

Wilhelm-Ostwald-Schule, Gymnasium der Stadt Leipzig

Dokumentation zur Besonderen Lernleistung

Im Fachbereich: Informatik

Thema: Untersuchung zur KI-basierten Objekterkennung auf
eingebetteten Systemen

Vorgelegt von: Merten Jakobza

Schuljahr: 2023/24

Externer Betreuer: Prof. Dr.-Ing. Axel Klarmann
Hochschule für Technik, Wirtschaft und Kultur Leipzig
Fakultät Digitale Transformation

Interner Betreuer: Robert Karl

Leipzig, 21.12.2023

Abstract

Der erste Schritt in die Programmierung erfolgt bei den meisten Informatikeinstiegern mit Scratch oder ähnlichen benutzerfreundlichen beziehungsweise grafischen Programmiersprachen. Jedoch wird für diesen Einstieg ein Computer benötigt. Genau dies soll mit dieser besonderen Lernleistung “Untersuchung zur KI-basierten Objekterkennung auf eingebetteten Systemen” geändert werden.

Schüler sollen verschiedene Programmierblättchen anordnen. Diese geben durch die abgebildeten Formen, wie Richtungspfeile oder Schleifensymbole, sowie ihre Reihenfolge und logische Zusammensetzung einem Roboter Anweisungen, um beispielsweise durch ein Labyrinth zu gelangen. Dies erfolgt durch die Aufnahme eines Fotos mit einer Kamera auf einem eingebetteten System. Die Blättchen auf dem Foto werden mit einem neuronalen Netzwerk erkannt und deren Anordnung interpretiert. Letztendlich wird dies in Befehle umgewandelt, damit der Roboter sich bewegen kann. Die künstliche Intelligenz (KI) wird dabei mit Hilfe eines kostenlosen Dienstes namens “Edge Impulse” (siehe Kapitel 4) trainiert.

Dies ermöglicht das Heranführen an die Programmierung allein mit der Hilfe des relativ preiswerten eingebetteten Systems und koppelt das Lernen somit vollständig von Computern ab.

Insgesamt ist das Ziel der BeLL, eine Künstliche Intelligenz zu trainieren, diese auf verschiedenen eingebetteten Systemen unterschiedlicher Preisklassen zu testen und einen Schluss bezüglich eines optimalen Preis-Leistung-Verhältnisses zu ziehen.

Inhaltsverzeichnis

1 Einleitung und Zielstellungen	1
2 Grundbegriffe	2
2.1 Eingebettetes System	2
2.2 Deep Learning	2
2.3 Metriken des Machine Learnings	3
3 Setup - Genutzte Hardware	5
3.1 Raspberry Pi 3 A+	5
3.1.1 Installatiton von PiOS	5
3.1.2 Aufnahme eines Bildes mit der Kamera	6
3.2 ESP-32-CAM	7
3.2.1 Laden einer Firmware in den Flash	8
3.3 M5Stack UnitV	9
3.3.1 MaixPy und Flashen einer Firmware	10
3.4 Theoretischer Vergleich	10
4 Nutzung von Edge Impulse	11
5 Deployment auf den eingebetteten Systemen	15
5.1 Raspberry Pi	15
5.1.1 Installation	15
5.1.2 Nutzung des Modells	15
5.2 AI Thinker ESP-32-CAM	17
5.2.1 Edge Impulse Firmware	17
5.2.2 MicroPython und TensorFLowLite	17
5.2.3 C++-Bibliothek	18
5.2.4 Arduino-Bibliothek	18
5.3 UnitV mit K210	19
5.3.1 Umwandlung eines TFLite-Modells in ein KModel und Flashen	19
5.3.2 Beispielprogramm	20
6 Data-Augmentation-Tool	21
6.1 Nutzung	21
6.2 Programmierung	22
6.2.1 Programmierung - Hintergrundveränderung	22
6.2.2 Programmierung - Helligkeit- und Kontrastveränderung	23
6.2.3 Programmierung - Perspektivische Verzerrung	23
6.2.4 Programmierung - Unschärfe-Filter	24
6.2.5 Programmierung - Generation JSON-Datei	24
6.2.6 Programmierung - Dateienverschiebung	25
7 Untersuchungen mit finaler KI	25

8 Finales Projekt / Auswertungsprogramm	27
8.1 Nutzung	27
8.1.1 Nutzung - Allgemeine Hinweise	28
8.1.2 Nutzung - Erklärung der Blättchen	28
8.1.3 Nutzung - Labyrinth-Editor	30
8.1.4 Nutzung - Hauptprogramm	31
8.2 Programmierung	32
8.2.1 Programmierung - Labyrinth-Editor	32
8.2.2 Programmierung - Pathfinding-Algorithmus	33
8.2.3 Programmierung - Klassifikation	33
8.2.4 Programmierung - Umwandeln der Bounding-Boxen in 2D-Darstellung des Programms	34
8.2.5 Programmierung - Simulation des Programmes	36
9 Ergebnisse	38
10 Erweiterungsmöglichkeiten	39

Abbildungsverzeichnis

1	Funktionsweise des FOMO-Modells	3
2	Metriken des Machine Learnings	4
3	Raspberry Pi	5
4	ESP-32 CAM	7
5	M5Stack UnitV	9
6	Edge Impulse -Dashboard	11
7	Edge Impulse -Data Aquisition	12
8	Edge Impulse -Gestalten des Impulses	12
9	Edge Impulse -Training des Modells	13
10	Edge Impulse -Live Classification	14
11	Edge Impulse -Model Testing	14
12	Verschobene Ergebnisse bei 250x250 Pixel Bild	16
13	Richtige Ergebnisse bei 224x224 Pixel Bild	16
14	Nutzung der kflash-GUI	20
15	Zahlen mit untersch. Farben	25
16	Genauigkeit vor der Farbenänderung der Zahlen	26
17	Genauigkeit nach der Farbenänderung der Zahlen	26
18	Zahlen mit untersch. Graustufen	26
19	F1-Score der Finalen KI	26
20	Zielerkennung	28
21	Beispiel-Programm	29
22	Beispielprogramm-Schleife mit Sub	30
23	Beispielprogramm-Schleife mit Methodenaufruf	30
24	Labyrinth-Editor nach Start	30
25	Labyrinth-Editor	30
26	Hauptseite des Programms nach Editor	31
27	Hauptseite des Programms nach Klassifikation	31
28	Hauptseite des Programms nach Simulation	31
29	Sub-Blättchen	35
30	Blättchen mit eckigen Konnektor	35
31	Beispiel Verkettung von Konnektoren	36
32	Blink-Blättchen	i
33	C01-Loop-Blättchen	i
34	C02-Wait-Blättchen	i
35	C01-If-Blättchen	i
36	CF-A-Blättchen	i
37	CF-Sub-Blättchen	i
38	DF-A-Blättchen	i
39	M01-Forward-Blättchen	i
40	M02-Backward-Blättchen	i
41	M03-Left-Blättchen	i
42	M04-Right-Blättchen	i

43	N01-Blättchen	i
44	Not-Blättchen	ii
45	S01-Start-Blättchen	ii
46	S02-End-Blättchen	ii
47	S04-Return-Blättchen	ii
48	W01-Wand-Blättchen	ii
49	Ziel-Blättchen	ii

Tabellenverzeichnis

1	Technische Daten Raspberry Pi 3 Modell A+	5
2	Technische Daten AI Thinker ESP 32	8
3	Technische Daten UnitV	9

Verzeichnis der Quellcodes und Commands

1	Installation d. Webservers über SSH (WinTotal.de , 01.04.2019)	6
2	Herausfinden der IP-Adresse (roshalmoraes , 10.05.2023)	6
3	Hilfreiche Befehle (TutorialsForRaspberryPi , o.D.)	6
4	Installation von PiCamera	7
5	Nutzung von PiCamera (Rosebrock , 20.03.2015)	7
6	Bauen / Flashen einer Firmware in den Flash	9
7	Thonny installieren	9
8	Installation Edge Impulse CLI (Edge Impulse Inc. , 14.09.2023)	15
9	Beispielprogramm für Nutzung des Modells (Edge Impulse Inc. , 14.09.2023) . .	16
10	Beispielprogramm zur Ausführung eines TensorFlow-Lite-Modells mit Micropython	17
11	Änderung an Beispielprogramm Arduino	19
12	Änderungen in Platformio-Einstellungen	19
13	Umkompilieren in ein .kmodel	19
14	Beispielprogramm für die Verwendung eines kmodels	20
15	Beispielparameter für die Nutzung des Tools	22
16	Hintergund-Austausch	22
17	Kontrast-/Helligkeitveränderung	23
18	Perspektivische Verzerrung	24
19	Unschärfe-Filter	24
20	Beispiel-JSON-Datei (Edge Impulse Inc. , 28.08.2023)	24
21	Speichern der JSON-Datei	24
22	Erstellen des Ordner / Verschieben der Bilder	25
23	Installation PyQt5	27
24	Beispiel-PyQt5-Programm	27
25	Installation von Edge-Impulse-Linux	27
26	Button-Klasse für Labyrinth-Editor	32

27	Label-Klasse für Hauptprogramm	33
28	Code zum Darstellen von Bildern in der GUI	34
29	Ergebnis der Klassifikation	34
30	Ergebnis-Liste	36

1 Einleitung und Zielstellungen

Die Welt ist von künstlicher Intelligenz (KI) nachhaltig geprägt. Bereits 13,3 Prozent der deutschen Firmen nutzen künstliche Intelligenz und knapp 46 Prozent diskutieren zumindest über ein zukünftigen Einsatz vgl. (Wolf , 02.08.2023). KI wird bereits von 68 Prozent der Schüler genutzt um Hausarbeiten zu schreiben oder sich die Schulaufgaben erklären zu lassen vgl. (Zeit-Online , 06.11.2023). Künstliche Intelligenzen können dabei nicht nur mit Texten arbeiten, sondern auch Objekte in Bildern erkennen. In dieser besonderen Lernleistung soll mit Hilfe von KI für Bildverarbeitung eine günstige Version des Programmiererstiegs erstellt werden. Das soll unabhängig von einem vollwertigen Computer geschehen, indem eingebettete Systeme genutzt werden. Es werden Blättchen im physischen Leben zu einem Programm zusammengelegt und abfotografiert. Als Anwendungsbeispiel soll das vorliegende Programm einen Roboter durch ein Labyrinth manövrieren.

Ziel dieser Arbeit ist die Entwicklung einer Applikation mit grafischer Benutzeroberfläche. Sie soll gelegte Programme aus Blättchen erkennen und Möglichkeiten für das Testen anbietet. Gleichzeitig werden verschiedene eingebettete Systeme betrachtet. Es soll die Erkennungsrate und -geschwindigkeit, sowie Benutzerfreundlichkeit von Raspberry Pi, ESP-32-CAM und M5Stack UnitV verglichen werden. Das erfolgt zunächst anhand theoretischer Aspekte (Prozessor-, Speichergeschwindigkeit usw.), gefolgt von der konkreten Umsetzung. Zum Trainieren der KI soll Edge Impulse (siehe Kapitel 4) genutzt werden.

2 Grundbegriffe

2.1 Eingebettetes System

Eingebettete Systeme sind leistungsschwache Computer-Systeme, die häufig in größere System eingebunden sind. Jedoch können sie auch einzeln genutzt werden. Beispielsweise werden eingebettete Systeme in Bankautomaten, Routern und Mikrowellen genutzt vgl. (Awati , 10.2023). Oft handelt es sich um Einplatinencomputer. Das heißt alles, was der Computer zum Funktionieren braucht, ist auf einer Leiterplatte integriert. Beispielsweise sind RAM, Speicher und IO-Controller meist direkt mit dem Prozessor auf einem Chip verbunden vgl. (ITWissen.info , 09.10.2017).

Die hier genutzten Systeme sind alleinstehende Computersysteme, welche nur eine Stromverbindung zum Funktionieren brauchen und nicht unbedingt in ein größeres System integriert werden.

2.2 Deep Learning

'Deep Learning (tiefes Lernen) ist ein Teilgebiet von maschinellem Lernen, welches sich auf künstliche neuronale Netze und große Datenmengen fokussiert. Deep Learning wird dazu genutzt, Bilder zu erkennen, Texte zu verstehen und Entscheidungen genauer zu tätigen.' Wuttke (2023). Hierbei werden sogenannte Raw-Features (unklassifizierte Eigenschaften) genutzt, welche beispielsweise bei einem Bild die RGB-Werte der Pixel sind und in Classified-Features umgewandelt, welche das Ergebnis darstellt. Im Falle der Objekterkennung von einem Bild ist das Ergebnis beispielsweise die erkannten Objekte.

Wichtig zu unterscheiden ist Image Classification und Object Detection. Während die Image Classification jedem Bild nur ein Ergebnis zuordnet, können bei der Object Detection auf einem Bild mehrere Objekte gefunden werden.

Beim Lernen wird das neuronale Netz basierend auf dem erhobenen Datensatz angepasst. Dies kann 'überwacht' und 'unüberwacht' geschehen. Überwachtes Lernen nutzt dabei gekennzeichnete Daten, bei denen das Ergebnis angegeben ist. Beim Lernen wird das Ergebnis der künstlichen Intelligenz mit dem des Datensatz abgeglichen und Änderung am neuronalen Netzwerks vorgenommen. Beim unüberwachten Lernen, bekommt das zu trainierende Modell keine Ergebnisse. Hier wird ein Algorithmus genutzt, der das Modell selbst trainieren lässt auch als 'Clustering' bezeichnet vgl. (Ines Bahr , 13.12.2021). Edge Impulse nutzt dabei eine Kombination aus beidem vgl. (Edge Impulse Inc. , 22.05.2023).

Edge Impulse nutzt für die Object Detection das sogenannte FOMO-Modell, was für 'Faster Objects, More Objects' vgl. (Edge Impulse Inc. , 06.09.2023) steht. Zum Erkennen der Objekte wird das Original-Bild in 8x8 Pixel große Teilbilder aufgeteilt, welche jeweils einzeln klassifiziert werden. Nebeneinanderliegende Teilbilder, welche gleich klassifiziert wurden, werden zu einem Objekt zusammengefügt. Dieser Vorgang ist in Abbildung 1 veranschaulicht. Vorteil dieser Herangehensweise ist die sehr niedrige Erkennungszeit im Vergleich zu anderen Ansätzen. Nachteil ist jedoch, dass dieses Modell nicht erkennen kann, ob ein Objekt von einem anderem Objekt verdeckt ist. Ist beispielsweise ein Auto von einem anderem Objekt in der Mitte vollständig verdeckt, werden die Enden des Autos vom FOMO-Modell als zwei unterschiedliche Autos

aufgefasst, da die Zellen mit den gleichen Ergebnissen sich nicht mehr berühren. Es wird also die Geschwindigkeit auf Kosten der Genauigkeit verbessert.

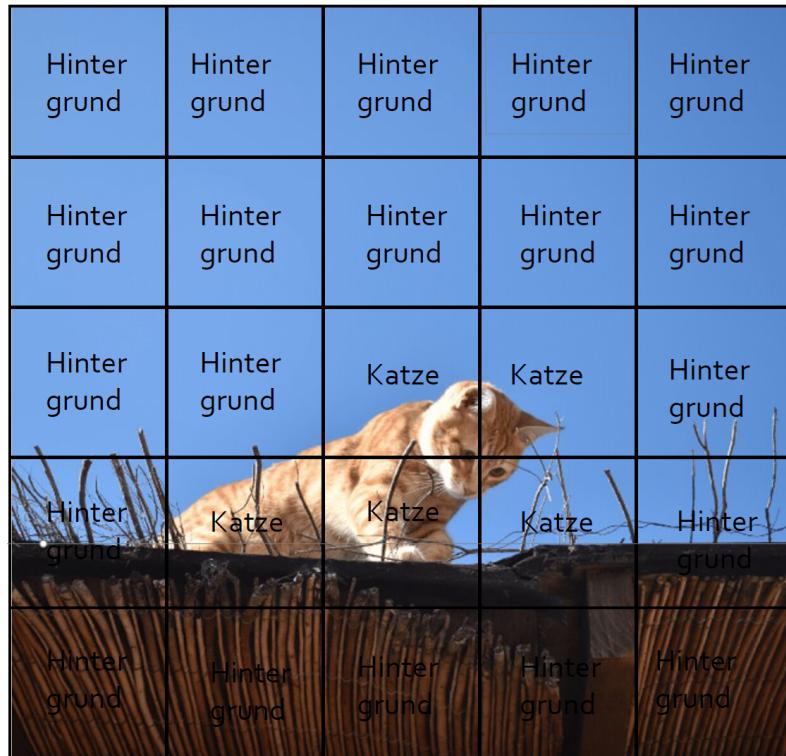


Abb. 1: Funktionsweise des FOMO-Modells

Eine weitere Möglichkeit für die Verbesserung der Geschwindigkeit ist die Quantisierung. Es werden dabei die 32-Bit Gleitkommazahlen in 8-Bit Ganzzahlen gerundet, welche schneller verarbeitet werden können. Gleichzeitig braucht auch jede 8-Bit Zahl weniger Speicherplatz als 32-Bit Zahlen, wodurch die Modellgröße reduziert wird. Somit können auch komplexe Modelle auf leistungsschwachen Systemen laufen. Dies reduziert jedoch auch die Genauigkeit dieser Argumente und somit des Modells vgl. (Edge Impulse Inc. , 10.11.2023).

Edge Impulse nutzt außerdem eine Technologie namens 'Transfer Learning'. Dabei wird ein vortrainiertes Modell lediglich auf ein neues und verwandtes Problem angepasst. Dies ermöglicht deutlich kürzere Trainingszeiten und spart Ressourcen vgl. (Edge Impulse Inc. , 07.09.2022).

2.3 Metriken des Machine Learnings

Von Edge Impulse werden 3 Metriken genutzt um dem Nutzer die Genauigkeit, Sensitivität und eine Gesamteinschätzung einer trainierten KI zu geben. Hierzu wird 'Precision' für die Genauigkeit und 'Recall' für die Sensitivität genutzt. Wie in Abbildung 2 zu sehen ist, ist Precision der Quotient aus der Anzahl der 'True Positives' und der Summe aus der Anzahl der 'True Positive' und der Anzahl der 'False Positive' vgl. (Kundu , 16.12.2022). Eine hohe Genauigkeit wird dadurch erreicht, dass keine Objekte gefunden werden, die nicht gesucht werden. Sie ist also 100 Prozent wenn alle erkannten Objekte, richtig erkannt wurden. Wird nur die 'Precision' genutzt,

könnte die KI nur wenige Objekte erkennen. Sind diese Objekt jedoch richtig erkannt, ist die Precision sehr hoch, was ein falsches Bild über die Qualität abgeben würde.

Deshalb wird die Sensitivität eingeführt, was der Quotient aus der Anzahl der 'True Positives' und der Summe aus der Anzahl 'True Positives' und der Anzahl 'False Negatives' ist vgl. (Kundu , 16.12.2022). Eine hohe Sensitivität wird erreicht, wenn alle Objekte richtig gefunden werden und dabei keine Objekte übersehen werden. Somit ist 'Recall' 100 Prozent, wenn es keine 'False Negatives' gibt. Wird nur 'Recall' genutzt, dann würde sie aber auch 100 Prozent werden, wenn zu viele Objekte erkannt wurden,

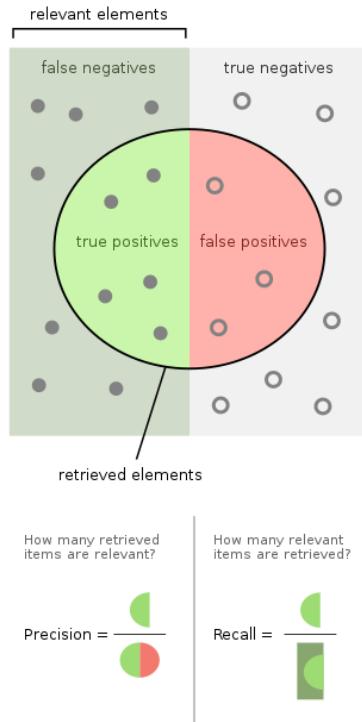


Abb. 2: Metriken des Machine Learnings
(Walber , 22.11.2014)

Somit muss eine Kombination aus beiden Metriken genutzt werden. In diesem Fall ist dies der F1-Score. Er lässt sich mit der Formel 1 berechnen vgl. (Kundu , 16.12.2022). Wird der F1-Score genutzt muss die KI möglichst alle gesuchten Objekte finden (Recall maximieren) und dabei müssen die Voraussagen stimmen (Precision maximal).

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad (1)$$

3 Setup - Genutzte Hardware

3.1 Raspberry Pi 3 A+



Abb. 3: Raspberry Pi
(Geizhals.de , o.D.)

Die technischen Daten sind in Tabelle 1 einzusehen.

Prozessor	64-bit Quad Core: 1400 MHz
RAM	512 MB SDRAM
Speicherplatz	MicroSD-Karte (max. 32GB)
Kamera	Raspberry Pi Camera Rev 1.3 (OV5647) 5MP 1920x1080@30fps, 1280x720@60fps vgl. (OmniVision Technologies Inc. , 11.03.2009)
Preis	ab 28€
Verbindungsstellen	1x USB2.0, 1x HDMI, 2.4/5 GHz WLAN, Bluetooth, 40 GPIO-Pins vgl. (Raspberry Pi Foundation , 11.2018)

Tabelle 1: Technische Daten Raspberry Pi 3 Modell A+

3.1.1 Installatiton von PiOS

Das Linux-basierte RaspberryPi-eigene Betriebssystem, PiOS, kann mit Hilfe des Tools Raspberry Pi Imager auf eine microSD-Karte geladen werden. Das Tool ist auf der offiziellen Webseite zu finden: <https://www.raspberrypi.com/news/raspberry-pi-imager-imaging-utility/>. Für die Installation kann der Nutzer die microSD-Karte auswählen, sowie die gewünschte Version von PiOS. Für den Raspberry Pi 3 A+ ist die 32-bit-Version optimal, da er über zu wenig Arbeitsspeicher verfügt, um die 64-bit Version zu nutzen. Außerdem sind weitere Einstellungen

sinnvoll, die über das Zahnrad in der unteren rechten Ecke veränderbar sind. Wenn der Nutzer beispielsweise plant, auf den Raspberry Pi über WLAN zuzugreifen, sollte hier das zu nutzende WLAN mit dem Passwort hinterlegt werden, und 'SSH' aktiviert werden. Außerdem muss ein sinnvoller Hostname gewählt werden. Mit dem Klicken auf 'Schreiben' wird auf der SD-Karte das Betriebssystem gespeichert und so formatiert, dass der Raspberry Pi direkt von der SD-Karte starten kann. Wenn dies abgeschlossen ist, kann die SD-Karte entfernt und in den Raspberry Pi eingesteckt werden. Nach dem Anbinden eines Netzteils startet der Raspberry Pi von der SD-Karte und ist für den Nutzer einsatzbereit.

Beim Betrieb über WLAN muss zunächst auf dem Raspberry Pi ein Webserver installiert werden. Hierzu wird eine SSH-Verbindung benötigt. Diese kann über die Kommandozeile entweder über eine konkrete lokale IP-Adresse oder mit dem Hostname des Raspberry Pi aufgebaut werden. Daraufhin können jegliche Befehle auf dem Raspberry Pi ausgeführt werden. Für den Webserver muss zunächst der RealVNC-Server deinstalliert, dann der XRDP-Server installiert werden (siehe Befehle 1).

Nun kann das von Windows eingebaute Programm 'Remotedesktopverbindung' genutzt werden, um auf den Desktop vom Raspberry Pi zuzugreifen. Hierzu wird nach dem Starten des Programms entweder die IP-Adresse oder der Hostname des Raspberry Pis eingegeben. Um die IP-Adresse herauszufinden, kann in der Liste aller mit dem Netzwerk verbundenen Geräte nach dem Raspberry Pi gesucht werden. Wenn kein Zugriff auf diese Liste möglich ist, kann der Nutzer nach dem Verbinden über HDMI den Befehl 2 ausführen.

Einige wichtigen Befehle für die allgemeine Nutzung der Kommandozeile des Raspberry Pis sind in Quellcode 3 aufgeführt.

```
1 ssh [username]@[IP-Address]
2 sudo apt-get purge realvnc-vnc-
   server
3 sudo apt-get install xrdp
```

Quellcode/Command 1: Installation d. Webservers über SSH (WinTotal.de , 01.04.2019)

```
1 hostname -I
```

Quellcode/Command 2: Herausfinden der IP-Adresse (roshalmoraes , 10.05.2023)

```
1 cd [Ordner] #wechseln des aktuellen Ordners
2 ls #auflisten aller Dateien und Ordner im aktuellen Ordner
3 sudo [befehl] #Ausfuehren des Befehls mit Admin-Berechtigungen
4 sudo apt-get update && sudo apt-get upgrade #Update
5 sudo raspi-config #Anpassen verschiedenster Einstellungen
6 raspistill -o img.jpg #Aufnehmen und speichern eines Bildes
```

Quellcode/Command 3: Hilfreiche Befehle (TutorialsForRaspberryPi , o.D.)

3.1.2 Aufnahme eines Bildes mit der Kamera

Ein Bild kann entweder mit dem Befehl raspistill oder mit der Bibliothek PiCamera aufgenommen werden. Hierzu muss zunächst der Befehl 4 ausgeführt werden. Der Code für die Nutzung ist in Quellcode 5 dargestellt.

```
1 pip install "picamera[array]"
```

Quellcode/Command 4: Installation von PiCamera

```
1 from picamera.array import PiRGBArray
2
3 from picamera import PiCamera
4 import time
5 import cv2
6
7 camera = PiCamera()
8 rawCapture = PiRGBArray(camera)
9 time.sleep(0.1) # damit die Kamera sich aufwaermen kann
10 camera.capture(rawCapture, format="bgr")
11 image = rawCapture.array # Bild als numpy-array
12
13 cv2.imshow("Image", image) #Anzeigen des Bildes
14 cv2.waitKey(0)
```

Quellcode/Command 5: Nutzung von PiCamera (Rosebrock , 20.03.2015)

3.2 ESP-32-CAM

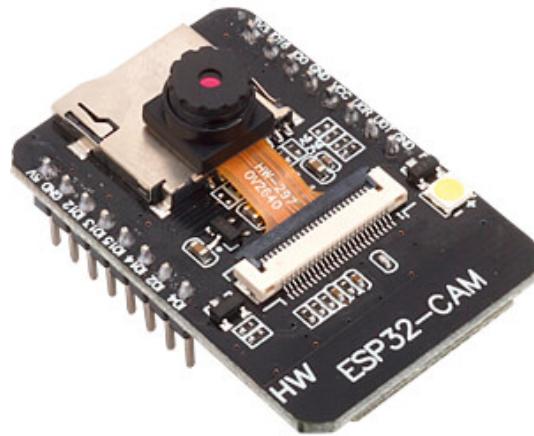


Abb. 4: ESP-32 CAM

(Reichelt.de , o.D.)

Die technischen Daten sind in Tabelle 2 einzusehen.

Prozessor	32-bit Dual Core: 240 MHz
RAM	520KB SRAM + 8MB PSRAM
Speicherplatz	32MB Flash, MicroSD-Karte (max. 32GB)
Kamera	OV2640 2MP 1600x1200@15fps, 800x600@30fps vgl. (OmniVision Technologies Inc. , 23.02.2007)
Preis	ab 5€
Verbindungsstellen	1x UART (microUSB), 2.4 GHz WLAN, Bluetooth vgl. (Ai-Thinker Technology , o.D.)

Tabelle 2: Technische Daten AI Thinker ESP 32

3.2.1 Laden einer Firmware in den Flash

Firmwares können mit Hilfe des ESP-32-Tool gebaut und direkt mit dem Tool oder mit Thonny auf den ESP-32 gespielt werden. Zunächst muss das ESPTool von der ESP-IDF Webseite (<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/>) heruntergeladen und installiert werden. Dies kann auch über Zeile 1 im Befehl 6 in einer Befehlszeile geschehen. Durch Zeile 2 wird aus bspw. einem C++-Projekt eine .bin-Datei generiert, welche auf den ESP-32 mit den Zeilen 3 und 4 geladen werden kann.

Zunächst muss der Flash gelöscht werden, damit keine Fehler beim Überschreiben des Flashes auftreten. Danach wird die Firmware auf den Flash geschrieben. Der Port-Parameter muss für jedes Board neu ausgefüllt werden, da sich dieser ändern kann. Wenn kein Port angegeben wird, dann sucht das ESPTool automatisch nach einem Port an dem ein ESP-32 angeschlossen ist. Die Baud-Rate stellt die Geschwindigkeit des Schreibens ein. Hier darf kein zu großem Wert genutzt werden, aber auch kein zu kleinem Wert, da das Aufspielen sonst sehr lang dauert. Die Standard-Einstellung ist hierbei 115200.

Mit Thonny kann dieser Vorgang direkt in der IDE vorgenommen werden. Zum Installieren von Thonny kann in einer Befehlszeile Befehl 7 ausgeführt werden. Zum Aufspielen wird in der unteren rechten Ecke die aktuell ausgewählte Python-Version ausgewählt, gefolgt von 'Configure Interpreter', und 'Install or update MicroPython'. Nachdem der Port nun ausgewählt wurde sowie die gewünschte .bin-Datei, ist es empfehlenswert die Check-Box 'Erase flash before installing' zu aktivieren, damit der Flash-Speicher vor dem Flash-Vorgang gelöscht wird. Beim Klicken des 'Install'-Buttons wird die Firmware auf den ESP-32 geladen.

```

1 pip install esptool #ESP-Tool installieren
2 python -m esptool build #Die Firmware bauen
3 python -m esptool --chip esp32 --port "COM3" erase_flash
4 python -m esptool --chip esp32 --port "COM3" --baud 460800 write_flash -z 0x1000
firmware.bin #Die Firmware flashen

```

Quellcode/Command 6: Bauen / Flashen einer Firmware in den Flash

```
1 pip install thonnyapp
```

Quellcode/Command 7: Thonny installieren

3.3 M5Stack UnitV



Abb. 5: M5Stack UnitV
(M5Stack , o.D.)

Die technischen Daten sind in Tabelle 3 einzusehen.

Prozessor	64-bit Dual Core: 400 MHz Neural Network Processor (K210) 0.8 TFLOPS
RAM	8MB SRAM
Speicherplatz	16MB Flash, MicroSD-Karte (max. 32GB)
Kamera	OV7740 0.4MP 640x480@60fps vgl. (OmniVision Technologies Inc. , 29.07.2008)
Preis	ab 28€
Verbindungsstellen	1x USB-C, 1x 4 Pin-Anschluss vgl. (M5Stack , o.D.)

Tabelle 3: Technische Daten UnitV

3.3.1 MaixPy und Flashen einer Firmware

Bei MaixPy handelt es sich um eine angepasste Version von Python, welche von Sipeed entwickelt wird. Diese Version ist direkt für die Verwendung auf zum Beispiel dem K210 ausgelegt, da bereits einige wichtige und nützliche Bibliotheken eingebaut sind. Es ist die Standardprogrammiersprache, welche bei den meisten Geräte bereits vorinstalliert ist.

Für das Flashen wird das Tool kflashGUI genutzt welches von folgender Webseite heruntergeladen und installiert werden kann: https://github.com/sipeed/kflash_gui. Außerdem wird eine Firmware mit der jeweilig gebrauchten Form von MaixPy benötigt. Diese sind unter <http://dl.sipeed.com/MAIX/MaixPy/release/master/> zu finden.

3.4 Theoretischer Vergleich

Anhand von den technischen Daten lässt sich vermuten, dass auf dem ESP-32-CAM das Modell nur langsam laufen kann, da 240MHz und 2 Kerne in heutigen Verhältnissen keine hohe Leistung verspricht. Dagegen sind die 1,4 GHz und 4 Kerne des Raspberry Pis deutlich schneller. Der M5Stack UnitV wird sich zwischen dem ESP und dem Raspberry Pi befinden, da auch hier nur ein langsamer Prozessor vorhanden ist. Dafür besitzt der UnitV eine eigene Tensor-Processing-Unit (TPU), welche die Ausführungsgeschwindigkeit von neuronalen Netzwerken deutlich erhöht.

Ein weiterer nicht zu verachtender Faktor ist, dass Edge Impulse den Raspberry Pi offiziell unterstützt und somit die Modelle (KIs) deutlich optimierter laufen werden, als bei dem UnitV. Auf diesem läuft das Modell nur über Umwege.

Bezüglich der Benutzerfreundlichkeit lässt sich abschätzen, dass sich der Raspberry Pi am besten nutzen lässt, da dieser ein vollwertiges Betriebssystem besitzt. Dagegen müssen der ESP-32-CAM und der UnitV mit umständlichen Firmwares programmiert werden.

4 Nutzung von Edge Impulse

Edge Impulse ist ein im Jahr 2019 von Zach Shelby und Jan Jongboom gegründeter Dienst zur Unterstützung bei der Entwicklung von KIs für eingebettete Systeme. Hierfür sind fast keine Vorkenntnisse in der Programmierung und dem Machine Learning nötig. Hierzu existiert eine gleichnamige Webseite, auf der der vollständige Prozess vom Aufnehmen der Daten bis zum Deployment auf den Systemen benutzerfreundlich vom Nutzer durchgegangen werden kann vgl. (Edge Impulse Inc. , 2023).

Nach dem Erstellen eines Projekts befindet sich der Nutzer auf der 'Dashboard'-Seite. Auf dieser sind beim Herunterscrollen alle mit dem Projekt verbundenen Dateien, wie trainierte Modelle, zu finden. Außerdem befinden sich hier alle Einstellungen rund um das Projekt. Ganz am Ende des Dashboards gibt es auch die Option, alle Daten zu löschen, wenn ein vollständig neues Modell trainiert werden soll (Abbildung 6).

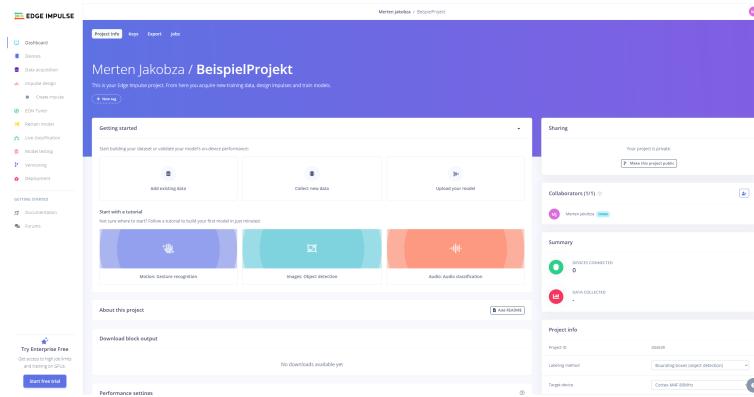


Abb. 6: Edge Impulse -Dashboard

Der erste Schritt ist das Aufnehmen von Daten. Es können hierzu verschiedenste Geräte mit dem Projekt verbunden werden. Dazu gehören unter anderem Smartphones, welche durch das Scannen von einem QR-Code Bilder direkt aufnehmen können. Außerdem können mehrere unterstützte Boards über bestimmte Firmwares mit dem Projekt verbunden werden. Alternativ können große Mengen von Daten direkt hochgeladen werden, indem neben den Bildern optional eine JSON-Datei hochgeladen wird, welche die Bounding-Boxen oder die Labels enthält. Wenn diese nicht enthalten ist, müssen alle Bilder händisch mit Label oder Bounding-Boxen versehen werden (Abbildung 7).

Anschließend ist es möglich, seinen 'Impulse' zu gestalten. Dazu kann die Auflösung bei Bild-Verarbeitungsprojekten eingegeben werden. Außerdem kann eingestellt werden, wie die Daten verarbeitet werden sollen, also das Feature-Processing. An dieser Stelle ist der 'Image-Processing-Block' für Bildverarbeitungsprojekte zu wählen. Dann kann in Form des 'Learning-Blocks' entschieden werden, ob das Modell auf Basis eines Object-Detection-Modells direkt von Edge-Impulse oder einem von BrainChip trainiert werden soll. Letzterer bringt jedoch nur Vorteile, wenn das Modell auf einem BrainChip AI-Accelerator genutzt werden soll. Dies ist bei den meisten Anwendungsfällen nicht der Fall. Deshalb ist das Modell von Edge-Impulse sinnvoller. Außerdem bietet Edge Impulse eine Option der 'Data Augmentation' an, welche ei-

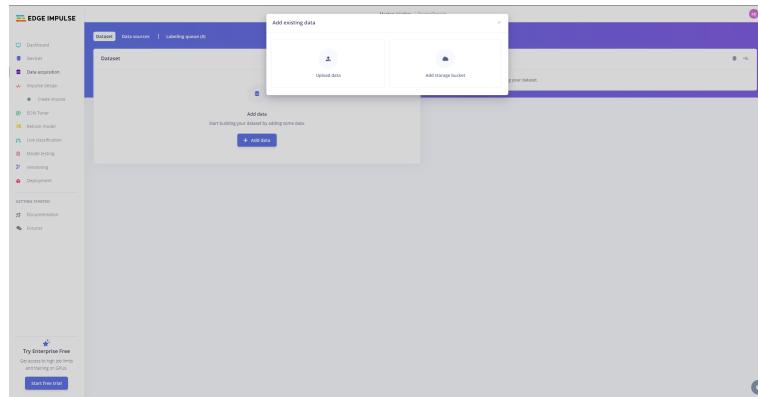


Abb. 7: Edge Impulse -Data Aquisition

ne native Implementation der Data Augmentation (siehe Kapitel 6) bereitstellt. Als 'Learning block' ist also 'Object Detection (Images)' von Edge Impulse für Object Detection und 'Transfer Learning (Images)' für Image Classification für nahezu alle Anwendungsbereiche zu empfehlen. Hier sind auch Optimierungen durch Veränderung des 'Learning-Blocks' möglich. Diese Learning Blocks bauen auf Transfer-Learning (Kapitel 2.2) auf (Abbildung 8).

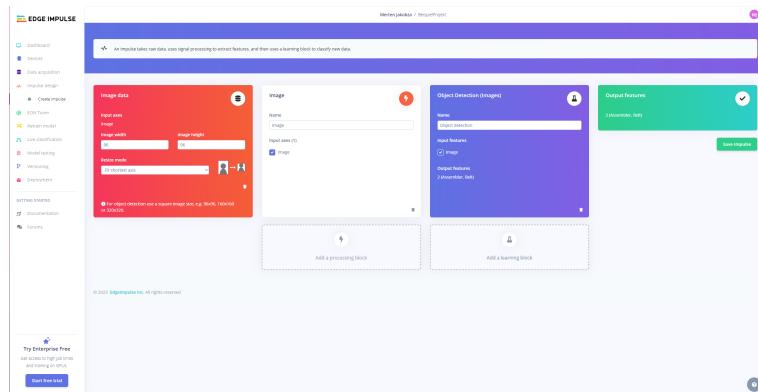


Abb. 8: Edge Impulse -Gestalten des Impulses

Nach Abschluss müssen nun die Processing- und Learning-Blöcke in den nächsten beiden Schritten konfiguriert werden. Beim Processing-Block wird ausgewählt, ob das Modell mit RGB oder nur Graustufen-Daten arbeiten soll. Das FOMO-Object-Detection-Modell nutzt hierbei nur Graustufen-Bilder. Die Features der Trainingsdaten werden unter dem 'Generate Features' Punkt für alle Bilder generiert und anschließend in einem Diagramm dargestellt.

Beim Konfigurieren des Learning-Blocks können die Anzahl der Training-Zyklen, die Learning-Rate und die Option der 'Data-Augmentation' an die Präferenzen angepasst werden. Außerdem können die Größenumfangverhältnis der Training- und Learning-Menge verändert werden. Die weiteren Einstellungen sind für den Standard-Benutzer meist unwichtig. Durch das Auswählen des 'Keras-Expert-Mode' kann der Benutzer außerdem direkte Veränderungen im Code für das Training vornehmen. Wie der Name jedoch schon sagt, ist hierfür ein gutes Verständnis von Python und der Bibliothek 'Tensorflow.Keras' nötig (Abbildung 9). Edge Impulse liefert am Ende des Trainings eine Confusion-Matrix. In dieser kann für jede Klasse die Genauigkeit, Sensitivität

tät und der F1-Score nachvollzogen werden. Dies ist hilfreich um zu erkennen, welche Klassen bei der Klassifizierung problematisch sind.

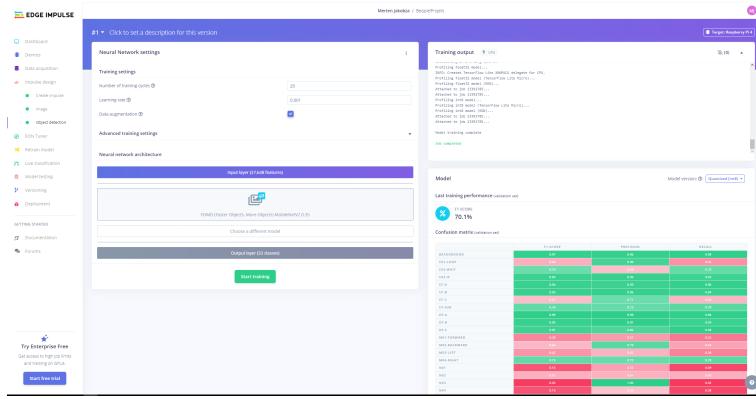


Abb. 9: Edge Impulse -Training des Modells

Da in der kostenlosen Version von Edge-Impulsen ein Zeit-Limit von 20 Minuten für das Training existiert, kann durch das Verändern der Trainings-Parameter die Zeit optimiert werden und somit mehr Daten zum Training genutzt werden. Die Anpassung des verwendeten Modells ist ebenfalls möglich. Für die Object Detection ist hier 'FOMO (Faster Objects, More Objects)' von MobileNetV2' in der neusten Version empfehlenswert. Für Image Classification sind die Modelle von MobileNet in der jeweilig benötigten Auflösung sinnvoll.

Es gibt verschiedene Möglichkeiten die Trainings-Zeit zu verringern, um unter die 20-Minuten Grenze zu gelangen. Zum einen kann die Datenmenge und zum anderen die Trainings-Zyklen verringert werden. Damit das Training trotzdem effektiv ist und am Ende trotzdem ein Ergebnis herauskommt, sollte die Learning Rate daraufhin auch angehoben werden. Als Letztes kann die Pixel-Größe der Eingabe-Bilder verringert werden, also von 300x300 Pixel auf zum Beispiel 200x200 Pixel vgl. (Edge Impulse Inc., 19.05.2022). Alternativ kann auch die Premium-Version erworben werden, welche für Firmen ausgelegt ist. Diese starten bei einem Preis von 3000€ pro Monat und somit keine Option für private Entwickler ist. Es gibt jedoch eine um rund 80 Prozent reduzierte Version für Schulen und andere Bildungseinrichtungen.

Im Bereich 'Live classification' kann entweder mit einer Live-Kamera oder einzelnen Bildern das Modell getestet werden (Abbildung 10). Um die Test-Daten zu nutzen, steht der Bereich 'Model testing' zu Verfügung. Hier werden beim Anklicken des 'Classify All'-Buttons alle Testing-Dateien klassifiziert und überprüft, ob die vorhergesagten Ergebnisse mit den eingespeicherten übereinstimmen. Edge Impulse liefert letztendlich eine Prozentzahl, die einen Durchschnitt für alle F1-Scores angibt (Abbildung 11).

Wenn weitere Dateien hinzugefügt werden, kann unter dem Punkt 'Retrain model' das Generieren der Features und das Trainieren des Modells direkt am Stück vorgenommen werden.

Unter 'Versioning' gibt Edge Impulse dem Nutzer die Möglichkeit, Zwischenergebnisse zu speichern, bevor große Änderungen vorgenommen werden. Dabei werden auch alle Bilder und Daten gespeichert, welche in einem neuen Projekt wiederhergestellt werden können.

Im letzten Punkt 'Deployment' kann letztendlich ausgewählt werden, auf welchem Gerät das

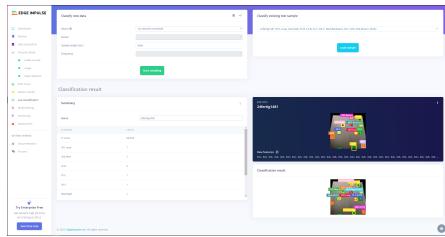


Abb. 10: Edge Impulse -Live Classification

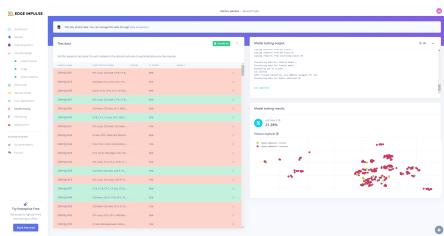


Abb. 11: Edge Impulse -Model Testing

Modell genutzt werden soll. Hier sind auch für jedes dieser Geräte Anleitungen bereitgestellt. Diese schlüsseln auf, wie das Deployment für das gewählte Board funktioniert. Außerdem ist hier eine erste Einschätzung über die Klassifikationsdauer von Edge Impulse angegeben.

Bei Problemen stehen auf dem Forum zahlreiche Communitymitglieder und Entwickler zur Verfügung, welche technische Fragen aktiv beantworten: <https://forum.edgeimpulse.com>

5 Deployment auf den eingebetteten Systemen

5.1 Raspberry Pi

5.1.1 Installation

Um das Modell auf dem Raspberry Pi 3 A+ laufen zu lassen, muss mit dem Raspberry Pi OS das Edge Impulse Command-Line-Interface (CLI) installiert werden. Hierzu werden die Befehle 8 im Terminal des Raspberry Pi ausgeführt. Daraufhin kann das Modell mit den Befehlen 8 in Form einer EIM-Datei heruntergeladen werden. Es wird eine EIM-Datei (Edge-Impulse-Model) genutzt, da in dieser das vollständige Projekt gespeichert ist und keine weiteren Abhängigkeiten, wie Bibliotheken gibt. Außerdem kann das Modell für das genutzte Board optimiert werden vgl. (Edge Impulse Inc. , 28.09.2023).

Dabei können einige Fehler auftreten. Zum Beispiel wird das Installieren der Edge-Impulse-CLI manchmal dadurch aufgehalten, dass 'opencv-python' nicht installiert werden kann. Dies kann dadurch gelöst werden, dass das 'OpenCV'-Modul separat installiert wird. Außerdem wird das Modul 'sharp' im falschen Ordner installiert, was dadurch zu Stande kommt, dass der Ordner 'Node-modules' zum einem an der systemweiten Position zu finden ist und zum anderem als Unterordner in 'Edge-Impulse'. Das 'sharp'-Modul wird dabei in den systemweiten Ordner geladen und nicht in dem von Edge Impulse.

Edge Impulse bietet in einem GitHub-Repository einige Beispielprogramme an, welche heruntergeladen und direkt genutzt werden können: <https://github.com/edgeimpulse/linux-sdk-python/tree/master/examples/image>

```
1 sudo apt-get install libatlas-base-dev libportaudio0 libportaudio2  
    libportaudiocpp0 portaudio19-dev  
2 pip3 install edge_ impulse_linux -i https://pypi.python.org/simple # Installation  
    der Edge-Impulse-CLI  
3 edge-impulse-linux-runner --download modelfile.eim # Herunterladen der .eim-Datei
```

Quellcode/Command 8: Installation Edge Impulse CLI (Edge Impulse Inc. , 14.09.2023)

5.1.2 Nutzung des Modells

Der Quellcode 9 bezieht sich lediglich auf die Nutzung auf dem Raspberry Pi. Letztendlich lässt sich das Modell nutzen, indem zuerst die Raw-Features aus dem Bild generiert wird und danach diese Liste an Raw-Features klassifiziert werden. Aktuell muss das Ausgangsbild bei einem Modell, welches in Graustufen trainiert ist, im BGR-Farbformat übergeben werden, anstatt des RGB-Formats. Ebenfalls wichtig ist, dass das Modell auf Bilder mit einem Vielfachen von 32 Pixeln in Höhe und Breite trainiert wird. Ansonsten kommt es zu internen Rundungsfehlern und die Ergebnisse werden verschoben, wie in Abbildung 12 zu sehen ist. Wird die Bildgröße auf beispielsweise 224x224 Pixel geändert, stimmen die Ergebnisse wieder überein, wie in Abbildung 13 zu sehen ist. vgl. (Edge Impulse Inc. , 06.09.2023).

```

1 import os
2 import sys
3 import cv2
4 from edge_ impulse_linux.image import ImageImpulseRunner
5 dir_path = os.path.dirname(os.path.realpath(__file__))
6 modelfile = os.path.join(dir_path, model) #Hier Pfad zur Modell-Datei angeben
7 with ImageImpulseRunner(modelfile) as runner:
8     try:
9         model_info = runner.init()
10        print('Loaded runner for "' + model_info['project']['owner'] + ' / ' +
model_info['project']['name'] + '"')
11        labels = model_info['model_parameters']['labels']
12        img = cv2.imread(args[1])
13        #img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) -> Eigentlich noetig, jedoch
aktuell ein Fehler im Edge Impulse Runner
14        features, cropped = runner.get_features_from_image(img)
15        res = runner.classify(features)
16        print('Found %d bounding boxes (%d ms.)' % (
17            len(res["result"]["bounding_boxes"]), res['timing']['dsp'] + res['timing',
]['classification']))
18        for bb in res["result"]["bounding_boxes"]:
19            print('\t%s (%.2f): x=%d y=%d w=%d h=%d' % (
bb['label'], bb['value'], bb['x'], bb['y'], bb['width'], bb['height',
]))
20            cropped = cv2.rectangle(cropped, (bb['x'], bb['y']),
21                                    (bb['x'] + bb['width'], bb['y'] + bb['
height']), (255, 0, 0), 1)
22        cv2.imwrite('debug.jpg', cv2.cvtColor(cropped, cv2.COLOR_RGB2BGR))
23    finally:
24        if (runner):
25            runner.stop()

```

Quellcode/Command 9: Beispielprogramm für Nutzung des Modells (Edge Impulse Inc., 14.09.2023)

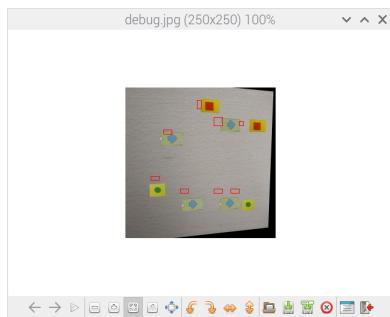


Abb. 12: Verschobene Ergebnisse bei 250x250 Pixel Bild

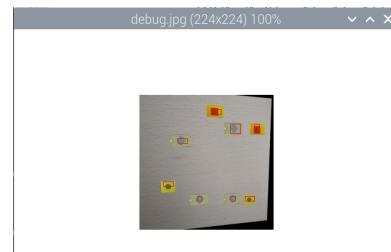


Abb. 13: Richtige Ergebnisse bei 224x224 Pixel Bild

5.2 AI Thinker ESP-32-CAM

5.2.1 Edge Impulse Firmware

Beim ESP-32 gibt es mehrere Ansätze, um das Edge-Impulse-Projekt laufen zu lassen. Zum einen gibt es die Option von Edge-Impulse direkt, das Projekt in eine Firmware für den ESP-32 umzuwandeln. Hierzu ist es wichtig, das ESP-Tool zu installieren, womit Firmwares auf den ESP-32 geladen werden. Das Problem hierbei ist, dass diese Firmware für den Espressif ESP-EYE gebaut ist und somit das PIN-Layout nicht mit dem des AI Thinker ESP-32 übereinstimmt. Außerdem wäre ein zweites Board zum Auswerten der Ergebnisse nötig, da keine anderen Funktionen hinzugefügt werden können. Somit fällt dieser Ansatz bei der vorliegenden Arbeit schnell heraus.

5.2.2 MicroPython und TensorFLowLite

Hier kommt der eigentlich angestrebte Ansatz mit Micropython ins Spiel. Micropython ermöglicht, dass das TFLite-Modell geladen wird und dann die Ergebnisse ausgegeben werden. Nach dem Importieren von tensorflow, numpy und pillow kann ein Bild wie in Quellcode 10 klassifiziert werden.

'Results' ist in diesem Fall (96x96 Pixel Bild, 3 Klassen) ein 12x12x4 großes Array. Dies liegt daran, dass FOMO eine Heatmap zurückgibt vgl. (Edge Impulse Inc., 06.09.2023). Der Wert jedes Elements gibt die Confidence-Werte für jedes Label an. Also gibt jedes Element wieder, wie sicher das Modell sich ist, dass hier einer der drei Klassen bzw. einfach nur Hintergrund gefunden wurde. Die Größe des Arrays ist dabei ein Achtel der Höhe und Breite des Eingabe-Bildes und die Confidence Werte sind im Bereich von -127 bis 127 angegeben. Ein Problem ist, dass die Ergebnisse nicht mit denen von der Edge-Impulse-Webseite übereinstimmen. Außerdem gibt es das Problem, dass es nicht möglich ist, ein Bild mit der Kamera aufzunehmen, da die meistens genutzte Camera-Bibliothek für den Espressif ESP-32-EYE ausgelegt ist, jedoch nicht für den AI Thinker ESP-32 CAM. Dadurch tritt der Fehler 'Camera Init failed' beim Initialisieren der Kamera wiederholt auf.

Nach der Klassifizierung müsste nun die Heatmap ausgewertet werden. Diese Auswertung fügt weitere Verarbeitungszeit hinzu, wodurch die Performance weiter verschlechtert wird. Da dies ein größerer Programmieraufwand nach sich ziehen würde, sind die nachfolgenden Ansätze sinnvoller.

```
1 interpreter = tf.lite.Interpreter(model_path={model_path})
2
3 input_details = interpreter.get_input_details()
4 output_details = interpreter.get_output_details()
5 input_data_type = input_details[0]["dtype"]
6
7 img = Image.open({image_path}).resize(({img_size, img_size}).convert("RGB")
8 img_np = np.array(np.expand_dims(img, axis = 0), dtype=input_data_type)
9 signature_fn = interpreter.get_signature_runner()
10 output = signature_fn(x = image)
11 results = output.get("output_0")
```

Quellcode/Command 10: Beispielprogramm zur Ausführung eines TensorFlow-Lite-Modells mit Micropython

5.2.3 C++-Bibliothek

Der nächste Ansatz ist das Bauen einer Firmware mit dem ESP-IDF-Tool basierend auf einem C++-Projekt. Diese Firmware wird dann auf den ESP-32 aufgespielt und die Ergebnisse ausgewertet. Auch hier steht ein funktionierendes C++-Projekt direkt von Edge Impulse zur Verfügung. Die Ausgangswerte müssen jedoch in einer Liste namens 'Features' gespeichert werden. Diese Features stellen die RGB-Werte des Ursprungsbildes dar. Um das Model wie gewollt laufen zu lassen, müsste ein Bild aufgenommen werden und dann in die Features umgewandelt werden. Doch dieser Ansatz ist deutlich ineffizienter als der folgende Ansatz, da der sämtliche Code zum Aufnehmen eines Bildes und Umwandeln in Features selbst geschrieben werden müsste. Auf diesen Edge-Devices ist der RAM und Speicher äußerst begrenzt und die Bildaufnahme zu programmieren ist deutlich schwerer umzusetzen, als folgender Ansatz vgl. (Edge Impulse Inc. , 28.06.2023), (Edge Impulse Inc. , 18.08.2023).

5.2.4 Arduino-Bibliothek

Der letzte Ansatz ist das Bauen einer Firmware auf Arduino-Basis. Hierzu wird die Erweiterung 'PlatformIO' bei Visual Studio Code installiert, welches erlaubt Arduino-basierte Firmwares für verschiedenste eingebetteten Systeme zu bauen. Beim Erstellen eines Projekts wird der Ai Thinker ESP-32-CAM ausgewählt, sowie Arduino als Framework. Das Edge-Impulse-Projekt wird dann als Bibliothek für das Arduino-Programm exportiert und kann dann in dem Library-Ordner gespeichert werden. Hier sind auch die verschiedenen Programme bereits vorprogrammiert und müssen nur noch angepasst werden. Zum einen muss eine Funktion namens 'ei_camera_get_data' im Programm-Ablauf hochbewegt werden, da sonst eine Fehlermeldung ausgegeben wird. Außerdem müssen die Änderungen aus Quellcode 11 im Programmablauf umgesetzt werden. Somit wird deklariert, dass hier der AI Thinker ESP-32-CAM mit PSRAM statt dem ESP-32-EYE genutzt wird.

In platformio.ini müssen die Parameter aus Quellcode 12 hinzugefügt werden, da sonst keine Ausgaben in dem Serial Monitor zu sehen sind. RTS steht für 'Request to Send' und DTR für 'Data terminal Ready' und führen einen Reset auf dem ESP-32 durch, wodurch der Serielle Monitor nicht funktioniert. vgl. (Weis , 23.06.2020). PlatformIO besitzt weiterhin Probleme mit Windows 11, weshalb das Hochladen der neuen Firmware unter Windows 11 in manchen Instanzen fehlschlägt vgl. (Edge Impulse Inc. , 18.08.2023). Letztendlich muss der Frame_Byte_Size-Parameter bei einem Graustufen-Modell auf 1 geändert werden, sowie die Kameraauflösung in camera_config und die raw_frame_buffer-Auflösung auf die Modellgröße angepasst werden.

```

1 #define CAMERA_MODEL_ESP_EYE Has
    PSRAM
2 // #define CAMERA_MODEL_AI_THINKER //
    Has PSRAM
3 // wird zu:
4 // #define CAMERA_MODEL_ESP_EYE //
    Has PSRAM
5 #define CAMERA_MODEL_AI_THINKER Has
    PSRAM

```

Quellcode/Command 11: Änderung an Beispielprogramm Arduino

```

1 monitor_speed = 115200
2 monitor_dtr = 0
3 monitor_rts = 0

```

Quellcode/Command 12: Änderungen in Platformio-Einstellungen

5.3 UnitV mit K210

5.3.1 Umwandlung eines TFLite-Modells in ein KModel und Flashen

Zur Umwandlung wird die unquantisierte Version des Modells gebraucht, welche mit einem Tool von 'nncase', namens 'ncc' in eine derartige Version, dass der K210 das Modell laufen lassen kann. Diese Version hat die Dateiendung .kmodel. Hierzu wird die neuste Version von ncc von <https://github.com/kendryte/nncase/releases> heruntergeladen und installiert. Außerdem wird die TFLite-Datei des Modells benötigt. Ist alles vorbereitet, kann mit dem Befehl 13 die TFLite-Datei in eine KModel-Datei umgewandelt werden. Hierbei ist zu beachten, dass die Netzstruktur nicht größer als 2MB ist, da ein K210 lediglich 2MB große Netze in den Arbeitsspeicher laden kann. Der restliche Speicher wird für Python benötigt. Für das genutzte Netz liegt das Maximum bei einer Eingabe-Bilder-Größe von rund 230x230 Pixeln. Beim Umwandeln wird zudem folgender Fehler ausgegeben: "Conv2D_13 Fallback to float conv2d due to weights divergence". Dies bedeutet, dass beim Quantisieren zu große Abweichen bei der Genauigkeit zu verzeichnen sind. Deshalb wird die unquantisierte Form genutzt. Da jedoch nicht nachvollzogen werden kann, worin der Fehler liegt, muss mit der unquantisierten Form weitergearbeitet werden.

Daraufhin muss das Modell auf eine SD-Karte gespielt werden, sowie die Firmware mit der gewünschten Form von MaixPy. Nach dem Starten des kflashGUI-Tools muss nun wie in Abbildung 14 gezeigt die .bin-Datei sowie die KModel-Datei ausgewählt werden. Es ist darauf zu achten, dass die KModel-Datei nach der Firmware steht. Sonst wird das Modell von der MaixPy-Firmware überschrieben vgl. (vowstar , 20.06.2022).

```

1 .\ncc compile model.lite model.kmodel -i tflite -o kmodel -t k210 --inference -
    type uint8 --dataset-format image --input-type uint8 --dataset #Pfad zu
    Datenset zum Quantisieren

```

Quellcode/Command 13: Umkomplizieren in ein .kmodel

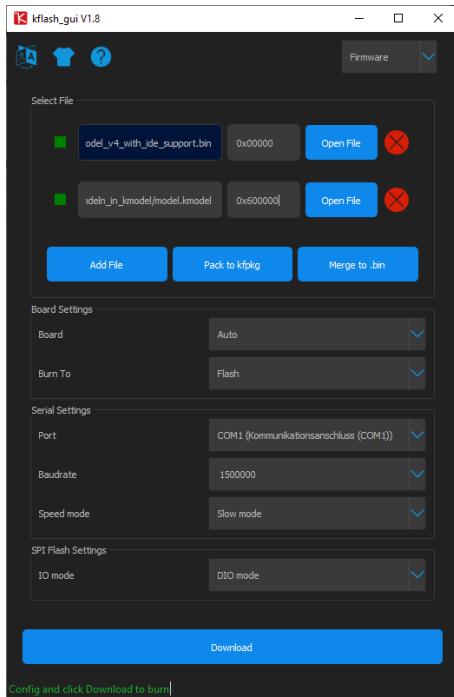


Abb. 14: Nutzung der kflash-GUI

5.3.2 Beispielprogramm

In Quellcode 14 ist die Verwendung eines KModels aufgezeigt. Zunächst wird das Modell geladen und die Kamera vorbereitet. Daraufhin wird in einer Schleife wiederholt ein Bild aufgenommen und dieses durch das Modell klassifiziert. Die Ausgabe ist eine Feature-Map, ähnlich mit der aus Kapitel 5.2.2. Jedoch ist es hier bisher noch nicht mal möglich diese Ergebnisse in irgendeiner Weise zu verarbeiten. Bisher werden nur Image-Classification-Modelle offiziell unterstützt. Alternativ werden auch YOLOv2-Modelle nativ für die Object-Detection unterstützt, jedoch können mit Edge Impulse nur YOLOv5 Modelle trainiert werden vgl. (sipeed , o.D.). Die Performance lässt sich trotzdem einschätzen, da die Klassifikation trotzdem funktioniert, jedoch mit den Ergebnissen bisher nicht weitergearbeitet werden kann.

```

1 import KPU as kpu
2 import sensor
3
4 sensor.reset()
5 sensor.set_pixformat(sensor.RGB565)
6 sensor.set_framesize(sensor.QVGA)
7 sensor.set_windowing((224, 224))
8 model = kpu.load("0x600000")           # Modell laden
9
10 while(True):
11     img = sensor.snapshot()           # Bild aufnehmen mit Kamera
12     img = img.resize(224, 224)
13     out = kpu.forward(task, img)

```

Quellcode/Command 14: Beispielprogramm für die Verwendung eines kmmodels

Es lässt sich bereits jetzt abschätzen, dass nur der Raspberry Pi sinnvoll ist zu verfolgen. Hier ist die Benutzerfreundlichkeit am besten gegeben und es müssen keine großen Umwege genutzt werden.

Für die weiteren Tests und die finale Anwendung wurden zwei Programme in Python entwickelt.

6 Data-Augmentation-Tool

Wenn ein Benutzer ein Bild aufnimmt, kann dieses komplett verschieden sein zu den Trainingsbildern vom Neuronalen Netz. Dadurch kann es auf dem Bild nur noch schlecht die gesuchten Objekte finden. Beispielsweise kann die Kamera, mit dem das Bild aufgenommen wird anders eingestellt sein. Dadurch könnte sie hellere und kontrastreichere Bilder aufnehmen. Zudem könnte der Nutzer das Bild aus einem anderen Kamerawinkel aufnehmen oder aus einer größeren Entfernung.

Um möglichst viele dieser Eventualitäten abzudecken, können verschiedene Veränderungen an den Training-Bildern vorgenommen werden, auch Data Augmentation genannt. Zunächst kann der Hintergrund ausgewechselt werden. Außerdem kann die Helligkeit und der Kontrast verändert werden, da jeder Benutzer eine andere Belichtung und Kamera benutzt. Durch das zufällige Drehen bzw. Spiegeln des Bildes sowie die perspektivische Verzerrung wird ausgeglichen, dass der Nutzer nicht immer aus einem perfekten Winkel die Bilder aufnimmt. Ein Unschärfefilter, der über das ganze Trainings-Bild gelegt wird, kompensiert Bilder mit schlechtem Fokus. Da der Benutzer nicht immer aus der selben Entfernung aufnimmt, kann die Größe der Programmierblättchen auf den Trainings-Bildern variiert werden.

Hierzu wurde ein Tool programmiert, welches aus den Blättchen und verschiedenen Hintergründen Bilder generiert. Diese Bilder werden auf Edge Impulse hochgeladen und anschließend zum Training genutzt.

Die Option des Drehens oder Spiegelns wird nicht auf die Bilder angewandt, da es für die finale Applikation wichtig ist, dass die Blättchen ihre Richtung behalten. Beispielsweise soll das Modell zwischen Vorwärts und Rückwärts unterscheiden können.

6.1 Nutzung

Die Nutzung dieses Programms ist nur wichtig für die Entwicklung der finalen KI und sollte ursprünglich nicht von Dritten genutzt werden.

Beim Starten des Programms mit der Konsole kann mit dem ersten Parameter übergeben werden, wie viele Blättchen maximal pro Bild enthalten sein sollen. Der zweite Parameter gibt die Anzahl der Bilder, die generiert werden sollen, an. Die generierten Bilder werden zunächst im Ordner 'fertig' gespeichert. Zuvor in 'fertig' gespeicherte Bilder werden bei Übergabe von 0 als dritten Parameter gelöscht und für 1 in den Ordner 'dauer' verschoben. Im 'dauer'-Ordner wird ein weiterer Ordner mit der nächstgrößten, ganzen Zahl als Name erstellt. Beispielsweise können mit Befehl 15, 1000 Bilder generiert werden, welche alle maximal 20 Blättchen enthalten, wobei die bereits existierenden, zwischengespeicherten Bilder gelöscht werden. Wenn keiner

```
1 python main.py 20 1000 0
```

Quellcode/Command 15: Beispielparameter für die Nutzung des Tools

dieser Argumente übergeben wird, dann wird der Nutzer in der Konsole nach den Eingaben gefragt. Während des Generierens wird dem Benutzer mit Hilfe von Progress Bars angezeigt, in welchem Schritt sich das Programm aktuell befindet.

Alle für das Programm benötigten Bibliotheken sind in 'requirements.txt' aufgelistet und können in bestimmten IDEs direkt installiert werden.

6.2 Programmierung

6.2.1 Programmierung - Hintergrundveränderung

Für das Austauschen der Hintergründe wird die Bibliothek pillow genutzt. Bevor ein neuer Hintergrund geladen wird, muss zunächst überprüft werden, ob der Hintergrund bereits geladen wurde. Dies geschieht, damit nicht bei jedem generiertem Bild der Hintergrund neu geladen wird. Für neue Hintergründe wird Zeile 1 aus Quellcode 16 genutzt. Dies lädt den zufälligen Hintergrund mit der zufälligen Nummer bg_num und verkleinert die Bilder mit dem Tupel bg_size auf die gewollte Größe. Nun wird eine zufällige Anzahl an Blättchen generiert. Das Minimum ist die Hälfte des eingegebenen Maximums.

Nun wird für jedes Blättchen eine zufällige Position generiert. Vor dem Einfügen des Blättchens muss überprüft werden, ob das Blättchen sich mit bereits existierenden Blättchen überschneiden würde. Hierzu wird für jedes Blättchen eine separate Koordinatenliste mitgeführt, welche die Position jedes Blättchen speichert. Beim Einfügen wird nun überprüft, ob die Eckpunkte und die Seitenhalbierenden des neuen Blättchens mit bereits existierendem Blättchen aus der mitgeführten Liste in Konflikt gerät. Ist das nicht der Fall, wird das Blättchen mit Zeile 2 aus Quellcode 16 auf den Hintergrund gelegt, wobei 'background' und 'blaettchen' pillow-Bilder sind und 'koordinaten' ein Tupel aus X-Koordinate und Y-Koordinate ist.

```
1 Image.open(f"bg/bg[{bg_num}].jpg").resize(bg_size)
2 background.paste(blaettchen, koordinaten, blaettchen)
```

Quellcode/Command 16: Hintergrund-Austausch

Außerdem werden die Bounding-Boxen, also die Position der Objekte, in einer Liste der Form

```
[[img_name, [label_name, x1_coord, y1_coord, x2_coord, y2_coord], [label_name2, ...]], [img_name2, [label_name1, ...]]]
```

mitgeführt. Dabei ist 'img_name' der Name des Bildes, 'label_name' der Name des Labels, 'x1_' und 'y1_coord' die X- und Y-Koordinate des Punktes in der oberen linken Ecke des Blättchens, 'x2_coord' die X-Koordinate des oberen rechten Eckpunktes des Blättchens und 'y2_coord' die Y-Koordinate des unteren linken Eckpunktes des Blättchens. Diese Liste wird

benötigt, um am Ende eine JSON-Datei zu erstellen. Diese JSON-Datei wird von Edge Impulse genutzt, um intern die Bounding-Boxen für das Training zu erstellen. Wenn keine solche JSON-Datei hochgeladen wird müssen alle Blättchen händisch mit Klassen an der richtigen Stelle versehen werden. Bei mehreren Tausend Labels pro Datensatz kann dies mehrere Stunden dauern. Dies wird mit der Liste automatisiert.

6.2.2 Programmierung - Helligkeit- und Kontrastveränderung

Nach dem Einfügen wird für jedes Bild die Helligkeit und der Kontrast verändert. Hierzu wird 'ImageEnhance' von pillow (Zeile 1 Quellcode 17) genutzt. Zunächst muss der 'enhancer' mit dem Bild initialisiert werden. Danach wird das Bild nun um einen Faktor zwischen 0,8 und 1,2 heller bzw. dunkler gemacht und in einem neuen Bild 'enhanced_img' gespeichert. Dieser Vorgang wird für den Kontrast mit 'ImageEnhancer.Contrast()' wiederholt.

Das pillow-Bild wird daraufhin in ein opencv-Bild umgewandelt. Da Pillow-Bilder das RGB-Format und opencv-Bilder das BGR-Format nutzen, wird das Bild zunächst in ein Numpy-Array umgewandelt. Daraufhin wird das Farbformat verändert vgl. (Steins , 12.09.2021). Das opencv-Bild wird in einer Liste gespeichert und das pillow-Bild aus dem Arbeitsspeicher mit Zeile 7 und 8 gelöscht.

```
1 enhancer = ImageEnhancer.Brightness(img) #Initialisieren
2
3 enhanced_img = enhancer.enhance(factor) #veraendern
4 img_cv = numpy.array(background) #Bild in numpy-array umwandeln
5 img_cv_converted = cv2.cvtColor(img_cv, cv2.COLOR_RGB2BGR) # Farbformat aendern
6
7 del background
8 gc.collect()
```

Quellcode/Command 17: Kontrast-/Helligkeitveränderung

6.2.3 Programmierung - Perspektivische Verzerrung

Ab diesem Zeitpunkt wird opencv genutzt. Diese Bibliothek bietet eine leichte Möglichkeit die Perspektivische Verzerrung zu implementieren. Hierzu werden zunächst 8 Zufallszahlen generiert. Dann werden 4 Ausgangspunkte in der Liste 'pts1' gespeichert und eine zweite Liste 'pts2' erstellt. Die X- und Y-Koordinaten jedes Punktes werden dabei mit den Zufallszahlen verändert. Daraufhin kann mit Zeile 1 aus Quellcode 18 eine Transformationsmatrix erstellt werden. Jedem Ursprungs-Punkt werden neue Koordinaten zugeordnet, die der Punkt nach der perspektivische Verzerrung hat. Mit Zeile 2 wird das Bild verzerrt. Dabei ist 'M' die Matrix, 'width' und 'height' die jeweilige Breite und Höhe des Bildes und 'origin_img' das Ursprungsbild. Da sich die Bounding-Boxen mit der Verzerrung ändern können, werden die Koordinaten der Label in der Bounding-Box-Liste angepasst. Hierzu können die Koordinaten eines Eingabepunktes nach der Verzerrung mit Zeile 4 herausgefunden werden. Dabei ist M die Matrix und oldPoint ist eine Numpy-Liste, welche mit Zeile 3 erstellt werden kann.

```

1 M = cv2.getPerspectiveTransform(pts1, pts2)
2 distorted_img = cv2.warpPerspective(origin_img, M, (width, height))
3 oldPoint = np.array([[[x_coord_old, y_coord_old]]], dtype="float32")
4 newPoint = cv2.perspectiveTransform(oldPoint, M)

```

Quellcode/Command 18: Perspektivische Verzerrung

6.2.4 Programmierung - Unschärfe-Filter

Final wird bei manchen Bildern ein Unschärfe-Filter angewandt. Hierzu kann eine Funktion von opencv, wie in Quellcode 19, genutzt werden. Abschließend wird das finale Bild gespeichert (Zeile 2).

```

1 img = cv2.blur(img, (staerkex, staerkey))
2 cv2.imwrite(f"fertig/fertig[{img_num}].png", img)

```

Quellcode/Command 19: Unschärfe-Filter

6.2.5 Programmierung - Generation JSON-Datei

Nun muss die Bounding-Box-Liste in eine JSON-Datei umgewandelt werden. Damit Edge Impulse diese nutzen kann muss sie bounding_boxes.labels heißen und in der Form wie in Quellcode 20 aufgebaut sein.

```

1 {
2     'version': 1,
3     'type': 'bounding-box-labels',
4     'boundingBoxes': [
5         'fertig0.png': [
6             {'label': 'label1', 'x': 100, 'y': 100, 'width': 120, 'height': 112
7             }, {
8                 'label': 'label2', 'x': 132, 'y': 254, 'width': 300, 'height': 298
9             }],
10        'fertig1.png': [
11            {'label': 'label1', 'x': 100, 'y': 100, 'width': 120, 'height': 112
12            }, {
13                'label': 'label2', 'x': 132, 'y': 254, 'width': 300, 'height': 298
14            }]
15    }

```

Quellcode/Command 20: Beispiel-JSON-Datei (Edge Impulse Inc., 28.08.2023)

Zur Umsetzung wird ein Dictionary genutzt, da dieses mit der json-Bibliothek von Python direkt in eine JSON-Datei umgewandelt werden kann. Dabei wird außerdem überprüft, dass die Bounding-Boxen nicht zu weit außerhalb des Bildes liegen. Eine Konstante beschreibt, welcher Teil der Bounding-Box außerhalb liegen muss, damit diese gelöscht wird. Ist die X-Koordinate der oberen linken Ecke plus die Breite mal der Konstante kleiner als Null, liegt die Bounding-Box zu weit außerhalb am linken Rand und wird deshalb nicht mitgezählt. Dieser Vorgang wird für jeden weiteren Rand und für jede Bounding-Box wiederholt. Mit Quellcode 21 wird das Dictionary in einer JSON-Datei gespeichert und auch direkt leserlich formatiert.

```
1 json.dump(dictionary, "fertig/bounding_boxes.labels", indent=4)
```

Quellcode/Command 21: Speichern der JSON-Datei

6.2.6 Programmierung - Dateienverschiebung

Nun können die Bilder in den Dauerspeicher verschoben werden. Ein weiteres Skript (Ausschnitt in Quellcode 22) lädt zunächst die JSON-Datei und ändert darin alle Bildernamen von fertig[img_num].png zu [index]fertig[img_num].png. Mit Zeile 1 wird der Name des nächsten Ordners bestimmt. Mit Zeile 2 wird ein neuer Ordner mit dem herausgefundenen Index als Namen erstellt. In diesen Ordner werden die Bilder dann verschoben.

```
1 index = max([next(os.walk("dauer"))[1]]) + 1
2 os.mkdir(f"dauer/{index}")
3 shutil.move("fertig/fertig[img_num].png", "dauer/[index]/[index]fertig[img_num].
png")
```

Quellcode/Command 22: Erstellen des Ordner / Verschieben der Bilder

Nun können diese Bilder auf Edge Impulse unter Data Acquisition hochgeladen werden, indem alle Bilder sowie die JSON-Datei ausgewählt werden.

7 Untersuchungen mit finaler KI

Im Folgenden wurden verschiedenste Versuchen unternommen, um die Genauigkeit zu maximieren und alle möglichen Situationen in der realen Welt bestmöglich zu simulieren. Durch das in Kapitel 6 beschriebene Data-Augmentation Tool konnten nun schnelle Änderungen vorgenommen werden.

Die Erkennungsrate war anfangs nicht besonders gut. Oft wurden Blättchen, mit gleichen Hintergrundfarben, verwechselt. Deshalb wurden zunächst die Blättchen so angepasst, dass sich die Zeichen auf den Blättchen farblich unterscheiden. Dies ist in Abbildung 15 zu sehen. Dies erhöhte die Genauigkeit deutlich. Beispielsweise ist der F1-Score der Erkennung der '6' von 48 Prozent (siehe Abbildung 16) auf nahezu 100 Prozent (siehe Abbildung 17) gestiegen. Hierbei wurden die Farben bei all den anderen Blättchen bereits angepasst. Deshalb ist die Genauigkeit hier bereits sehr hoch.



Abb. 15: Zahlen mit untersch. Farben

Da das FOMO-Modell bereits nur Graustufen nutzt, wurden die Farben später in die bis zum Ende genutzten Graustufen (siehe Abbildung 18) geändert. Dadurch gibt es keine Änderung bei der Performance der KI selbst, jedoch sind verschiedene Graustufen für den Nutzer deutlich angenehmer zu nutzen, als unterschiedliche Farben.

Ein Test, die KI auf YOLO-Basis zu trainieren, schlug mehrfach fehl. Da jedoch YOLO noch rechenintensiver ist, hätten sowieso nur weniger Trainingsdaten genutzt werden können. Gleichzeitig hätte sich die Performance auf den Boards weiter verschlechtert.

Die Änderungen an den Graustufen wurden noch verfeinert, indem die Graustufenunterschiede bei den Blättchen, die von der KI oft verwechselt wurden, erhöht wurden.

	F1-SCORE	PRECISION	RECALL
BACKGROUND	1.00	1.00	1.00
C01-LOOP	0.99	0.98	1.00
C02-WAIT	0.99	0.99	0.99
C03-IF	0.99	0.99	1.00
CF-A	1.00	1.00	1.00
CF-B	0.98	0.96	1.00
CF-C	1.00	1.00	1.00
CF-SUB	0.98	0.96	1.00
DF-A	1.00	1.00	1.00
DF-B	1.00	0.99	1.00
DF-C	0.99	0.99	0.99
M01-FORWARD	0.99	0.98	1.00
M02-BACKWARD	0.98	0.97	1.00
M03-LEFT	1.00	0.99	1.00
M04-RIGHT	1.00	0.99	1.00
N01	0.98	0.99	0.97
N02	0.76	0.71	0.85
N03	0.68	0.63	0.81
N04	0.90	0.92	0.88
N05	0.73	0.94	0.60
N06	0.48	0.56	0.42
N08	0.83	0.93	0.75
N09	0.58	0.61	0.67
S01-START	1.00	1.00	0.99
S02-END	0.98	0.97	1.00
S03-BREAK	0.99	0.98	1.00
S04-RETURN	0.98	0.96	0.99
W01-WAND	0.99	0.97	1.00

Abb. 16: Genauigkeit vor der Farbenänderung der Zahlen

	F1-SCORE	PRECISION	RECALL
BACKGROUND	1.00	1.00	1.00
C01-LOOP	0.99	0.98	1.00
C02-WAIT	0.98	0.98	0.98
C03-IF	1.00	0.99	1.00
CF-A	1.00	1.00	1.00
CF-B	0.99	0.99	0.99
CF-C	1.00	0.99	1.00
CF-SUB	1.00	1.00	1.00
DF-A	0.99	0.99	1.00
DF-B	1.00	1.00	1.00
DF-C	0.99	0.98	1.00
M01-FORWARD	0.97	0.96	0.98
M02-BACKWARD	0.99	0.99	0.98
M03-LEFT	0.99	0.99	0.99
M04-RIGHT	0.99	0.99	1.00
N01	1.00	1.00	1.00
N02	1.00	0.99	1.00
N03	0.97	0.96	0.99
N04	0.98	0.98	0.99
N05	1.00	1.00	0.99
N06	1.00	0.99	1.00
N08	0.98	0.96	1.00
N09	1.00	1.00	0.99
S01-START	1.00	1.00	1.00
S02-END	0.99	0.98	0.99
S03-BREAK	1.00	1.00	0.99
S04-RETURN	0.98	0.98	0.98
W01-WAND	1.00	1.00	1.00

Abb. 17: Genauigkeit nach der Farbeänderung der Zahlen



Abb. 18: Zahlen mit untersch. Graustufen

Mit all diesen Änderungen konnte beim Model-Testing ein F1-Score von **98,4 Prozent**, mit konsistent hohem F1-Score bei allen Blättchen (siehe Abbildung 19) erreicht werden. 80 Prozent der Daten wurden zum Training genutzt und 20 Prozent zum Testen.



Abb. 19: F1-Score der Finalen KI

8 Finales Projekt / Auswertungsprogramm

Das Programm ist für den Raspberry Pi entwickelt. Auf anderen Linux-Systemen wurde es bisher nicht getestet. Eine Windows-Demo-Version ist ebenfalls auf GitHub zu finden. Diese beinhaltet alles, bis auf der Bild-Aufnahme- sowie Klassifikationfunktion.

Für das finale Auswertungsprogramm wurde Python und die PyQt5-Bibliothek genutzt, da auch PyQt5 auf dem Raspberry Pi installiert werden kann und somit das Programm vollständig auf dem Raspberry Pi laufen kann, wodurch kein weiterer Computer nötig ist.

PyQt5 kann mit Befehl 23 installiert werden. Nach dem Ausführen von Zeile 2 kann der pyqt5-designer zum schnellen Erstellen der GUI genutzt werden. Nach Ausführen von Zeile 4, können die gewünschten GUI-Widgets auf das Fenster gezogen werden und das Ganze dann als .ui-Datei gespeichert wird. Mit pyuic5 kann die .ui-Datei in eine .py-Datei umgewandelt werden, indem Zeile 5 in der Konsole ausgeführt wird. Pyuic5 erstellt hierbei eine Python-Datei 'window.py', welche in das Hauptprogramm wie in Quellcode 24 eingebunden werden kann. Über 'mainui' kann auf die GUIWidgets zugegriffen werden. Zum Beispiel kann mit Zeile 7 beim Klick auf den Button eine Funktion aufgerufen werden.

```
1 pip install pyqt5
2 pip install pyqt5-tools
3
4 pyqt5-tools designer
5 pyuic5 window.ui -o window.py
```

Quellcode/Command 23: Installation
PyQt5

```
1 from window import Ui_{window_name}
2 import sys
3 app = QApplication(sys.argv)
4 mainwin = QMainWindow()
5 mainui = Ui_{window_name}()
6 mainui.setupUi(mainwin)
7 mainui.button.clicked.connect(lambda
8     : function())
9 mainwin.show()
9 sys.exit(app.exec())
```

Quellcode/Command 24: Beispiel-PyQt5-
Programm

8.1 Nutzung

Allgemein wird zunächst ein Labyrinth mit dem Labyrinth-Editor erstellt, anschließend ein Programm aus den 3D-gedruckten Blättchen im physischen Leben gelegt. Davon wird dann ein Bild aufgenommen. Dieses Bild wird daraufhin klassifiziert und dargestellt. Als letztes kann das gelegte Programm getestet werden, indem das Programm für das Labyrinth durchsimuliert wird.

Alle benötigten Bibliotheken sind in 'requirements.txt' aufgelistet und können in bestimmten integrierten Entwicklungsumgebungen direkt installiert werden. Damit das Programm auf dem Raspberry-Pi nativ läuft, müssen ebenfalls die Befehl 25 ausgeführt werden.

```
1 sudo apt-get install libatlas-base-dev libportaudio0 libportaudio2
   libportaudiocpp0 portaudio19-dev
2 pip3 install edge_ impulse_linux -i https://pypi.python.org/simple
```

Quellcode/Command 25: Installation von Edge-Impulse-Linux

8.1.1 Nutzung - Allgemeine Hinweise

Es gibt einige automatische Funktionen die bei der Simulation eingebaut sind:

1. Die Ausführung des Programms wird automatisch abgebrochen, wenn der Roboter sich nach einem Durchlauf des gelegten Programmes an einer Stelle befindet, an der er bereits war und dabei auch in die selbe Richtung schaut. Kommt es zwischendurch zu einer Endlosschleife, wird dies nicht abgefangen.
2. Die Überprüfung, ob der Roboter sich auf dem Ziel befindet muss vom Benutzer im gelegten Programm vorgenommen werden. Dies ist mit dem Beispielprogramm wie in Abbildung 20 möglich.
3. Da explizite Pixel-Werte zum Bestimmen der Position der Blättchen in einer Reihe genutzt werden, muss das gelegte Programm parallel zu den Bildrändern verlaufen, da sonst bei einem langen Programm die Blättchen nicht mehr als in einer Reihe erkannt werden.
4. Die Simulation startet immer damit, dass der Roboter nach rechts schaut.
5. Um jedes erstellte Labyrinth wird automatisch eine Mauer erstellt.
6. Insgesamt ist das Programm nicht für alle Randfälle getestet und sollte möglichst in vorgesehener Weise genutzt werden.
7. Weitere Änderungen am Programm sind dem Entwickler vorbehalten und können unter dem GitHub (<https://github.com/MertenJakobza/BeLL>) abgerufen werden.

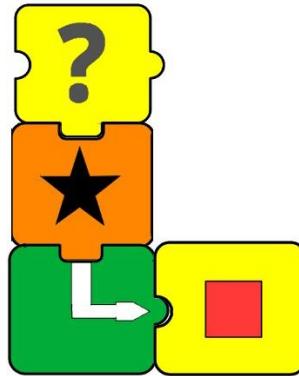


Abb. 20: Zielerkennung

8.1.2 Nutzung - Erklärung der Blättchen

Die meisten Blättchen sind selbsterklärend. So startet das Start-Blättchen (Abbildung 45) das Programm und das Stop-Blättchen (Abbildung 46) stoppt es. Hierbei zu beachten ist, dass das Stop-Blättchen nur die Funktion oder Programm stoppt, in welchem sich der Simulation-

Algorithmus befindet. Beispielsweise wird in Abbildung 21 in Methode A bei der Ausführung durch das Stopp-Blättchen nur die Methode A beendet und im Hauptprogramm wird die ganze Simulation beendet. Das Return-Blättchen (Abbildung 47) kehrt bei der Simulation an den Anfang des aktuellen Programmes zurück. Somit ist eine dauerhafte Ausführung des Programms möglich. Auch hier ist zu beachten, dass nur zum Anfang des aktuellen Programms zurückgekehrt wird. In Abbildung 21 wird also zum Anfang des Hauptprogramms zurückgekehrt. Wenn also ein Return-Blättchen sich in einer Methode befinden würde, würde der Simulation-Algorithmus an den Anfang der Methode zurückkehren.

Die Standardbewegungs-Blättchen, umfassen ein Vorwärts- (Abbildung 39), Rückwärts- (Abbildung 40), ein Links - (Abbildung 41) und ein Rechts - (Abbildung 42) Blättchen. Welche den Roboter vorwärts bzw. rückwärts bewegen oder ihn nach links oder rechts drehen.

An das If-Blättchen (Abbildung 30) können mehrere Bedingungen angehangen werden. Zum Beispiel könnte mit einem Blättchen abgefragt werden, ob der Roboter vor einer Wand (Abbildung 48) steht und auf dem Ziel (Abbildung 49) steht. Mit dem Not-Blättchen (Abbildung 44) kann der Wahrheitswert der Bedingungen umgedreht werden. Werden keine Bedingungen angegeben ist der Wahrheitswert standardmäßig wahr. An das Blink-Blättchen (Abbildung 32), das Schleifen-Blättchen (Abbildung 33) und das Warte-Blättchen (Abbildung 34) können Zahlen (bspw. Abbildung 43) angehangen werden. Mehrere Zahlen hintereinander werden dabei zusammen addiert, wodurch größere Zahlen möglich sind. In Abbildung 22 würde der Befehl 'gerade aus zu gehen' sieben mal ausgeführt werden. Das Blink-Blättchen ist dafür gedacht, dass der Roboter in der realen Welt blinkt. Da es jedoch keine Umsetzung in der realen Welt gibt, wird bisher nur in der Konsole 'Blink' ausgegeben. Die Anzahl der ausgegebenen 'Blinks' kann mit einem angehängten Zahlenblättchen verändert werden (bspw. sichtbar in Abbildung 21). Bei dem Schleifen-Blättchen handelt es sich bisher lediglich um eine Zählschleife, welche die folgenden Anweisungen so oft ausführt, wie mit den Zahlenblättchen angegeben (siehe Abbildung 22 bzw. 23).

Mit den DF-Blättchen (Abbildung 38) können Methoden definiert werden, welche dann mit den CF-Blättchen (Abbildung 36) aufgerufen werden können. Die Nutzung ist beispielsweise in Abbildung 21 zu sehen.

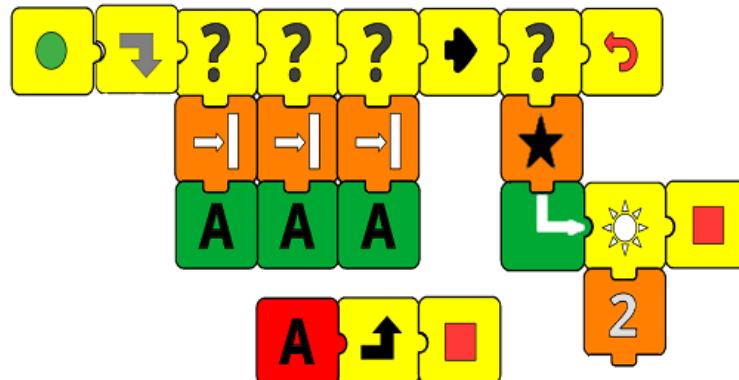


Abb. 21: Beispiel-Programm

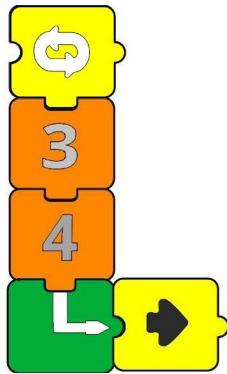


Abb. 22: Beispielprogramm-Schleife mit Sub



Abb. 23: Beispielprogramm-Schleife mit Methodenaufruf

8.1.3 Nutzung - Labyrinth-Editor

Als Erstes kann mit Hilfe des Labyrinth Editors ein Labyrinth erstellt werden. Der Editor ist über die Menubar unter dem Punkt Tools aufrufbar. Beim Klicken des Abbrechen-Buttons schließt sich das Fenster wieder, ohne eine Änderung am Hauptfenster vorzunehmen (Abbildung 24).

Mit der Höhe und Breite können die Dimensionen des Labyrinths eingestellt werden. Nach Klicken des 'Vorbereiten'-Button wird ein Feld aus Buttons erstellt, die im unteren Teil des Fensters dargestellt werden. Beim Klicken einer dieser Buttons wird der Button zunächst dunkelgrau, dann grün, gefolgt von gelb und wieder hellgrau. Dabei steht Dunkelgrau für eine Wand, Grün für den Start, Gelb für das Ziel und Hellgrau für nichts (Abbildung 25).

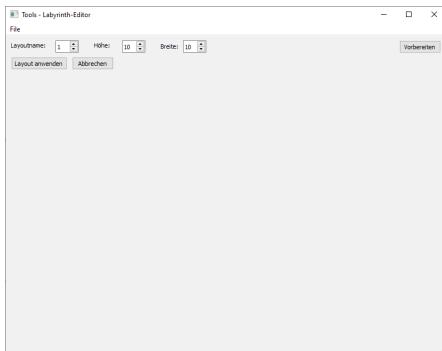


Abb. 24: Labyrinth-Editor nach Start

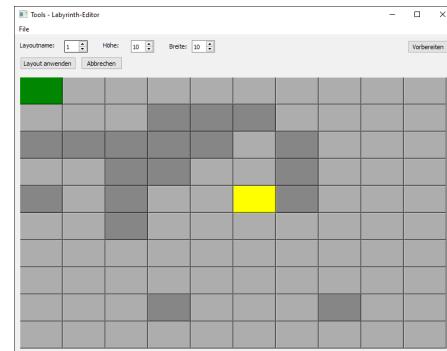


Abb. 25: Labyrinth-Editor

Unter 'File' kann ein Labyrinth-Slot geladen, gespeichert oder gelöscht werden. Dazu muss zunächst eines der 10 Slots beim 'Layoutname' ausgewählt werden. Nach dem Laden wird das Labyrinth mit den Buttons dargestellt.

Nach dem Anwenden des Layouts wird zunächst überprüft, ob es mindestens ein Ziel sowie genau einen Start gibt. Außerdem wird überprüft ob das Labyrinth überhaupt machbar ist. Außerdem wird dem Benutzer angezeigt, wie lang der kürzeste Weg vom Start zu einem Ziel ist.

8.1.4 Nutzung - Hauptprogramm

Nach dem Erstellen oder Laden eines Labyrinths, wird es mit Labels auf der rechten Seite dargestellt. Hier werden auch Texturen wie eine Zielflagge genutzt. Mit dem 'Bild aufnehmen'-Button auf der linken Seite kann ein neues Bild aufgenommen werden, welches in der oberen linken Ecke dargestellt wird (Abbildung 26). Mit dem 'Klassifizieren'-Button wird nun das Bild klassifiziert und die erkannten Blättchen unterhalb dargestellt. Das Ursprungsbild wird mit den erkannten Bounding-Boxen außerdem geupdated (Abbildung 27).

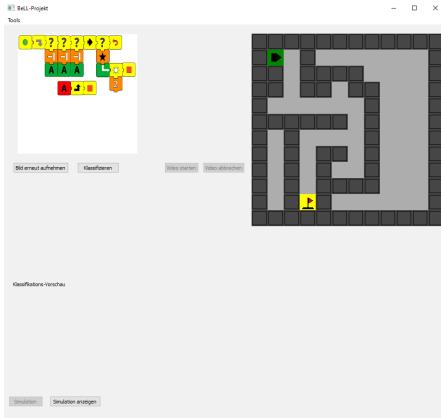


Abb. 26: Hauptseite des Programms nach Editor

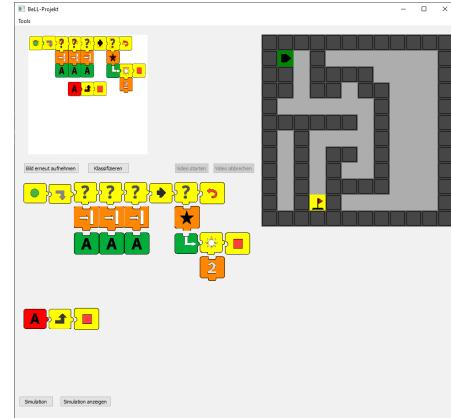


Abb. 27: Hauptseite des Programms nach Klassifikation

Beim Klicken des Buttons Simulation wird der Ablauf simuliert und das Ergebnis in dem Labyrinth rechts dargestellt (Abbildung 28).

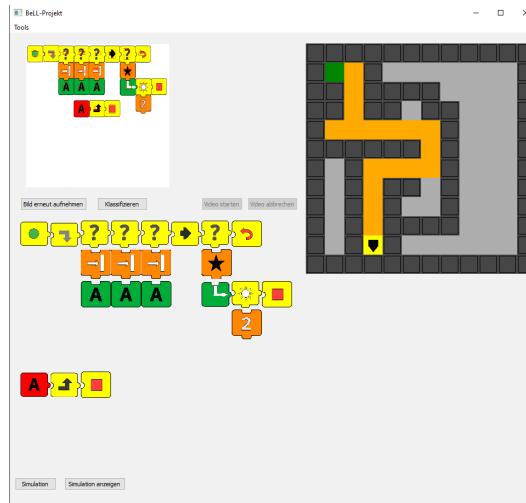


Abb. 28: Hauptseite des Programms nach Simulation

Mit dem Button 'Simulation anzeigen' wird die Simulation Schritt für Schritt angezeigt.

8.2 Programmierung

8.2.1 Programmierung - Labyrinth-Editor

Für das Einstellen der Dimensionen des Labyrinths, sowie die Auswahl des Slots wird eine Spin-Box genutzt. Der Nutzer kann das Programm dadurch nur mit der Maus benutzen. Es ist für die meisten jüngeren Nutzer deutlich intuitiver und einfacher, eine Maus zu nutzen als eine Tastatur. Für die Buttons zum Erstellen des Labyrinths wird eine eigene Klasse genutzt, welche von der PyQt5-Button-Klasse vererbt ist. Die Klasse kann wie in Quellcode 26 gezeigt, erstellt werden.

```
1 class mybutton(QPushButton):
2
3     colors = [[0, "darkgrey"], [1, "grey"], [2, "green"], [3, "yellow"]]
4
5     def __init__(self, posx, posy, width, height, editorwin: QMainWindow, px, py):
6         :
7         QPushButton.__init__(self, editorwin)
8         self.setGeometry(px, py, width, height)
9         self.setStyleSheet(f"background-color: {self.colors[0][1]}")
10        self.show()
11        self.clicked.connect(lambda: self.anpassen())
```

Quellcode/Command 26: Button-Klasse für Labyrinth-Editor

Dabei wird im Konstruktor zunächst der Konstruktor des QPushButtons aufgerufen. Es wird der Button auf das QMainWindow des Editors geladen und die Größe und Farbe angepasst. Zum Schluss wird festgelegt, dass beim Klicken auf den Button die Funktion 'anpassen' aufgerufen wird, damit die Farbe des Buttons beim mehrmaligen Klicken angepasst werden kann. Dies geschieht indem der Button die Farben in der Farbenliste nacheinander annimmt. Interessanterweise ist in PyQt5 die Darstellung für 'darkgrey' heller als 'grey', weshalb 'darkgray' für den Hintergrund und 'gray' für die Wände genutzt wird.

Beim Anklicken des Anwenden-Buttons wird basierend auf den Farben der Buttons das Labyrinth in Form einer zweidimensionalen Liste erstellt. Wobei eine Null für Nichts bzw. Hintergrund, eine Eins für eine Wand, eine Zwei für einen Start und eine Drei für ein Ziel steht. Außerdem wird hier auch überprüft, dass es genau einen Start, mindestens ein Ziel und mindestens einen möglichen Weg vom Start zu einem Ziel gibt. Hierzu wurde ein Algorithmus entwickelt (siehe Kapitel 8.2.2), um einen Weg zu finden.

Für die Darstellung des Labyrinths im Hauptprogramm werden Labels (siehe Quellcode 27) genutzt, welche auch Bilder anzeigen können. Die Farbe des Labels kann geändert werden indem erst der 'Value' des Labels geändert wird. Daraufhin wird die Funktion update_color() aufgerufen, welche die Farbe ändert.

```

1 class mylabel(QLabel):
2     colors = ["darkgrey", "grey", "green", "yellow", "", "orange", ""]
3     def __init__(self, posx, posy, width, height, editorwin: QMainWindow, px, py,
4                  color):
5         QLabel.__init__(self, editorwin)
6         self.posx = posx
7         self.posy = posy
8         self.height = height
9         self.width = width
10        self.setGeometry(px, py, width, height)
11        self.value = color
12        self.update_color()
13        self.show()
14    def update_color(self):
15        self.setStyleSheet(f"background-color: {self.colors[self.value]}")
16    def set_image(self, img_dir, scale_by_height: bool):
17        pixmap = QPixmap(img_dir)
18        if scale_by_height:
19            self.setPixmap(pixmap.scaledToHeight(self.height))
20        else:
21            self.setPixmap(pixmap.scaledToWidth(self.height))

```

Quellcode/Command 27: Label-Klasse für Hauptprogramm

8.2.2 Programmierung - Pathfinding-Algorithmus

Am Anfang werden von der Startposition aus alle Felder, in denen sich keine Wand befinden, mit einer 4 markiert. Nun startet eine Schleife, die in jeder Iteration zunächst die Koordinaten aller Felder zwischenspeichert, in denen sich am Anfang eine 4 befunden hat. Dann werden für jede dieser Koordinaten wiederum alle umliegenden Felder, die nicht eine Wand sind mit einer 4 markiert. Die Koordinate selbst wird als 5 markiert, damit klar ist, dass der Algorithmus an dieser Stelle bereits war. Wenn das Ziel mit diesem Algorithmus erreicht wird, kann das Labyrinth als machbar markiert und die Schleife abgebrochen werden. Um herauszufinden, ob ein Labyrinth nicht machbar ist, wird nach jeder Iteration überprüft, ob es noch mindestens eine 4 im Labyrinth gibt. Wenn dies der Fall ist, hat der Algorithmus noch nicht das ganze Labyrinth durchlaufen. Wenn keine 4 mehr zu finden ist, wurden alle möglichen Wege bereits abgelaufen und das Labyrinth ist somit nicht machbar. Um die Länge des Weges herauszufinden, wird eine Zählvariable in der Schleife mitgeführt. Der Wert dieser Variable gibt die Anzahl der Schritte, die der Algorithmus bereits gelaufen ist, an. Am Ende der Schleife kann also diese Zählvariable als Länge des kürzesten Weges genutzt werden.

Ist das Labyrinth nicht machbar, wird dies dem Nutzer über einen Dialog mitgeteilt. Dieser kann Verbesserungen am Labyrinth vornehmen oder ein völlig anderes laden.

8.2.3 Programmierung - Klassifikation

Nach dem Anwenden des Labyrinth-Layouts wird dieses auf der rechten Seite mit Hilfe einer von QLabel abgeleiteten Klasse dargestellt. Das Erstellen dieser Klasse verläuft analog zur Button-Klasse. Mit den Labels ist es nun auch möglich Bilder darzustellen. Somit können kleine

eigens gezeichnete Bilder zur Illustration des Zielfeldes, des Roboters und der Wände genutzt werden. Diese werden, wie in Quellcode 28 dargestellt, geladen.

```
1 pixmap = QPixmap(img_dir)
2 self.setPixmap(pixmap.scaledToHeight(self.height))
```

Quellcode/Command 28: Code zum Darstellen von Bildern in der GUI

Beim Klicken des 'Bild aufnehmen' Buttons wird ein Bild mit dem in Kapitel 3.1.2 bereits beschriebenen Code aufgenommen und mit dem Klicken auf den 'Klassifizieren'-Button mit dem Code aus Kapitel 5.1.2 klassifiziert.

Ein gekürztes Ergebnis der Klassifikation von Abbildung 21 ist beispielsweise in Quellcode 29 zu sehen. Die Ergebnisse werden an der Stelle des Ursprungsbildes dargestellt. Der Entwickler kann somit sehen, welche Blättchen eventuell nicht erkannt wurden.

```
1 res = [{"height": 8, "label": "S01-Start", "value": 0.941252589225769, "width": 8, "x": 16, "y": 16},
2         {"height": 8, "label": "M04-Right", "value": 0.9913778305053711, "width": 8, "x": 40, "y": 16},
3         {"height": 8, "label": "C03-If", "value": 0.998925507068634, "width": 16, "x": 64, "y": 16},
4         {"height": 8, "label": "C03-If", "value": 0.9999614953994751, "width": 8, "x": 96, "y": 16},
5         {"height": 8, "label": "C03-If", "value": 0.9995933175086975, "width": 8, "x": 120, "y": 16},
6         {"height": 8, "label": "M01-Forward", "value": 0.9994913339614868, "width": 8, "x": 152, "y": 16},
7         {"height": 8, "label": "C03-If", "value": 0.999528169631958, "width": 8, "x": 176, "y": 16},
8         {"height": 8, "label": "S04-Return", "value": 0.9992904663085938, "width": 16, "x": 200, "y": 16},
9         {"height": 16, "label": "Ziel", "value": 0.9997374415397644, "width": 8, "x": 176, "y": 40}]
```

Quellcode/Command 29: Ergebnis der Klassifikation

8.2.4 Programmierung - Umwandeln der Bounding-Boxen in 2D-Darstellung des Programms

Bevor das Programm umgewandelt wird, werden alle Bounding-Boxen entfernt, dessen Confidence-Wert unter eine eingestellte Grenze von standardmäßig 80 Prozent fallen.

Für das Umwandeln wird zunächst nach der Position des Start-Blättchens gesucht. Nun kann der Nutzer darauf hingewiesen werden, falls nicht genau ein Start-Blättchen erkannt wurde. Nun wird eine rekursive Methode genutzt. Diese Funktion nutzt eine X- und eine Y-Koordinate als Übergabewert, um durch die erkannten Blättchen zu iterieren und in einer zweidimensionalen Liste zu speichern. Diese Methode fängt an mit dem übergebenen Y-Wert alle Blättchen zu suchen, dessen Y-Wert innerhalb eines Toleranzbereichs von +16 Pixeln liegt.

Kommt die Methode beim Durchzählen an ein Blättchen, welches einen eckigen Konnektor nach unten besitzt, (Abbildung 30), dann beginnt das Programm sich an der Stelle nach unten zu bewegen. Hierfür wird der X-Wert des zum Beispiel If-Blättchens genutzt. Dazu wird eine weitere Liste erstellt, welche die Blättchen nach unten fortlaufend darstellt.

Trifft die Methode nun auf ein 'Sub'-Blättchen, welches die Schnittstelle zwischen einem nach unten verlaufendem Teil und einem nach rechts laufendem Teil darstellt (Abbildung 29), wird an der X- und Y-Koordinate des 'Sub'-Blättchens die Methode selbst wieder aufgerufen. Dadurch ist eine theoretisch unendlich lange Verkettung des Programms möglich (Abbildung 31).



Abb. 29: Sub-Blättchen



Abb. 30: Blättchen mit eckigen Konnektor

Ist die Methode mit jeglicher Bewegung nach unten fertig, liegt eine zweidimensionale Liste vor. Diese stellt den Teil dar, welcher sich nach unten bewegt. Sie würde in dem Fall von oben wie folgt aussehen.

`[[19, 21, 23], [0, 0, 19, 21, 23]]`

Diese Sub-Liste wird nun mit dem großen Programm zusammengefügt. Hierzu wird eine weitere Methode genutzt, welche zwei Listen als Parameter erhält und dabei die erste Liste in die zweite an eine angegebene Stelle einfügt. Dazu wird die zweite Liste zunächst so erweitert, dass sie groß genug ist. Anschließend werden die Werte von der ersten Liste in die zweite Liste übertragen.

Das Ergebnis der Methode ist eine Liste, welche das Programm zweidimensional darstellt. Beispielsweise ist das Ergebnis für das Beispielprogramm in Abbildung 21 die Liste aus Quellcode 30.

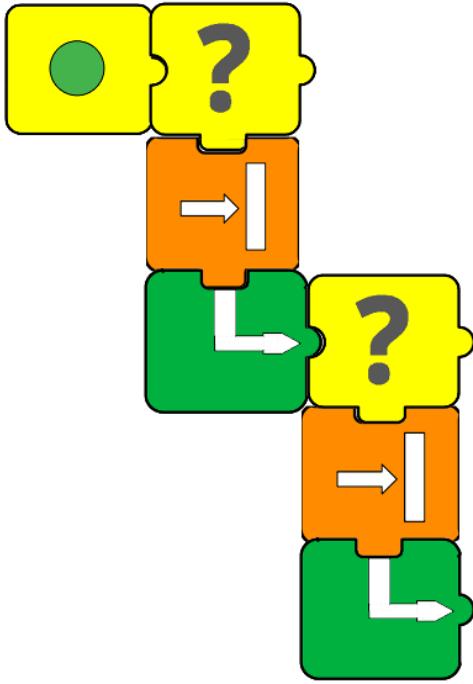


Abb. 31: Beispiel Verkettung von Konnektoren

```

1 [[1, 0, 0, 0, 0, 11, 0],
2 [28, 0, 0, 0, 0, 27, 0],
3 [7, 8, 10, 0, 0, 2, 0],
4 [7, 8, 10, 0, 0, 0, 0],
5 [7, 8, 10, 0, 0, 0, 0],
6 [25, 0, 0, 0, 0, 0, 0],
7 [7, 9, 29, 0, 0, 0, 0],
8 [3, 0, 4, 17, 0, 0, 0],
9 [0, 0, 2, 0, 0, 0, 0],
10 [0, 0, 0, 0, 0, 0, 0]]

```

Quellcode/Command 30: Ergebnis-Liste

8.2.5 Programmierung - Simulation des Programmes

Hierbei handelt es sich um eine deutlich erweiterte Version des Umwandel-Algorithmus.

Der Simulations-Funktion wird immer die Position des Roboters, die Position im Prorammblauf, die aktuelle Ausrichtung des Roboters und ob sich die Simulation im Hauptprogramm befindet übergeben.

Beim ursprünglichen Start der Simulation wird die Methode mit den Koordinaten des Start-Blättchens in der 2D-Darstellung des Programms aufgerufen. Nun läuft der Algorithmus die Blättchen horizontal ab und führt die Aktionen aus, die die Blättchen definieren. Trifft das Programm beispielsweise auf ein If-Blättchen, bewegt sich die Simulation nach unten. Es werden alle Bedingungen gesammelt und evaluiert, ob alle Bedingungen wahr oder falsch sind. Ist ein Not-Blättchen innerhalb dieser Bedingungen, wird der Wahrheitswert der zu vorigen Bedingungen umgedreht. Nur wenn das Ergebnis wahr ist, wird das darauf folgende ausgeführt. Dies kann entweder der Aufruf einer Methode sein oder ein Sub-Blättchen. In beiden Fällen wird die Funktion selbst wieder aufgerufen und dabei die nötigen Parameter übergeben.

Trifft die Simulation auf ein Return-Blättchen wird ein Parameter 'repeat' auf True gesetzt. Die Funktion, die die aktuelle Funktion aufgerufen hat, weiß nun, dass die Funktion wiederholt werden soll. Beim Aufruf einer Methode A, B oder C wird diese direkt wiederholt. Bei einem Sub-Blättchen wird die Funktion wiederholt, in der sich das Sub-Blättchen befindet. Hier wird auch überprüft, ob sich der Roboter in einer Endlosschleife befindet, indem die aktuelle Position mit einer Liste aller besuchten Positionen verglichen wird. Wenn dies nicht der Fall ist, wird die aktuelle Position, sowie Ausrichtung in einer Liste 'Weg' gespeichert. Diese Liste wird erstens

zum eben beschriebenen Abbrechen bei einer Endlosschleife genutzt und zweitens später zur Visualisierung der Simulation.

Bei den Bewegungs-Blättchen (Vorwärts und Rückwärts) wird ebenfalls die alte Position in dem Labyrinth vermerkt, damit am Ende dargestellt werden kann, wo sich der Roboter entlang bewegt hat.

Läuft die Simulation aus, wird überprüft, ob sich der Roboter auf dem Ziel befindet. Dem Nutzer wird zurückgemeldet, ob der Roboter es auf das Ziel geschafft hat. Außerdem wird der gesamte Weg, den der Roboter genommen hat, dargestellt. Alle besuchten Felder werden orange eingefärbt (siehe Abbildung 28).

Für das Anzeigen der Simulation wird mit jedem Button-Klick ein Schritt weiter in der Weg-Liste gegangen. Die aktuelle Situation wird auf der rechten Seite dargestellt.

9 Ergebnisse

Die neusten Versionen der entstandenen Programme sind unter <https://github.com/MertenJakobza/BeLL> zu finden.

Im Laufe der Arbeit hat sich gezeigt, dass das Arbeiten mit dem Raspberry Pi am leichtesten fällt. Hier steht ein vollständiges Betriebssystem zur Verfügung, auf dem mit Hilfe von Programmen direkt gearbeitet werden kann. Dagegen muss auf dem ESP-32 und dem UnitV nach jeder Änderung des Programmes oder der KI eine neue Firmware geladen werden.

Die Erkennungszeit für ein Modell, was 224x224 Pixel große Bilder klassifiziert, beträgt auf dem ESP-32-CAM rund 4300 ms (0.23 FPS). Bei größeren Bildern reicht der interne Arbeitsspeicher nicht mehr aus, um die Bilder zwischenzuspeichern. Das Maximum liegt hier bei 230x230 Pixel. Zur Erkennungsrate lassen sich keine Aussagen treffen. Es kann nicht eingesehen werden, was auf dem Bild zu sehen ist und was das Modell erkennen sollte.

Bei dem UnitV dauert die Ausführung des selben Modells 500 Millisekunden (2 FPS). Auch hier lässt sich, wie beim ESP-32-CAM, nicht sagen, wie hoch die Erkennungsrate ist.

Auf dem Raspberry Pi liegt die Erkennungszeit hingegen nur bei 40 ms (25 FPS) für 224x224 Pixel große Bilder. Auf den Test-Bildern lässt sich sehen, dass die Ergebnisse meist mit den Ergebnissen von der Edge Impulse -Webseite übereinstimmen und somit die Erkennung gut funktioniert.

Somit lässt sich sagen, dass der Raspberry Pi nicht nur das am besten zu nutzende eingebettete System ist, sondern gleichzeitig auch das schnellste. Dies kann auf die gute Optimierung durch Edge Impulse zurückzuführen sein. Die niedrige Erkennungszeit erlaubt eine intuitive Benutzung der Bilderkennung. Der UnitV liefert trotz der Tensor-Processing-Unit keine deutlich Verbesserung der Erkennungszeit. Die Vermutungen aus Kapitel 3.4 zum ESP-32 sind bestätigt, da hier die Geschwindigkeit nicht besonders hoch ist.

Das daraus trainierte Modell erkennt mit vom Raspberry Pi aufgenommenen Bildern das Test-Programms (Abbildung 21) sicher nach ca. 10 Versuchen (Visualisierung in Abbildung 27). Die Grundlage für die Erkennung, ist das mit Edge Impulse erstellte Modell. Dieses erreicht eine theoretische Erkennungsrate von 98.4 Prozent mit künstlich erzeugten Bildern (siehe Kapitel 6).

10 Erweiterungsmöglichkeiten

Das Ergebnis-Programm ließe sich in verschiedenster Weise verbessern. Zum einen könnten Live-Interaktionen implementiert werden. Damit ist nicht nur gemeint, dass Änderungen am gelegten Programm direkt erkannt und simuliert werden, sondern Live-Interaktion. Beispielsweise könnte ein Drücken auf das grüne Symbol des Start-Blättchen die Simulation starten. Eine Erweiterung könnte erlauben, dass schiefe Aufnahmen durch den Nutzer möglich wären. Als letztes könnte ein leeres Blättchen in den Blättchen-Satz aufgenommen werden, damit alle Varianten gelegt werden können.

Für die Erstellung des Modells könnte neben der Nutzung von Edge Impulse zum KI-Training auch 'TensorFlow Model Maker' genutzt werden um die KI selbst zu trainieren, was mehr Kontrolle über die KI selbst erlauben würde. Somit wäre die Implementierung auf dem UnitV leichter, da wahrscheinlich weniger Fehler beim Umwandeln in das .kmodel auftreten würde.

Weitere Ansätze zur Auswertung eines Bildes umfassen beispielsweise das Feature Extracting und das 'Histogram of oriented Gradients', kurz HOG. Diese Ansätze wurden bisher noch gar nicht betrachtet, sehen jedoch äußerst vielversprechend aus.

Literaturverzeichnis

- Ai-Thinker Technology (o.D.). ESP32-CAM camera development board. <https://docs.ai-thinker.com/en/esp32-cam>. (18-12-2023).
- Rahul Awati (10.2023). embedded device. <https://www.techtarget.com/whatis/definition/embedded-device>. (18-12-2023).
- Edge Impulse Inc. (07.09.2022). Transfer learning (Keyword Spotting). <https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/transfer-learning-few-shot>. (18-12-2023).
- Edge Impulse Inc. (10.11.2023). Edge Impulse Python SDK. <https://docs.edgeimpulse.com/docs/tools/edge-impulse-python-sdk#quantization>. (18-12-2023).
- Edge Impulse Inc. (22.05.2023). Supervised or unsupervised? <https://forum.edgeimpulse.com/t/supervised-or-unsupervised/7643>. (18-12-2023).
- Edge Impulse Inc. (06.09.2023). FOMO: Object detection for constrained devices. <https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection/fomo-object-detection-for-constrained-devices>. (18-12-2023).
- Edge Impulse Inc. (14.09.2023). Linux Python SDK. <https://docs.edgeimpulse.com/docs/tools/edge-impulse-for-linux/linux-python-sdk>. (18-12-2023).
- Edge Impulse Inc. (18.08.2023). Arduino library. <https://docs.edgeimpulse.com/docs/run-inference/arduino-library>. (18-12-2023).
- Edge Impulse Inc. (18.08.2023). C++ Library. <https://docs.edgeimpulse.com/docs/run-inference/cpp-library/deploy-your-model-as-a-c-library>. (18-12-2023).
- Edge Impulse Inc. (19.05.2022). Lower compute time. <https://docs.edgeimpulse.com/docs/tips-and-tricks/lower-compute-time>. (18-12-2023).
- Edge Impulse Inc. (2023). We're building the future of data-driven engineering. <https://edgeimpulse.com/about>. (18-12-2023).
- Edge Impulse Inc. (28.06.2023). As a generic C++ library. <https://docs.edgeimpulse.com/docs/run-inference/cpp-library/deploy-your-model-as-a-c-library>. (18-12-2023).
- Edge Impulse Inc. (28.08.2023). Uploader. <https://docs.edgeimpulse.com/docs/tools/edge-impulse-cli/cli-uploader>. (18-12-2023).
- Edge Impulse Inc. (28.09.2023). Edge Impulse for Linux. <https://docs.edgeimpulse.com/docs/tools/edge-impulse-for-linux>. (18-12-2023).
- Geizhals.de (o.D.). Rapsberry Pi 3 Modell A+ 512MB RAM. <https://geizhals.de/raspberry-pi-3-modell-a-a1926467.html>. (18-12-2023).
- Sierra Rogers Ines Bahr (13.12.2021). Überwachtes und unüberwachtes Lernen: Welches Machine Learning-Modell ist das Richtige für dich? <https://www.capterra.com.de/blog/2352/uberwachtes-und-unuberwachtes-lernen>. (18-12-2023).

ITWissen.info (09.10.2017). Einplatinencomputer. <https://www.itwissen.info/Einplatinencomputer-single-board-computer-SBC.html>. (18-12-2023).

Rohit Kundu (16.12.2022). F1 Score in Machine Learning: Intro & Calculation. <https://www.v7labs.com/blog/f1-score-guide>. (18-12-2023).

M5Stack (o.D.). UnitV. https://docs.m5stack.com/en/unit/unitv_ov7740. (18-12-2023).

OmniVision Technologies Inc. (11.03.2009). 1/4"color CMOS QSXGA (5 megapixel) image sensor with OmniBSI technology. https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/ov5647_full.pdf. (18-12-2023).

OmniVision Technologies Inc. (23.02.2007). OV2640 Color CMOS UXGA (2.0 MegaPixel) CameraChip Sensor with OmniPixel2 Technology. https://www.uctronics.com/download/OV2640_DS.pdf. (18-12-2023).

OmniVision Technologies Inc. (29.07.2008). 1/5"CMOS VGA (640x480) CameraChip sensor with OmniPixel3-HS technology. https://www.dragonwake.com/download/camera/vc0706_spi/0V7740_CSP.pdf. (18-12-2023).

Raspberry Pi Foundation (11.2018). Raspberry Pi 3 Model A+ Product Brief. https://www.raspberrypi.org/app/uploads/2018/11/Raspberry_Pi_3A_product_brief.pdf. (18-12-2023).

Reichelt.de (o.D.). DEBO CAM ESP32 Entwicklerboards
- ESP32-Kamera 2MP 25°. <https://www.reichelt.de/entwicklerboards-esp32-kamera-2mp-25--debo-cam-esp32-p266036.html>. (18-12-2023).

Adrian Rosebrock (20.03.2015). Accessing the Raspberry Pi Camera with OpenCV and Python. <https://pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/>. (18-12-2023).

roshalmoraes (10.05.2023). hostname command in Linux with examples. <https://www.geeksforgeeks.org/hostname-command-in-linux-with-examples/>. (18-12-2023).

sipeed (o.D.). Basic knowledge of MaixPy AI hardware acceleration. https://wiki.sipeed.com/soft/maixpy/en/course/ai/basic/maixpy_hardware_ai_basic.html. (18-12-2023).

Steins (12.09.2021). The Ultimate Handbook for OpenCV and Pillow. <https://medium.com/analytics-vidhya/the-ultimate-handbook-for-opencv-pillow-72b7eff77cd7>. (18-12-2023).

TutorialsForRaspberryPi (o.D.). Aufnahmen mit dem offiziellen Kamera Modul des Raspberry Pi. <https://tutorials-raspberrypi.de/aufnahmen-mit-dem-offiziellen-kamera-modul-des-raspberry-pi/>. (18-12-2023).

vowstar (20.06.2022). Cross platform GUI wrapper for kflash.py. https://github.com/sipeed/kflash_gui. (18-12-2023).

Walber (22.11.2014). Precision and recall. <https://commons.wikimedia.org/wiki/File:Precisionrecall.svg>. (18-12-2023).

Olga Weis (23.06.2020). Pinbelegung und Verwendung des RS-232 DB9. <https://www.eltima.com/de/article/9-pin-serial-port.html>. (18-12-2023).

WinTotal.de (01.04.2019). Zugriff auf den Raspberry Pi über den Remote Desktop von Windows 10. <https://www.wintotal.de/tipp/raspberry-pi-remote-desktop/>. (18-12-2023).

Dr. Anna Wolf (02.08.2023). 13,3 Prozent der Unternehmen in Deutschland nutzen Künstliche Intelligenz. <https://www.info.de/fakten/2023-08-02/unternehmen-deutschland-nutzen-kuenstliche-intelligenz>. (18-12-2023).

Laurenz Wuttke (2023). Deep Learning: Definition, Beispiele und Frameworks. <https://datasolut.com/was-ist-deep-learning/>. (18-12-2023).

Zeit-Online (06.11.2023). Schüler, Azubis und Studenten nutzen KI für Hausarbeiten. <https://www.zeit.de/news/2023-11/06/schueler-azubis-und-studenten-nutzen-ki-fuer-hausarbeiten>. (18-12-2023).

Anhang

Veranschauungsbilder für die Blättchen

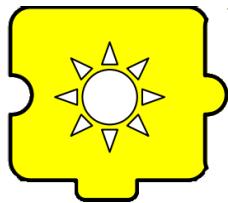


Abb. 32: Blink-Blättchen

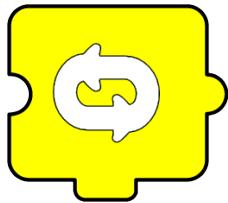


Abb. 33: C01-Loop-Blättchen

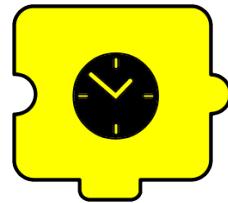


Abb. 34: C02-Wait-Blättchen



Abb. 35: C01-If-Blättchen



Abb. 36: CF-A-Blättchen



Abb. 37: CF-Sub-Blättchen

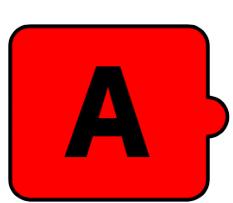


Abb. 38: DF-A-Blättchen

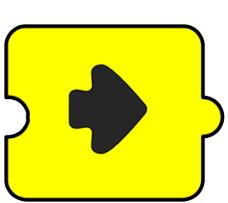


Abb. 39: M01-Forward-Blättchen

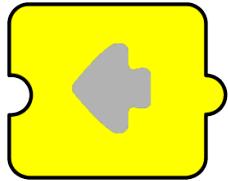


Abb. 40: M02-Backward-Blättchen

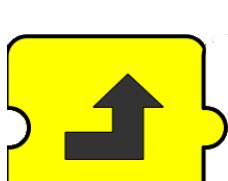


Abb. 41: M03-Left-Blättchen

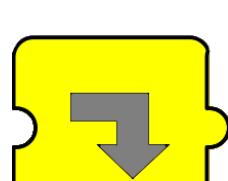


Abb. 42: M04-Right-Blättchen



Abb. 43: N01-Blättchen



Abb. 44: Not-Blättchen

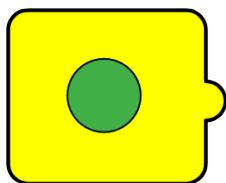


Abb. 45: S01-Start-Blättchen

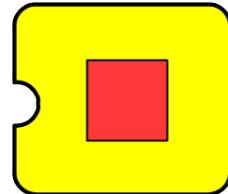


Abb. 46: S02-End-Blättchen

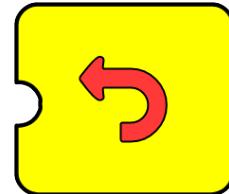


Abb. 47: S04-Return-Blättchen

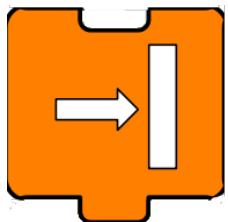


Abb. 48: W01-Wand-Blättchen

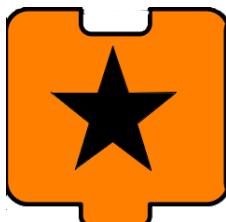


Abb. 49: Ziel-Blättchen

Danksagung

Mein besonderer Dank gilt meinem externen Betreuer Prof. Dr-Ing. Axel Klarmann. Bei ihm möchte ich mich für das interessante Thema bedanken, welches er bereitgestellt hat. Sowie für die besonders ausführliche Strukturierung des Themas, an die ich mich halten konnte. Er stand mir bei jedem Problem fachkundig zur Seite und gab mir auch äußerst wertvolle Hinweise für die konkrete Umsetzung meines Themas. Das Korrekturlesen half einige Fehler der schriftlichen Dokumentation auszubessern.

Die Hochschule für Technik, Wirtschaft und Kultur hat mir durch die Bereitstellung eines Arbeitsplatzes zur effektiven Arbeit verholfen.

Ebenfalls möchte ich mich bei meinem internen Betreuer Herrn Robert Karl bedanken. Er stand mir beim formellen Ausarbeiten der BeLL zur Seite und hat maßgeblich zur konkreten Umsetzung beigetragen. Außerdem stand er für immer für Rückfragen bereit.

Selbstständigkeitserklärung

Hiermit erkläre ich, Merten Jakobza, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Insbesondere versichere ich, dass wir alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche gekennzeichnet habe. Ich bin mit einer schulinternen Verwendung meiner Arbeit einverstanden.

Ort, Datum

Unterschrift