

# Nummernschilderkennung mit Python

---

Anne-Sophie Bollmann, Susanne Klöcker, Pia von Kolken, Christian Peters

19. Januar 2021

1. Einleitung
2. Extraktion des Nummernschildes
3. Nummernschild auslesen

# Einleitung

---

Ziel: Erkennen von Nummernschildern auf Fotos und Auslesen der Nummernschilder

Herausforderungen:

- Vielfältigkeit der Nummernschilder
- Rahmenbedingungen der Bildaufnahme (Beleuchtung)



**Abbildung 1:** Beschreibung

# Extraktion des Nummernschildes

---

# Convolutional Neural Networks

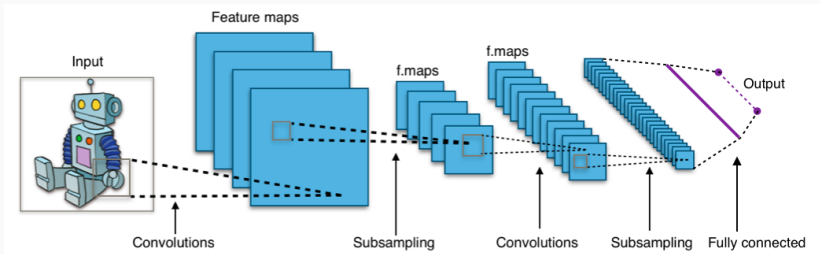


Abbildung 2: Convolutional Neural Network. <sup>1</sup>

**Input:** Bild mit Auto  $\mapsto$  **Output:** Bounding Box

<sup>1</sup>Bildquelle: [https://de.wikipedia.org/wiki/Convolutional\\_Neural\\_Network](https://de.wikipedia.org/wiki/Convolutional_Neural_Network)

## Netzarchitektur:

- Inspiriert durch YOLO (**Y**ou **O**nly **L**ook **O**nce) [?]
- Kann sowohl Klassen als auch Bounding Boxes vorhersagen
  - Wir brauchen nur Bounding Boxes von Nummernschildern, also Vereinfachung nötig

## Implementierung:

- Open Source Deep-Learning Bibliothek Keras
- Geschrieben in Python



# Nummernschild auslesen

---

Wie können wir Text auf Bildern auslesen?

- Tesseract: freie Software zur Texterkennung mit vielen vorimplementierten Sprachen
- Problem: Text wird größtenteils noch nicht richtig auf den unbearbeiteten Nummernschildern erkannt
- Lösung: das erkannte Nummernschild derart vorverarbeiten (preprocessing), dass das richtige Auslesen der einzelnen Elemente möglichst gut unterstützt wird

- Geeignetes Werkzeug: OpenCV
- OpenCV ist eine plattformübergreifende Bibliothek, für Echtzeit-Computer-Vision-Anwendungen
- beinhaltet Algorithmen für die Bildverarbeitung und im Rahmen von Computer Vision (CV) auch für maschinelles Lernen
- Nutzung für die Verarbeitung des erkannten Nummernschildes (z.B. Thresholding), um die Zeichen besser zu erkennen und richtig auszulesen

# Beispiel für die Anwendung von OpenCV

OpenCV wurde bereits auf Nummernschildverarbeitung verwendet:



Abbildung 3: Original



Abbildung 4: Graustufen



Abbildung 5: Thresholding



Abbildung 6: Konturen



**Abbildung 7:** Aussortierung



**Abbildung 8:** Schwarze Schrift aufweissem Hintergrund

Auf das finale Bild (Abbildung 7) wird anschliessend Tesseract angewendet, das die Nummern und Buchstaben ausgibt

## Validierung:

- Rastersuche über Parametereinstellungen für OpenCV
- Validierung über *character accuracy* [?]:

$$\frac{n - \#errors}{n},$$

wobei  $n$  Anzahl der Zeichen im Datensatz und  $\#errors$  Anzahl der fehlerhaft erkannten Zeichen

→ Parametereinstellungen mit höchstem *character accuracy* werden für Texterkennung verwendet

**Irgendwas zum Schluss**



