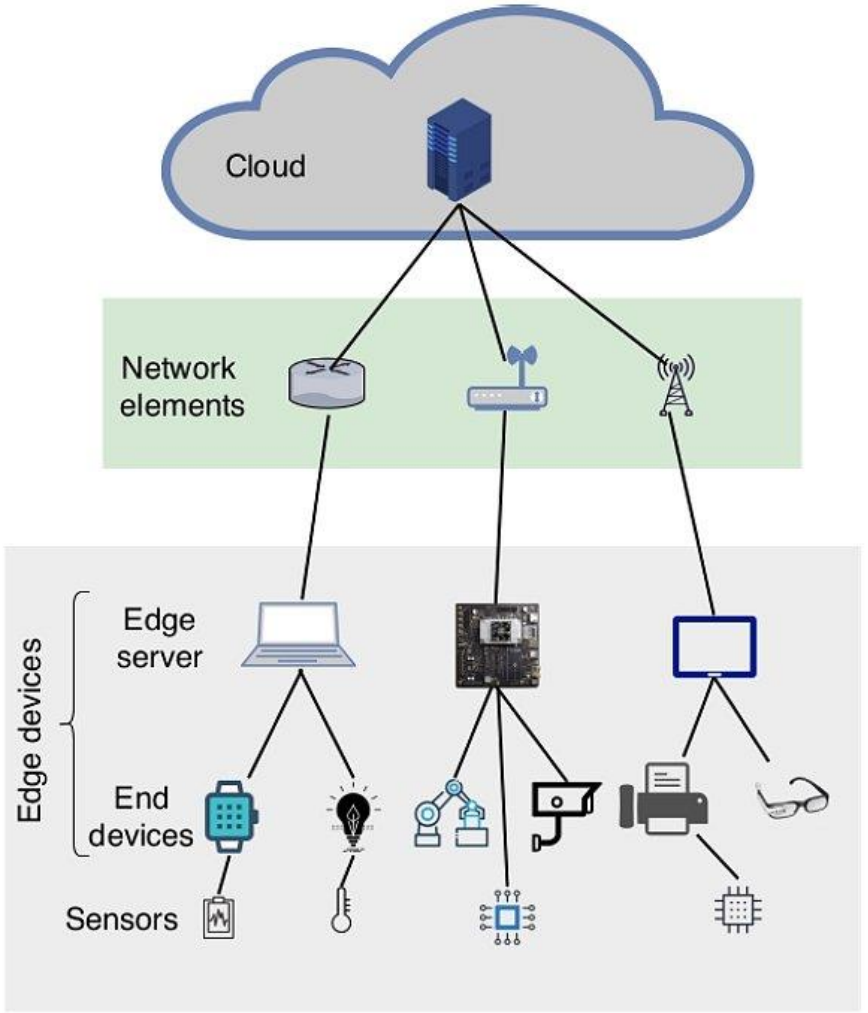
AI-on-the-edge-device

Sebastian Veigl

Cyber Physical Systems

Was ist Edge AI?

- „Intelligenz“ wir in Geräte am Rand des Netzwerks verlagert -> nah an Datengenerierung
- Sensoren etc. führen intelligente Datenanalyse selbstständig durch
- Ergebnisse werden per Kommunikationsschnittstelle zur Verfügung gestellt
- Training wird zentral durchgeführt



https://viso.ai/edge-ai/edge-ai-applications-and-trends/

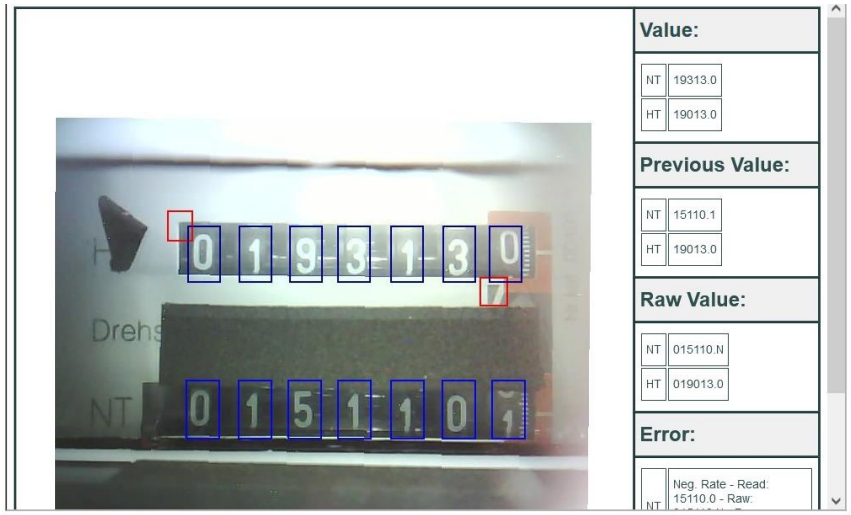
Vorteile / Nachteile von Edge AI

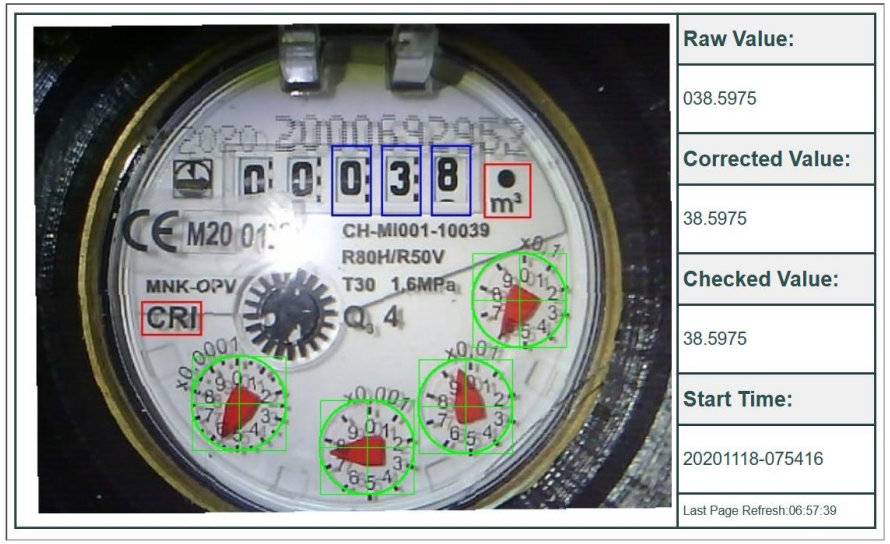
- **Vorteile**:
  - Latenzzeiten
  - Weniger Daten-Traffic
  - Sicherheit
  - Offline-fähig
  - Kosten
  - Energieverbrauch
- **Nachteile**:
  - Nur begrenzte Rechenleistung
  - Irreversibler Datenverlust durch Vorverarbeitung

AI-on-the-edge-device

- Open-source Repository zur Anwendung von Edge AI
- Ursprünglich: Analyse von Zählerständen auf Strom-/Wasser-/Gaszählern
- Auswertung von digitalen und analogen Anzeigeelementen

https://github.com/jomjol/AI-on-the-edge-device

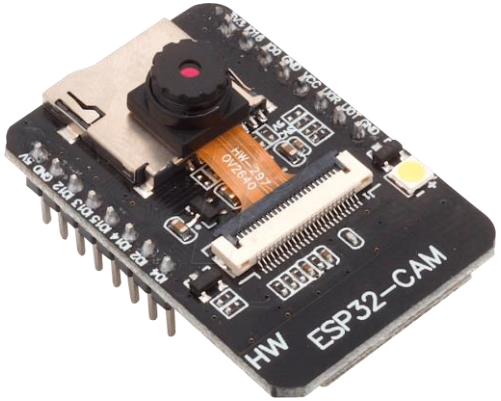




https://github.com/jomjol/AI-on-the-edge-device

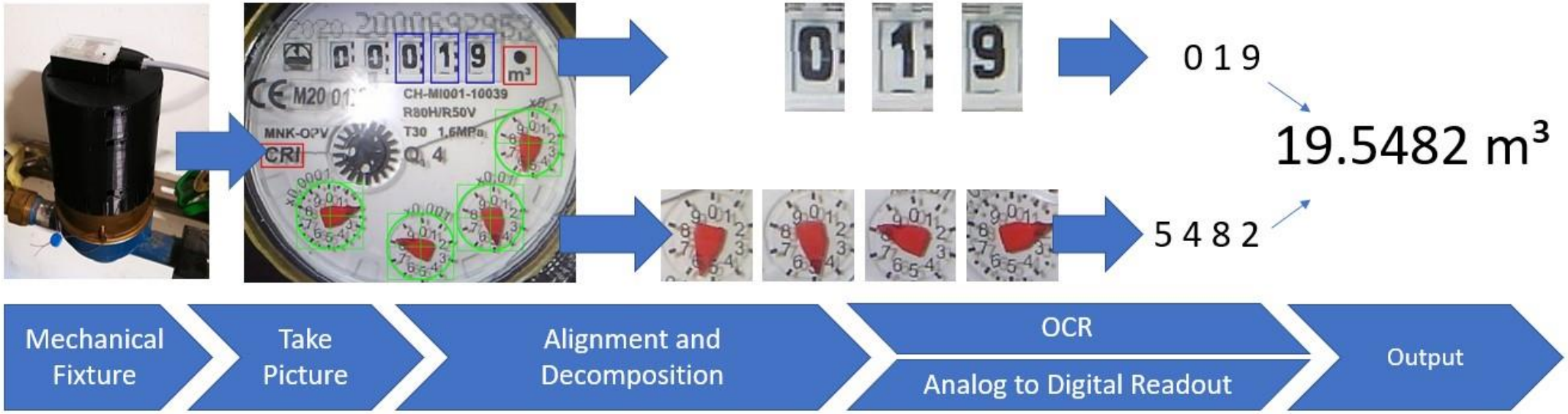
Hardware

- ESP32-Cam
  - ESP32
  - Kamera
  - SD-Kartenslot
  - Min. 4MB PSRAM
  - LED-Blitzlicht
- 5V Spannungsversorgung
- USB-TTL Adapter
  - zum Flashen der Firmware
- Halter



https://github.com/jomjol/AI-on-the-edge-device

Funktionsweise

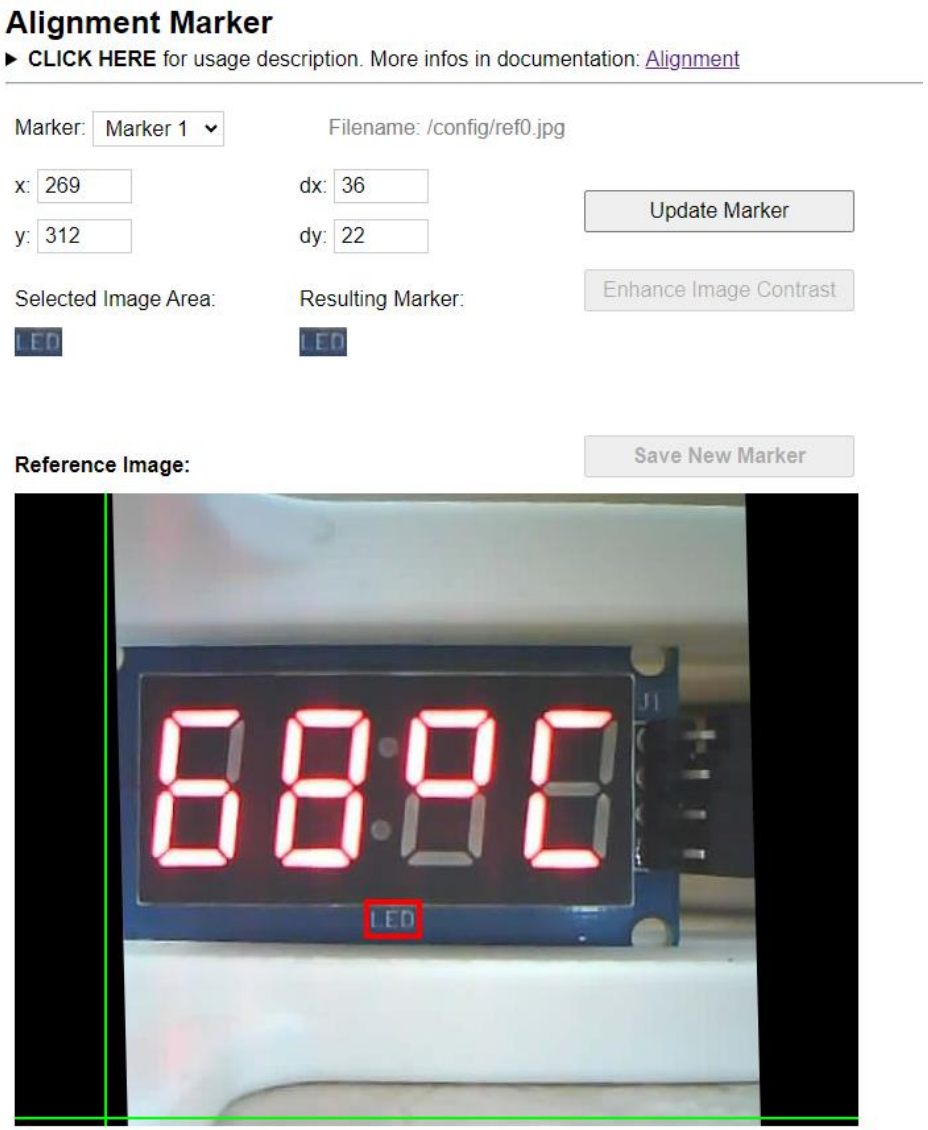


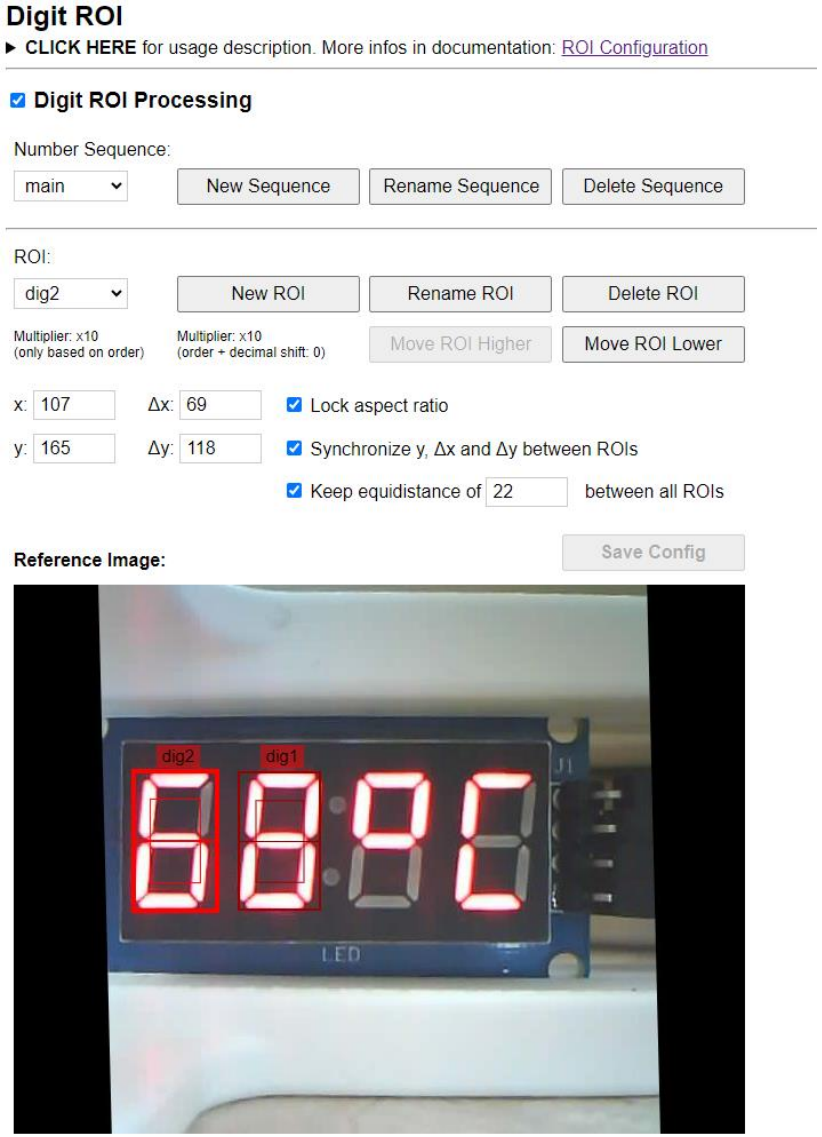
https://github.com/jomjol/AI-on-the-edge-device

Zusätzliche Features

- Tensorflow Lite Integration
- Webserver
- REST API
- MQTT
- Fileserver
- OTA-updatefähig
- …

Setup

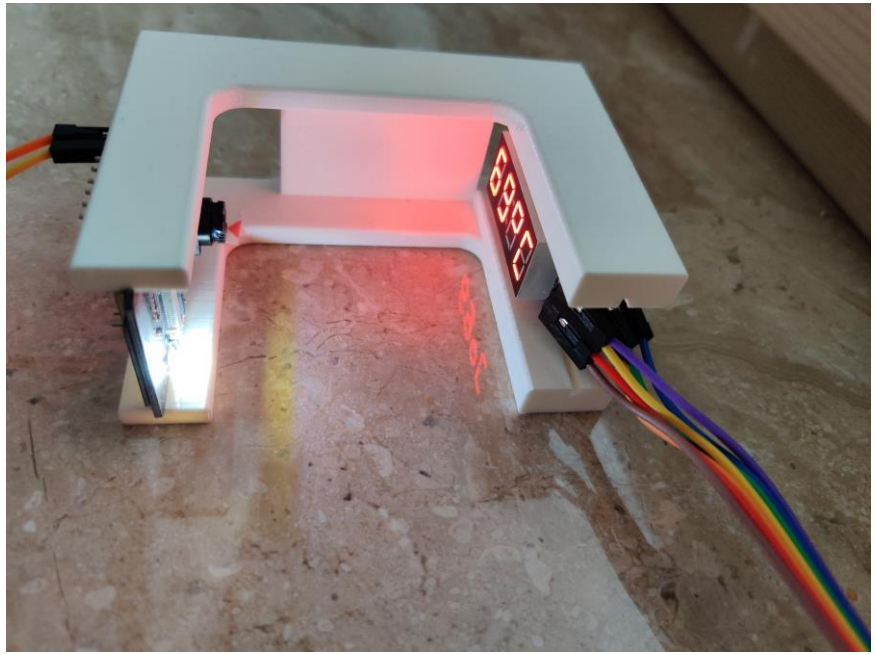

Alignment Marker
▸ CLICK HERE for usage description. More infos in documentation: Alignment
Marker: Marker 1
Filename: /config/ref0.jpg
x: 269
dx: 36
y: 312
dy: 22
Update Marker
Enhance Image Contrast
Selected Image Area:
Resulting Marker:
Save New Marker
Reference Image:


Digit ROI
▸ CLICK HERE for usage description. More infos in documentation: ROI Configuration
Digit ROI Processing
Number Sequence:
main
New Sequence
Rename Sequence
Delete Sequence
ROI:
dig2
New ROI
Rename ROI
Delete ROI
Multiplier: x10
(only based on order)
Multiplier: x10
(order + decimal shift: 0)
Move ROI Higher
Move ROI Lower
x: 107
Δx: 69
Lock aspect ratio
y: 165
Δy: 118
Synchronize y, Δx and Δy between ROIs
Keep equidistance of 22
between all ROIs
Save Config
Reference Image:
dig2
dig1

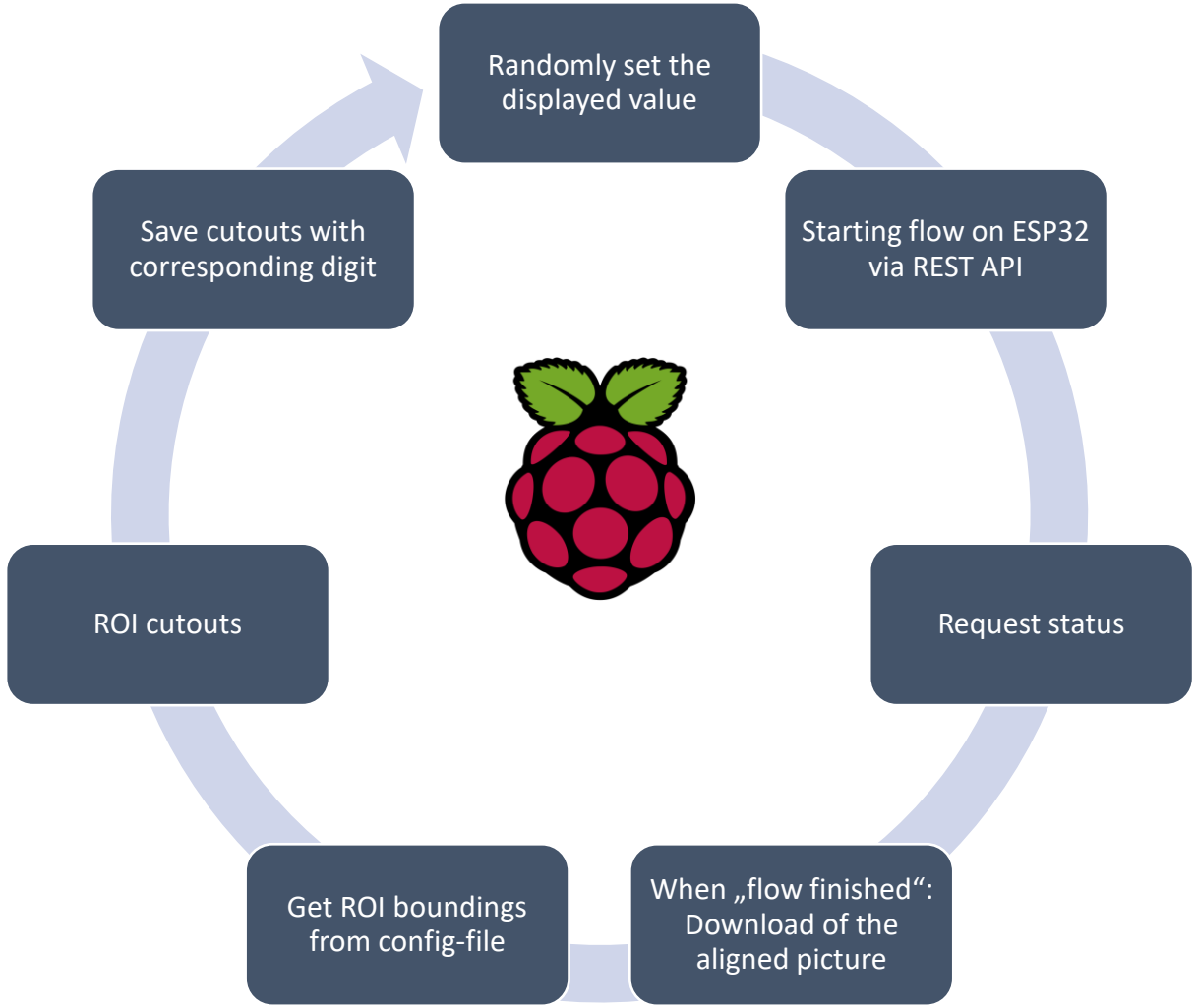
Versuche mit 7-Segment Display

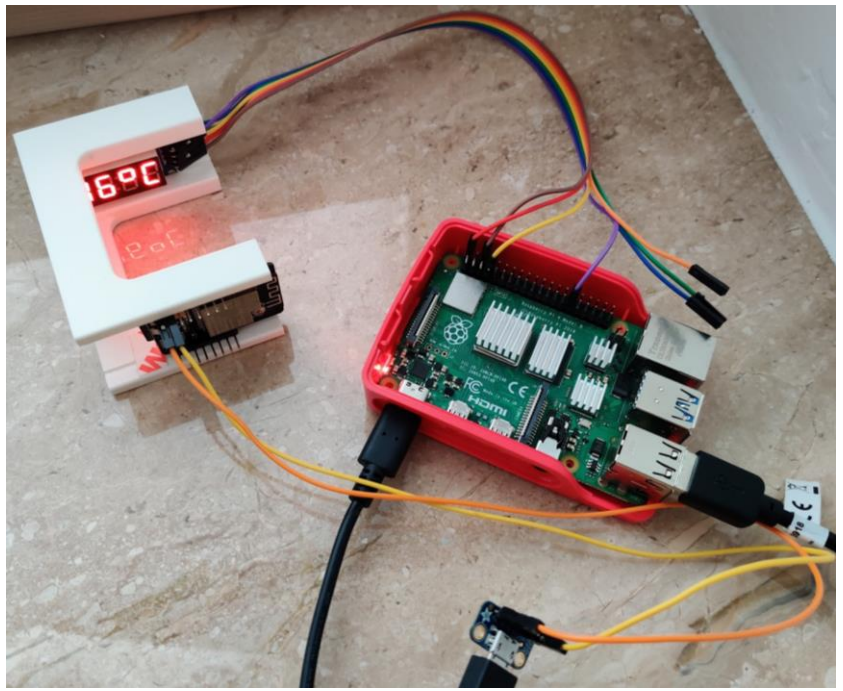
- Auswertung der angezeigten Temperatur auf einem 7-Segment Display mit I2C-Schnittstelle (TM1637)
- Mitgelieferte Modelle zu ungenau (~31%)

-> Training eines eigenen Modells

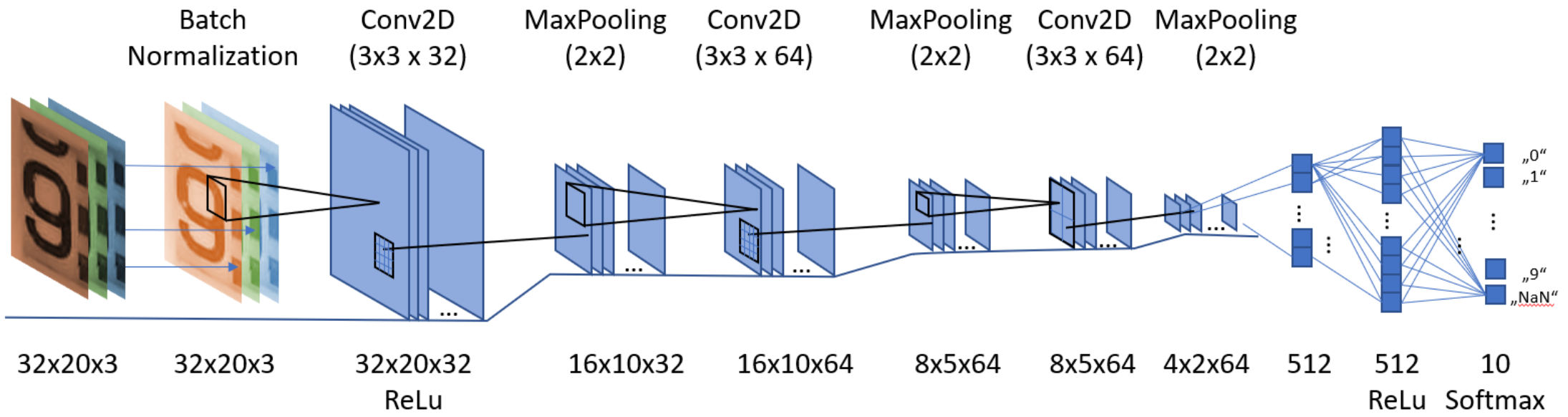


Trainingsaufbau





Aufbau des Netzes

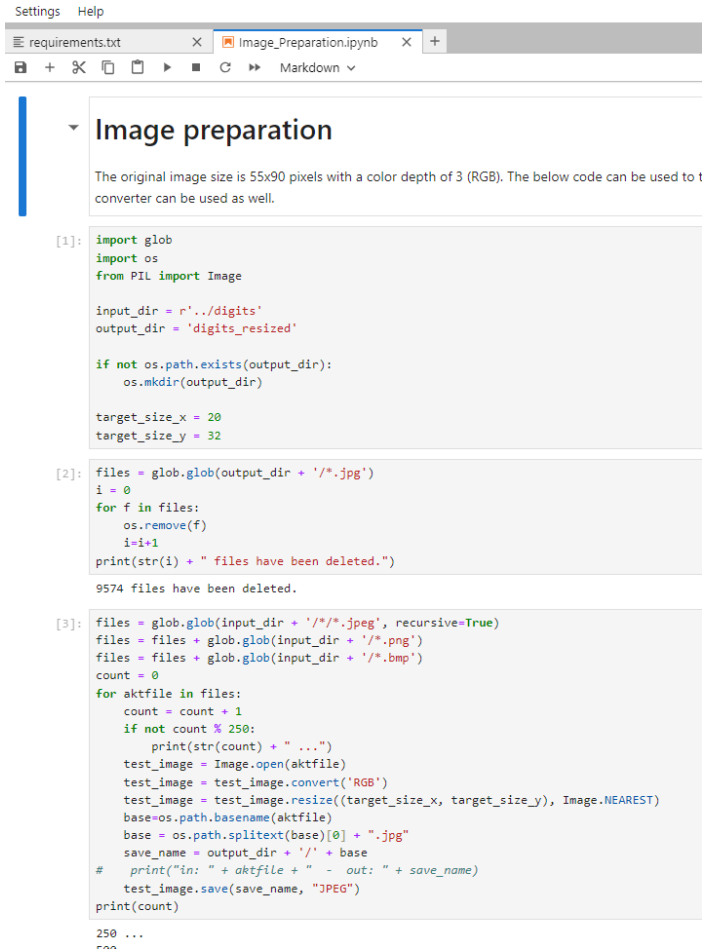


Starten der JupyterLab Umgebung

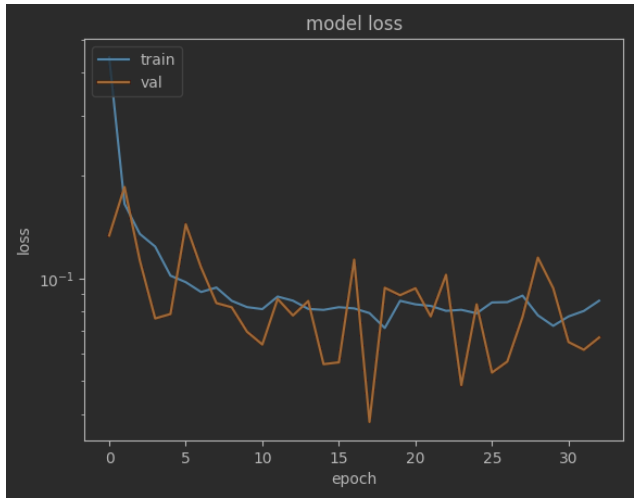
Zum Ausprobieren:

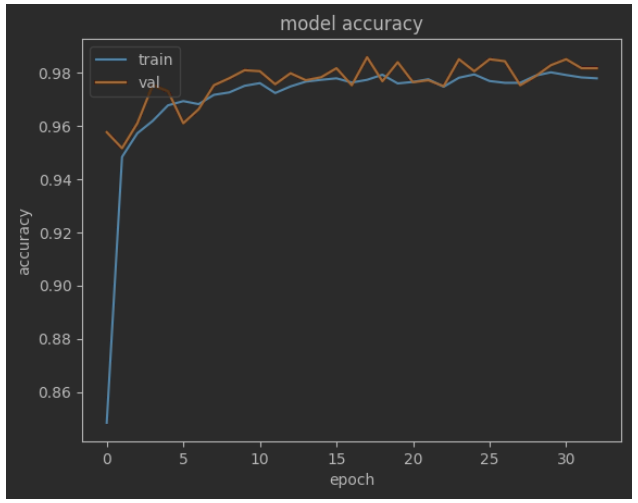
```
git clone https://github.com/SebastianVeigl/AI_on_the_edge_segment_train
cd AI_on_the_edge_segment_train/
pip install -r training/requirements.txt
```

```
jupyter-lab training
```

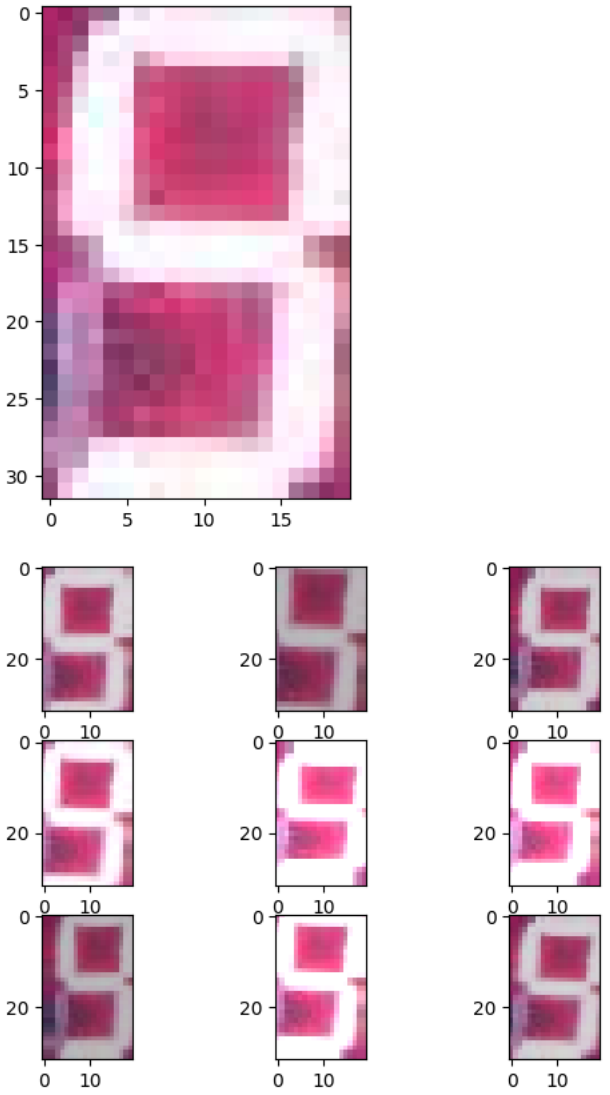


Training des neuralen Netzes





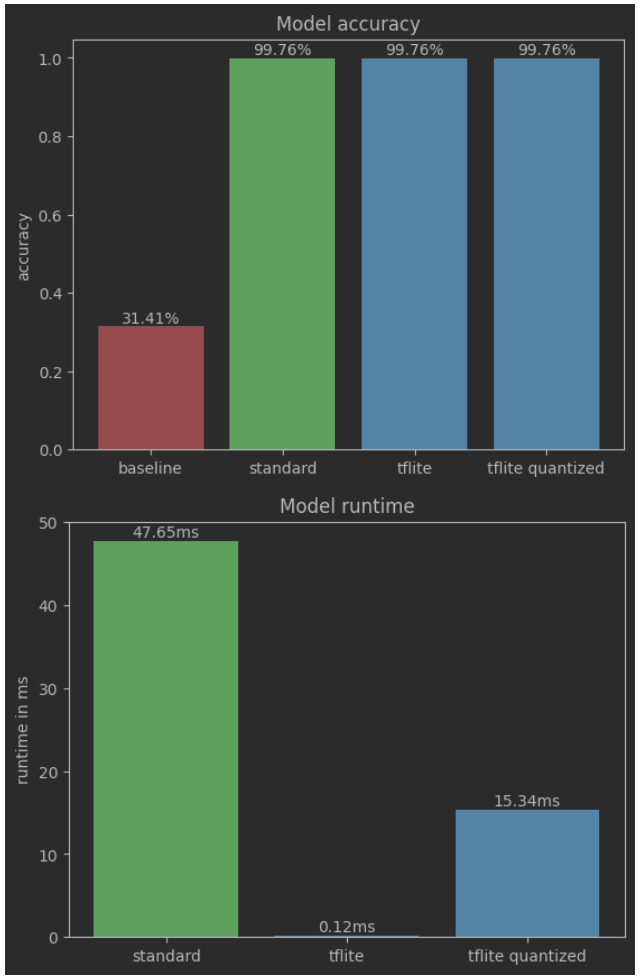
- Training auf dem Laptop
- Vorverarbeitung:
  - ROIs zu 32x20 Pixel ([Jupyter Notebook])
  - Augmentation
- Training
  - *Tensorflow* für den Aufbau + Training des Netzes
  - Train-Val-Split (70%/30%)
  - [Jupyter Notebook]
  - Umwandlung in TFLite
  - Post-training quantization

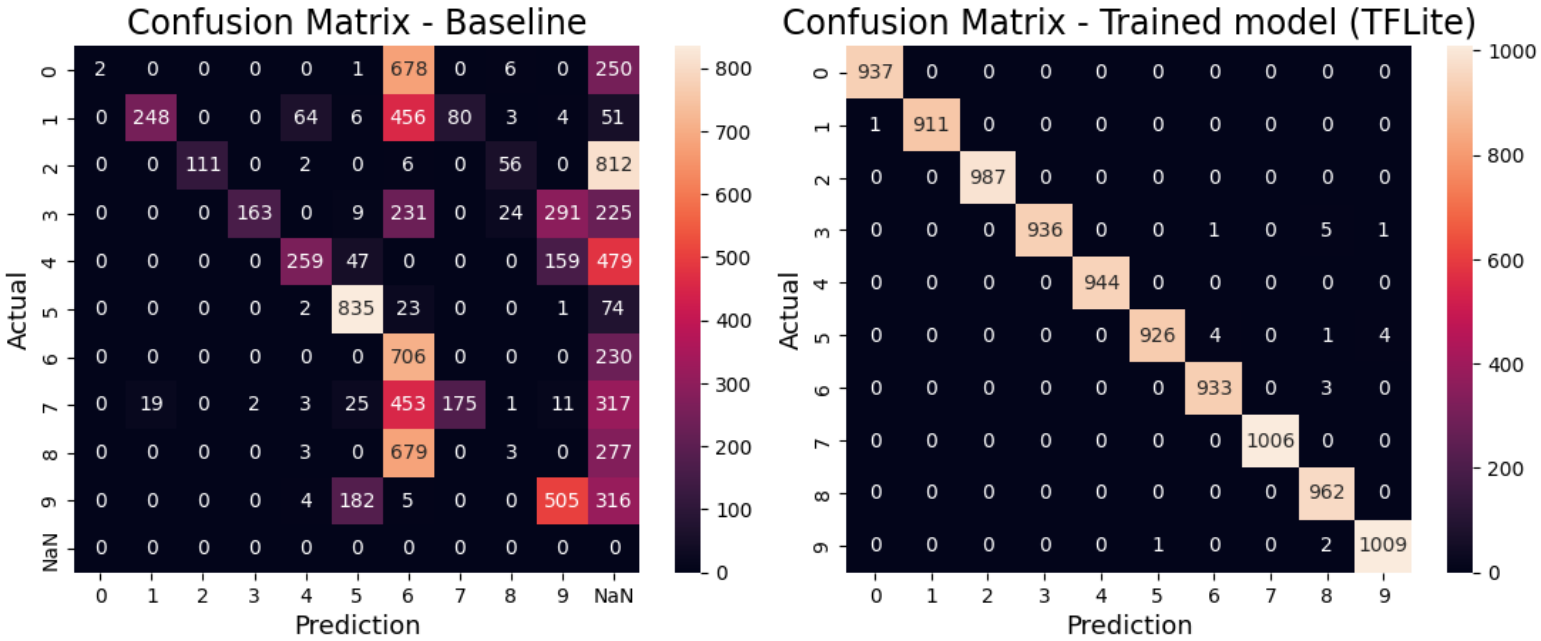


Sichern des Netzes

- Standard Format (.keras)
  - Ermöglicht Training + Inferenz
- TFLite
  - Nur Training
  - Geringere Laufzeit
  - Geringere Größe
  - Weniger RAM-Verbrauch
- TFLite (Post-training quantization)
  - Wie TFLite
  - Alle Gewichte + Aktivierungsfunktionen werden in Integer umgewandelt (oder andere Variablentypen z.B. float64)
  - Geringere Genauigkeit

Laufzeit/Genauigkeit der Modelle





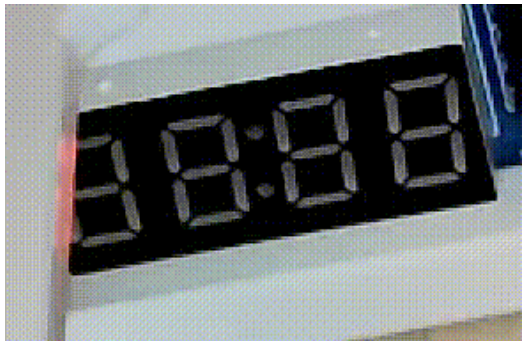
Ergebnisse

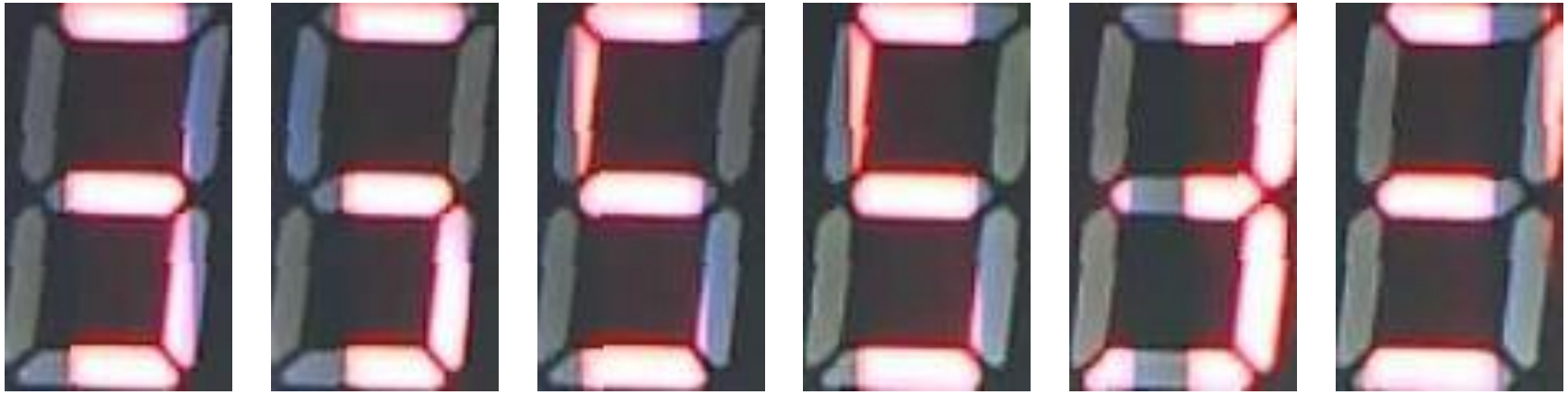
	Baseline (TFLite)	Standard model (keras)	TFLite	TFLite quantized
Size	349KB	692KB	215KB	62KB
Accuracy	31,43%	99,76%	99,76%	99,76%
Runtime	-	47,6 ± 7,9ms	0,12 ± 0,01ms	15,3 ± 3,3ms

Probleme bei der Klassifizierung

- Bei hoher Sonneneinstrahlung bilden sich Streifen auf den Bildern

-> Nicht mehr eindeutig





Inferenz auf anderen Geräten

z. B.: Raspberry Pi mit Python

- Modell im TFLite-Format
  - Installation von *tflite_runtime* (Python-Packet) mit *pip3 install tflite-runtime*
  - Siehe Code zum Ausführen der Klassifikation eines Bildes

Am PC kann auch direkt das *tensorflow.lite* Subpacket verwendet werden

```python
import tflite_runtime.interpreter as tflite
import numpy as np
from PIL import Image

TFLITE_FILE_PATH = '../models/7seg2912.tflite'

# Load the TFLite model in TFLite Interpreter
interpreter = tflite.Interpreter(TFLITE_FILE_PATH)
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

image_in = Image.open('../training/digits_resized/7_19_12_2023_18_41_26.jpg')
test_image = np.array(image_in, dtype="float32")
img = np.reshape(test_image, [1, 32, 20, 3])

# Test the model on random input data.
input_shape = input_details[0]['shape']
interpreter.set_tensor(input_details[0]['index'], img)

interpreter.invoke()

# The function `get_tensor()` returns a copy of the tensor data.
# Use `tensor()` in order to get a pointer to the tensor.
output_data = interpreter.get_tensor(output_details[0]['index'])
print(output_data)
print(output_data.argmax())
```

Quellen

- https://www.computacenter.com/de-de/what-we-do/cloud-and-data-center/ki/ai-the-edge
- https://viso.ai/edge-ai/edge-ai-applications-and-trends/
- https://github.com/jomjol/AI-on-the-edge-device
- https://jomjol.github.io/AI-on-the-edge-device-docs/
- https://github.com/jomjol/neural-network-digital-counter-readout/
- https://www.tensorflow.org/lite/
- https://www.tensorflow.org/lite/performance/post_training_quantization