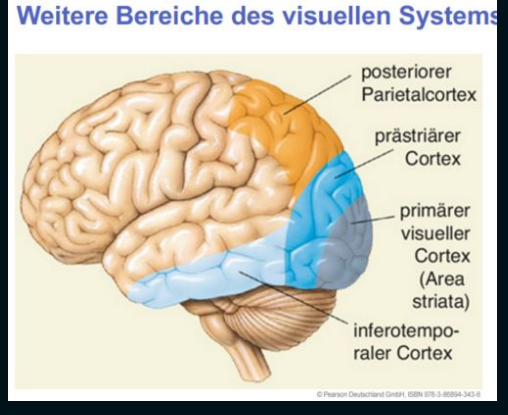
1. Objekterkennung
2. YOLO - Allgemein
3. YOLO - Algorithmus
4. YOLO - Performance und Modelle
5. Ultralytics
6. Wo ist Waldo?

Grundlagen

- Convolutional Neural Networks (CNNs)
- Max-Pooling und Convolutional Layer
- Funktion der Layer

Parallelen zum Menschlichen Sehen

- CNNs inspiriert vom visuellen Kortex
- einfache Merkmale in frühen Schichten extrahiert
- komplexere Merkmale in tieferen Schichten aufgebaut werden
- Neuronen im visuellen Kortex verbinden sich nur mit lokalen Bereich des Inputs
- Neuronen in CNN-Schicht nur mit lokaler Region der Eingabe verbunden



https://o.quizlet.com/2B.8q7YdvBhc9nB6e-dc0Q.png

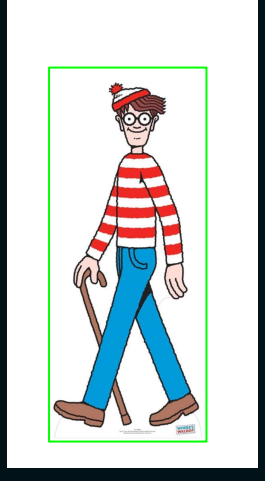
Parallelen zum Menschlichen Sehen

- Neuronen des visuellen Kortex: Erkennt Merkmale unabhängig von ihrer Position im Gesichtsfeld
- Pooling-Schichten in einem CNN fassen lokale Merkmale zusammen
- Neuronen im visuellen Kortex: nichtlineare Reaktionseigenschaften.
- CNNs erreichen Nichtlinearität durch Aktivierungsfunktionen wie ReLU

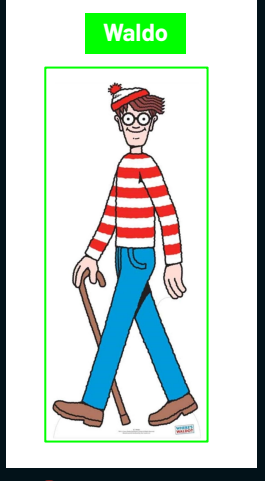
Object Detection/Objekterkennung

- Image Recognition: Klassifizierung von Bildern
- Image Localization: Identifizierung der Position
- Objekterkennung: Kombiniert beide Techniken

1 Image Recognition



2 Image Localisation



3 Object Detection

Single Stage / Two Stage Object Detection

Two Stage

- Vorschläge für Regionen werden in der Pipeline weiter zur Klassifikation getragen
- Sehr akkurat, langsamer
- Genauigkeit wichtiger als Geschwindigkeit

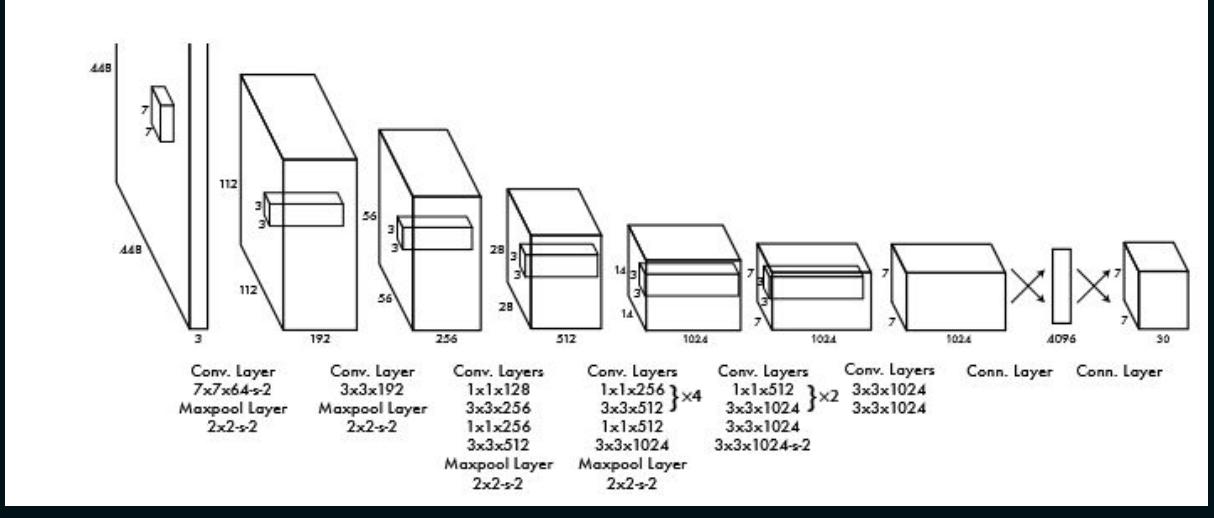
Single Stage

- Gefundene Regionen werden direkt klassifiziert
- Anwendungen, die Geschwindigkeit erfordern

YOLO

YOLO (You Only Look Once)

- Realzeit Algorithmus für Objekterkennung
- CNN
- verschiedene Größen
- 24 Convolutional Layer
- 4 Max Pooling Layer
- 2 Fully Connected Layer

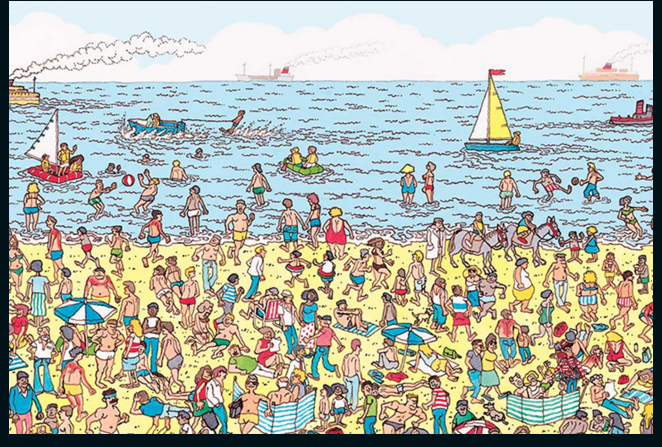


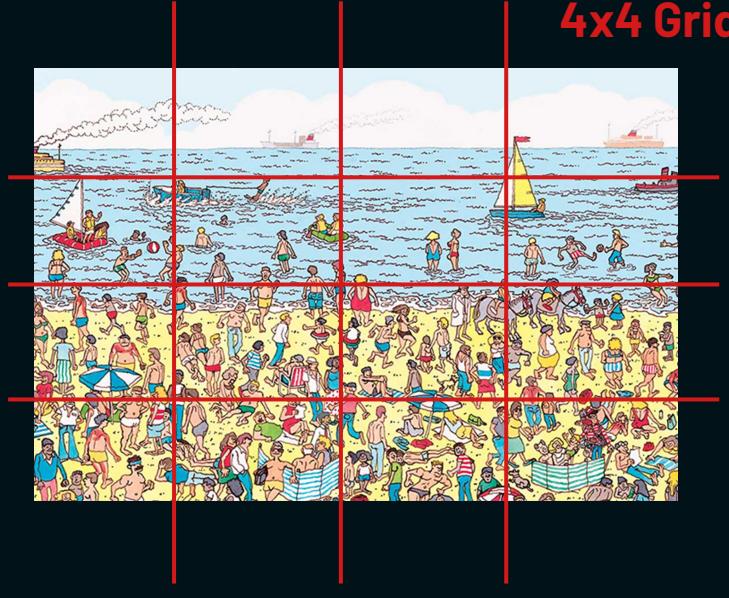
https://arxiv.org/pdf/1506.02640

YOLO Algorithmus

Rastererstellung

- Bild wird in NxN Zellen aufgeteilt
- Je Zelle: Objekterkennung in Bounding Boxen





https://www.paperflite.com/sites/default/files/inline-images/Where%27s%20Waldo_0.jpg

Bounding Box Regression

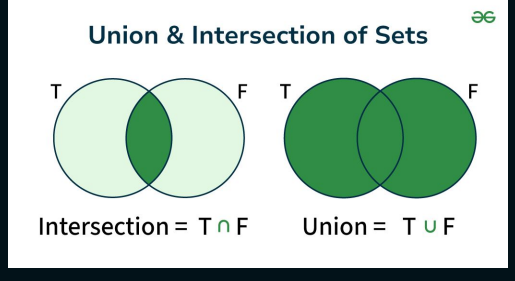
- Zelle: Erstellung von gewisser Anzahl an Bounding-Boxen
- Boxen werden klassifiziert

Finaler Vector durch Vektorgeneralisierung:

Y = [pc, bx, by, bh, bw, c1]

IoU + Non Maximum Suppresion

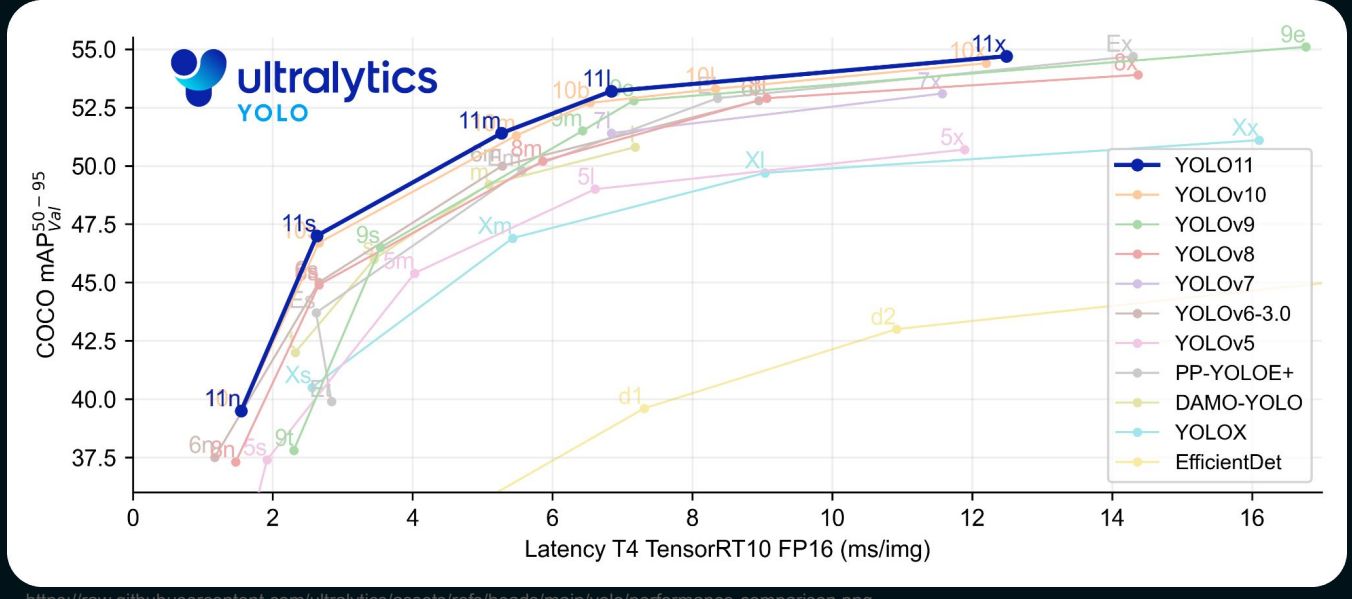
- Jede Bounding-Box hat einen Confidence-Score
- Boxen mit niedriger Wahrscheinlichkeit werden entfernt
- Objekte können in mehreren Zellen auftauchen
- Vergleich mit der besten Box
- YOLO errechnet den IoU-Wert (Intersektion / Vereinigung)
- Die Box mit IoU über dem Grenzwert wird ignoriert ⟶ Duplikat



https://media.geeksforgeeks.org/wp-content/uploads/20240711102224/Union-of-sets-Intersection-of-sets.png

YOLO Performance

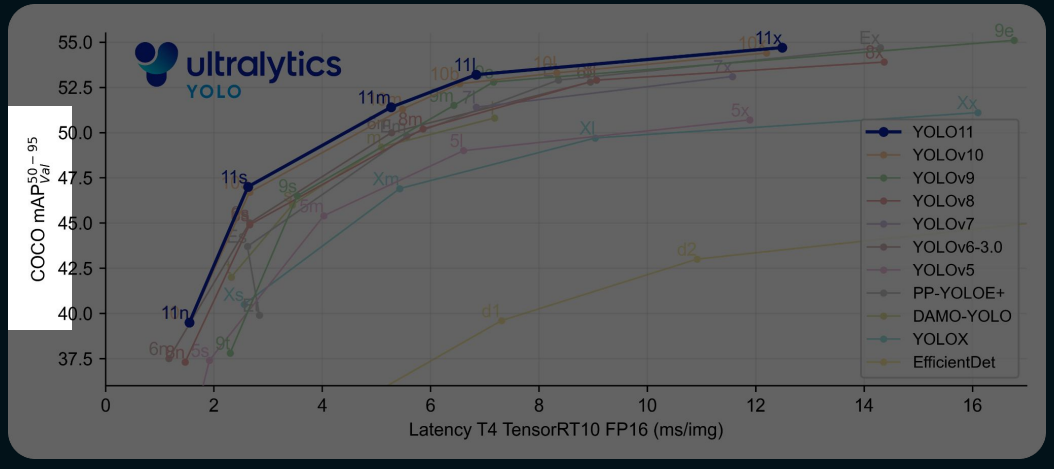
Performance

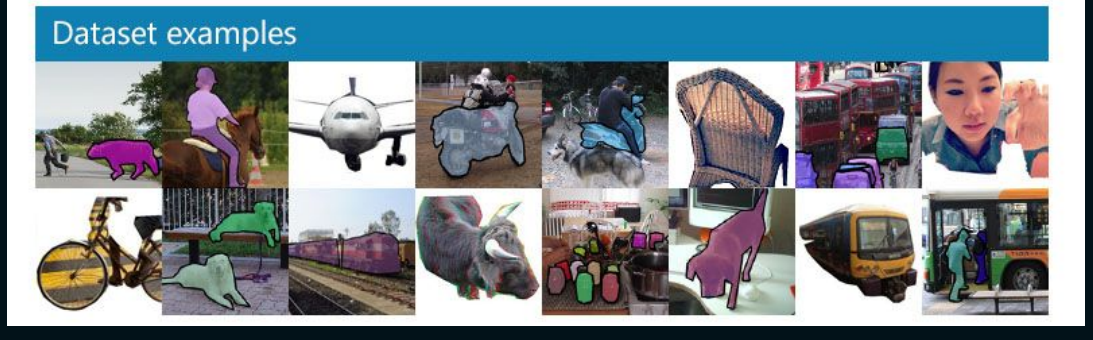


https://raw.githubusercontent.com/ultralytics/assets/refs/heads/main/yolo/performance-comparison.png

COCO (Common Objects in Context)

- COCO: Datensatz zum Benchmarking von Computer Vision Modellen
- AP (Average Precision): Vereint die Präzision und die Recall Leistung des Modells in einem Wert
- mAP (mean Average Precision): durchschnittlichen AP-Werte über mehrere Objektklassen
- 50 - 95: IoU Wert (Loose Match / Strict Match)

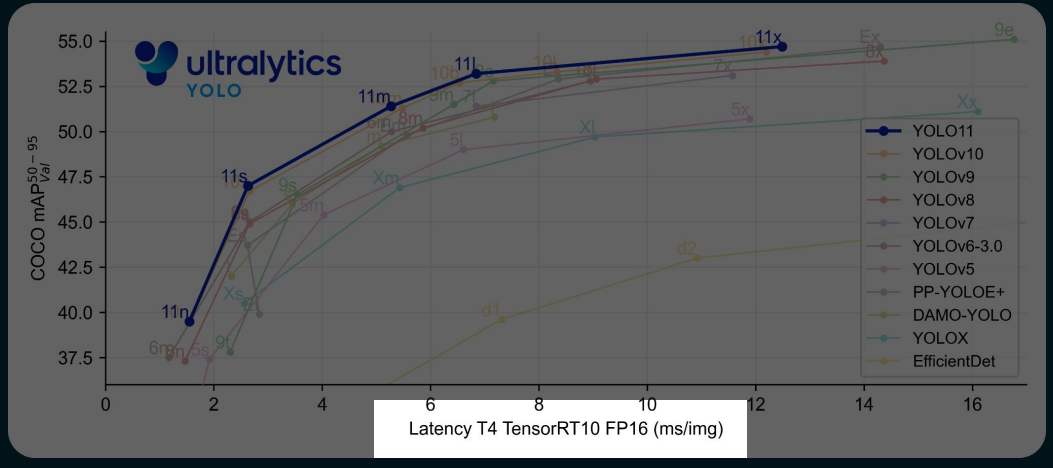




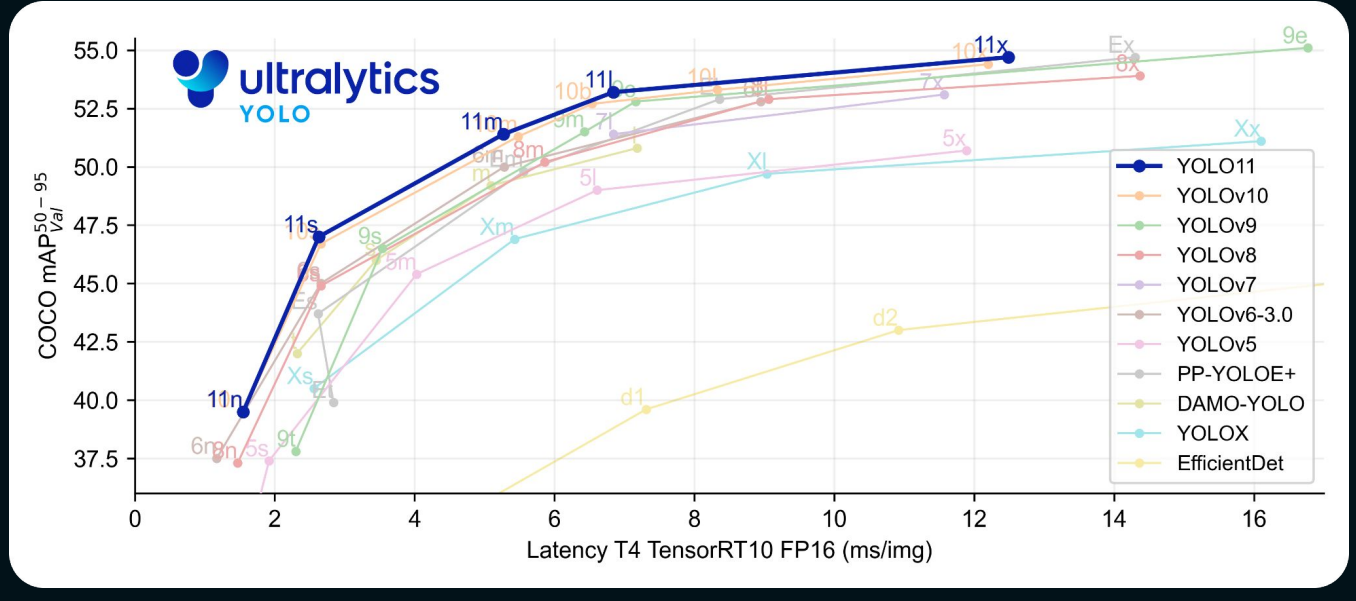
https://cocodataset.org/images/coco-examples.jpg

Latency T4 TensorRT10 FP16

- **Latency** : Wie lange dauert es, ein einzelnes Bild zu verarbeiten und eine Ausgabe zu erzeugen
- **T4**: Nvidia T4 GPU
- **TensorRT 10**: : Nvidia-Bibliothek zur Optimierung und Laufzeit von Deep-Learning-Modellen
- **FP16**: 16-Bit-Gleitkomma-Präzision für Berechnungen
- **ms/img**: Millisekunden pro Bild



Performance



YOLO 11

- yolo11-cls.yaml
- yolo11-obb.yaml
- yolo11-pose.yaml
- yolo11-seg.yaml
- yolo11.yaml

YOLO 11

- yolo11-cls.yaml
- yolo11-obb.yaml
- yolo11-pose.yaml
- yolo11-seg.yaml
- yolo11.yaml

- **cls:** Bildklassifizierungsmodell
- **obb:** Modell für orientierte Bounding Boxes (OBB-Modell)
- **pose:** keypoints/pose Schätzungs-Modell
- **seg:** **I**nstanzsegmentierungs-Modell
- Objekterkennung-Modell

ViTs (Vision Transformers)

- CNN Alternative
- Extension of the Transformer Architecture for Images

Ultralytics

Ultralytics - Firma

- Unternehmen hinter YOLO
- Entwicklung von Software für YOLO-Training (Ultralytics Hub)

Ultralytics - Package

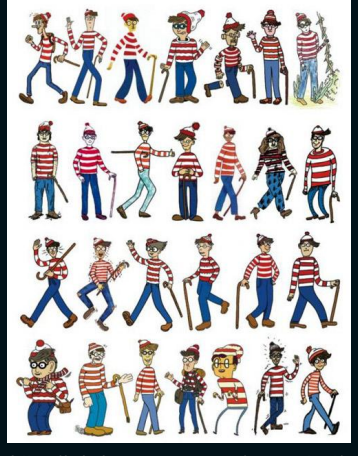
- Python-Bibliothek zur Arbeit mit YOLO
- Zugriff auf Modelle
- Training von Modellen

```python
from ultralytics import YOLO

# Load Model
model = YOLO('yolov10s.pt')

model.train(
    data='dataset.yaml',  # Dataset
    epochs=100,
    imgsz=256,  # Image Size
    batch=-1,
    name='custom_yolov10_model',  # Name of trained model
    pretrained=True  # Pretrained Weights (Transfer Learning)
)
```

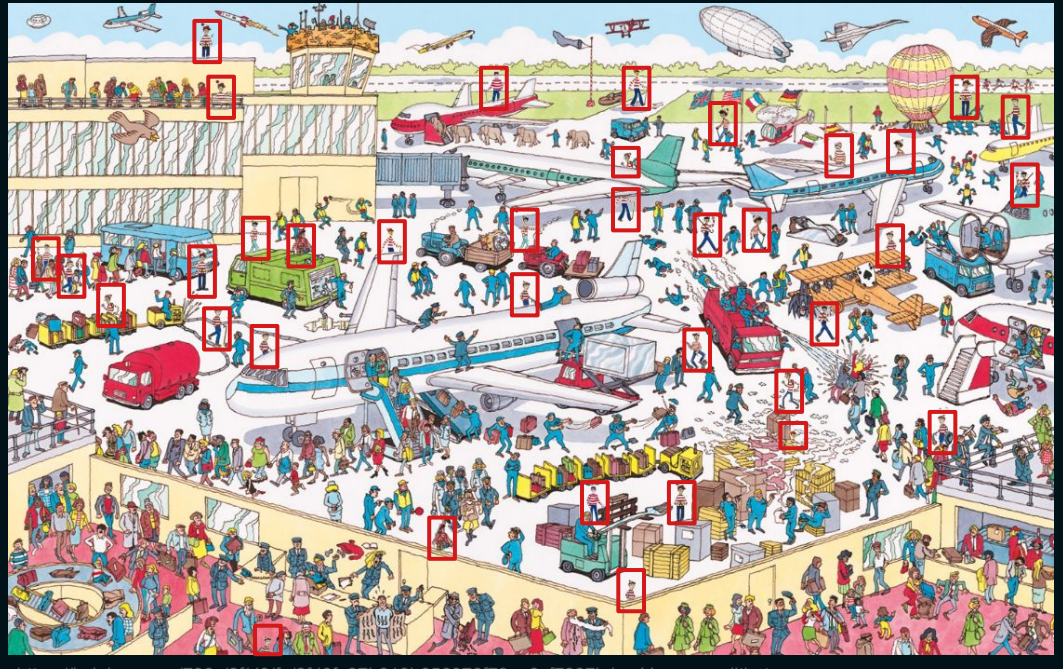
Project

Datensatz

- Datensatz von Kaggle
- Drei selbst erstellte Bilder



https://cdn.booooooom.com/wp-content/uploads/2011/09/wheres_waldo.jpg

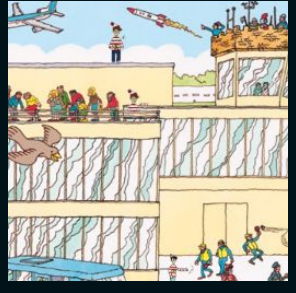


https://i.pinimg.com/736x/8f/d0/fa/8fd0fa97b313b656976f79ee8cf7997b.jpg Von uns editiert

Labeling Make Sense AI

Training

Images



Labels

dataset > 256 > labels > whereiswaldo.txt

You, 7 hours ago | 1 author (You)

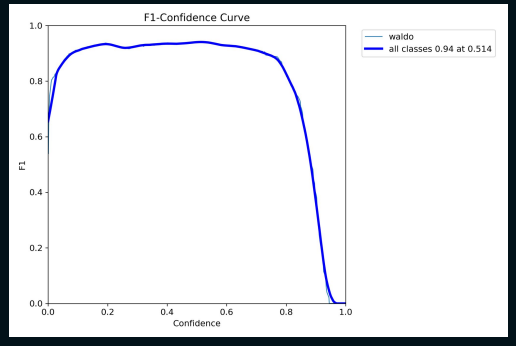
1	0	0.392901	0.141983	0.106487	0.168911
2	0	0.452876	0.360465	0.072215	0.111383
3	0	0.587515	0.943084	0.138311	0.113831
4	0	0.792534	0.965728	0.080783	0.068543

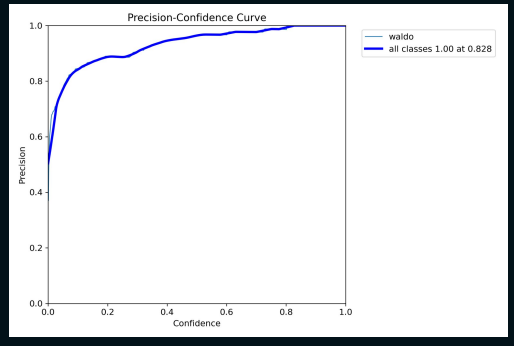
Klasse, x, y, Breite, Höhe

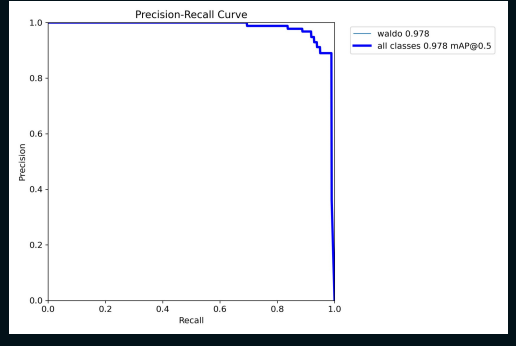
```
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Marvin\Documents\GitHub\where-is-waldo>python train.py
New https://pypi.org/project/ultralytics/8.3.50 available 🅄  Update with 'pip install -U ultralytics'
Ultralytics 8.3.23 🅄  Python-3.11.2 torch-2.0.0+cpu CPU (Intel Core(TM) i5-4670K 3.40GHz)
```

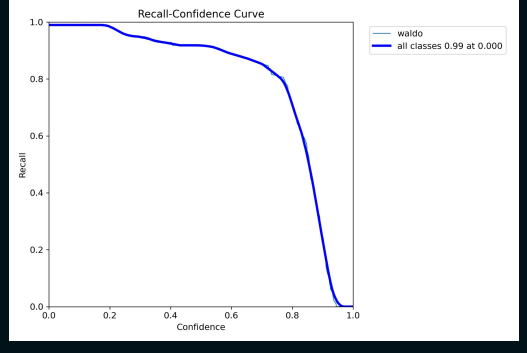
Metriken

- F1: harmonische Mittelwert aus Precision und Recall
- Precision: Von allen als positiv klassifizierten Objekten, wie viele sind tatsächlich korrekt
- Recall: Von allen vorhandenen Objekten, wie viele korrekt erkannt wurden
- Confidence: Wie sicher sich das Modell für eine Erkennung ist Schwellwert

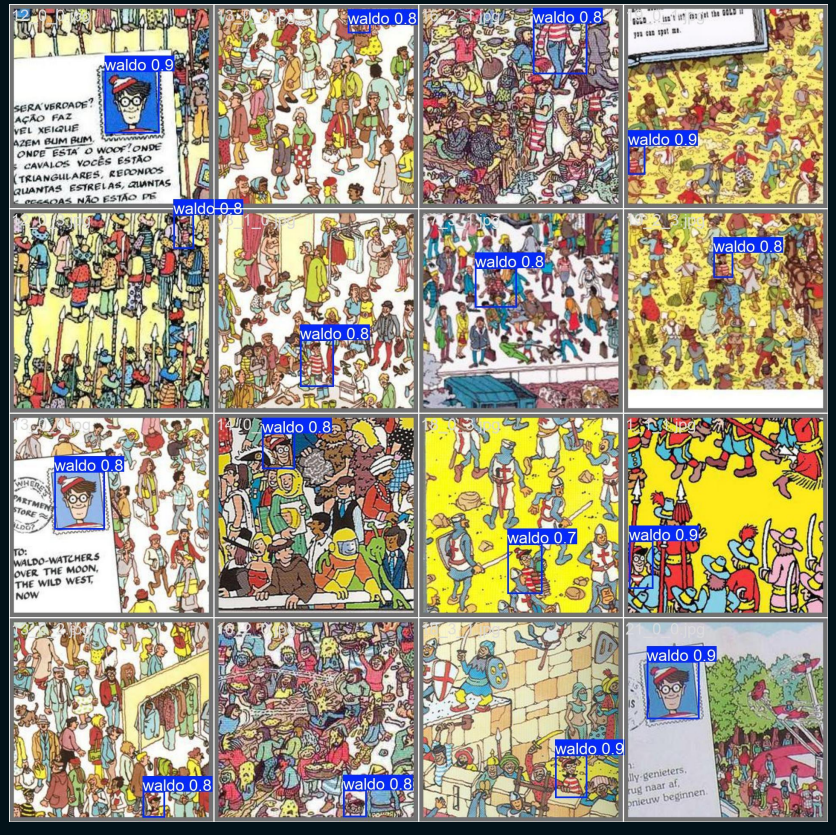








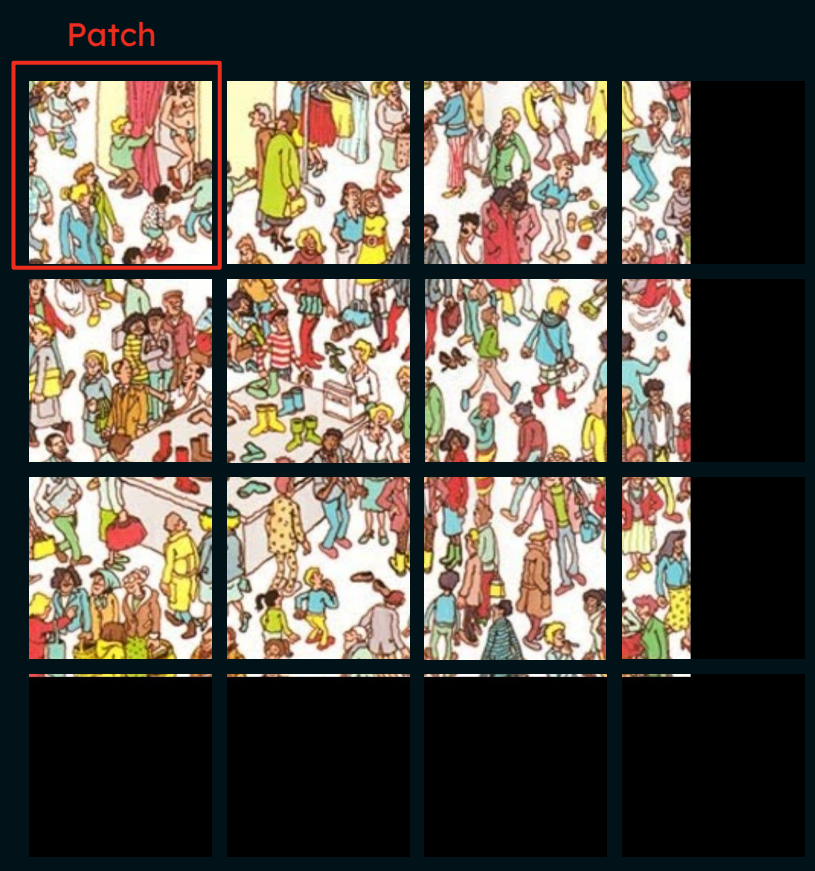
Beispiel Epoche



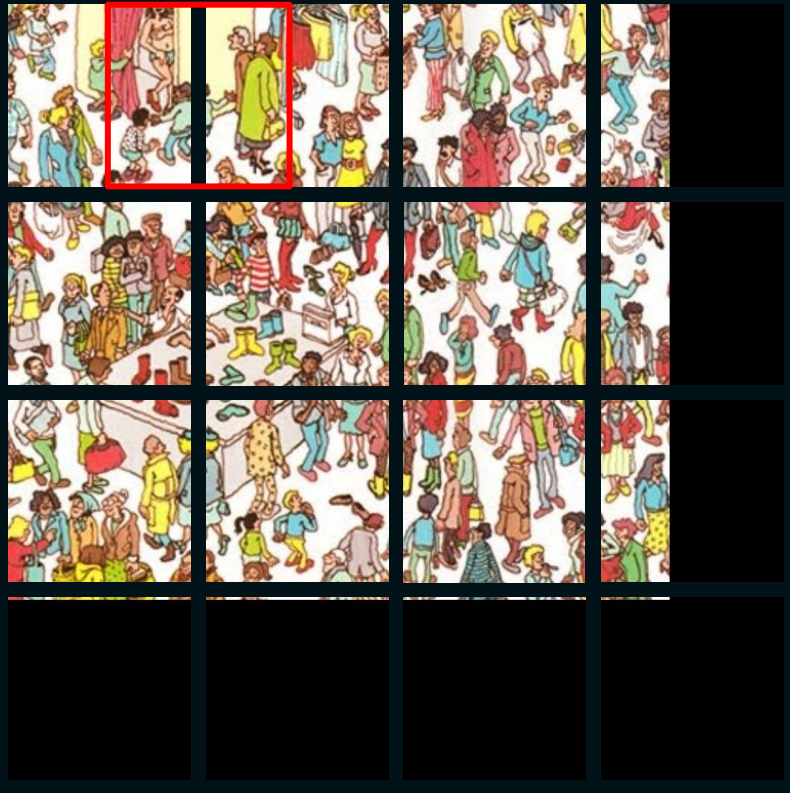
Input Image

Padding

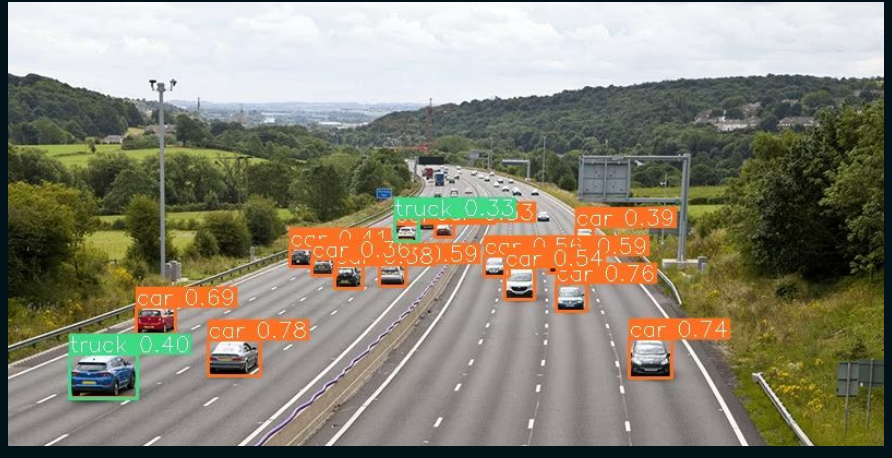
Schneiden



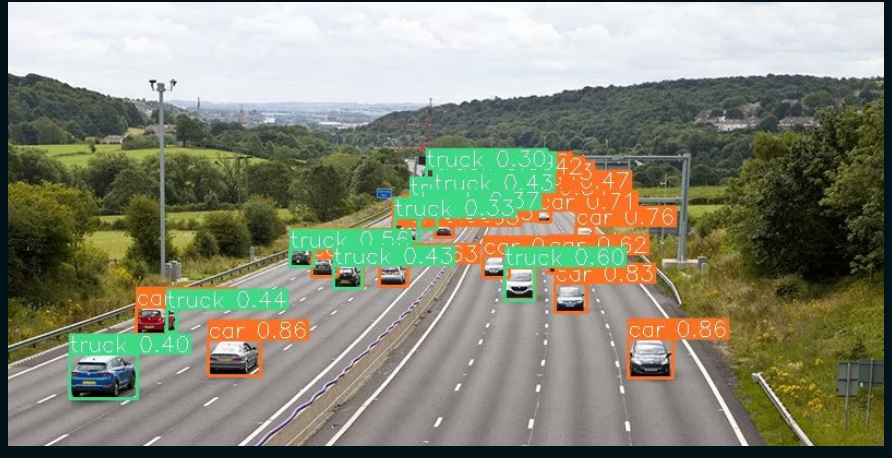
Schneiden Offset



Slicing Aided Hyper Inference



https://github.com/ultralytics/docs/releases/download/0/yolov8-without-sahi.avif

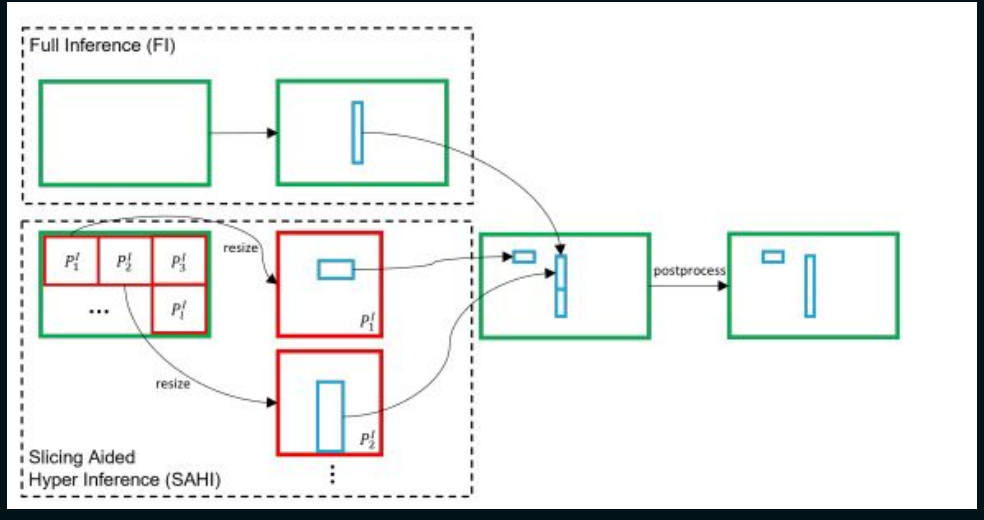


https://github.com/ultralytics/docs/releases/download/0/yolov8-with-sahi.avif

Slicing Aided Hyper Inference

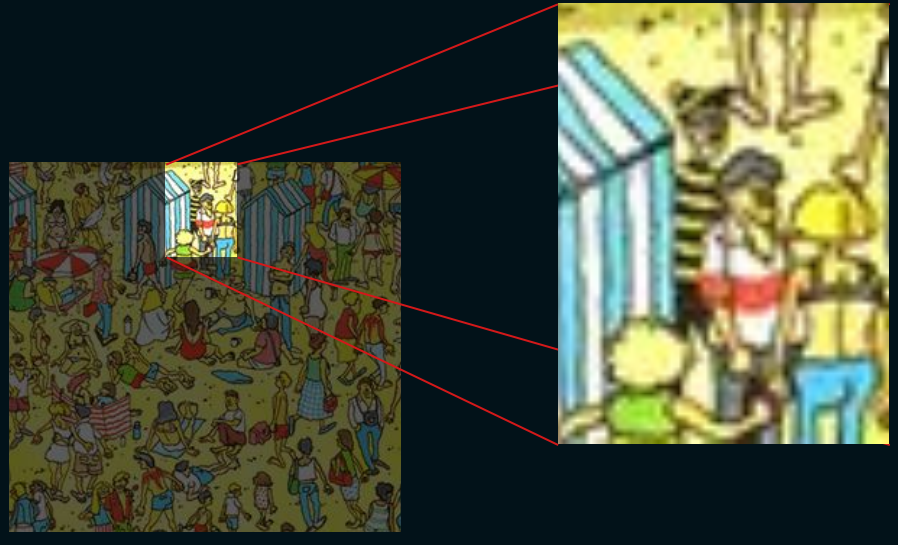
Unterschied unserer Implementierung und Paper “SLICING AIDED HYPER INFERENCE AND FINE-TUNING FOR SMALL OBJECT DETECTION”

- Bounding Boxes werden vereint
- Skalierung von Patches
- Erkennung im gesamten Bild wird vereint mit erkannten Bounding Boxes in den Patches



https://arxiv.org/pdf/2202.06934

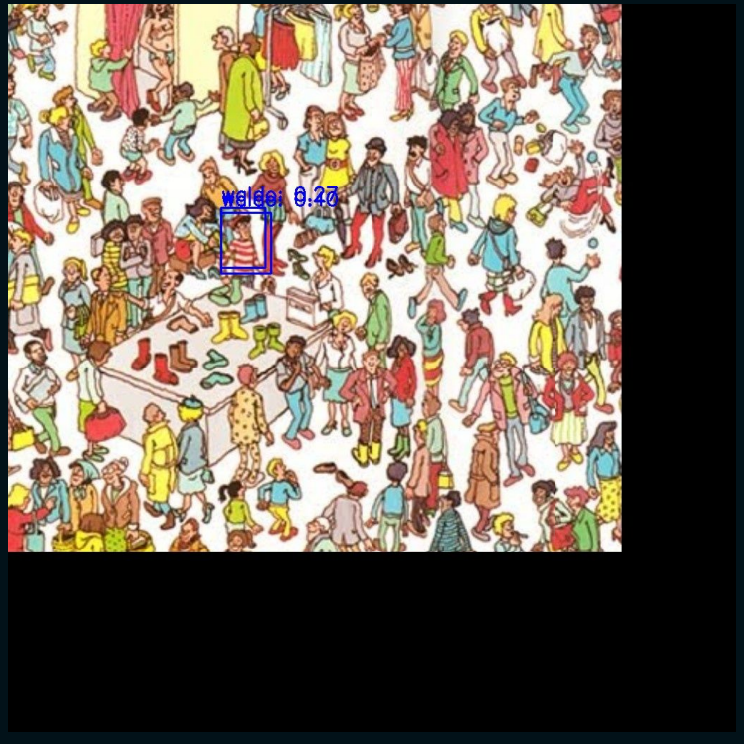
Slicing Aided Hyper Inference Unterschiede



Waldo erkennen


```
"bounding_boxes": [
    {
        "image_path": "output/cut/images/departmentstore_256_256.jpg",
        "x1": 43,
        "y1": 37,
        "x2": 113,
        "y2": 122
    },
    {
        "image_path": "output/cut/images/departmentstore_256_256.jpg",
        "x1": 43,
        "y1": 30,
        "x2": 105,
        "y2": 114
    },
    {
        "image_path": "output/cut/images/departmentstore_128_256.jpg",
        "x1": 43,
        "y1": 165,
        "x2": 113,
        "y2": 250
    },
    {
        "image_path": "output/cut/images/departmentstore_128_256.jpg",
        "x1": 43,
        "y1": 158,
        "x2": 105,
        "y2": 242
    }
]
```


Zusammenfügen



Verbesserungen

- Figuren, die ähnlich aussehen, aber nicht Waldo sind
- Oft niedrige Wahrscheinlichkeit mit Kaggle-Datensatz
- Erstellung von eigenen Bildern

Zusammenfassung

- Objekterkennung am Beispiel von YOLO
- Praktisches Beispiel

Habt Ihr noch Fragen?

Quelle - Datensatz

https://www.kaggle.com/datasets/residentmario/wheres-waldo?resource=download

Quellen (Vortrag)

YOLO Object Detection Explained: A Beginner's Guide | DataCamp

[2405.14458] YOLOv10: Real-Time End-to-End Object Detection

[1506.02640] You Only Look Once: Unified, Real-Time Object Detection
https://arxiv.org/pdf/1506.02640
https://docs.ultralytics.com/de

https://www.datacamp.com/de/tutorial/introduction-to-convolutional-neural-networks-cnns

Quellen cont’d

https://pypi.org/project/ultralytics/

Ultralytics HUB | YOLOv8 No-Code, End-to-End ML Lösung

https://www.ultralytics.com/de/yolo

https://github.com/ultralytics/ultralytics

Optimizing the Trade-off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction

https://encord.com/blog/yolo-object-detection-guide/

COCO - Ultralytics YOLO Docs

https://github.com/ultralytics/ultralytics/tree/main/ultralytics/cfg/models

YOLO Leistungsmetriken - Ultralytics YOLO Docs

https://arxiv.org/pdf/2202.06934

Bildquellen

Waldo-Bild im Folien-Design:

https://pbs.twimg.com/profile_images/561277979855056896/4yRcS2Zo_400x400.png

Die jeweiligen Abbildungen sind mit Quellen gekennzeichnet. Jegliche weitere Abbildungen wurden selbst erstellt.