THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik
TECHNISCHE HOCHSCHULE MITTELHESSEN

Grundlagen der KI

Abschlussprojekt

Kaffeeliste

Lucas Weißmann, Lennart Zinn

Inhaltsverzeichnis

1. Projektbeschreibung und Aufgabenstellung
2. Training von Objekterkennungsmodell
3. Entwicklungsprozess
   - 3.1. Datensatz & Labeling
   - 3.2. Namens - & Objekterkennung
   - 3.3. Auswertung & Exportierung
   - 3.4. Mobile-App
4. App Demo
5. Ausblick
   - 5.1. Probleme
   - 5.2. Verbesserungsmöglichkeiten
   - 5.3. Fazit
6. Quellenverzeichnis

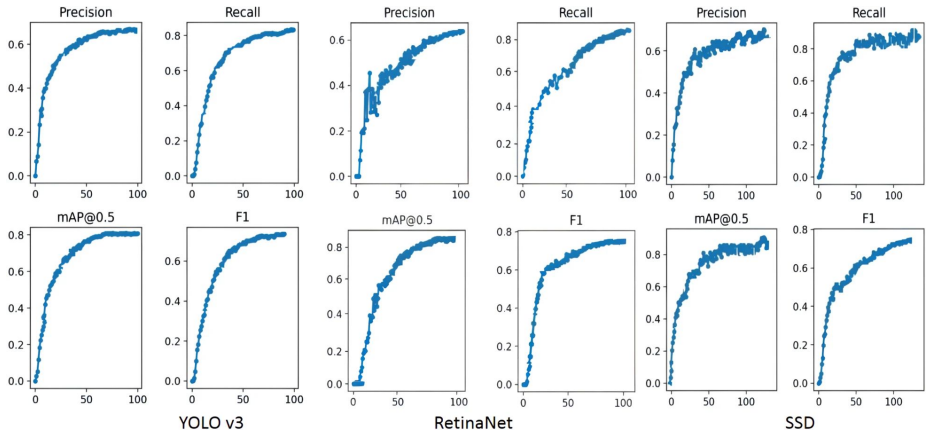
Projektbeschreibung

- Anwendung die Kaffeelisten automatisch auswertet und exportiert
- Nutzer macht Bild von Kaffeeliste und erhält Excel Sheet
- Auswertung geschieht durch trainiertes Machine Learning Modell

- Technische Anforderungen und Lösungen:
  - Namenserkennung (via Tesseract Library)
  - Objekterkennung (durch selbst trainiertes Modell)
  - Exportierung (mit Pandas Library)
  - GUI (durch Kivy Library)

Training von Objekterkennungsmodell (Research)

- Informationen über Objekterkennung & Modelle gesammelt
  - TF2 Detection Model Zoo & YOLO Modelle
  - Two-stage detectors (z.B. R-CNN Netzwerke) vs One-stage detectors (z.B. SSD)
  - Schnelligkeit vs Präzision
- Simples Projekt mit nur Bilderkennung und Fokus auf Mobile
  - ➔ YOLO, SSD oder RetinaNet
  - Entscheidung für SSD MobileNet 640x640 (ausbalanciert und geeignet für mobile)



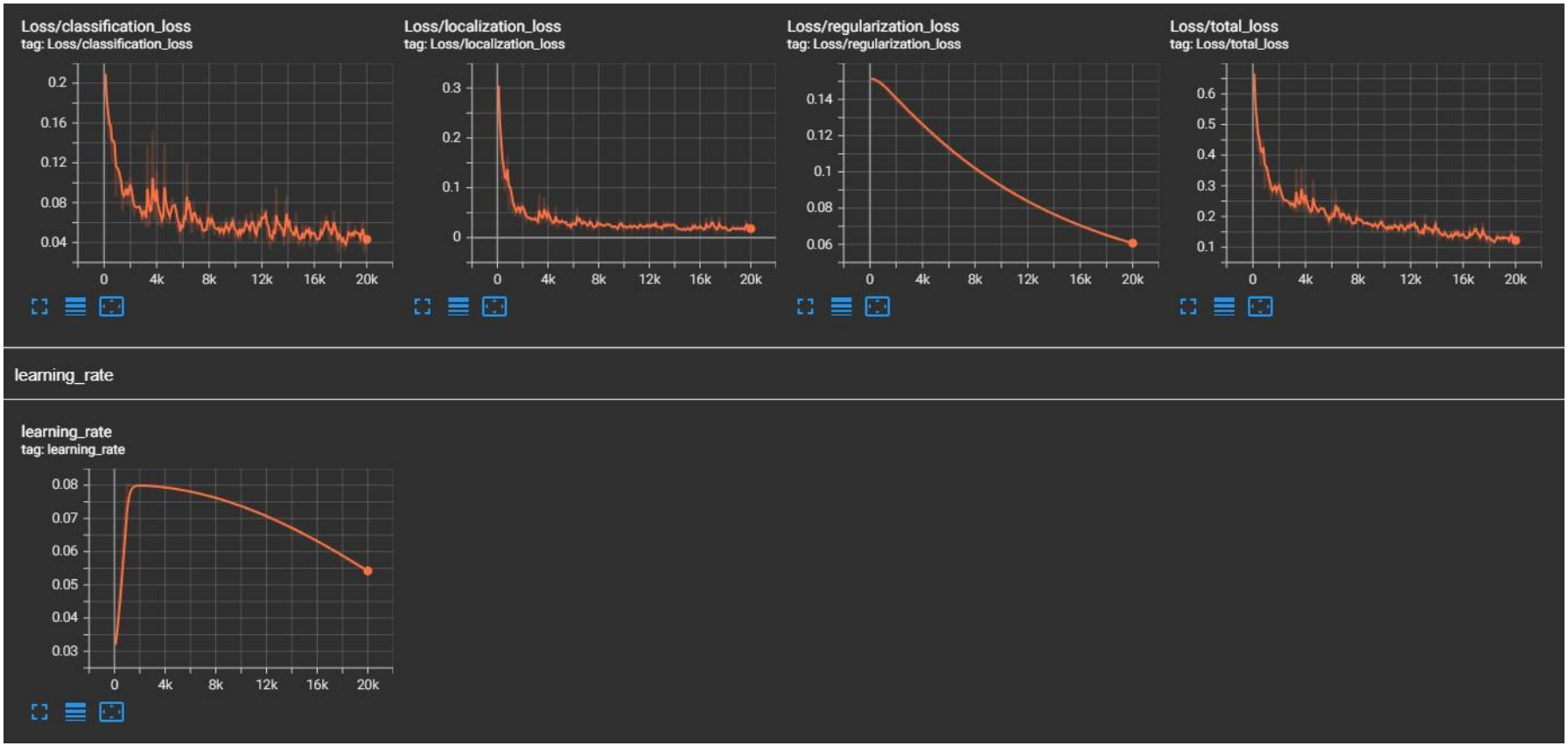
https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01691-8#Sec12

Training von Objekterkennungsmodell (Prozess)

- Setup und Installierung von diversen Bibliotheken in venv
  - Tensorflow Object Detection API
  - TF2 Model Zoo Models
  - CUDA + cuDNN von NVIDIA
- 6 verschiedene Modelle trainiert
  - Solide Erkennung von Blöcken (meist mit 100% Confidence)
  - Erkennung von einzelnen Strichen schwach
  - Datensatz mit jedem Training angepasst
  - Inkorrekte und ungenaue Labels verbessert

Training von Objekterkennungsmodell (Prozess)

Training von finalem Modell



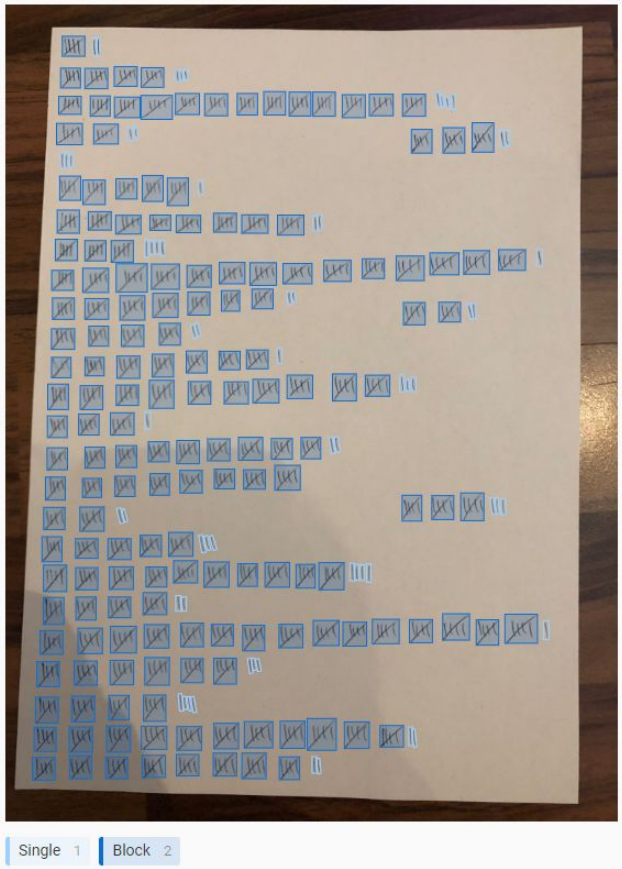
Entwicklungsprozess I – Datensatz

- Anlegen von eigenen Datensätzen mit möglichst vielen Variablen:
  - Papiere
  - Stifte
  - Hintergründe
  - Winkel
  - Beleuchtung
- Papiere mit möglichst vielen Strichen befüllen
- Unterschiedliche Größen und Ausrichtungen
- Verschiedene Entfernungen zueinander

Beispiel Datensatz für Einzelstriche

Entwicklungsprozess I – Labeling

- Verwendung von Label Studio für labeling
- Zwei verschiedene Labels
  - Single (Einzelne Striche)
  - Block (5er Blöcke)
- Export als Pascal VOC XML
  - Beliebtes Format für Objekterkennung und Bildsegmentierung



Beispiel labeled Datensatz

Entwicklungsprozess II – Namenserkennung

- Simple erkennung durch Tesseract Library
- Unterstützt keine Handschrift
- Viele überflüssige Daten
  - Begrenzung auf Alphabet & Leertaste
  - Confidence Rate
  - Durchschnittliche Position der Namen
- Exportierung der übrigen Daten in Liste
  - [Name, x, y, width, height, stroke count]
  - Vor & Nachname zusammengefügt (via y-Position)
  - Für spätere Verwendung bei Auswertung

Entwicklungsprozess II – Objekterkennung

- Importieren von trainierten Modell
  - config
  - label_map
  - aktuellster checkpoint

```
@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

- Simple Objekterkennung durch Hilfsmethode von Tensorflow
- Gibt bounding boxes, labels & confidence Wert zurück
- Erstellen von Objektliste mit ähnlicher Struktur wie Namensliste
  - [Label, x, y, width, height]

Entwicklungsprozess III – Auswertung & Exportierung

Auswertung

- Durch gleiche Struktur von Objekt- & Namenslisten sehr simpel
- Iterieren über alle Namen und vergleichen von y-Koordinaten
  - Objekt in Buffer Bereich von Name?
  - Objekt rechts von Name?
- Erhöhe stroke count in namensliste und gebe liste zurück

Exportierung

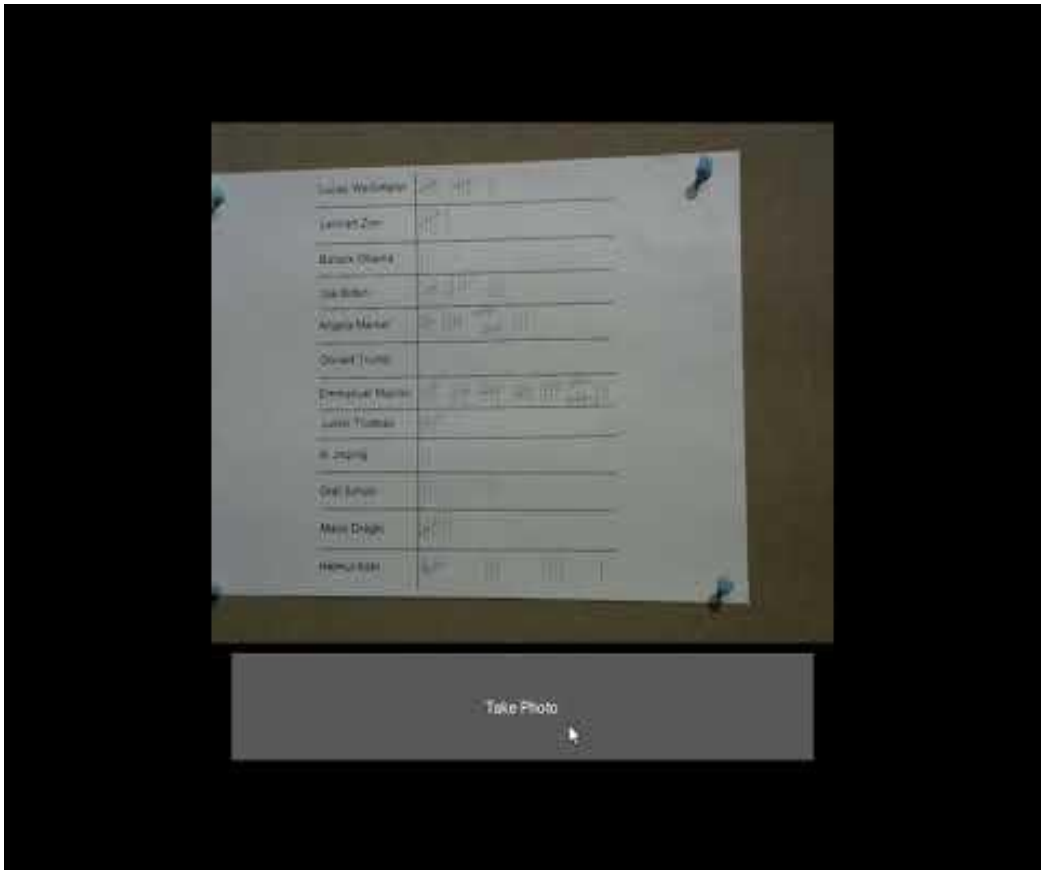
- Einfache Umsetzung durch Pandas
  - DataFrame von names array erstellt
  - Unnötige Daten entfernt (x, y, width, height)
  - Zu Excel exportiert

Entwicklungsprozess IV – Mobile-App

- Möglichkeiten für Entwicklung:
  - App in anderer Sprache entwickeln mithilfe von TFLite
  - Python Backend Server hosten
  - GUI per Python entwickeln
- Entscheidung für GUI via Python
  - Code stand bereits vollständig in Python
  - GUI steht nicht im Fokus, also eher simple Lösung suchen

- Kivy Library als Lösung (erlaubt Desktop + Mobile App Export)
  - Kamera Stream eingebaut um Bilder zu machen
  - Bestätigung des Bildes zum auswerten
  - Export als Excel Sheet, welches lokal gespeichert wird

App Demo



	A	B	C
1		**Names**	**Coffee count**
2	**0**	Lucas Weißmann	12
3	**1**	Lennart Zinn	6
4	**2**	Barack Obama	1
5	**3**	Joe Biden	14
6	**4**	Angela Merkel	23
7	**5**	Donald Trump	0
8	**6**	Emmanuel Macron	40
9	**7**	Justin Trudeau	5
10	**8**	Xi Jinping	3
11	**9**	Olaf Scholz	4
12	**10**	Mario Draghi	11
13	**11**	Helmut Kohl	17

Ausblick I – Probleme

- Tesseract unterstützt keine Handschrift
  - Lösung: User muss Computer gedruckte Namen verwenden
- Installation von Libraries, TF, CUDA, cuDNN war schwierig, erstes Training lief auf CPU, weil GPU lange nicht erkannt wurde
  - Lösung: Langes rumprobieren und neues aufsetzen von Virtual Environment
- Erkennung von einzelnen Strichen war mangelhaft
  - Lösung: Datensatz verbessert (false positives entfernt), aber noch unzufrieden
- Mobile Export hat lange zeit nicht geklappt wegen Windows, Libraries etc.
  - Lösung: WSL, Object Detection API in Projekt inkludiert, Dateien reduziert
- Probleme mit schlechten Inputs (Verschwommen/ungewöhnliche Einträge)
  - Lösung: Datensatz verbessern

Ausblick II – Verbesserungsmöglichkeiten

- Erkennung einzelner Striche
  - Eventuell waren Daten zu Abwechslungsreich, aber dafür zu wenig
  - Mehr konsistente & “saubere” Datensätze
  - False Positives wurden vielleicht nicht alle entdeckt beim prüfen
- App-Design
  - App nochmal in anderer Sprache entwickeln
  - Komplett das Design überarbeiten
- Mehr Optionen bieten
  - Export zu verschiedenen Tabellentypen bieten
  - Bildupload & Aufnahme anbieten

Ausblick III – Fazit

- Projekt war relativ gut gelungen, aber brauch noch sehr viel Arbeit am Datensatz & Design
- Größte Hürde war das aufsetzen der Libraries und das Modell zu trainieren
- Bei nächsten Projekten wird Training leichter und Zeit kann mehr auf Datensatz & Code konzentriert werden
- Viel gelernt über das trainieren von Modellen, wie viel nur kleine Fehler im Datensatz ausmachen und viel mehr
- Nächstes mal GUI nicht in Python programmieren, da das sehr limitiert ist
- Mit bisschen mehr Arbeit wäre die App jedoch nützlich, momentan noch nicht

Quellenverzeichnis

- **Github Projekt**
  - https://github.com/lwsm99/Coffeelist
- **Datensatz & Labeling**
  - https://labelstud.io/guide/
  - https://keymakr.com/blog/what-are-bounding-boxes/
  - https://albumentations.ai/docs/getting_started/bounding_boxes_augmentation/
- **Objecterkennung**
  - https://tesseract-ocr.github.io/tessdoc/
  - https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01691-8#Sec12
  - https://arxiv.org/ftp/arxiv/papers/1003/1003.5893.pdf
  - https://github.com/tensorflow/models/tree/master/research/object_detection
  - https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/
  - https://github.com/nicknochnack/TFODCourse
  - https://viso.ai/deep-learning/object-detection/
- **Daten Exportierung**
  - https://pandas.pydata.org/docs/
  - https://www.delftstack.com/de/howto/python-pandas/export-pandas-dataframe-to-excel-file/
- **Mobile-App**
  - https://realpython.com/mobile-app-kivy-python/
  - https://kivy.org/doc/stable/api-kivy.html