Verkehrszeichenerkennung für die Stadt Rostock 22.10.2024

STATUS-UPDATE 02

BISHERIGE FORTSCHRITTE:

Seit dem letzten Status-Update

1. Sichtung der bereitgestellten Beispielbilder
2. Festlegung des Annotationsformats
3. Festlegung und Installation des Annotationstools (lokale Entwicklungsumgebung)
4. Beispielhafte Annotation weniger dieser Beispielbilder mit der Klasse „Traffic Sign“
5. Erstellung eines Beispiel-Datensatz aus diesen Bildern und Annotationen
6. Erstes Finetuning eines einfachen YOLO-basierten neuronalen Netzes zur Objekterkennung mithilfe dieser beschrifteten Beispielbilder → Nachtrainiertes Modell erhalten
7. Erste Inferenz (= Erkennung von Verkehrszeichen) auf diesem nachtrainierten Modell

NÄCHSTE SCHRITTE:

1. Weiterentwicklung des PoC-Prototyps
2. Dokumentation des bestehenden Ansatzes
3. Evaluierung verschiedener vortrainierter Modelle
4. Logging & Visualisierung des Trainingsverlaufs

BENÖTIGTES FEEDBACK:

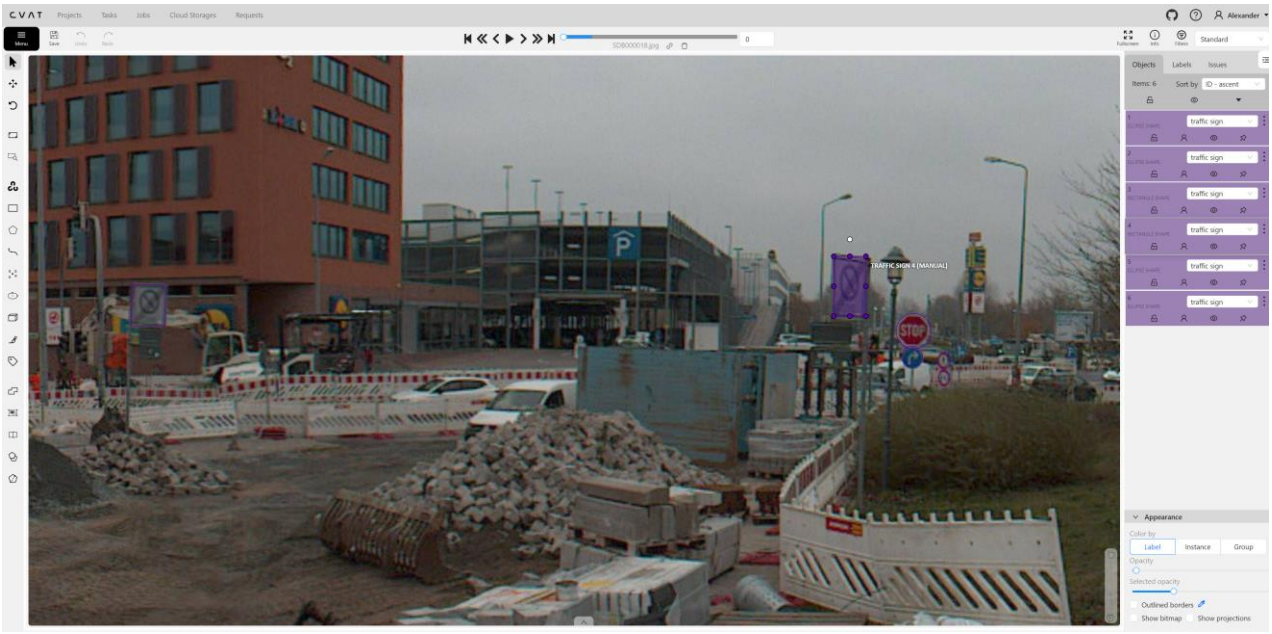
1. Welche Schilder sollen am Ende unterschieden werden können?
   → Liste von Klassen erstellen, z.B. anhand von dieser Wikipedia Liste, ggf. sinnvoll gruppieren. Für jede dieser Klassen werden ausreichend viele Beispiele benötigt. Die genaue Anzahl ist abhängig von der Qualität des Pretrained-Modells, grob geschätzt: 50+ (verschiedene) Beispiele je Schild wären gut, es gilt aber: Je mehr Bilder pro Klasse, desto besser.

STATUS-UPDATE 02

ANHANG

1. Annotationstool

Als Annotationstool habe ich mich für „CVAT“ entschieden, da dieses kostenlos, open source und schon seit mehreren Jahren etabliert ist, immer noch weiterentwickelt wird und trotz der schnell verständlichen Primärfunktionen sehr umfangreich ist.



Bildquelle: Screenshot selbsterstellt, Bild aus euren Beispieldaten

Rahmen für Annotationen können mit der Maus einfach via „Drag & Drop“ platziert und in der Größe und Anzahl Eckpunkte verändert werden, und Klassen-Labels können über Dropdowns ausgewählt werden.

Für eine fortgeschrittenere und effizientere Bedienung lassen sich auch über mehrere Bilder hinweg verfolgte Objekte einfügen (auf Wunsch auch mit KI-Unterstützung für die Objektverfolgung / Vorschläge), es gibt zahlreiche Tastenkürzel für die schnelle Abarbeitung großer Bildmengen und es ist ein Mehrbenutzer-Betrieb mit separaten Accounts möglich.

2. Annotationsformat

Für die Daten habe ich das Format „YOLO V8 Segmentation“ vorgesehen. Es ermöglicht es, Bilder und Annotationen gemeinsam in einer einfachen Ordnerstruktur abzulegen und ist sowohl für den Computer als auch für Menschen gut verständlich.

Das Format ist sowohl mit dem für diesen vorgesehenen Annotierungstool „CVAT“ als auch mit dem geplanten Framework „YOLO“ kompatibel.

Grundsätzlich wird bei diesem Ansatz für jede Bilddatei eine gleichnamige Textdatei angelegt, die die jeweiligen Annotationen enthält. Jede Zeile dieser Dateien beschreibt dabei eine Objektinstanz mit Informationen wie dem Klassen-Label, Bounding-Box Koordinaten sowie optional weiteren Attributen.

Darüber hinaus gibt es noch wenige den Datensatz und die Datensatz-Splits beschreibende Dateien.

STATUS-UPDATE 02

3. Erstes Finetuning & erste Inferenz

Für das Finetuning wurde ein vortrainiertes YOLOv8 Model geladen und in 100 Epochen mit 10 Bildern aus dem eigens erstellten Datensatz nachtrainiert. In diesem ersten Prototyp kam in diesem Datensatz nur eine einzige Klasse „Traffic Sign“ vor, die vom Hintergrund unterschieden werden konnte. Eine Unterscheidung innerhalb der Verkehrszeichen war hier noch nicht möglich. Es sollte in einem nächsten Schritt problemlos möglich sein, weitere Klassen zu annotieren und beim Finetuning zu berücksichtigen.

Bildquelle: Rahmen eingezeichnet durch eigenes Python-Skript, Bild aus euren Beispieldaten

Aufgrund der kleinen Datenmenge ist noch kein aussagekräftiges Qualitätsurteil möglich, das Experiment diente erstmal nur dazu zu schauen, wie Finetuning in diesem Framework codetechnisch überhaupt funktioniert.