

Nummernschilderkennung mit Python

Anne-Sophie Bollmann, Susanne Klöcker, Pia von Kolken, Christian Peters
18. Februar 2021

Pipeline

Pipeline

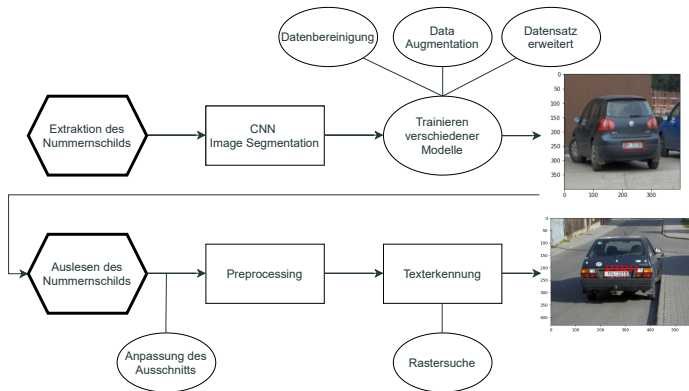
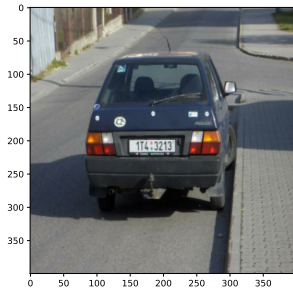
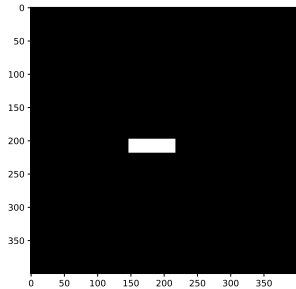


Image Segmentation

Trainingseingaben



(a) Eingabe



(b) Ziel

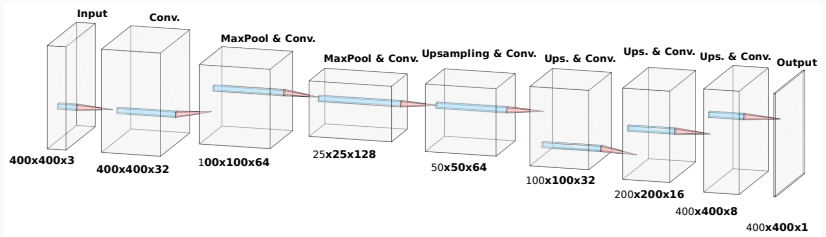
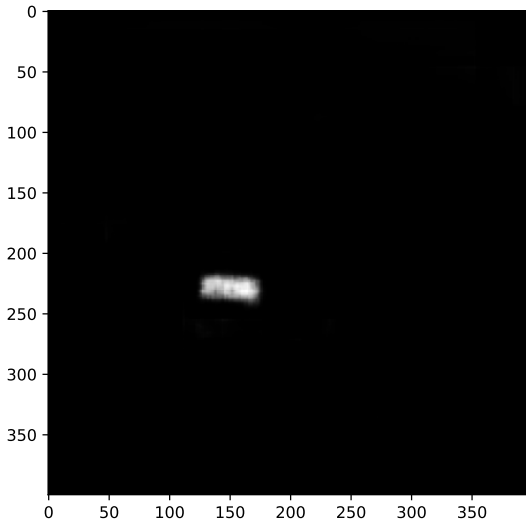
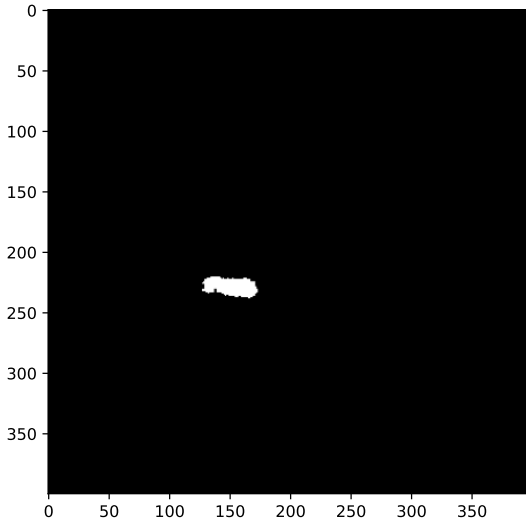


Abbildung 2: Das Ergebnis nach wochenlangem Ausprobieren. Insgesamt 191.297 trainierbare Parameter.

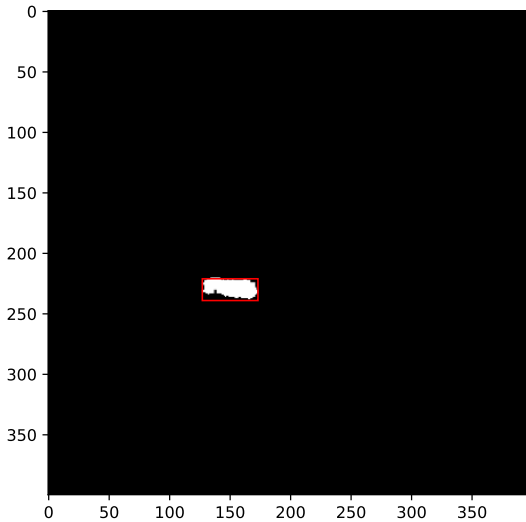
Modellvorhersage



Schwellenwert



Umschließendes Rechteck



Resultat



Implementierung in Tensorflow, Training auf Google Colab GPU

- 534 Zusätzliche Trainingsbilder hinzugefügt¹
- Data Augmentation: 949 \Rightarrow 22.776 Bilder
 - Horizontal Flip, Random Cropping, Random Contrast, Random Brightness
 - **Benötigt >14GB GPU Speicher!**
- Mini Batch Stochastic Gradient Descent mit ADAM Optimizer
- Loss: Binary cross entropy

$$- \sum_{i \in \text{Pixel}} y_{\text{true}}^{(i)} \log(y_{\text{pred}}^{(i)}) + (1 - y_{\text{true}}^{(i)}) \log(1 - y_{\text{pred}}^{(i)})$$

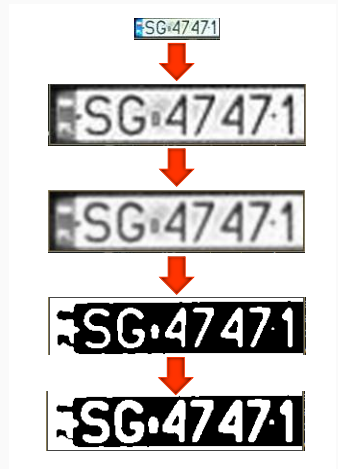
- Zur Validierung 20 Bilder aus EU/RO per Hand selektiert
- „Early Stopping“ nach 19 Epochen

¹<https://github.com/RobertLucian/license-plate-dataset>

Optical Character Recognition

Bearbeitungsschritte

1. Vergrößerung und Graustufen
2. Blurring (Bildglättung)
 - Gaußsche Unschärfe und Median Unschärfe
3. Thresholding (Schwellenwertverfahren)
 - Otsu: wählt automatisch einen geeigneten Schwellenwert aus
 - Einfaches Verfahren mit Schwellenwerten {60, 80, 100, 120}
 - Binary-Inv.: Die Werte werden derart getauscht, dass schwarz zu weiß wird und umgekehrt → führt zur besseren Erkennung von Konturen
4. Dilation (Morphologische Transformation)



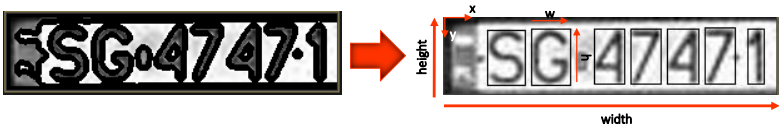
Aussortierung der Konturen

- Genutzt werden die Werte x , y , w , h (Werte der Kontur) sowie $width$ und $height$ (Breite und Höhe des Bildes)
- Es werden nur Konturen berücksichtigt, die folgende Bedingungen erfüllen:

$$\frac{height}{h} > 3$$

$$\frac{h}{w} < 1.2$$

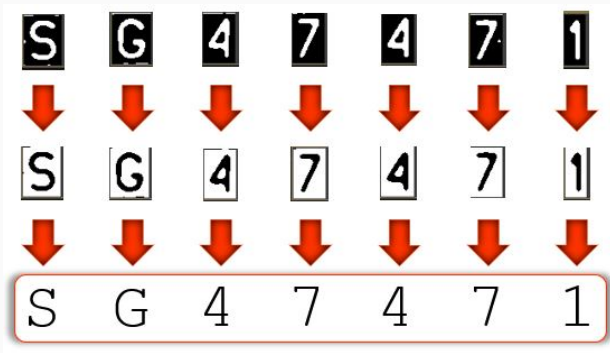
$$\frac{width}{w} > 50$$



Character auslesen

Einstellungen für das Auslesen mit Tesseract 5:

- Jeder Character wird einzeln ausgelesen → Page Segmentation Mode (`--psm10`)
- Engine Mode `--oem3`
- Zeichen-Whitelist (Großbuchstaben + Zahlen 0-9)



Boundingboxes verschieben

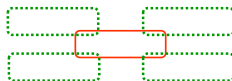
Verschiebung der Bounding-Box in acht Richtungen → Character, die nicht in ursprünglicher Bounding-Box enthalten sind, werden noch ausgelesen



oben/unten



links/rechts



oben/unten links,
oben/unten rechts

Tabelle 1: Levenshtein-Distanz

ohne Verschiebung	mit Verschiebung	mit Verschiebung & Thresholding-Variierung
4.05	3.55	3.2

Learnings

Erfolgreiches Programmieren im Projekt erfordert Rahmenbedingungen

- Requirements festlegen (welche Pakete werden benötigt, was muss installiert sein) → `requirements.txt`
- Versionskontrolle (in unserem Fall Git)²

Verbesserungen:

- Die OCR-Pipeline ist manchmal noch recht fehleranfällig
 - ⇒ Schwellenwertverfahren verbessern
 - ⇒ Vielleicht Image Segmentation auch für OCR?
- Geschwindigkeit kann noch verbessert werden

²Besucht uns auf https://github.com/cxan96/license_plate_detection

Evaluator Results

Dies und das...



A. et al.

Systematic review on vehicular licence plate recognition framework in intelligent transport systems.

The Institution of Engineering and Technology, 2019.



T. P. I. P. Ltd.

Learn opencv, 2021.



J. Redmon and A. Farhadi.

Yolov3: An incremental improvement.

arXiv, 2018.



J. F. R. Rice, Stephen R. and T. A. Nartker.

The fifth annual test of ocr accuracy.

Information Science Research Institute, 1993.



R. Smith.

An overview of the tesseract ocr engine.

Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2:629–633, 2007.