



University of Applied Sciences and Arts Northwestern Switzerland FHNW

Master of Science in Engineering

Edge ML Camera for Citizen Science

Edge ML Kamera für Citizen Science

Author: Andri Wild

Supervisor: Prof. Thomas Amberg

Advisors: Prof. Thomas Amberg

Start Date: 01.01.2024

Submission Date: 01.01.2024

I confirm that this Master of Science in Engineering's thesis is my own work and
I have documented all sources and material used.

Munich, 01.01.2024

Andri Wild

Acknowledgements

First, I'd like to thank coffee for fueling my brain cells and making this thesis possible.

A big shoutout to my advisor for your patience and for not laughing (too hard) at my wild ideas.

To my family, your snack supplies and constant reminders to "just finish it already" were invaluable.

Finally, to my pet, your keyboard sit-ins ensured I took breaks, whether I wanted to or not.

Abstract

Note:

1. **paragraph:** What is the motivation of your thesis? Why is it interesting from a scientific point of view? Which main problem do you like to solve?
2. **paragraph:** What is the purpose of the document? What is the main content, the main contribution?
3. **paragraph:** What is your methodology? How do you proceed?

Zusammenfassung

Note: Insert the German translation of the English abstract here.

Inhaltsverzeichnis

1 Einleitung	9
1.1 Ausgangslage	9
1.2 Zielsetzung	9
1.3 Abgrenzung	10
1.4 Aufbau des Berichts	10
2 Grundlagen	11
2.1 Citizen Science	11
2.2 Biodiversität Monitoring	14
2.3 Machine Learning Grundlagen	15
2.3.1 ML Frameworks	16
2.3.1.1 TensorFlow	16
2.3.1.2 TensorFlow Lite	16
2.3.1.3 PyTorch	16
2.3.1.4 ONNX	16
2.3.1.5 NCNN	17
2.3.1.6 Hailo	17
2.3.1.7 Ultralytics	17
2.3.1.8 Trade-Off's	18
2.3.2 Anatomie von Machine Learning Modellen	18
2.3.3 Geschwindigkeit einer Inferenz	20
2.4 Edge Computing	21
2.4.1 Edge ML	22
2.4.2 Host Systeme	22
2.4.2.1 Raspberry Pi 4	22
2.4.2.2 Raspberry Pi 5	23
2.4.2.3 BeagleY-AI	23
2.4.3 Hardware Beschleuniger	24
2.4.3.1 Coral USB Accelerator	24
2.4.3.2 Coral PCI Accelerator	25

2.4.3.3 Hailo TPU	25
2.4.3.4 AI-Kamera	26
3 Analyse	28
3.1 Zielgruppe	28
3.2 Use Cases	29
3.2.1 Referenz Use Case	30
3.3 Anforderungen	30
3.3.1 Funktionale	31
3.3.2 Nicht Funktionale	31
3.4 Evaluation	31
3.4.1 Methodik	31
3.4.2 ML Framework	32
3.4.2.1 Vergleich Inferenzzeiten	32
3.4.3 Hardware	33
3.4.3.1 Raspberry Pi Vergleiche	33
3.4.3.2 Coral Accelerator	34
3.4.3.3 Hailo	35
3.5 Edge ML Setups	36
3.6 Schlussfolgerung	37
4 Umsetzung	38
4.1 System Architektur	38
4.1.1 Komponenten Diagram	38
4.1.2 Qualitäten	39
4.1.3 Konfiguration	39
4.2 Software Design	40
4.3 Prototypen: verschiedene System Setups	42
4.3.1 Prototyp 1	42
4.3.2 Prototyp 2	43
4.3.3 Prototyp 3	43
4.3.4 Prototyp 4	43
5 Validierung	43
Abbildungsverzeichnis	44
Tabellenverzeichnis	46

Appendix A: Supplementary Material	47
Bibliographie	48

1 Einleitung

1.1 Ausgangslage

- Mitwelten Projekt (Ziel, Kontext)
- Analyse Bestäuber (Kurzbeschreibung der Mitwelten Pipeline wie sie eingesetzt war)

Das SNF Projekt „Mitwelten - Medienökologische Infrastrukturen für Biodiversität“ hat IoT-Technologie in urbanen Naturgebieten installiert, um dort vorhandene Tiere, Pflanzen und Umweltbedingungen genauer zu untersuchen. Ein Teilprojekt beschäftigte sich mit der Erkennung von Bestäuber-Arten auf Blumenblüten mittels Raspberry Pi Kameras. Die Machine-Learning-basierte Detektion und Kategorisierung geschieht zurzeit in einem zentralen Cloud-Backend.

Diese Architekturlösung bringt folgende Konsequenzen mit sich: Zum Einen muss der Betrieb eines Backends sichergestellt sein. Dies erfordert einen gewissen Wartungsaufwand und insbesondere ein erhöhtes technisches Verständnis für die Projektumsetzung. Für Forschende bedeutet dies, dass zusätzliches IT-Fachpersonal für den Aufbau und Betrieb der IT-Infrastruktur benötigt wird. Zum Anderen werden die aufgenommenen Bilder zur Analyse über Mobilfunknetze versendet, wodurch die Betriebskosten erhöht werden und das Einsatzgebiet auf dessen Abdeckung begrenzt. Beide der genannten Aspekte sind mit zusätzlichen Kosten verbunden und erhöhen die Gesamtkomplexität des Projekts.

1.2 Zielsetzung

- Fragestellung / Hypothese

Das Ziel ist die Portierung der ML-Pipeline des SNF Projekts Mitwelten auf eine Edge ML Kamera, um Citizen Science Projekte zu ermöglichen:

Dank der fortschreitenden Entwicklung von Edge-Computing-Hardware sind auf Machine Learning spezialisierte Geräte zu einem vergleichbaren Preis wie konventionelle Edge Computer erhältlich. Dies wirft die Frage auf, ob durch diesen technologischen Fortschritt die IT-Infrastruktur eines Edge-Kamera basierenden Systems so weit reduziert werden kann, dass die selbe Funktionalität ohne zentrales Backend umsetzbar ist. Der Einsatz von Edge-basierten Systemen könnte den Zugang zu Machine Learning Projekten für Forschende erheblich erleichtern, da ein System ohne zentrale Abhängigkeit mit weniger

Aufwand betreibbar ist. Diese stand-alone Lösung öffnet auch die Türen für Citizen Science Projekte. Einzelpersonen oder Gemeinschaften können so eigenständig und unabhängig Forschungsprojekte durchführen.

- Welche Hardware und Frameworks gibt es, um Machine Learning (ML) dezentral, im Feld, mittels Edge Computing umzusetzen?
- Was sind Anforderungen an eine ML-Kamera für typische Citizen Science Anwendungen im Bereich Biodiversitätsmonitoring?
- Wie sieht eine konkrete Edge ML Kamera aus, für Monitoring von Biodiversität, wie im Projekt SNF Mitwelten angewendet?

1.3 Abgrenzung

Ein System zur Analyse der Biodiversität mittels Machine Learning beinhaltet viele verschiedene Stellschrauben. In diesem Projekt liegt der Fokus auf der aktuellsten Hardware und wie diese für Edge ML eingesetzt werden kann. Nicht Teil dieses Projekt ist das Trainieren von Modellen.

1.4 Aufbau des Berichts

Der Bericht besteht im wesentlichen aus vier Teilen. Im ersten teil werden die grundlagen vermittelt, welche relevant sind für das Verständnis des Berichts. Anschliessend folgt die Analyse, welche die Zielgruppen und Anforderungen definiert. In der Umsetzung wird die konkrete Umsetzung des Projekts beschrieben und abschliessend wird die Evaluation der Resultate vorgenommen.

RPi5 mit 16GB wäre verfügbar

2 Grundlagen

(Dinge die man auf wikipedia lesen kann, was ist schon hier, die Dinge, die ich nachher brauche)

Dieses Kapitel beschreibt den Theoretischen Inhalt dieser Arbeit. Um die späteren Analysen und Evaluationen verstehen zu können, werden die grundlegenden Konzepte, Technologien und Frameworks vorgestellt, die in diesem Projekt verwendet werden. Dazu gehören die Prinzipien des Machine Learnings, die Rolle von Edge Computing sowie die eingesetzten Hardware- und Softwarekomponenten. Ebenso werden relevante Metriken und Methoden erläutert, die zur Bewertung der entwickelten Systeme herangezogen werden.

2.1 Citizen Science

- Was ist das?
- Motivation (Warum machen Leute mit?)
- Warum ist es wichtig / notwendig?

Eine treffende Definition von Citizen Science liefert das Grünbuch Citizen Science Strategie 2020 für Deutschland:

„Citizen Science beschreibt die Beteiligung von Personen an wissenschaftlichen Prozessen, die nicht in diesem Wissenschaftsbereich institutionell gebunden sind. Dabei kann die Beteiligung in der kurzzeitigen Erhebung von Daten bis hin zu einem intensiven Einsatz von Freizeit bestehen, um sich gemeinsam mit Wissenschaftlerinnen bzw. Wissenschaftlern und/oder anderen Ehrenamtlichen in ein Forschungsthema zu vertiefen. Obwohl viele ehrenamtliche Forscherinnen und Forscher eine akademische Ausbildung aufweisen, ist dies keine Voraussetzung für die Teilnahme an Forschungsprojekten. Wichtig ist allerdings die Einhaltung wissenschaftlicher Standards, wozu vor allem Transparenz im Hinblick

auf die Methodik der Datenerhebung und die öffentliche Diskussion der Ergebnisse gehören.“

— A. Bonn *u. a.* [1]

Bürgerinnen und Bürger können also einen Beitrag zur Wissenschaft leisten, indem sie Daten sammeln, analysieren und interpretieren. Dies kann von der einfachen Datenerhebung bis hin zur intensiven Auseinandersetzung mit einem Forschungsthema reichen. Die Beteiligung an Citizen Science Projekten ist nicht an eine akademische Ausbildung gebunden, jedoch ist die Einhaltung wissenschaftlicher Standards unabdingbar. Es stellt sich die Frage, worin die Motivation liegt sich freiwillig solchen Projekten zu widmen. Ein treibender Faktor kann der Wissenszuwachs für ein bestimmtes Thema sein. Die Auseinandersetzung in einem Bereich, für den man sich interessiert. Das Buch *The Science of Citizen Science* [2, p 283] diskutiert die verschiedenen Aspekte des Lernens in einem Citizen Science Projekt.

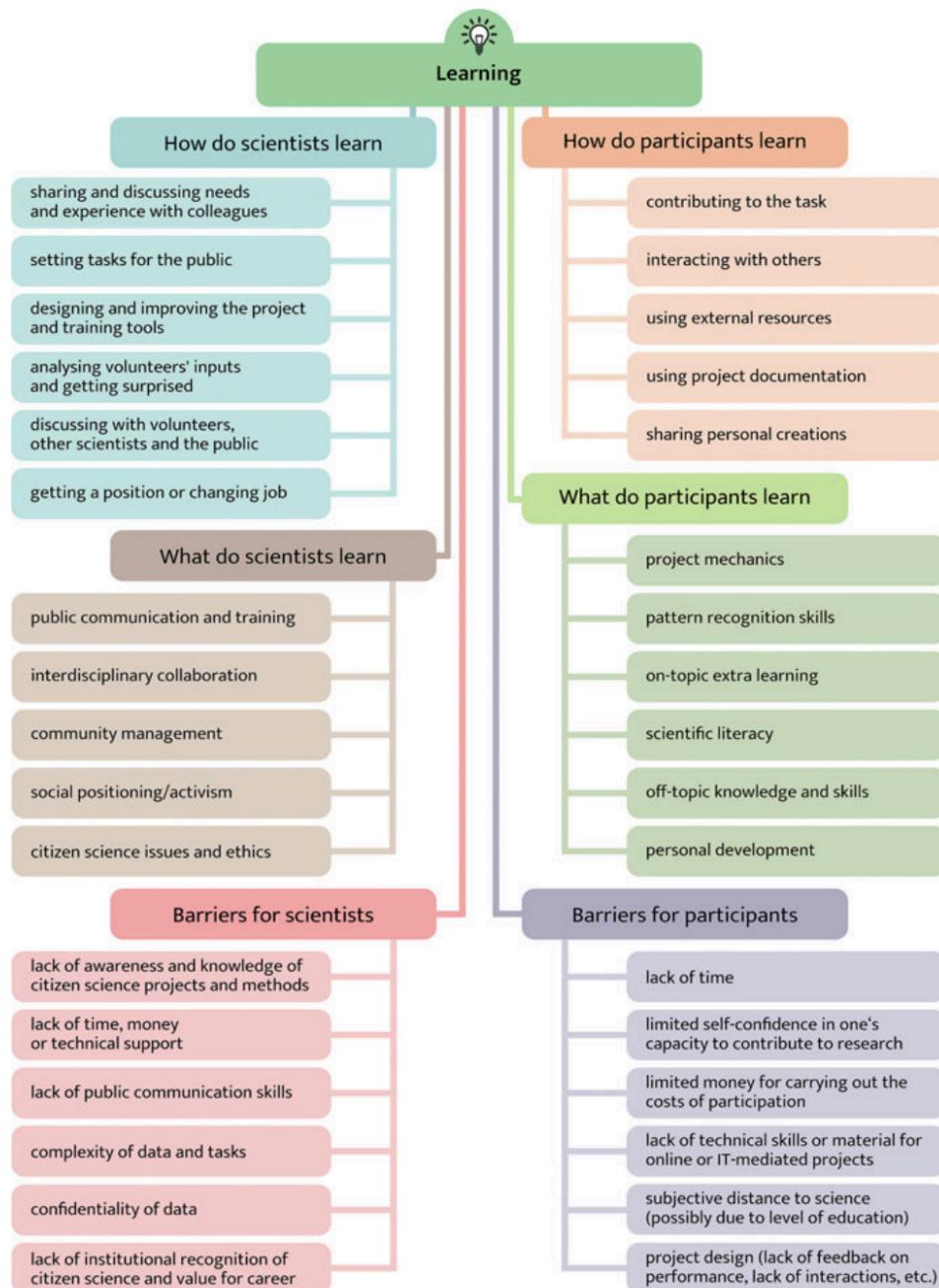


Abbildung 1: Erweiterte thematische Karte zum Lernen von Freiwilligen

(Quelle: The Science of Citizen Science, S.300)

Aus der Illustration geht hervor, dass verschiedene Aspekte die Bürgerinnen und Bürger zu einer aktiven Beteiligung motivieren kann. Nebst den Fachlichen sind auch Soziale Aspekte von Bedeutung. Auf der anderen Seite priorisieren auch die Projekt initianten. Die

Organisation und Zusammenarbeit mit Aussenstehenden kann eine spannende Aufgabe sein.

2.2 Biodiversität Monitoring

(Timeo's Arbeit zitieren - nicht das selbe machen)

- Sinn und Zweck
- Mögliche Anwendungszwecke / Szenarien / Perspektiven
- Mitwelten Pipeline (Von der Kamera zu den Daten)

Biodiversitäts Monitoring befasst sich grundsätzlich mit dem Beobachten und Analysieren von Tier - und Pflanzenwelt.

„Die biologische Vielfalt bildet eine Lebensgrundlage der Schweiz.
Deshalb ist es wichtig, ihren Zustand und ihre Entwicklung zu
kennen.“

— bdm [3]

Aus diesem Grund ist es von grosser Bedeutung die Umwelt zu beobachten und Veränderungen festzustellen. Um diese Arbeit zu vereinfachen, sind automatisierte Lösungen für ein Monitoring gefragt.

Ein System zur automatisierten Datenerfassung wurde im Rahmen des Projekts Mitwelten entwickelt [4], [5]. Hierfür wurden verschiedene Sensoren zu einem IoT-Toolkit kombiniert. Dieses Toolkit ermöglicht es, Daten von dezentralen Systemen zu erfassen und an ein zentrales Backend weiterzuleiten.

Das IoT-Toolkit umfasst unter anderem eine Kamera, die in regelmässigen Abständen Fotos von Blumentöpfen aufnimmt. Im Rahmen des Projekts wurden so insgesamt 1,5 Millionen Bilder generiert. Abbildung Abbildung 2 zeigt eine im Feld aufgestellte Kamera.



Abbildung 2: Pollinator Kamera im Feld
(Quelle: mitwelten.ch, aufgerufen am 23.01.2025)

Die enorme Menge an Bilddaten lässt sich nicht manuell analysieren. Deshalb entwickelte das Projekt-Team eine Machine-Learning-Pipeline, die Bestäuber automatisch erkennt. Die technische Architektur des Systems verbindet die Kamera mit einem leistungsstarken Backend. Die Kamera, bestehend aus einem Raspberry Pi und einer angeschlossenen Kameraeinheit, erfasst die Bilder und überträgt diese über einen Access Point an das Backend. Auf dem Server führt eine Machine-Learning-Pipeline die Analyse der empfangenen Bilder durch, um Bestäuber zu erkennen. Im Kontext dieser Entwicklung wurden ebenfalls alternative Architekturen untersucht. Unter anderem ein System, auf dem die Machine-Learning-Pipeline auf einem Raspberry Pi 3 ausgeführt wurde. Aufgrund der zu langen Analysezeiten wurde jedoch ein Server basierter Ansatz gewählt.

2.3 Machine Learning Grundlagen

- Frameworks (Unterschiede, Vor- und Nachteile)
- Anatomie eines Modells
 - Architektur
 - welche parameter eines modells werden bei der Erstellung definiert
 - Input output tensor
 - Art: typen von ML modellen (object detection, segmentation, pose)
 - welche messdaten gibt es für object detection (iou, mAP, perfomance / inferenzzeit)
 - training eines models (sehr oberflächlich)
 - infernz, was ist das ?

2.3.1 ML Frameworks

Ein Machine-Learning-Framework bildet die Grundlage für die Entwicklung, das Training und die Bereitstellung von Modellen. Es bietet Werkzeuge und Bibliotheken, die den gesamten ML-Workflow unterstützen. Im Gegensatz dazu ist ein Machine-Learning-Modell das konkrete Ergebnis des Trainingsprozesses, das spezifische Aufgaben wie Klassifikation oder Objekterkennung ausführt. Während ein Framework die Infrastruktur bereitstellt, ist das Modell die fertige Anwendung innerhalb dieser Infrastruktur.

2.3.1.1 TensorFlow

TensorFlow [6] wurde ursprünglich von Google entwickelt und ist inzwischen ein Open Source Projekt. Das Framework ist eher für Systeme in Produktion gedacht.

- Vorteile: Weit verbreitet mit grosser Community und umfangreicher Dokumentation.
Stellt viele Features zu Verfügung.
- Nachteile: Steile Lernkurve, insbesondere für Einsteiger. Komplex durch die vielen Features.

2.3.1.2 TensorFlow Lite

- Vorteile: Speziell für den Betrieb auf ressourcenbeschränkten Geräten optimiert.
Reduzierte Speicher- und Rechenanforderungen für Echtzeit-Anwendungen.
- Nachteile: Eingeschränkte Unterstützung für komplexe Modelle, Quantisierung erforderlich.

2.3.1.3 PyTorch

PyTorch [7] ist ebenfalls Open Source und hat seinen Ursprung im Forschungsteam für künstliche Intelligenz von Facebook. Das Framework eignet sich für Experimente kann aber auch für Systeme in Produktion eingesetzt werden. ChatGPT von OpenAI arbeitet im Hintergrund mit dem Pytorch Framework [8].

- Vorteile: Hat eine gute Community und eine ausführliche Dokumentation. Viele Features und viele Tutorials.
- Nachteile: Für Einsteiger

2.3.1.4 ONNX

ONNX (Open Neural Network Exchange) [9] ist ein generalisiertes Format von Deep-Learning Modellen. Dies ermöglicht den Austausch von Modellen zwischen verschiedenen Frameworks. ONNX wird von Microsoft, Amazon, Facebook und weiteren Partners als Open-Source Projekt entwickelt.

- Vorteile: Austauschformat, das Modelle zwischen verschiedenen Frameworks kompatibel macht. Optimierte Laufzeitumgebungen, die speziell für Edge-Geräte geeignet sind.
- Nachteile: Begrenzte Funktionalität für Modelltraining, primär für den Einsatz trainierter Modelle gedacht. Kleinere Community im Vergleich zu TensorFlow und PyTorch.

2.3.1.5 NCNN

NCNN [10] ist eine high performance computing platform für mobile Endgeräte. Dieses Framework wird oft auf Smartphone verwendet, weil es optimiert für Geräte mit beschränkter Rechenleistung ist.

- Vorteile: Schnelle Inferenzzeiten auf der CPU.
- Nachteile: Im Vergleich zu anderen Frameworks weniger ausführlich dokumentiert.

2.3.1.6 Hailo

Hailo, gegründet im Jahr 2017, hat sich als führendes Unternehmen im Bereich leistungsfähiger KI-Prozessoren für Edge-Anwendungen etabliert. Mit ihrer eigens entwickelten Hardware-Architektur verfolgt Hailo das Ziel, Machine Learning auch außerhalb von Rechenzentren zugänglich und effizient nutzbar zu machen [11]. Die Nutzung der Hailo-Chips erfordert eine Konvertierung der Modelle, wofür das Unternehmen eine umfassende Software-Suite bereitstellt. Diese Suite enthält alle notwendigen Werkzeuge, um Modelle auf die spezifischen Anforderungen der Hailo-Hardware abzustimmen.

- Vorteile: Ermöglicht sehr schnelle Inferenzen und Hailo bietet eine gute Dokumentation und ein großes Ökosystem
- Nachteile: Eine intensive Einarbeitung ist notwendig und der Konvertierungsprozess erfordert Modulspezifisches Wissen..

2.3.1.7 Ultralytics

Das Ultralytics [12] Framework ist ein auf PyTorch basierendes Open-Source-Framework, das vor allem für die Entwicklung von Modellen zur Objekterkennung bekannt ist. Es wurde durch die Implementierung von YOLOv5 (You Only Look Once) populär und bietet eine benutzerfreundliche Umgebung für das Training und die Bereitstellung von Objekterkennungsmodellen. Inzwischen sind neuere Implementierungen der YOLO Familie verfügbar und mit Ultralytics anwendbar.

- Vorteile: Die Verwendung des Framework ist sehr einfach und dadurch geeignet für Einsteiger. Modelle können in andere Formate exportiert werden. Inferenzen werden für PyTorch Modelle angeboten.
- Nachteile: Die Flexibilität im Vergleich zu anderen Frameworks ist eingeschränkt.

2.3.1.8 Trade-Off's

- Flexibilität vs. Spezialisierung: Frameworks wie TensorFlow und PyTorch bieten umfassende Funktionen, benötigen jedoch mehr Ressourcen. Edge-spezifische Frameworks sind dagegen ressourcenschonender, aber eingeschränkter in ihrer Funktionalität.
- Einsatzgebiet: Die Wahl des Frameworks sollte sich an den spezifischen Anforderungen orientieren, wie etwa der Zielplattform (Cloud vs. Edge), der Modellkomplexität und der Verfügbarkeit von Ressourcen.

2.3.2 Anatomie von Machine Learning Modellen

Ein Machine-Learning-Modell besteht aus einer Architektur, die dessen Aufbau und Funktionsweise definiert, und einem Satz von Parametern, die während des Trainings optimiert werden. Die Architektur eines Modells legt grundlegende Eigenschaften fest, wie die Struktur der Neuronen und Layer, die Verbindungen zwischen ihnen sowie Eingabe- und Ausgabetensoren. Die Eingabetensoren bestimmen, welche Form und Dimensionen die Daten haben müssen (z. B. die Größe von Bildern), während die Ausgabetensoren das Ergebnis des Modells beschreiben, etwa Klassenetiketten oder Koordinaten für erkannte Objekte.

Es gibt verschiedene Typen von Machine-Learning-Modellen, die je nach Anwendungsfall eingesetzt werden. Zu den wichtigsten gehören Modelle für Objekterkennung, die Objekte in einem Bild lokalisieren und klassifizieren, Bildsegmentierung, die jedem Pixel eines Bildes eine Klasse zuordnet, und Posenschätzung, die die Körperhaltung von Personen oder Tieren analysiert.

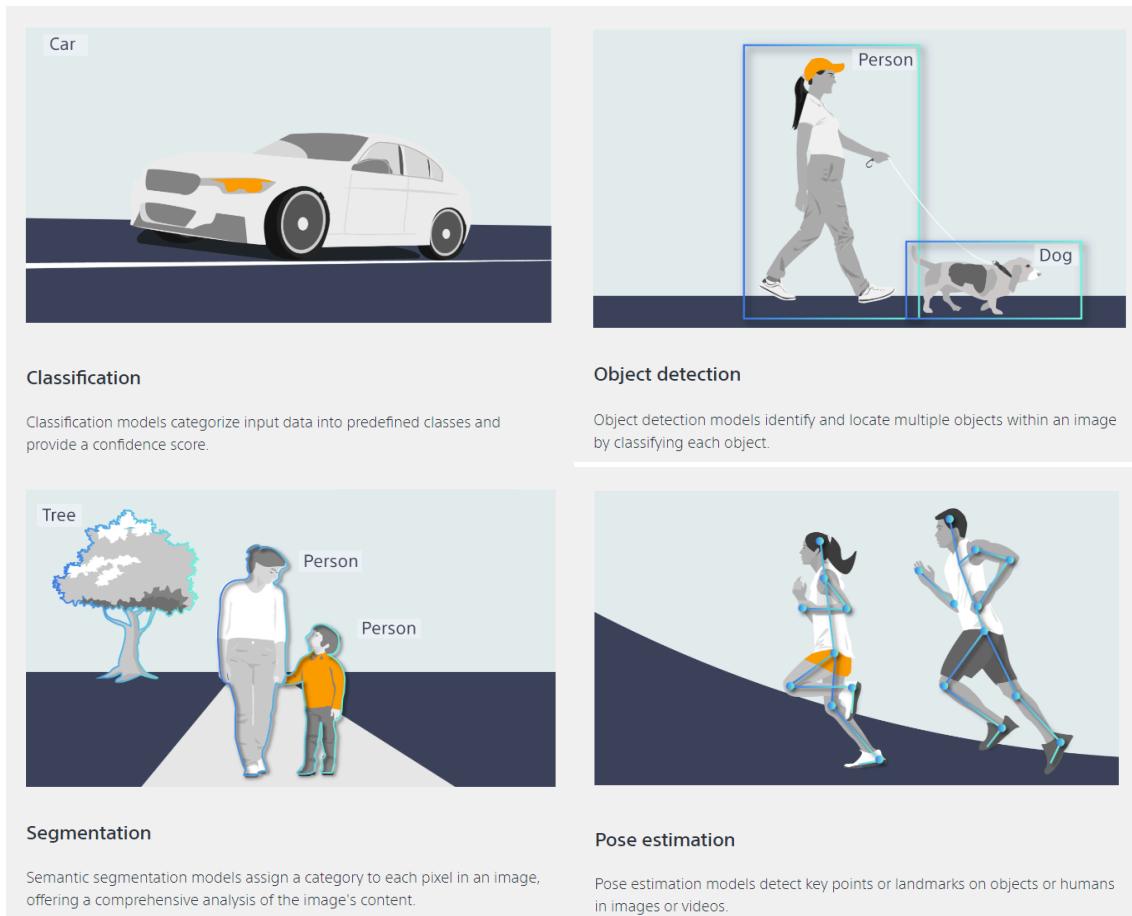


Abbildung 3: Machine Learning Tasks

(Quelle: https://github.com/sony/model_optimization, aufgerufen am 20.01.2025)

Für die Objekterkennung werden verschiedene Metriken verwendet, um die Modellleistung zu bewerten. Dazu zählen die Intersection over Union (IoU), die angibt, wie gut vorhergesagte und tatsächliche Direktionsfläche übereinstimmen, und die Mean Average Precision (mAP), die die Präzision über verschiedene Schwellenwerte hinweg aggregiert. Zusätzlich spielen Leistungskennzahlen wie die Inferenzzeit eine wichtige Rolle, insbesondere bei Echtzeitanwendungen.

Das Training eines Modells umfasst die Anpassung der Parameter auf Basis eines grossen Datensatzes, um Muster und Zusammenhänge zu lernen. Dabei kommen Optimierungsalgorithmen wie Gradient Descent zum Einsatz, die das Modell iterativ verbessern. Nach dem Training wird das Modell in der Inferenzphase genutzt. Inferenz bezeichnet

den Prozess, bei dem ein trainiertes Modell neue Eingaben verarbeitet und Vorhersagen generiert.

2.3.3 Geschwindigkeit einer Inferenz

Die Dauer einer Inferenz hängt massgeblich von den Berechnungen des Modells und der Leistungsfähigkeit des Systems ab, auf dem sie ausgeführt wird. Die benötigte Anzahl an Berechnungen wird durch die Neuronenanzahl im neuronalen Netz bestimmt. Mit zunehmender Anzahl an Neuronen – sei es durch mehr oder grössere Hidden Layers – steigt der Rechenaufwand. Die Abbildung 4 illustriert den symbolischen Aufbau eines neuronalen Netzes mit drei Hidden Layers. Ein Netz mit mehr als einem Hidden Layer wird als „Deep Neural Network“ bezeichnet.

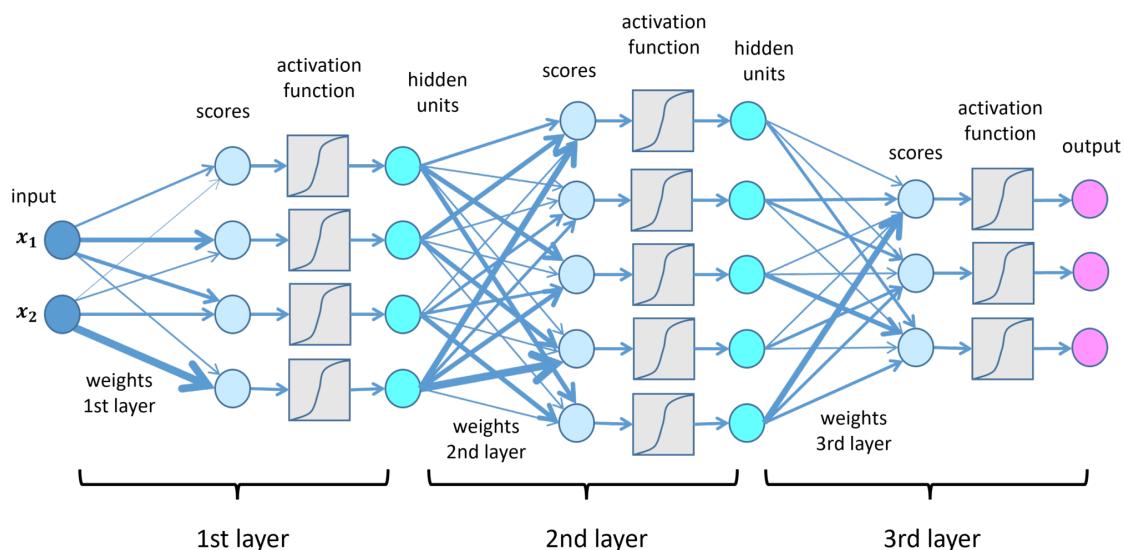


Abbildung 4: Symbolbild Neuronales Netzwerk

(Quelle: lamarr-institute.org/blog/deep-neural-networks, abgerufen am 23.01.2025)

Je mehr Hidden Layers und trainierte Gewichte ein Modell besitzt, desto präziser werden seine Vorhersagen. Allerdings steigt damit auch der Rechenaufwand. Aus diesem Grund existieren YOLO-Modelle, wie sie in dieser Arbeit verwendet wurden, in verschiedenen Größen [13].

Ein weiterer entscheidender Faktor ist die Hardware, auf der die Berechnungen durchgeführt werden. Durch das Parallelisieren von Berechnungen lassen sich erhebliche Geschwindigkeitsvorteile erzielen. Besonders geeignet sind GPUs (Graphics Processing Units), TPUs (Tensor Processing Units) und NPUs (Neural Processing Units). Da die Berechnungen in neuronalen Netzen denen in der Bildverarbeitung ähneln, sind diese

Hardwarelösungen sowohl für das Training als auch für die Anwendung von ML-Modellen optimal.

TPUs, von Google für das TensorFlow-Framework entwickelt [14], sind speziell für die benötigten Matrixoperationen optimiert. Ebenso wurde die NPU [15] entwickelt, um ML-Tasks hochgradig parallelisiert und effizient auszuführen. In bestimmten Szenarien kann eine NPU eine GPU in Sachen Geschwindigkeit deutlich übertreffen [16].

2.4 Edge Computing

(vllt auch in der Ausgangslage erwähnen)

Edge Computing und Edge Machine Learning markieren einen Paradigmenwechsel in der Systemarchitektur, indem die Datenverarbeitung von zentralen Backend Server oder Cloud-Systemen hin zu Geräten am Rand des Netzwerks (Edge) verlagert wird. Diese Architektur ermöglicht es, Berechnungen und Analysen direkt vor Ort durchzuführen, anstatt die Daten für die Verarbeitung zu versenden.

Dies bringt mehrere Vorteile mit sich. So ermöglicht die lokale Verarbeitung eine geringere Latenz, was vor allem für Echtzeitanwendungen entscheidend ist. Zudem reduziert sich der Datenverkehr zu einem Backend, was nicht nur die Betriebskosten senkt, sondern auch die Abhängigkeit von einer stabilen Internetverbindung minimiert. Nebst der Datenübertragung ist auch die Komplexität eines Systems von Bedeutung. Eine Systemarchitektur mit zentraler Einheit bedeutet nebst zusätzlichen Technologien auch einen erhöhten Wartungsaufwand, welcher gerade bei grosser Projektdauer nicht zu unterschätzen ist. Ein weiterer wesentlicher Vorteil liegt im Bereich der Datensicherheit. Sensible Informationen bleiben vor Ort und müssen nicht über Netzwerke übertragen werden, wodurch die Privatsphäre der Nutzer besser geschützt wird.

Allerdings bringt dieser Ansatz auch Herausforderungen mit sich. Edge-Geräte verfügen häufig über eingeschränkte Rechenleistung und Speicherressourcen, was die Ausführung komplexer Machine-Learning-Modelle erschweren kann. Darüber hinaus ist die Energieeffizienz ein zentraler Aspekt, da viele Edge-Geräte batteriebetrieben sind und die Optimierung des Energieverbrauchs entscheidend für den Betrieb sein kann. Schliesslich erfordert die Verwaltung und Skalierung einer Vielzahl von verteilten Geräten ohne zentrale Steuerung zusätzlichen Aufwand.

2.4.1 Edge ML

Edge Machine Learning kombiniert die Prinzipien des Edge Computing mit den Anforderungen moderner Machine-Learning-Modelle, indem die Rechenleistung direkt auf die Endgeräte verlagert wird. Dieser Ansatz wird entweder auf der CPU ausgeführt oder durch spezialisierte Hardware beschleunigt. Diese ist darauf ausgelegt, ML-Modelle performant, effizient und ressourcenschonend auszuführen. Beschleuniger Hardware ist dann erforderlich, wenn die Inferenz auf der CPU zu viel Zeit oder Ressourcen in Anspruch nimmt. Diese Hardware wird typischerweise mit einem Host System verbunden, welches mit der Beschleuniger Hardware kommunizieren kann.

Die Weiterentwicklung der kompakten Computer sowie die auf ML spezialisierte Hardware öffnet neue Möglichkeiten im Bereich des Edge ML. Zum Zeitpunkt dieser Arbeit sind die Entwicklungen im vollen Gange. Nicht nur auf Seite der Hardware, sondern auch im Umfeld des Machine Learning wird zur Zeit geforscht und entwickelt. Somit kann in diesem Projekt unter Anderem neuste Hardware eingesetzt werden, welche sich in der Industrie möglicherweise erst in der kommenden Zeit etablieren wird. Die im Projekt eingesetzte Hardware ist im folgenden aufgeführt.

2.4.2 Host Systeme

Der einfachste Weg Machine Learning im Bereich Edge zu betreiben ist die CPU eines Edge Computers. Um Erfahrungen zu sammeln sind verschiedene Einplatinen Computer evaluiert worden. Hauptsächlich hat sich die Verwendung von Raspberry Pi Computer durchgesetzt. Sie weisen ein gutes Preis Leistungsverhältnis auf und sind durch Dokumentation und Community extrem gut unterstützt.

2.4.2.1 Raspberry Pi 4

Das Raspberry Pi 4 8 GB ist für rund 85Fr. [17] erhältlich. Mit einer CPU Taktfrequenz von 1.5GHz lassen sich schon viele Anwendungen realisieren. Nachteil des Model 4 ist, dass keine PCI Schnittstelle vorhanden ist.



Abbildung 5: Raspberry Pi 4

(Quelle: <https://www.pi-shop.ch/raspberry-pi-4-model-b-4gb>, aufgerufen am 23.01.2025)

2.4.2.2 Raspberry Pi 5

Das Raspberry Pi 5 stellt den Nachfolger vom Model 4 dar und ist mit 8 GB RAM und einer CPU Taktfrequenz von 2.4GHz ausgerüstet. Ebenso ist eine PCI Schnittstelle verfügbar, wodurch das anschliessen von leistungsstarker Beschleuniger Hardware möglich wird. Dieses Setup ist für ca. 85 Fr. [18] Erhältlich.

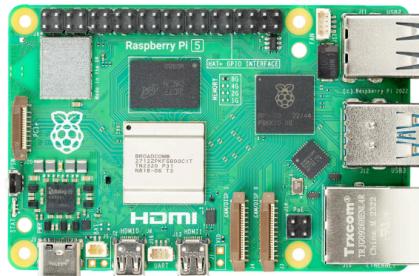


Abbildung 6: Raspberry Pi 5

(Quelle: <https://www.pi-shop.ch/raspberry-pi-5-16gb-ram?src=raspberrypi>, aufgerufen am 23.01.2025)

2.4.2.3 BeagleY-AI

Das BeagleY-AI Board hat eine CPU Taktfrequenz von 1.5GHz und 4 GB RAM. Das besondere an diesem Board ist der integrierte AI Accelerator mit 4 TOPS (Terra Operation per Second). Das Board ist für run 70 Fr. [19] Erhältlich.



Abbildung 7: BeagleY-AI Board

(Quelle: <https://www.seeedstudio.com/BeagleYr-AI-beagleboard-orgr-4-TOPS-AI-Acceleration-powered-by-TI-AM67A.html>, aufgerufen am 23.01.2025)

2.4.3 Hardware Beschleuniger

Machine-Learning-Beschleuniger sind spezialisierte Hardwarekomponenten, die entwickelt wurden, um die Ausführung von Machine-Learning-Modellen zu optimieren. Sie sind insbesondere für rechenintensive Aufgaben wie Matrixmultiplikationen und Tensorberechnungen ausgelegt, die in vielen ML-Algorithmen eine zentrale Rolle spielen. Zu den bekanntesten Typen solcher Beschleuniger gehören GPUs (Graphics Processing Units) und TPUs (Tensor Processing Units).

Eine GPU wurde ursprünglich für die Verarbeitung von Grafikanwendungen entwickelt, hat sich jedoch aufgrund ihrer Fähigkeit zur parallelen Verarbeitung tausender Operationen gleichzeitig als ideal für Machine-Learning-Aufgaben erwiesen. GPUs kommen häufig bei grossen Modellen und im Training von neuronalen Netzwerken zum Einsatz, da sie eine hohe Rechenleistung bieten.

Eine TPU ist ein speziell entwickelter Prozessor, der ausschliesslich für Machine-Learning-Workloads optimiert wurde. Sie wurde von Google entwickelt und ist besonders effizient bei der Verarbeitung von Tensoroperationen, wie sie in Frameworks wie TensorFlow genutzt werden. TPU's unterscheiden sich von GPUs durch ihre Fokussierung auf deterministische Operationen, die typischerweise in ML-Modellen vorkommen, was sie besonders leistungsstark und energieeffizient macht. TPU's sind für das Training und die Inferenz geeignet, wobei sie bei letzterem aufgrund ihrer geringen Latenz und Effizienz eine herausragende Rolle spielen.

2.4.3.1 Coral USB Accelerator

Der USB-Dongle von Google, Coral USB Accelerator, hat eine dedizierte TPU (Tensor Processing Unit). Solche Geräte lassen sich leicht in bestehende Systeme integrieren und

ermöglichen die schnelle Ausführung von ML-Algorithmen ohne signifikante Anpassungen der Infrastruktur. Der Preis bewegt sich um 89.90Fr.[20].



Abbildung 8: Coral USB Accelerator

(Quelle: <https://www.coral.ai/products/accelerator>, aufgerufen am 22.01.2025)

2.4.3.2 Coral PCI Accelerator

Nebst den USB-Dongles stellt Google auch über die PCI Schnittstelle angeschlossene Beschleuniger her. Diese sind als single oder dual TPU erhältlich. Der Preis für die signle Lösung mit 4TOPS liegt bei 47.90Fr. [21], für die dual Lösung mit 8 TOPS bezahlt man 96.90 Fr.[22].



Abbildung 9: Coral PCI Accelerator

(Quelle: <https://www.coral.ai/products/m2-accelerator-dual-edgetpu>, aufgerufen am 22.01.2025)

2.4.3.3 Hailo TPU

Hailo ist eine Firma die sich auf Edge-KI Prozessoren spezialisiert hat. Diese sind moderner als die Lösungen von Google und erreichen eine höhere Leistungsfähigkeit. Die in diesem Projekt verwendeten Beschleuniger haben 13 und 26 TOPS zu 79.90 Fr.[23] resp. zu 122.90 Fr.[24]. Die folgende Abbildung zeigt ein Hailo Brschleuniger auf einem Raspberry Pi Abbildung 10.



Abbildung 10: Hailo Accelerator

(Quelle: <https://www.pi-shop.ch/raspberry-pi-ai-hat-13t>, aufgerufen am 22.01.2025)

2.4.3.4 AI-Kamera

Eine weitere Kategorie sind AI-Kameras, die mit integrierten ML-Chips ausgestattet sind. Diese Kameras, wie auf Abbildung 11 dargestellt, können nicht nur Bilder aufnehmen, sondern auch direkt auf der Kamera Inferenzdaten bereitstellen, beispielsweise durch die Erkennung und Klassifikation von Objekten. Raspberry Pi stellt eine solche Kamera zur Verfügung für 77.90 Fr. [25].



Abbildung 11: Raspberry Pi AI Camera

(Quelle: <https://www.berrybase.ch/raspberry-pi-ai-camera>, aufgerufen am 22.01.25)

Dieser Ansatz verlagert die intensiven Berechnungen womit der Edge Computer weniger Ressourcen bereitstellen muss. Die Abbildung 12 verdeutlicht, wie die Architektur eines solchen Systems aufgebaut ist. Die linke Seite zeigt den Aufbau herkömmlicher Kamera Architekturen, während rechts die Lösung mittels AI Accelerator dargestellt ist.

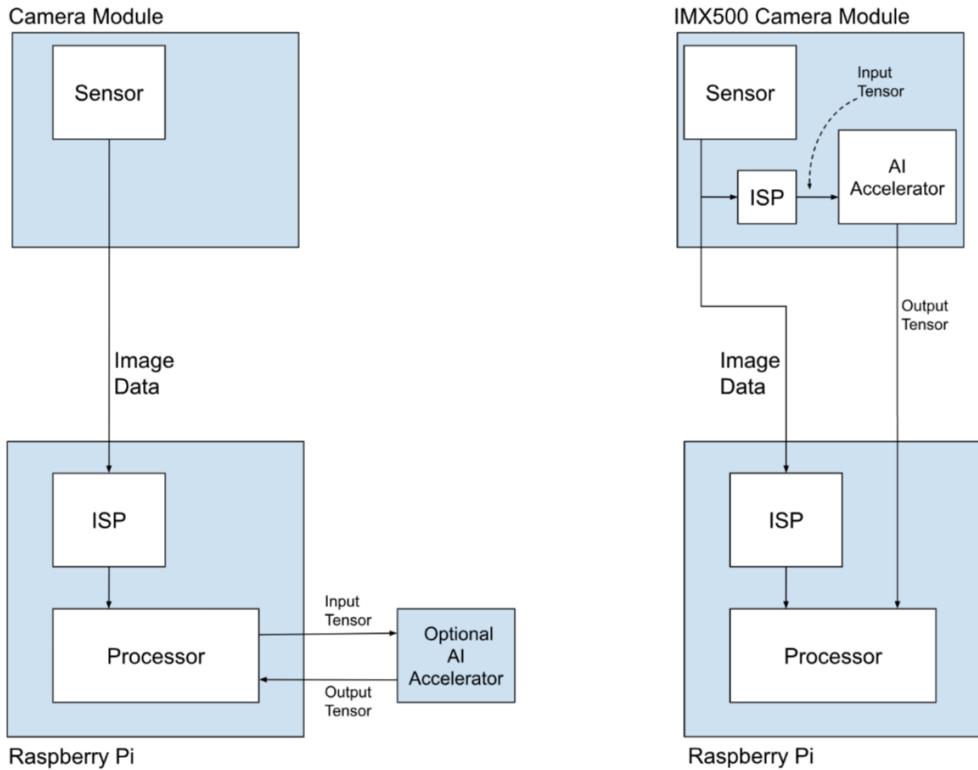


Abbildung 12: Raspberry Pi AI Camera Architektur

(Quelle: <https://www.raspberrypi.com/documentation/accessories/ai-camera.html>, aufgerufen am 20.01.2025)

Beim Sensor handelt es sich um einen Sony IMX500 Chip [26], welcher ein Bild aufnehmen kann. Die Daten gelangen dann als RAW-Image zum ISP, einem kleinen Image Signal Processor. Dieser wandelt das RAW-Image zu einem Input Tensor, welcher vom AI-Accelerator als Input Größe entgegen genommen wird. Nach durchführen der Inferenz gelangen die Resultate, in diesem Fall der Output Tensor zum Processor des Host Systems. Gleichzeitig gelangt auch das Image auf das Host-System, um wenn nötig die Resultate anzuzeigen oder weiterverarbeitet zu werden. Die Bilder können wahlweise mit 10 FPS bei einer Auflösung von 4056x3040 oder bei 30 FPS mit 2028x1520 aufgenommen werden.

3 Analyse

(Was braucht es für mich)

Dieses Kapitel befasst sich mit der Analyse und Bewertung der Möglichkeiten für die Umsetzung einer Edge ML Kamera. Zu diesem Zweck wurden anhand von Experimenten verschiedene Messungen durchgeführt. Für die Analyse dieser Experimente ist ein Dashboard [27] entwickelt worden, um interaktiv verschiedene Komponenten miteinander zu vergleichen. Die folgende Abbildung 13 zeigt ein Screenshot der Applikation.

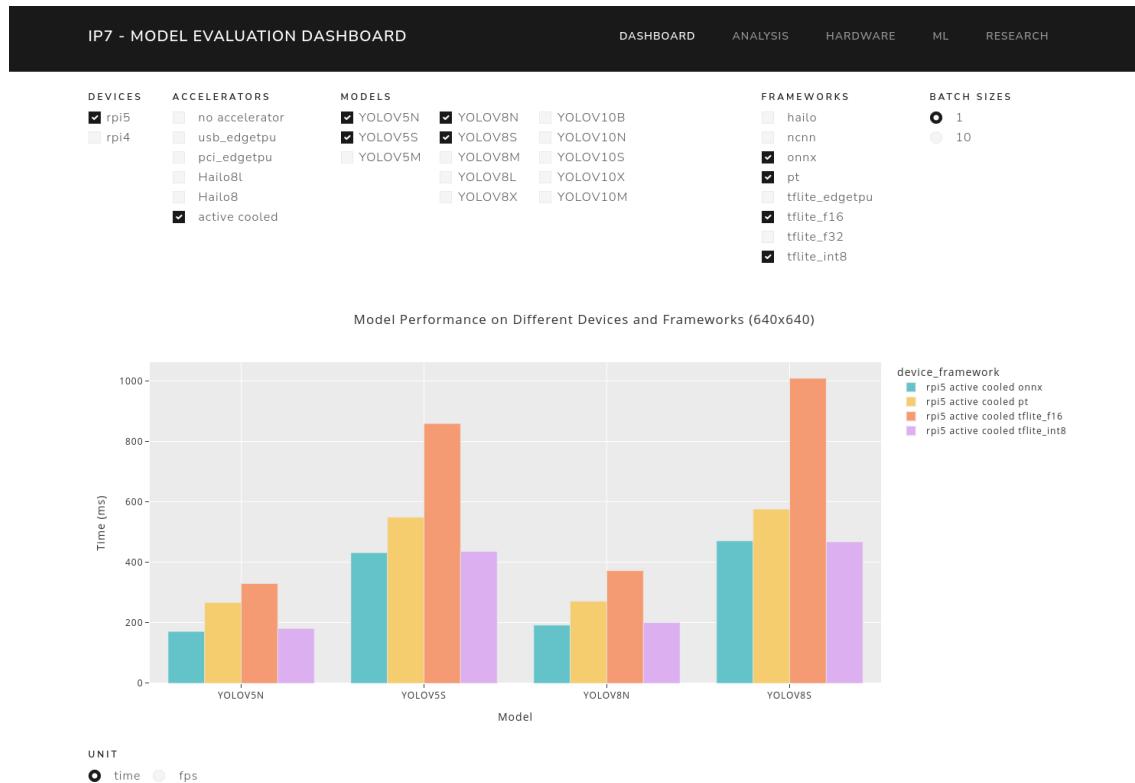


Abbildung 13: ML Exploration Dashboard

(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)

3.1 Zielgruppe

- Was müssen Bediener können

Ein Kamera System, welches flexibel für verschiedene Zwecke einsetzbar ist, kann interessant sein für eine Vielzahl an Anwendungszwecken und deckt daher auch eine breite Zielgruppe ab:

- Forschende: Wissenschaftlerinnen und Wissenschaftler aus den Bereichen Biodiversität, Ökologie oder Umweltwissenschaften.
- Citizen Scientists: Ehrenamtliche, die sich an wissenschaftlichen Projekten beteiligen.
- Naturschutzorganisationen: Institutionen, die Artenschutz- oder Umweltprojekte durchführen.
- Bildungseinrichtungen: Schulen, Universitäten und andere Lernstätten für Lehrzwecke und studentische Forschungsprojekte.
- Landwirtschaftliche Betriebe: Landwirte, die ihr Wissen über Bestäuber und Schädlingsbekämpfung erweitern möchten.
- Regierungs- und Umweltbehörden: Organisationen, die Umweltschutzmaßnahmen überwachen oder Berichte erstellen.
- Technologie- und Umweltenthusiasten: Personen mit Interesse an IoT und nachhaltiger Technologie.

3.2 Use Cases

Durch die grosse Anzahl an verschiedenen Zielgruppen gibt es auch viele verschiedene Scenarien, in dem der Einsatz einer Edge ML Kamera vorstellbar ist.

- Biodiversitätsforschung: Überwachung von Bestäubern, Vögeln, Säugetieren oder Pflanzenwachstum.
- Artenschutz: Identifikation und Überwachung gefährdeter Tier- oder Pflanzenarten.
- Umweltüberwachung: Aufzeichnung und Analyse des Einflusses von Umweltbedingungen wie Temperatur, Feuchtigkeit oder Lichtverhältnissen auf die Pflanzen oder Tierwelt.
- Landwirtschaft: Erkennung von Schädlingen, Optimierung des Pflanzenwachstums.
- Bildungsprojekte: Praktische Anwendungen für Schüler und Studierende in Naturwissenschaften und Technik, als Ausbildungs Objekt für eine Machine Learning Applikation.
- Citizen Science: Ermöglicht Gemeinschaften, eigene wissenschaftliche Projekte durchzuführen, z. B. zur lokalen Artenvielfalt oder sich grösseren Projekten anzuschliessen.

- Was sollen Personen fähig sein zu machen

3.2.1 Referenz Use Case

Um bei der Entwicklung der Edge ML Kamera zielgerichtet arbeiten zu können, ist die Mitwelten Bestäuber Analyse das zu erfüllende Scenario. Durch die grosse Anzahl an verfügbaren und teilweise bereits gelabelten Bildern ist der Weg ein neues Modell zu trainieren geebnet. Die Bestäuber Detektion erfolgt in folgenden wesentlichen Schritten. Ein Foto von Blüten durchläuft in einem ersten Schritt eine Inferenz zur Detektion von Blüten. Die auf dem Bild detektierten Blüten werden anschliessen ausgeschnitten und somit zu kleineren Fotos. Jedes dieser Fotos wird anschliessend mit einer weiteren Inferenz und einem anderen Machine Learning Modell auf Bestäuber untersucht. Eine grosse Herausforderung in diesem Setup ist, dass die Anzahl Bestäuber-Inferenzen mit der Anzahl detektierten Blüten steigt. Dies kann bei grossen Blütenzahlen zu langen Analysen eines einzigen Bildes führen.



Abbildung 14: Mitwelten Machine Learning Pipeline

(Quelle: Automated Analysis for Urban Biodiversity Monitoring, Timeo Wullschleger)

3.3 Anforderungen

- Funktionale
 - Features (Cam Preview, Inferenz ausführen, Modelwahl, Framework wahl)
 - Hardware
- Nicht funktionale
 - Inferenz Zeitlimit
 - Einsatzgebiet
- Citizen Science bezogen

Wie in den Kapitel zuvor diskutiert, gibt es eine breite Anwendung für eine Edge ML Kamera. Der Referenz Use Case für die Entwicklung des Systems ist die in der Mitwelten Projekt eingesetzte Bestäuber Analyse. Dieser Anwendungsfall dient somit zur Definierung der Anforderungen an das System.

3.3.1 Funktionale

Die folgenden funktionalen Anforderungen an eine Edge ML Kamera wurden identifiziert:

- Datenerfassung: Das System muss anhand einer angeschlossenen Kameraeinheit Bilder kontinuierlich erfassen können
- Analyse Resultate: Das System muss die Analysedaten für den jeweiligen Anwendungszweck zur Verfügung stellen
- Benutzerinteraktion: Das System muss vom Benutzer konfiguriert werden können
- Energieversorgung: Während des Betriebs muss eine stabile Energieversorgung vorhanden sein

3.3.2 Nicht Funktionale

Die folgenden nicht funktionalen Anforderungen an eine Edge ML Kamera wurden identifiziert: Analysedauer: Im Mitwelten Projekt ist definiert, dass die Analyse von Bestäubern in 15 Sekunden erfolgen muss [5, p 62]. Die Zeit ist von der Verweildauer von Bestäuber auf einer Blume limitiert.

3.4 Evaluation

Im folgenden werden Resultate der Untersuchungen bezüglich ML Frameworks und Hardware aufgezeigt. Dabei sind Inferenzen mit PyTorch, Tensorflow lite, ONNX und Hailo Modellen implementiert worden. Die Inferenzen wurden auf Raspberry Pi's mit und ohne Beschleuniger Hardware ausgeführt und gemessen.

3.4.1 Methodik

Bei den Verwendeten Modellen handelt es sich um die YOLOv5, v8 und v10 Modellreihen in verschiedenen Größen. Die Größe des Modells bestimmt die Anzahl der trainierten Gewichte eines Modells. In diesen Versuchen werden die kleineren Modelle verwendet, weil diese grundsätzlich schneller sind. Der Preis für weniger Gewichte und somit schnelleren Inferenzen ist die geringere Genauigkeit.

Die YOLO Modelle wurden gewählt, weil sie zum Zeitpunkt dieser Arbeit gute Ergebnisse in den Bildanalysen liefern und breit eingesetzt werden. Zudem ist die Pipeline des Referenz Use Case Abschnitt 3.2.1 mit Modellen der YOLO Generation 5 umgesetzt. Die Messungen wurden jeweils auf den von Ultralytics vor trainierten YOLO Modellen mit Bildern des COCO Standard Datenset durchgeführt.

- Abkürzungsschlüssel für Legenden
 - YOLO Input size
 - Genauigkeit

3.4.2 ML Framework

Im folgenden werden die Inferenzzeiten gleicher Modelle mit unterschiedlichen ML Frameworks verglichen. Dies auf einem Raspberry Pi 5 8GB mit einem aktiven Kühllement.

3.4.2.1 Vergleich Inferenzzeiten

Die folgende Abbildung Abbildung 15 zeigt auf, wie sich die Inferenzzeiten der jeweiligen ML Frameworks in Bezug auf die YOLO Generation verhält. Ersichtlich ist, dass hauptsächlich die Inferenzzeit der PyTorch Inferenz mit jeder Generation signifikant gestiegen ist. Die anderen Frameworks weisen nur geringe Unterschiede auf, wobei die Tendenz des Anstieges bei jeder Inferenzzeit feststellbar ist. Ein weiteres Merkmal ist, dass die Inferenz mit dem ONNX Model bei allen Versuchen am wenigsten Zeit benötigt.

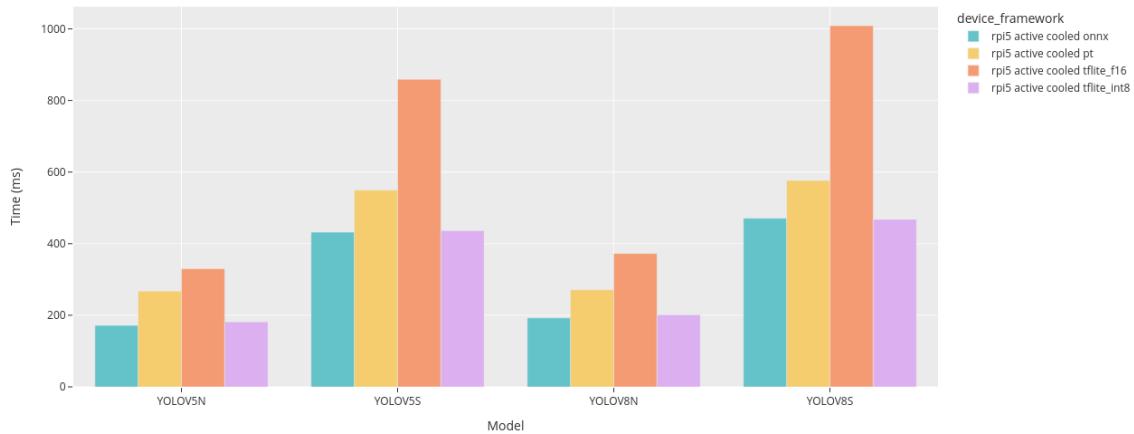


Abbildung 15: Vergleich: Inferenzzeit YOLOv5, v8 auf Raspberry Pi 5

In einem nächsten Schritt wird ONNX mit NCNN verglichen. Wie in Abschnitt 2.3.1.5 erwähnt, ist NCNN für Geräte mit begrenzter Rechenleistung optimiert. Dies zeigt sich auch in der folgenden Abbildung Abbildung 16. Die Inferenzzeiten halbieren sich nochmals gegenüber der Inferenz mit ONNX.

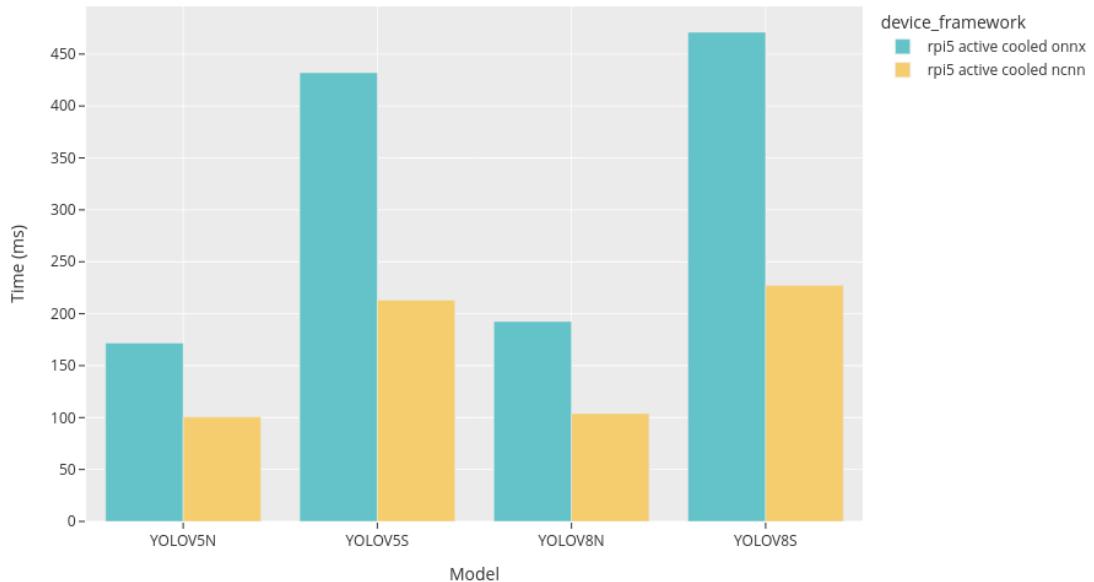


Abbildung 16: Vergleich: ONNX und NCNN auf Raspberry Pi 5

Aus diesen Analysen geht hervor, dass mit NCNN auf einer Raspberry Pi 5 CPU die schnellsten Ergebnisse erzielt werden können. Um nun die Inferenzzeiten weiter zu beschleunigen, wird zusätzliche Hardware benötigt.

3.4.3 Hardware

- Recherche Hardware
 - Beschleuniger (TPU, AI-Camera)
 - Kamera (Raspberry Pi Cam, Webcam)
 - Edge Computer (Raspberry Pi 4,5)

Im folgenden werden verschiedene Hardware Setups miteinander verglichen. Teilweise beansprucht spezifische Hardware auch definierte ML Frameworks. Somit ist der Vergleich von verschiedener Hardware nicht mit gleichen Frameworks möglich. Dennoch können die Inferenzzeiten verglichen werden.

3.4.3.1 Raspberry Pi Vergleiche

Der erste Vergleich zeigt den Fortschritt der Raspberry Pi Generationen auf. Die Inferenzzeiten eines ONNX Models ist auf einem Raspberry Pi 5 mehr als doppelt so schnell.

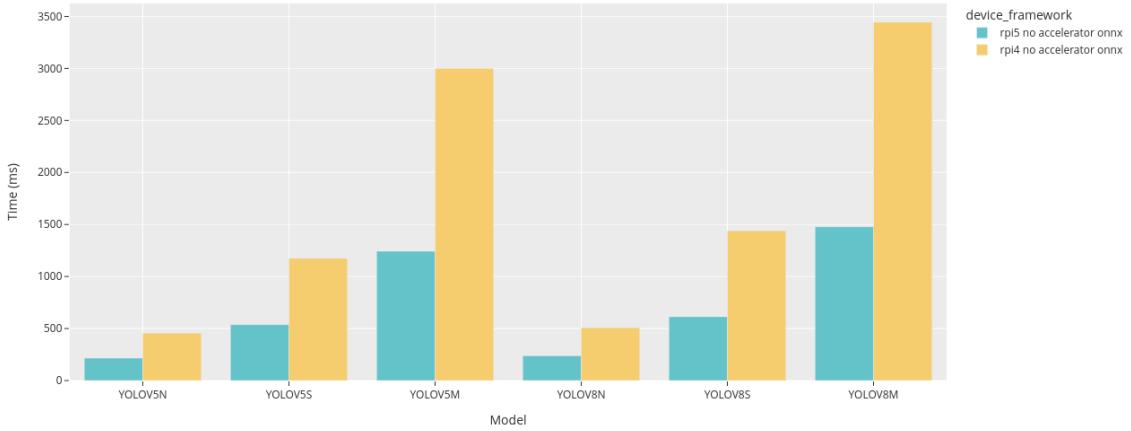


Abbildung 17: Vergleich: Raspberry Pi 4 vs. Pi 5

(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)

Ein weiterer wichtiger Punkt ist die Kühlung des Systems. Ansonsten wird die CPU Leistungs des Rechners gedrosselt. In der folgenden Abbildung 18 ist ersichtlich, dass die Inferenzzeiten mit Kühlung rund 20% Prozent schneller gegenüber der selben Inferenz auf dem Vorgängermodell Raspberry Pi 4.

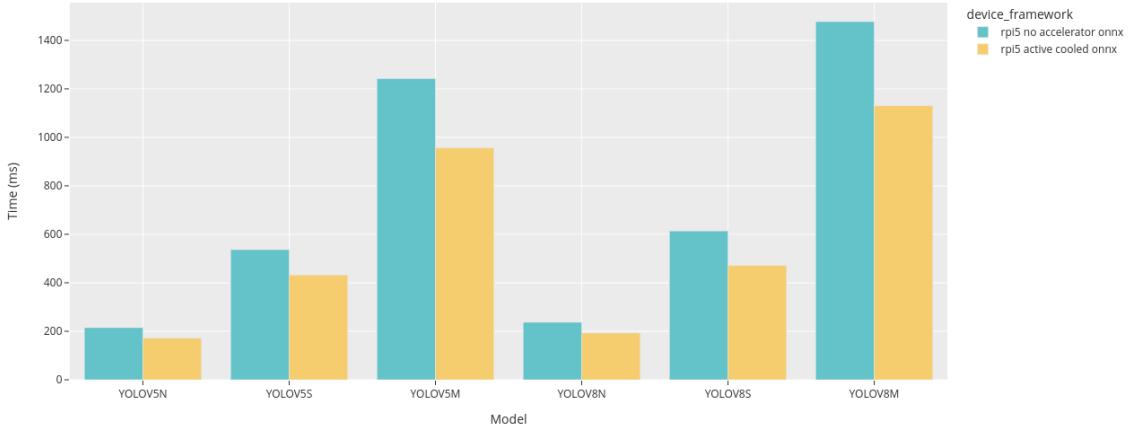


Abbildung 18: Vergleich: Raspberry Pi 5 mit und ohne Kühlung

(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)

3.4.3.2 Coral Accelerator

Das Toolkit Coral [28] wurde von Google entwickelt und stellt Beschleuniger verschiedener Art zur Verfügung. Im Kontext dieser Arbeit wurden zwei dieser Beschleuniger untersucht: Ein USB Dongle mit einer Edge TPU von 4 TOPS und eine ein über PCI verbundenes Board mit einer TPU von 8 TOPS. Bei den Versuchen hat sich herausgestellt,

dass der USB Dongle unzuverlässig ist. Es kam während länger laufenden Tests wiederholt zu Verbindungsunterbrüchen. Die Edge TPU über PCI funktionierte zuverlässig.

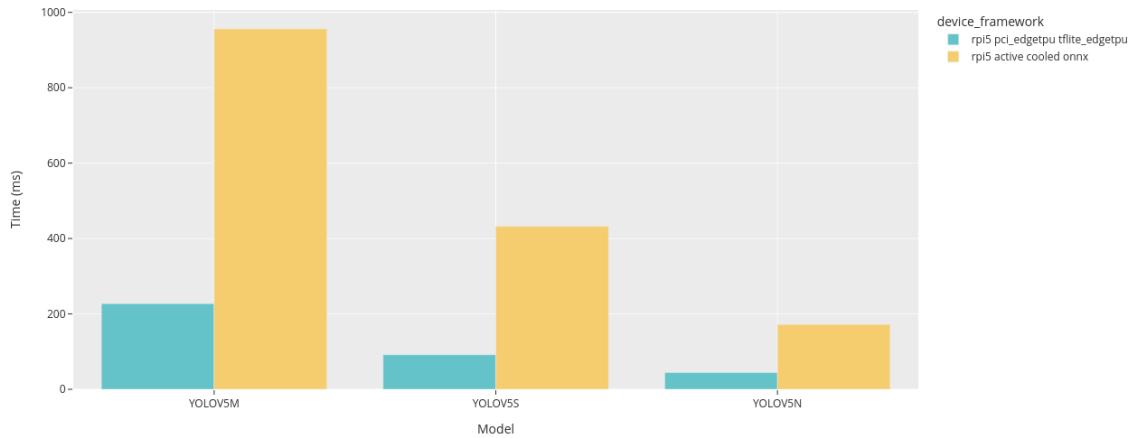


Abbildung 19: Vergleich: Raspberry Pi 5 vs. Coral PCI Edge TPU

Es ist sofort ersichtlich, dass die Inferenz auf der Edge TPU um ein Vielfaches schneller ist als auf der CPU des Raspberry Pi 5.

3.4.3.3 Hailo

Hailo [29] ist ein Unternehmen, welches auf Hochleistungs KI-Edge-Prozessoren spezialisiert hat. In dieser Untersuchung wurden die beiden Modelle Hailo8l (13 TOPS) und Hailo8 (26 TOPS) eingesetzt. Die beiden Module werden jeweils über die PCI Schnittstelle mit dem Raspberry Pi 5 verbunden. Aufgrund der benötigten PCI Schnittstelle kann das Raspberry Pi 4 Modell nicht verwendet werden. Für die Messungen der Inferenzzeit ist das von Hailo zur Verfügung gesetzte ML-Framework verwendet worden. Zudem müssen die Modelle im Hailo eigenen Format .hef sein, um sie auf den Accelerator auszuführen. Hailo stellt ein Model Zoo [30] zur Verfügung, welcher die Modelle YOLOv5s und YOLOv5m beinhaltet. Die folgende Abbildung 20 zeigt den Vergleich mit der Coral Edge TPU.

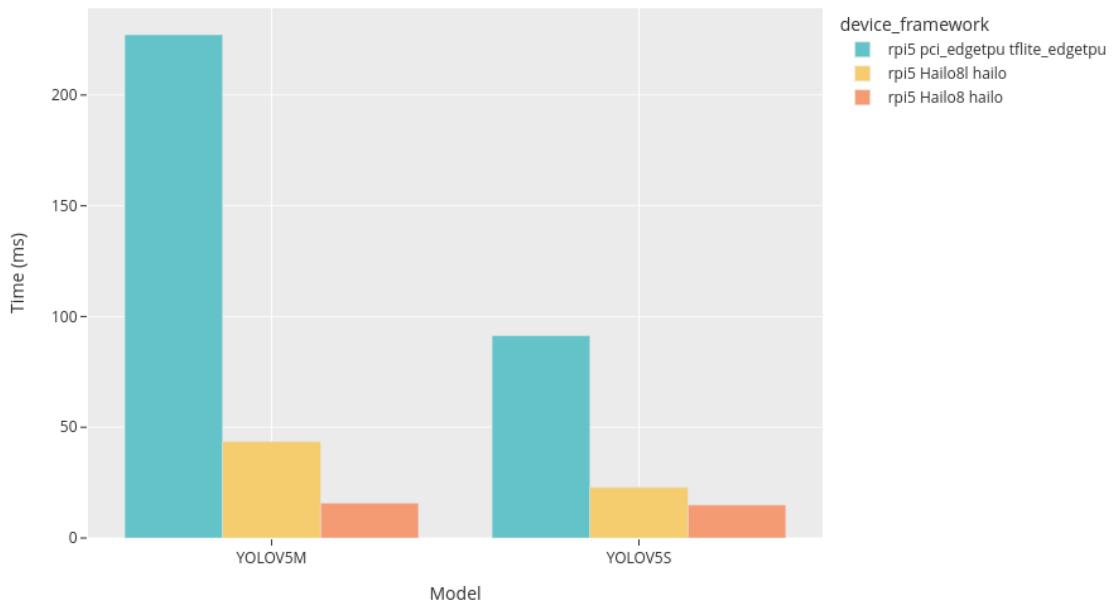


Abbildung 20: Vergleich Hailo Accelerator vs. Coral PCI Edge TPU

(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)

Die beiden Hailo Accelerator sind signifikant schneller als der schon etwas ältere Coral Accelerator. Ebenso ist ersichtlich, dass sich die doppelte Anzahl TOPS direkt auf die Inferenz Geschwindigkeit auswirkt. Die schnellste Inferenz wird somit mit dem Hailo8 Accelerator erzielt und beträgt rund 15ms.

3.5 Edge ML Setups

- billig Setup
- Performance Setup
- AI-Camera Setup
- Chart mit Trade Off

Aus der Selektion der Hardware und den in diesem Kapitel aufgeführten Evaluationen von Inferenzen lassen sich nun verschiedene Konstellationen mit unterschiedlichen Stärken, resp. Schwächen definieren. Die Abbildung 21 zeigt das Verhältnis von Kosten zu Inferenzgeschwindigkeit der vielversprechendsten Kombinationen mit dem YOLOv8n Model und Batch Size 1.

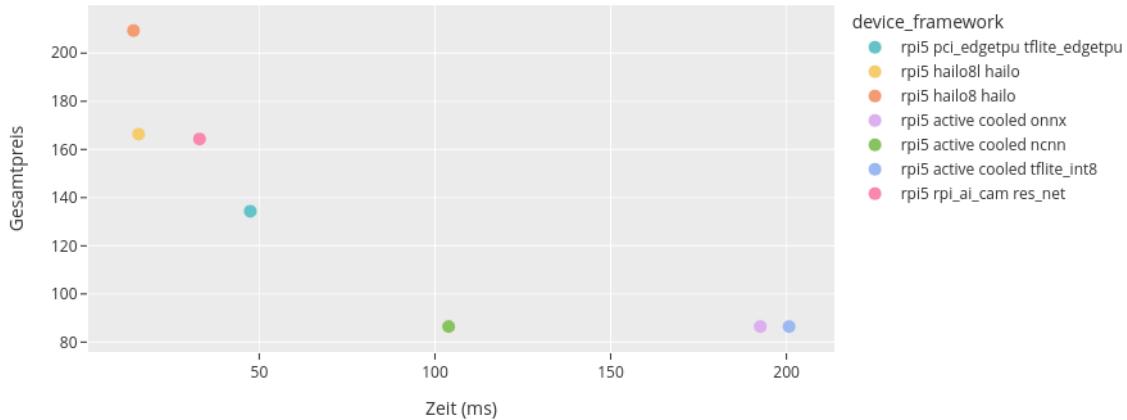


Abbildung 21: Vergleich: Kosten vs. Geschwindigkeit mit YOLOv8n

(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)

Weil das Raspberry 4 kaum billiger ist als sein Nachfolger Pi 5 ist das Model 4 für diese Bewertung auszuschliessen. Die mit Abstand am Kostengünstigsten Varianten sind jene ohne zusätzliche Beschleuniger-Hardware. Dabei schneidet das NCNN Framework auf dem Raspberry Pi 5 mit ca 9.6 FPS am besten ab gefolgt von ONNX mit rund 5.2 FPS.

Bei den Setups mit Beschleuniger sehen wir, dass mit höheren Ausgaben entsprechend mehr FPS zu erzielen sind. Die schnellsten Inferenzen liefert der teuerste Beschleuniger Hailo8 mit 26 TOPS von Hailo. Der kleinere Hailo Beschleuniger befindet sich auf Stufe der AI-Camera von Raspberry Pi. Etwas abgeschlagen, dafür auch kostengünstiger ist der M.2 PCI Edge TPU von Coral.

Um auf dem Raspberry Pi die beste Performance zu erzielen, empfiehlt sich ein Chip von Hailo. Eine Edge TPU von Google ist aufgrund des abnehmenden Supports und der weniger ausführlichen Dokumentation nicht empfehlenswert.

Lässt es die Applikation zu, ist auch die AI-Kamera von Raspberry Pi eine gute Wahl. Dabei muss berücksichtigt werden, dass in Systemen mit mehr als einer Inferenz nur die erste auf der Kamera durchgeführt werden kann. Darauffolgende Inferenzen müssten auf der CPU oder auf einem weiteren Beschleuniger ausgeführt werden. Ein Setup mit einem weiteren Beschleuniger würde natürlich auch die Kosten erhöhen.

3.6 Schlussfolgerung

Paper referenzieren

4 Umsetzung

(Was ich tatsächlich Umgesetzt habe)

Dieses Kapitel beschreibt die Umsetzung einer Software zum Betreiben einer Machine Learning Pipeline. Das System umfasst die Erfassung von Bildern, ausführen einer Machine Learning Analyse und der Weiterverarbeitung deren Resultate. Die Entwicklung erfolgte drei Iterationen, bei welchen jeweils die erstellte Architektur reflektiert und anhand der gewünschten Qualitäten angepasst wurden.

4.1 System Architektur

- Qualitäten (Erweiterbarkeit, Robustheit, Self-contained, Einfachheit)
- Modularisierung (Komponenten: Cam, ML Framework)

Damit das System Citizen Science fähig ist, muss es möglichst leicht für neue Szenarien anpassbar sein. Bei den auswechselbaren Komponenten handelt es sich um: Die Quelle zum erfassen von Bildern mit einem Buffer, die Analyse der einzelnen Bilder und die Verarbeitung der Analysedaten. Diese drei Komponenten müssen ohne Anpassung von bestehendem Code auswechselbar sein, um die Komplexität für das betreibende Projektteam gering zu halten. Da ein Machine Learning Modell sehr individuell ist, verunmöglicht es eine generelle Implementierung einer Pipeline für verschiedene Modelle.

Ein weiterer wichtiger Punkt ist der Buffer für erfasste Bilder. Grundsätzlich ist das System zur Biodiversitätsüberwachung ausgelegt, weshalb die Zeit zur Erfassung von Bildern limitiert auf auf die Dauer des Sonnenscheins ist. Somit ist für die Analyse potentiell mehr Zeit zur Verfügung als für die Aufnahme der Bilder.

4.1.1 Komponenten Diagram

Die folgende Abbildung 22 zeigt eine übersicht des Systems. Eine Source repräsentiert die Quelle eines Bildes, wie zum Beispiel eine Kamera. Als Buffer zwischen der Analyse und der Source dient eine Queue. Bei der Operation handelt es sich um die Komponente der Analyse. Dies kann eine Machine Learning Inferenz sein oder eine andere Operation, die ein Bild verarbeitet und ein Resultat generiert. Anschliessend gelangt der Output der Operation in eine oder mehrere Sinks. Die Sink ist jener Komponenten, welches das Resultat weiterverarbeitet. Dies ist beispielsweise eine Datenbank.

Frames weglassen

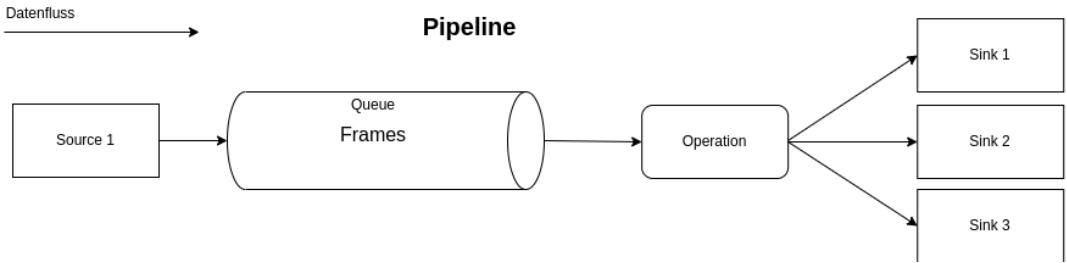


Abbildung 22: Pipeline Komponenten

(Quelle: Screenshot eigene Darstellung, Andri Wild, 2025)

4.1.2 Qualitäten

Da die Software für verschiedenste Anwendungen zum Einsatz kommen kann und dafür jeweils auch angepasst oder erweitert werden muss, ist das Hauptaugenmerk auf die folgenden Qualitäts Kriterien gelegt worden:

- Einfachheit: Der Code ist in Python geschrieben, eine Programmiersprache die gerade bei Forschenden hohen Bekanntheitsgrad geniesst. Implementierungen sind jeweils einfach gehalten, so dass sie beim lesen des Codes leicht verständlich und interpretierbar sind.

beispiel, einfacher, komplexer code

1 `print("hello world")`

`python`

- Modularisierung: Alle Komponenten die austauschbar sind, sind durch abstrakte Klassen ohne Implementierung der Methoden vorgegeben. Interfaces, wie man es aus anderen Sprachen kennt gibt es in Python nicht.

4.1.3 Konfiguration

Das System lässt sich mittels Konfigurations File definieren. Die Konfiguration enthält im wesentlichen drei Komponenten, einen für jeden auswechselbaren Teil der Software. Somit lassen sich mehrere Sources, Operations und Sinks definieren. Diese Einzelteile können dann zur Laufzeit angepasst werden. Dieser Mechanismus erleichtert das angenehme beobachten und anpassen einer Pipeline. Das folgende Listing 1 zeigt eine Beispiel Konfiguration mit jeweils einer Option für jeden Komponenten Typ.

```

1   sources:
2     - name: webcam
3       file_path: ./source/impl/webcam.py
4       class_name: Webcam
5
6       parameters:
7         device: "/dev/video0"
8         width: 640
9         height: 640
10      operations:
11        - name: detect coco objects
12          class_name: UlDetect
13          file_path: ./pipe/impl.ulDetect.py
14          parameters:
15            model_path: ./resources/ml_models/yolol1n.onnx
16            label_path: ./resources/labels/coco.txt
17            confidence_threshold: 0.5
18            nms_threshold: 0.5
19      sinks:
20        - name: console
21          class_name: Console
22          file_path: ./sink/impl/console.py

```

Listing 1: Beispiel Pipeline Konfiguration

Jede Komponente hat die folgenden Werte, die definiert sein müssen:

- **name**: Der Name des Komponenten, damit man ihn später über das Konfigurations Interface erkennen kann.
- **file_path**: Der Pfad zum File, in der die zu ladende Klasse implementiert ist.
- **class_name**: Die zu ladende Klasse aus dem definierten File. In Python können mehrere Klassen in einem File implementiert sein.
- **parameters** (optional): Ist ein Satz an Parameter, die der Klasse beim initialisieren übergeben werden. Weil diese Parameter sehr Anwendungsspezifisch sind, ist die Struktur dieses Objekts offen gelassen.

Auf diese Weise lassen sich unabhängig vom Rest der Software Klassen definieren und in die Pipeline einbinden.

4.2 Software Design

(Aufteilung in Klassen)

- Klassen Diagramm
- Sequenz Diagramm

In diesem Abschnitt wird aufgezeigt, wie die einzelnen Komponenten mit einander kommunizieren. Zu diesem Zweck ist folgend ein Klassen Diagramm mit den relevanten Elementen darstellt. Die zentrale Komponente stellt die Klasse Pipeline dar. Diese hat eine Beziehung zu der Source, Operation und Sink und orchestriert den Datenfluss zwischen den Komponenten. Mit der Applikation wird auch ein Konfiguration Server gestartet. Mittels setter-Methoden auf der Pipeline lassen sich die austauschbaren Komponenten konfigurieren. Der Konfigurations-Server liefert ein simples index.html File aus, welches die Optionen aus dem Konfigurationsfile anzeigt. Durch Auswählen und Bestätigen gelangt die Information an den Server zurück, worauf die Pipeline angepasst wird.

Kardinalität im Diagramm

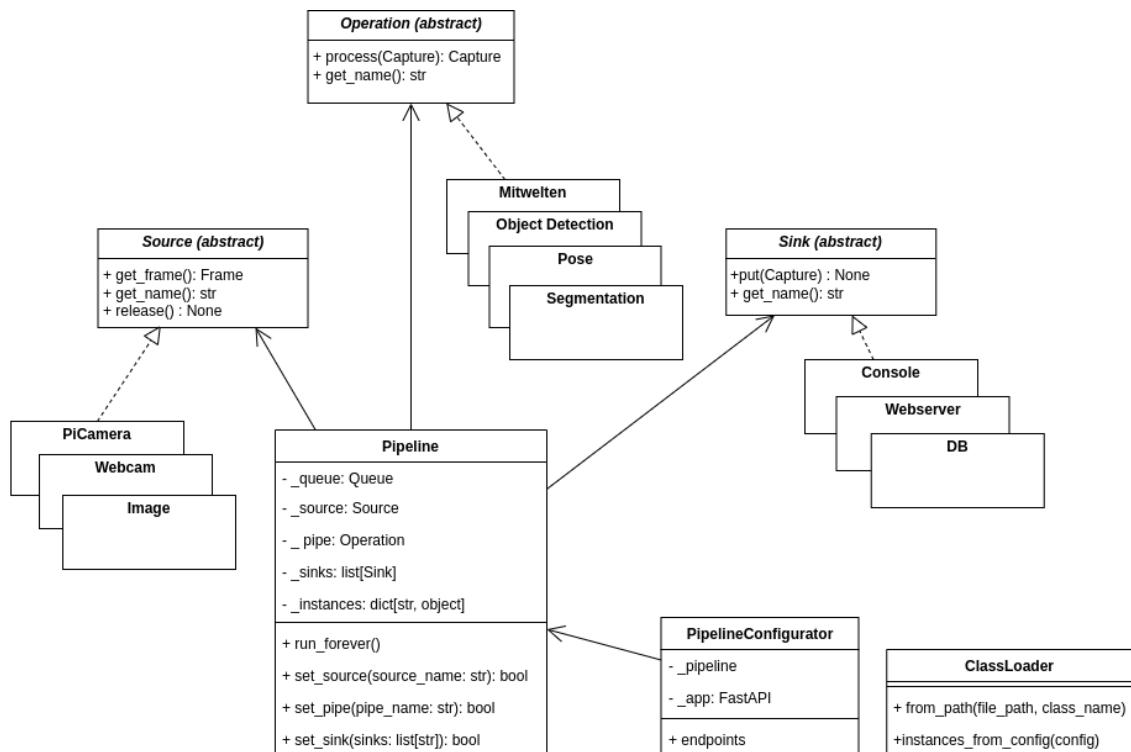


Abbildung 23: Klassen Diagramm

(Quelle: Screenshot eigene Darstellung, Andri Wild, 2025)

Um die zeitlichen Abläufe besser dazustellen ist folgend ein vereinfachtes Sequenz Diagramm dargestellt. Die Pipeline ist dafür verantwortlich, dass ein Thread zur Erfassung

von Bildern gestartet wird. Diese Bilder werden in eine Thread-Safe Queue zwischengespeichert. Im Haupt-Thread der Applikation holt die Pipeline ein Bild von Queue ab und führt die Operation aus, wobei es sich zum Beispiel um die Mitwelten Analyse handelt. Anschliessend übergibt die Pipeline die Resultate der Analyse einer oder mehreren Sinks. Es kann mehrere Sinks geben, weil die Möglichkeit bestehen soll, Resultate abzuspeichern und gleichzeitig auf der Konsole oder auf einem Web Interface visuell darzustellen.

Mehrzahl Sinks, Thread1 -> Thread

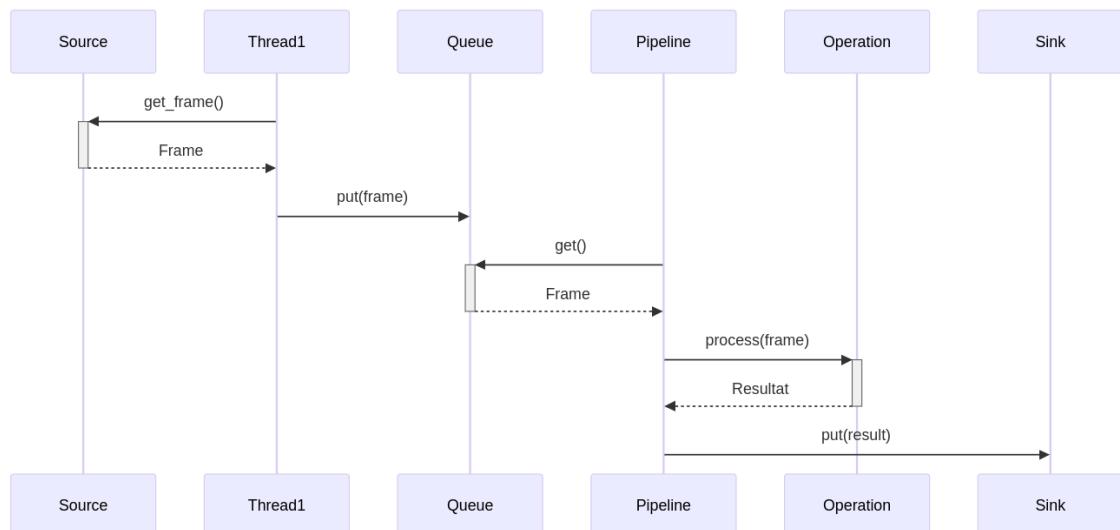


Abbildung 24: Pipeline Komponenten

(Quelle: Screenshot eigene Darstellung, Andri Wild, 2025)

4.3 Prototypen: verschiedene System Setups

(welche Konfiguration wurde verwendet)

- Prototyp 1
- Prototyp 2
- FPS und andere Metriken

Der folgende Abschnitt beschreibt verschiedene Setups, welche für den Einsatz der Mitwelten Analyse möglich wären. Alle Ansätze erfüllen die nötigen Anforderungen. (nicht nur MW)

4.3.1 Prototyp 1

- Mitwelten Analyse auf CPU, ONNX Model, nicht NCNN weil kein export support

4.3.2 Prototyp 2

- Mitwelten Analyse auf edge TPU

4.3.3 Prototyp 3

- AI-Camera Setup

4.3.4 Prototyp 4

- Hailo Object Detection

5 Validierung

(Auf MW Pipeline, mit Blumen Bilder)

- Performance
- Energie
- Konkurrenz (Node Red)
- reCamera

Abbildungsverzeichnis

Abbildung 1: Erweiterte thematische Karte zum Lernen von Freiwilligen (Quelle: The Science of Citizen Science, S.300)	13
Abbildung 2: Pollinator Kamera im Feld (Quelle: mitwelten.ch, aufgerufen am 23.01.2025)	15
Abbildung 3: Machine Learning Tasks (Quelle: https://github.com/sony/model_optimization , aufgerufen am 20.01.2025) ...	19
Abbildung 4: Symbolbild Neuronales Netzwerk (Quelle: lamarr-institute.org/blog/deep-neural-networks, abgerufen am 23.01.2025) ..	20
Abbildung 5: Raspberry Pi 4 (Quelle: https://www.pi-shop.ch/raspberry-pi-4-model-b-4gb , aufgerufen am 23.01.2025)	
23	
Abbildung 6: Raspberry Pi 5 (Quelle: https://www.pi-shop.ch/raspberry-pi-5-16gb-ram?src=raspberrypi , aufgerufen am 23.01.2025)	23
Abbildung 7: BeagleY-AI Board (Quelle: https://www.seeedstudio.com/BeagleYr-AI-beagleboard-orgr-4-TOPS-AI-Acceleration-powered-by-TI-AM67A.html , aufgerufen am 23.01.2025)	24
Abbildung 8: Coral USB Accelerator (Quelle: https://www.coral.ai/products/accelerator , aufgerufen am 22.01.2025)	25
Abbildung 9: Coral PCI Accelerator (Quelle: https://www.coral.ai/products/m2-accelerator-dual-edgetpu , aufgerufen am 22.01.2025)	25
Abbildung 10: Hailo Accelerator (Quelle: https://www.pi-shop.ch/raspberry-pi-ai-hat-13t , aufgerufen am 22.01.2025) .	26
Abbildung 11: Raspberry Pi AI Camera (Quelle: https://www.berrybase.ch/raspberry-pi-ai-camera , aufgerufen am 22.01.25) .	26
Abbildung 12: Raspberry Pi AI Camera Architektur (Quelle: https://www.raspberrypi.com/documentation/accessories/ai-camera.html , aufgerufen am 20.01.2025)	27
Abbildung 13: ML Exploration Dashboard	

(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)	28
Abbildung 14: Mitwelten Machine Learning Pipeline	
(Quelle: Automated Analysis for Urban Biodiversity Monitoring, Timeo Wullschleger)	30
Abbildung 15: Vergleich: Inferenzzeit YOLOv5, v8 auf Raspberry Pi 5	32
Abbildung 16: Vergleich: ONNX und NCNN auf Raspberry Pi 5	33
Abbildung 17: Vergleich: Raspberry Pi 4 vs. Pi 5	
(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)	34
Abbildung 18: Vergleich: Raspberry Pi 5 mit und ohne Kühlung	
(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)	34
Abbildung 19: Vergleich: Raspberry Pi 5 vs. Coral PCI Edge TPU	35
Abbildung 20: Vergleich Hailo Accelerator vs. Coral PCI Edge TPU	
(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)	36
Abbildung 21: Vergleich: Kosten vs. Geschwindigkeit mit YOLOv8n	
(Quelle: Screenshot eigene Entwicklung, Andri Wild, 2025)	37
Abbildung 22: Pipeline Komponenten	
(Quelle: Screenshot eigene Darstellung, Andri Wild, 2025)	39
Abbildung 23: Klassen Diagramm	
(Quelle: Screenshot eigene Darstellung, Andri Wild, 2025)	41
Abbildung 24: Pipeline Komponenten	
(Quelle: Screenshot eigene Darstellung, Andri Wild, 2025)	42

Tabellenverzeichnis

Appendix A: Supplementary Material

– Supplementary Material –

Bibliographie

- [1] A. Bonn *u. a.*, „Grünbuch Citizen Science Strategie 2020 für Deutschland“, Berlin, 2016.
- [2] K. Vohland *u. a.*, Hrsg., „The Science of Citizen Science“. Springer International Publishing, Cham, 2021. doi: 10.1007/978-3-030-58278-4.
- [3] bdm, „Verlässliche Daten über unsere Lebensgrundlage“. Zugegriffen: 15. Januar 2025. [Online]. Verfügbar unter: <https://www.biodiversitymonitoring.ch/index.php/de/>
- [4] T. Wullschleger, „Data Acquisition for Urban Biodiversity Monitoring“.
- [5] T. Wullschleger, „Automated Analysis for Urban Biodiversity Monitoring“.
- [6] „TensorFlow“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://www.tensorflow.org/>
- [7] „PyTorch“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://pytorch.org/>
- [8] „OpenAI standardizes on PyTorch“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://openai.com/index/openai-pytorch/>
- [9] „ONNX | Home“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://onnx.ai/>
- [10] H. Ni und The ncnn contributors, „ncnn“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://github.com/Tencent/ncnn>
- [11] „Das führende KI-Chip-Unternehmen für Edge-Geräte“. Zugegriffen: 27. Januar 2025. [Online]. Verfügbar unter: <https://hailo.ai/de/company-overview/>
- [12] Ultralytics, „Home“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://docs.ultralytics.com/>
- [13] Ultralytics, „YOLOv5“. Zugegriffen: 23. Januar 2025. [Online]. Verfügbar unter: <https://docs.ultralytics.com/models/yolov5>
- [14] „TPU (Tensor Processing Unit)“. Zugegriffen: 23. Januar 2025. [Online]. Verfügbar unter: <https://www.ultralytics.com/glossary/tpu-tensor-processing-unit>
- [15] „What is a Neural Processing Unit (NPU)? | IBM“. Zugegriffen: 23. Januar 2025. [Online]. Verfügbar unter: <https://www.ibm.com/think/topics/neural-processing-unit>

- [16] „NPU vs GPU: What's the Difference? | IBM“. Zugegriffen: 23. Januar 2025. [Online]. Verfügbar unter: <https://www.ibm.com/think/topics/npu-vs-gpu>
- [17] „Raspberry Pi 4 Model B - 8GB“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.pi-shop.ch/raspberry-pi-4-model-b-8gb>
- [18] „Raspberry Pi 5 Model B 8GB“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.pi-shop.ch/raspberry-pi-5-8-gb>
- [19] „BeagleY®-AI“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.seeedstudio.com/BeagleYr-AI-beagleboard-orgr-4-TOPS-AI-Acceleration-powered-by-TI-AM67A.html>
- [20] „Coral USB Accelerator“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.pi-shop.ch/coral-usb-accelerator>
- [21] „Pineboards Hat AI! Dual, Edge Coral TPU für Raspberry Pi 5, Bundle - kaufen bei BerryBase“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.berrybase.ch/pineboards-hat-ai-dual-edge-coral-tpu-fuer-raspberry-pi-5-bundle>
- [22] „Pineboards Hat AI!, Coral Edge TPU Erweiterung für Raspberry Pi 5, Bundle - kaufen bei BerryBase“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.berrybase.ch/pineboards-hat-ai-coral-edge-tpu-erweiterung-fuer-raspberry-pi-5-bundle>
- [23] „Raspberry Pi AI HAT+ (13T)“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.pi-shop.ch/raspberry-pi-ai-hat-13t>
- [24] „Raspberry Pi AI HAT+ (26T)“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.pi-shop.ch/raspberry-pi-ai-hat-26t>
- [25] „Raspberry Pi AI Camera - kaufen bei BerryBase“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.berrybase.ch/raspberry-pi-ai-camera>
- [26] „AI Camera - Raspberry Pi Documentation“. Zugegriffen: 20. Januar 2025. [Online]. Verfügbar unter: <https://www.raspberrypi.com/documentation/accessories/ai-camera.html>
- [27] awild, „andriwild/ip7-ml-model-eval“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://github.com/andriwild/ip7-ml-model-eval>
- [28] „Products“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://coral.ai/products/#prototyping-products>

- [29] „KI-Edge-Prozessoren für Höchstleistung - Hailo KI Chip“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: <https://hailo.ai/de/>
- [30] T. Tapuhi *u. a.*, „Hailo Model Zoo“. Zugegriffen: 16. Januar 2025. [Online]. Verfügbar unter: https://github.com/hailo-ai/hailo_model_zoo