h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

Fachbereiche
Mathematik und Naturwissenschaften
&
Informatik

Personenzählung on the Edge mit neuronalen Netzen als Teil von Gebäudesensorik

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

im Studiengang Data Science

vorgelegt von

Robert Hecker

Matrikelnummer: 760742

Referentin	:	Prof. Dr. Elke Hergenröther
Korreferentin	:	Prof. Dr. Antje Jahn

Ausgabedatum	:	20.06.2024
Abgabedatum	:	18.12.2014

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 16. Dezember 2024

Robert Hecker

ABSTRACT

Heating and ventilation account for significant costs in buildings, both public and private. Inefficient heating wastes energy, inadequate ventilation may be detrimental to a person's well-being. In private buildings, factors such as a known work schedule can be used to turn off heating and ventilation when nobody is present. Public buildings and office spaces usually don't offer such possibilities to detect whether parts of a building or certain rooms are being used. Determining usage e. g. through timetables in universities is inaccurate and may not overlap with actual usage. Instead, classic occupancy detection sensors can be used. Simple detectors such as motion sensors can only detect whether people are present but cannot provide information about their quantity, while advanced sensors such as depth cameras are prohibitively expensive, making widespread use unattractive. Additionally, video-based counting techniques that transfer live video from each room to a central location for counting raise significant privacy concerns and offer an attack surface for cybercriminals to extract highly-sensitive data. To alleviate these concerns and to allow for widespread use, this thesis has the central theme of implementing an occupancy detection system using an overhead mounted camera. Cost-efficient embedded hardware shall be used to count people on-device using neural networks. The main research question of this work asks how such a system could be implemented on real hardware and which techniques and frameworks aid such an implementation. To answer this question, the specific challenges of running neural networks on resource-constrained embedded systems are illustrated based on a selection of devices. Existing neural network architectures are researched and chosen based on whether they fit these constraints. The selection of networks is then trained on several datasets and the networks are evaluated regarding both their object detection metrics and their performance on the counting task itself. Techniques such as quantization and teacher-student training are also evaluated in regards to whether they aid in preparing neural networks for the specific challenges of this resource-constrained context. Results of the experiments with these object detection networks show that training on a large, but not representative dataset results in better counting results on the used test dataset, but the significance of these results remains unclear due to the low size and variance of the given dataset. Quantization can be confirmed as a valuable tool to lessen resource demands on storage for network parameters, while teacher-student training does not offer benefits. The implementation on real hardware uses several different frameworks and runtimes to convert the neural networks and illustrates a strong fragmentation of the technologies used, while training specialized neural nets for this task is hindered by a lack of freely-available, representative datasets.

ZUSAMMENFASSUNG

Das Beheizen und Belüften von Gebäuden stellt einen sehr kostspieligen Aspekt des Betreibens öffentlicher als auch privater Gebäude dar. Ein unnötiges Heizen verursacht hohe Kosten und verschwendet Energie, während eine unzureichende Belüftung zu Problemen wie Konzentrationsstörungen bei anwesenden Personen führen kann. In privaten Wohngebäuden lässt sich die Gebäudeklimatisierung mit Konzepten des Smart Homes intelligent nach Bedarf steuern, in öffentlichen Gebäuden oder Büros ist es hingegen schwerer, die Auslastung eines Gebäudeteils oder einzelner Räume zu erkennen. Das Heizen nach Belegplänen ist ungenau, klassische Gebäudesensorik wie Bewegungsmelder können nur die Anwesenheit von Personen, aber nicht deren genaue Anzahl bestimmen. Vorhandene Systeme zur Personenzählung nutzen kostspielige Sensortypen wie Tiefenkameras, welche einen flächendeckenden Einsatz unattraktiv machen. Ebenfalls kommen bei Systemen, welche Videodaten zur Auswertung zu einer Zentrale übertragen, Datenschutzbedenken auf. In dieser Arbeit soll aus diesem Grund die Personenzählung durch eine zentral in einem Raum an der Decke montierte Kamera betrachtet werden. Hierbei soll die Zählung durch neuronale Netze auf kostengünstiger und somit leistungsschwacher eingebetteter Hardware durchgeführt und untersucht werden. Es wird sich der Frage gewidmet, wie neuronale Netze zur Personenzählung unter diesen Bedingungen implementiert werden können. Zur Beantwortung dieser Frage wird die Problemstellung anhand einer Hardwareauswahl mehrerer geeigneter Geräte illustriert. Durch eine Recherche existierender neuronaler Netze zur Objektdetektion werden solche identifiziert, welche unter den Rahmenbedingungen der geringen Leistung der Geräte einsetzbar wären und diese Netze mit geeigneten Datensätzen auf die Erkennung von Personen angepasst. Weiterhin werden problemspezifische Techniken wie Quantisierung und Teacher-Student-Training hinsichtlich ihrer Wirksamkeit zur Verkleinerung neuronaler Netze untersucht. Nach Evaluation der erhaltenen Netze hinsichtlich ihrer Zählqualität und Implementierung auf echter Hardware kann festgestellt werden, dass Netze, welche auf einem großen, aber für die Problemstellung nicht repräsentativen Datensatz trainiert wurden, eine ausreichend gute Personenzählung ermöglichen. Das Konzept der Quantisierung kann als nützliche Technik zur Verkleinerung neuronaler Netze bestätigt werden, während Teacher-Student-Training keine leistungsfähigeren Netze als ein direktes Training ergibt. Es kann identifiziert werden, dass eine Implementierung auf realer Hardware durch eine starke Fragmentierung des Marktes in viele verschiedene Laufzeitumgebungen und Frameworks herausfordernd ist, während das Training der Modelle durch einen Mangel an öffentlichen Datensätzen erschwert wird.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

GLOSSAR

Neural Processing Unit Koprozessor zur Beschleunigung üblicher mathematischer Operationen innerhalb Machine-Learning-Modellen, besonders neuronaler Netze.

Post-Training-Quantization Eine Quantisierungsmethode, welche auf ein bereits trainiertes neuronales Netz angewandt wird, anstatt während des Trainings.

Single-board computer Computersysteme, welche alle Bauteile eines Computer auf einer Platine integrieren, sich dabei von Mikrocontrollern durch eine höhere Leistung abgrenzen. [34, S. 2]

TensorFlow Lite Ein auf TensorFlow basierendes Framework zur Ausfuehrung neuronaler Netze auf mobilen (aber nicht zwingend eingebetteten) Geraeten.

TensorFlow Lite Micro Eine Variante von TensorFlow Lite, welche durch einen reduzierten Funktionsumfang und besondere Optimierungen auf eingebettete Geräte wie Mikrocontroller zugeschnitten ist.

ABKÜRZUNGSVERZEICHNIS

AP Average Precision

CNN Convolutional Neural Network

IoU Intersection over Union

MAE Mean Absolute Error (dt. mittlerer absoluter Fehler)

NPU Neural Processing Unit

PTQ Post-Training-Quantization

RAM Random-access Memory

SBC Single-board computer
SSD Single Shot MultiBox Detector

TFL TensorFlow Lite
TFLM TensorFlow Lite Micro

YOLO You Only Look Once

Teil I

THESIS

1 EINLEITUNG

Moderne Gebäude können mit einer Vielzahl an Sensoren ausgestattet werden, um verschiedenste Funktionen zu automatisieren. Mit dieser Sensortechnik können viele Ziele verfolgt werden, eines davon ist eine intelligente und bedarfsgerechte Klimatisierung. Durch kontrolliertes Heizen und Belüften von Räumlichkeiten können Kosten gespart und die Umwelt geschützt werden.

In dieser Arbeit soll das Konzept einer deckenmontierten Kamera zur Zählung von anwesenden Personen als Teil einer modernen Gebäudesensorik untersucht werden. Um die Relevanz dieses Konzeptes zu zeigen, wird im Folgenden in den Hintergrund und die Motivation der Arbeit eingeführt. Daran anschließend werden die Forschungsfragen, welche untersucht und beantwortet werden sollen, vorgestellt. Abschluss dieses Kapitels bildet eine kurze Darstellung des Aufbaus der Arbeit.

1.1 MOTIVATION UND HINTERGRUND

Nach Einschätzungen des Umweltbundesamtes entfiel im Jahr 2021 mehr als ein Viertel des gesamten Energieverbrauchs in Deutschland auf die Beheizung und Kühlung von Gebäuden [65]. Eine falsch konfiguriertes oder ineffizientes Klimatisieren von Gebäuden führt somit zu massiven Energieaufwänden, welche Kosten verursachen und der Umwelt schaden. Das effiziente Heizen und Klimatisieren von Gebäuden ist somit sowohl von einem wirtschaftlichen als auch von einem gesellschaftlichen Standpunkt ein erstrebenswertes Ziel. Besonders in öffentlichen Gebäuden bieten sich hinsichtlich der Kontrolle der Gebäudetemperatur und Klimatisierung viele Möglichkeiten an. Ein Beispiel bildet die Steuerung der Heizung eines jeden einzelnen Raumes abhängig von der tatsächlichen Nutzung der Fläche. Ein Raum, der nicht aktiv genutzt wird, muss dementsprechend nicht oder nicht so stark beheizt werden wie ein aktiv genutzter Raum.

Das Erkennen dessen, ob Teile eines Gebäudes durch Personen genutzt werden, ist kein neues Problem. Mit klassischer Sensorik existieren verschiedenste Ansätze, um die Notwendigkeit der Beheizung oder Klimatisierung eines Raumes zu erkennen. Beispielsweise könnten Sensoren zur Messung der Luftqualität signalisieren, ab wann ein Raum belüftet werden muss, oder ein Bewegungsmelder erkennt die Anwesenden durch ihre Bewegung im Raum. Während solche Sensorik bewährt ist und das Ziel einer adäquaten Beheizung und Klimatisierung durchaus erfüllen kann, leiden die genutzten Sensoren unter verschiedenen Problemen. Messungen der Luftqualität besitzen sehr hohe Latenzzeiten; eine Verschlechterung des Raumklimas entsteht nicht direkt bei Eintreten einer Person in den Raum und ist erst nach ei-

ner längeren Zeit messbar. Bewegungsmelder erkennen keine Personen, die sich nicht oder nur wenig bewegen, gleichzeitig können Objekte im Blickfeld des Sensors, welche sich bewegen, aber keine Menschen sind, zu Fehlerkennungen führen. Beide aufgeführten Beispiele besitzen ebenfalls das Problem, dass die Erkennung der Anwesenheit von mindestens einer Person möglich ist, aber eine Differenzierung zwischen einer und mehreren Personen nur schwer bzw. gar nicht erfolgen kann. Es ist anzunehmen, dass ein Bewegungsmelder keine Zählung der Personen vornehmen kann, eine Messung der Luftqualität könnte basierend auf der Geschwindigkeit, mit der sich die Luftwerte verändern, eventuell grobe Schlüsse auf die Anzahl der Personen ziehen.

Für eine effiziente Kontrolle von Gebäuden ist es hilfreich, neben der Erkenntnis, ob ein Raum überhaupt belegt ist, Informationen über die Höhe der Auslastung des Raumes zu sammeln, also die Anzahl der Personen in einem Raum zu zählen. Eine einfach umzusetzende Möglichkeit ist hierbei, den Raum mit einer Kamera zu erfassen und Personen auf dem Kamerabild zu erkennen. Indem die unterschiedlichen Erkennungen gezählt werden, erhält man so die Anzahl der anwesenden Personen. Während algorithmische Verfahren zur Erkennung von Personen auf Bildern existieren, wird das Problem des Zählens von Personen üblicherweise mit Techniken des maschinellen Lernens angegangen. Hierzu werden neuronale Netze eingesetzt, welche die Kamerabilder verarbeiten und eine Ausgabe erzeugen, welche entweder direkt die Anzahl der Personen enthält oder aus derer sich diese Zahl ableiten lässt. Ein Nachteil neuronaler Netze besteht darin, dass die Berechnungen, welche im Zuge dieser Inferenz nötig sind, mitunter aufwändig sind und eine gewisse Mindestanforderung an die zur Berechnung verwendete Hardware stellen. Bedingt durch die immer wachsende Verfügbarkeit an Rechenkapazität benötigen viele zeitgemäße Implementierungen neuronaler Netze spezielle Hardware wie Grafikkarten, um effizient ausgeführt werden zu können. Eine Möglichkeit, dieses Problem anzugehen, besteht darin, die Aufnahmen von Kameras an eine zentrale Recheneinheit zu übertragen und die Personenzählung dort auf leistungsstarker Hardware umzusetzen. Während ein solches Vorgehen sehr einfach umzusetzen ist, entstehen Bedenken hinsichtlich des Schutzes der Privatsphäre der aufgenommenen Personen. Selbst in dem Falle, dass die Aufnahmen für Menschen nicht einsehbar sind und nur für das eingesetzte Machine-Learning-System sichtbar sind, stellt alleine die Übertragung der Daten über weite Strecken von der Kamera zu einer Zentrale einen möglichen Angriffsvektor für Cyberkriminelle dar. Sollte es möglich sein, Zugriff auf Kameras für jegliche Räume eines Gebäudes zu erhalten, so wäre dies eine massive Verletzung der Privatsphäre der Aufgenommenen.

Eine Alternative ist die Verarbeitung der Kamerabilder möglichst nahe an der Quelle der Daten. Da die Kameraaufnahmen nicht mehr an eine zentrale Verarbeitungseinheit transportiert werden müssen, sinkt die Wahrscheinlichkeit, dass die Daten abgegriffen werden können, enorm. Nachteil dieser Verarbeitung "on the Edge" ist, dass Einschränkungen hinsichtlich

der Leistungsfähigkeit der verwendeten Hardware gegeben sind. Falls beispielsweise sowohl die Kamera als auch die Hardware zur Bildverarbeitung zusammen in einem Gehäuse verbaut werden sollen, kann die verarbeitende Hardware keine beliebige Größe annehmen, was ebenfalls die Rechenleistung einschränkt. Beim Designprozess solcher auf einen Zweck zugeschnittener Hardware werden sogenannte Embedded-Systeme eingesetzt, bei denen ein auf eine bestimmte Funktion zugeschnittenes Computersystem umgesetzt wird.

Diese Arbeit entstand im Rahmen eines Forschungsprojektes, welches als Ziel die Entwicklung eines neuen Hardwaresystems für eine Personenzählung in Innenräumen hatte. Die in dieser Arbeit getätigten Untersuchungen besaßen somit teilweise zusätzliche Einschränkungen, beispielsweise sollten die Kosten für ein aus Kamera und Prozessor bestehendes Gesamtsystem unter 100 Euro gehalten werden. Ebenfalls soll die Erkennung von Personen auf möglichst niedrigauflösenden Bildern geschehen, um eventuelle Bedenken hinsichtlich der Identifizierbarkeit der abgebildeten Personen anzusprechen.

1.2 ZIEL DER ARBEIT

Ziel dieser Arbeit ist es, die Personenzählung mit Hilfe einer im Raum an der Decke angebrachten Kamera auszuarbeiten. Die Kamera ist hierbei zentral an der Decke montiert und erfasst den Raum somit aus einer Vogelperspektive. Es ist nicht das Ziel, ein vollständig ausgereiftes System zu erschaffen, sondern nur mit einem Proof-of-Concept zu illustrieren, dass die Problemstellung im Rahmen der gegebenen Einschränkungen lösbar ist.

Als Forschungsfrage gilt: "Mit welchen Verfahren und Frameworks können neuronale Netze zur Personenerkennung und -zählung auf Embedded-Geräten implementiert werden?". Um diese Frage zu beantworten, soll auf folgende Teilfragen eingegangen werden:

1. Welche Vorteile bietet eine Personenerkennung mit neuronalen Netzen auf Farbbildern gegenüber anderen Verfahren, welche keine neuronalen Netze oder andere Sensorarten verwenden?

2. Welche Besonderheiten bestehen im Kontext von eingebetteten Systemen und welche Auswirkungen besitzen diese auf Verfahren der Computer Vision, besonders die Personenzählung mit neuronalen Netzen?

3. Ist es mit vertretbarem Aufwand möglich und sinnvoll, vortrainierte Objektdetektoren, die mehrere Klassen unterscheiden, auf die Erkennung von Personen anzupassen?

4. Mit welchen Techniken können neuronale Netze auf die besonderen Rahmenbedingungen von eingebetteten Geräten angepasst werden?

1.3 AUFBAU

In dieser Arbeit wird die Implementierung neuronaler Netze auf eingebetteten Systemen behandelt. Nachdem in dieser Einleitung Motivation und Hintergrund erläutert wurden, wird im folgenden Abschnitt die Abgrenzung zu existierender Forschung vorgenommen, um diese Arbeit zu legitimieren. Das Erfassen und Zählen von Personen ist ein typisches Problem, wie es in an die Gebäudesensorik angrenzenden Fachgebieten, beispielsweise bei der Überwachungs- und Sicherheitstechnik, üblich ist. Dementsprechend existieren bereits viele vorhandene Untersuchungen, welche in diesem Kapitel vorgestellt und von dieser Arbeit abgegrenzt werden sollen. Hierbei soll betont werden, inwiefern sich diese Arbeit durch den Fokus auf sehr kleine, schwache eingebettete Systeme und die genauere Betrachtung der Qualität der eigentlichen Personenzählung von anderen Arbeiten unterscheidet. Mit diesem Kapitel kann die erste Teilfrage, welche sich mit alternativen Verfahren der Personenzählung befasst, beantwortet werden.

Bevor in den Hauptteil der Arbeit übergegangen werden kann, werden im Grundlagenteil die wichtigsten Konzepte zum Verständnis der Arbeit vermittelt. Hierbei wird eine Einführung in das Fachgebiet der Objekterkennung, auf der eine Personenzählung basiert, gegeben. Zusätzlich werden die wichtigsten Konzepte von Convolutional Neural Networks beschrieben, wobei ein Fokus auf Konzepten liegt, welche im Rahmen der Objekterkennung und den vorgenommenen Untersuchungen im Hauptteil benötigt werden.

Im Rahmen der Methodik wird das weitere Vorgehen im Hauptteil begründet. Der Hauptteil teilt sich in zwei Teile auf. Um die Problemstellung einer Personenzählung mit neuronalen Netzen auf eingebetteten Geräten zu untersuchen, wird dieses Problem in aufeinander aufbauenden Teilen untersucht. Als erster Teil wird die Problemstellung in einem Lösungskonzept schrittweise ausgearbeitet. Es werden zuerst die Anforderungen von auf neuronalen Netzen basierenden Objektdetektoren an die zur Ausführung genutzte Hardware illustriert. Hierzu werden erst theoretische Überlegungen dargelegt, welche Minimalanforderungen an Hardware gelten. Auf diese Erkenntnisse aufbauend wird anschließend eine Auswahl mehrerer Geräte getroffen, die diese Anforderungen erfüllen. Mit diesen Überlegungen kann die zweite Teilfrage beantwortet werden. Unter Beachtung der getroffenen Hardwareauswahl wird mit einer Literaturrecherche nach geeigneten neuronalen Netzarchitekturen gesucht, welche auf den Geräten lauffähig sind. Um ein späteres Training dieser Netzarchitekturen auf dem Anwendungsfall der Personenzählung zu ermöglichen, werden verschiedene Datensätze vorgestellt, welche Personen enthalten und deswegen für das Training geeignet sind. Die gewählten Netze werden in mehreren Varianten auf den gegebenen Datensätzen trainiert und schließlich evaluiert. Hierbei werden verschiedene Untersuchungen vorgenommen; es wird die Übertragbarkeit des Wissens der Objektdetektoren zwischen den Datensätzen geprüft, der Einfluss der Bildauflösung wird betrachtet und die Genauigkeit der eigentlichen Personenzählung wird gemessen. Mit den Ergebnissen lässt sich darstellen, in-

wiefern das Neutrainieren von Objektdetektoren gegenüber vortrainierten Netzen Vorteile bietet und welche Zusammenhänge hierbei zu den genutzten Trainingsdaten und Bildauflösungen bestehen, was die dritte Teilfrage beantwortet. Um die letzte Teilfrage zu beantworten, werden die Techniken der Quantisierung und des Teacher-Student-Trainings untersucht und auf einige erhaltene Netze angewandt. Es wird gemessen, inwiefern Quantisierung eine Verkleinerung des Speicherbedarfs der Modellparameter eines neuronalen Netzes im Kontext der Objektdetektion ermöglicht und wie stark eventuelle Leistungsverluste ausfallen. Für Teacher-Student-Training wird untersucht, ob die Technik in diesem Falle in der Lage ist, kleinere Objektdetektoren mit der Performance eines größeren Detektors zu trainieren. Mit diesem Abschnitt wird die letzte verbleibende Teilfrage beantwortet.

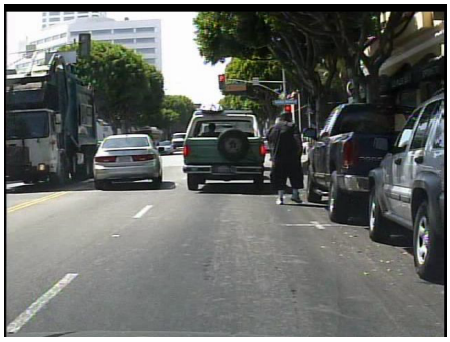
Die Beantwortung der Teilfragen ermöglicht eine Beantwortung der ersten Hälfte der Forschungsfrage, da nun Techniken zur Implementierung neuronaler Netze auf eingebetteter Hardware bekannt sind. Es wurden jedoch noch keine Messungen auf echter Hardware durchgeführt, ebenso verbleibt der Aspekt der Forschungsfrage, welcher sich auf die Frameworks einer Implementierung bezieht. Um diesen fehlenden Teil zu beantworten, wird eine Auswahl der performantesten erhaltenen Netze auf den ausgewählten Hardwaregeräten implementiert, wobei ein Fokus auf den genutzten Frameworks und Laufzeiten liegt. Mit Messungen zur Laufzeit und anderen, gerätespezifischen Metriken, wird gezeigt, wie unterschiedlich gut die Geräte für den Anwendungsfall geeignet sind.

Im Anschluss werden die Ergebnisse im Rahmen der Diskussion kritisch betrachtet und in den Kontext anderer Arbeiten gesetzt, zusätzlich wird ein Ausblick auf zukünftige Forschung gegeben, welche an diese Arbeit anknüpfen könnte. Den Abschluss der Arbeit bildet die Zusammenfassung mit einem Gesamtfazit.

2 ABGRENZUNG ZU BISHERIGER FORSCHUNG

Computer Vision ist ein Fachgebiet, welches eine große Menge an aktiver Forschung zum maschinellen Lernen und zur künstlichen Intelligenz vorweisen kann. Dementsprechend ist das Problem der Personenzählung kein neues. Es existieren sehr viele Arbeiten, welche sich auf verschiedene Weise mit der Zählung von Personen außerhalb und innerhalb von Räumen beschäftigen. Auch Arbeiten, in denen leistungsschwache Hardware zur Zählung von Personen genutzt wird, existieren. Ein Auszug solcher Arbeiten soll im Folgenden vorgestellt und untersucht werden, um diese Arbeit von bisheriger Forschung abgrenzen zu können.

Die Erkennung von Personen auf Kamerabildern ist eine weit verbreitete Problemstellung, welche in verschiedensten Anwendungsfällen umgesetzt werden muss. So ist beispielsweise die Erkennung von Fußgängern ein solcher Anwendungsfall. Die korrekte Erkennung von Fußgängern und das Verfolgen ihrer Bewegungen ist eine Disziplin, welche unter anderem für autonome Fahrzeuge von Bedeutung sein kann, um Kollisionen im Straßenverkehr zu verhindern. Bilder von Fußgängern sind oftmals aus einer Bodenperspektive aufgenommen, sodass die abgebildeten Personen frontal und mitunter mit Verdeckungen durch andere Personen oder Objekte abgebildet sind (vgl. [72, S. 15]). Zur Fußgängererkennung existieren bereits viele Arbeiten, auch solche, welche vor der Popularisierung von Deep-Learning-Methoden entstanden [72, S. 15]. Ein Beispiel für ein Verfahren der klassischen Computer Vision bildet die Histogrammbildung von Farbgradienten in Schwarz-Weiß-Bildern und anschließende Klassifizierung mit statistischen Verfahren wie Entscheidungsbäumen [14]. Verfahren, welche auf Methoden des maschinellen Lernens basieren, nutzen für diese Problemstellung Verfahren wie Deep Neural Networks, um automatisiert aus den Farbbildern mögliche Positionen für Fußgänger zu extrahieren [72, S. 15]. Das Problem der Fußgängererkennung ähnelt insofern der Personenerkennung, wie sie in dieser Arbeit angestrebt wird, dass Personen auf Farbbildern lokalisiert werden müssen. Unterschiede zur Erkennung von Fußgängern bestehen vor allem in der Perspektive der Bildaufnahmen, in dieser Arbeit soll eine Zählung aus der "Vogelperspektive" einer deckenmontierten Kamera betrachtet werden, während Fußgänger typischerweise von der Seite betrachtet werden. Dies kann in Abbildung 2.1a erkannt werden, wobei ein Beispielbild aus dem Caltech-Pedestrian-Datensatz, einem bekannten Datensatz zur Evaluation von Verfahren zur Fußgängererkennung, abgebildet ist. Ebenso kann angenommen werden, dass die Zählung der erkannten Fußgänger üblicherweise nicht Ziel typischer Anwendungsfälle der Fußgängerzählung ist, sondern die eigentliche Erkennung im Vordergrund steht.



(a) Beispielbild aus dem Caltech-Pedestrian-Datensatz [13, 15]

(b) Beispielbild aus dem ShanghaiTech-Datensatz [71]

Abbildung 2.1: Beispielbilder aus Datensätzen verwandter, jedoch im Detail zum Kontext dieser Arbeit abgegrenzter Themengebiete.

Ein weiterer ähnlicher Anwendungsfall besteht in der Zählung von Menschenmengen (engl. Crowd Counting). Ziel des Crowd Counting ist, einzelne Menschen in den großen Menschenmengen zu lokalisieren und die Dichte der Menge abzuschätzen [40, S. 1]. Ähnlich wie bei der Erkennung von Fußgängern werden Personen möglicherweise verdeckt, sodass sie nur aufgrund der nicht verdeckten Körperteilen erkannt werden müssen [40, S. 1]. Nach einer Untersuchung von Li et al. [40] erzielen auf neuronalen Netzen basierende Methoden beim Crowd Counting gute Ergebnisse. Li et al. untersuchen ebenfalls, welche Datensätze für Crowd Counting zur Verfügung stehen. Hierbei fallen zwei Umstände auf, in denen sich typische Bilder beim Crowd Counting von der angestrebten Personenzählung unterscheiden. Einerseits enthalten viele Datensätze pro Bild eine sehr große Anzahl an Personen. Beispielsweise enthält der ShanghaiTech-Datensatz aus [71] (Beispielbild in Abbildung 2.1b) in seinen zwei Teilen durchschnittlich 501 bzw. 123 Personen pro Bild; dies sind eine oder zwei Größenordnungen mehr als in einem Büroraum zu erwarten wären. Andererseits fokussieren sich die Datensätze oftmals auf die Anzahl der erkennbaren Köpfe auf einem Bild. Da in großen Menschenmengen praktisch alle Personen zumindest in Teilen verdeckt sind, stellt der Kopf den einzigen Körperteil dar, welcher sinnvoll erkannt werden kann. Für eine Zählung in Gebäuden ist anzunehmen, dass nur wenige Personen im Sichtfeld einer Deckenkamera abgebildet sind, diese jedoch möglicherweise nur in Teilen und ohne Sichtbarkeit des Kopfes erkennbar sind. Abgesehen von diesen Unterschieden besteht beim Crowd Counting jedoch ein ähnlicher Fokus auf der Zählung der Personen, die genaue Lokalisation ist zweitrangig.

Abseits von Crowd Counting und Fußgängererkennung existieren weitere Arbeiten, welche sich mit der Erkennung von Personen beschäftigen. Del Pizzo et al. [11] befassen sich in ihrer Arbeit mit der Zählung von Personen, welche eine virtuelle Linie auf dem Kamerabilder überschreiten. Die Autoren fokussieren sich hierbei auf das Zählen der Überschreitungen von Personen über die Linie, weshalb die Erfassung auf Videoaufnahmen mit 30 Bildern pro Sekunde durchgeführt wird. Die Zählperformance wird dadurch gemes-

sen, dass jedes Überqueren der Linie als Ereignis gezählt werden soll. Die Autoren vergleichen somit die Anzahl der gezählten Überquerungen der Linie mit der echten Anzahl. Für die Aufnahmen wurden sowohl Farbbilder als auch Tiefenbilder erfasst. Del Pizzo et al. vergleichen die Auswirkungen dessen, ob nur Farb-, nur Tiefen- oder eine Kombination aus Farb- und Tiefenbildern genutzt werden, auf die Qualität ihrer Zählmethode. Gleichzeitig führen sie einen Vergleich zwischen drei verschiedenen Hardwaregeräten durch, von denen eines ein üblicher PC und die verbleibenden zwei kleinere Geräte sind, welche (basierend auf ihren Eckdaten, genauere Angaben fehlen) als ähnlich zu leistungsstarken Embedded-Geräten angesehen werden können. Als Ergebnis erhalten die Autoren, dass ihre Zählmethode auf den von ihnen erstellten Aufnahmen gute Ergebnisse erzielt. Einzelne Personen, die die Linie überqueren, werden auf beiden Bildarten zuverlässig erkannt, während für Gruppen an Menschen eine Kombination von Farb- und Tiefenbild genutzt werden muss, um bessere Ergebnisse zu liefern. Durch Reduktion der Bildgröße erhalten die Autoren auch auf den schwächeren Geräten eine geringere Laufzeit mit einem Genauigkeitsverlust zwischen zwei und fünf Prozent [11, S. 11], sodass ihre Zählung auch auf diesen Geräten mit ausreichend vielen verarbeiteten Bildern pro Sekunde lauffähig ist.

Die Arbeit von Del Pizzo et al. vereint mehrere Aspekte, welche ähnlich zur angestrebten Arbeit sind. Es werden Personen auf den Bildern einer deckenmontierten Kamera erfasst, dabei handelt es sich um einzelne Personen oder kleinere Gruppen, welche auf den Aufnahmen abgebildet sind. Die Bewertung eines Zählaspekts fließt durch die Erfassung der Überquerungen der Linie ebenfalls in ihre Untersuchungen mit ein. Auch untersuchen die Autoren die Anwendbarkeit ihres Verfahrens auf leistungsschwachen Geräten, hierbei wird sogar explizit auf eingebettete Systeme, wie sie in einer intelligenten Kamera verbaut sein könnten, eingegangen [11, S. 12]. Unterschiede bestehen darin, dass sich die Autoren nicht tiefergehend mit den Besonderheiten der schwachen Systeme befassen. Ebenso werden Tiefenbilder genutzt, welche im Rahmen dieser Arbeit nicht zur Verfügung stehen. Außerdem ist das Ziel ihrer Arbeit das Zählen der Überquerungen der virtuellen Linie im Bild, weshalb eine Verarbeitung nahe Echtzeit angestrebt wird. Der Anwendungsfall der Zählung unterscheidet sich dahingehend, dass Überquerungen der Linie mit Bewegung verbunden sind und nicht, wie in dieser Arbeit, auf Momentaufnahmen eines Raumes basieren. Somit liegt der Fokus ihrer Zählung darauf, einzelne Personen in der Bewegung zu erkennen, während in dieser Arbeit die Zählung mehrerer Personen auf einem statischen Bild geschehen soll. Um das Ziel der hier angestrebten Arbeit zu erreichen, ist hingegen keine Verarbeitung von Bildern in Echtzeit notwendig. Es muss lediglich eine Messfrequenz erreicht werden, welche höher ist als die einer klassischen Gebäudesensorik. Eine Verarbeitung von Videos in Echtzeit, wie es von Del Pizzo et al. geschieht, ist nicht notwendig.

Weitere Arbeiten, welche sich mit dem Erkennen von Personen auf Kamerabildern einer deckenmontierten Kamera befassen, wurden von Ahmad et al. in ihrer Arbeit zusammengetragen [1]. Viele der aufgeführten Arbeiten nut-

zen eine Kombination aus Farb- und Tiefenbildern, welche mit algorithmischen Methoden der Computer Vision verarbeitet werden. Ein Beispiel hierfür ist die Arbeit von Filip Malawski [46], in welcher der Autor eine Kinect-Tiefenkamera zur Zählung von Personen, basierend auf ihren Köpfen, umsetzt. Hierzu werden, von den zur Kamera am nächsten gelegenen Punkten ausgehend, zusammenhängende Flächen, welche Köpfe abbilden könnten, identifiziert und gezählt. Weitere von Ahmad et al. identifizierte Arbeiten verwenden Methodiken, welche eine algorithmische Zählung auf Farb- oder Tiefenbildern vornehmen (vgl. [1, S. 5-7]). Mangels eines qualitativen Vergleiches der durch Ahmad et al. identifizierten Arbeiten kann keine Aussage zur Güte der vorgestellten Verfahren getroffen werden. Basierend auf den zu jeder Arbeit aufgeführten verwendeten Methoden ist jedoch anzunehmen, dass viele der vorgestellten Arbeiten mit Methoden der algorithmischen Computer Vision arbeiten. In dieser Arbeit sollen mit künstlichen neuronalen Netzen Verfahren angewendet werden, welche nicht algorithmisch beschrieben werden können. Auch sollen keine Tiefenkameras verwendet werden, da diese üblicherweise mehrere hundert Euro kosten, während Sensoren zur Erfassung von Farbbildern für einen Bruchteil dieser Kosten erhältlich sind. Für viele existierenden Arbeiten gilt somit, dass Tiefenbilder genutzt werden, was in dieser Arbeit nicht geschehen soll. Außerdem sollen neuronale Netze statt algorithmischer Methoden eingesetzt und die Besonderheiten der Ausführung auf eingebetteten Systemen untersucht werden.

Als letzte Gruppe existierender Forschung sollen Arbeiten betrachtet werden, welche eine Personenzählung mit neuronalen Netzen auf eingebetteten Systemen untersuchen. Diese entsprechen in ihrer Form den Zielen, welche mit dieser Arbeit verfolgt werden, weisen bei genauerer Betrachtung jedoch kleine Unterschiede auf, welche einen Neuheitswert für diese Arbeit ermöglichen. Eine Arbeit von Shen und Chu [58] befasst sich mit der Zählung von Wartenden vor einem Aufzug. Die Autoren nutzen zur Erkennung der Personen ein neuronales Netz, welches auf einem eingebetteten Gerät ausgeführt wird. Die Anzahl der gezählten Personen wird anschließend genutzt, um Wartende über die Auslastung der Aufzüge zu informieren. Ähnlich ist die Arbeit von Shen und Chu somit in der Nutzung neuronaler Netze zur Personendetektion und -zählung und in der Nutzung eines eingebetteten Geräts. Die Autoren nutzen als eingebettetes Gerät ein NVIDIA Jetson-Nano-Board, welches zwar ein eingebettetes System darstellt, aber als Hardwarebeschleuniger speziell auf die Anforderungen von neuronalen Netzen zugeschnitten ist und über deutlich mehr Leistung als weniger spezialisierte Geräte verfügt. Ebenfalls befindet sich das Jetson Nano in einer Preisklasse, welche für eine kostengünstige Integration in eine Kamera unerschwinglich wäre. Ein weiterer Unterschied zu dieser Arbeit besteht in den Anforderungen an die Verarbeitungszeit. Wartezeiten vor Aufzügen sind für die Beteiligten unangenehm und sollen aus diesem Grund minimal gehalten werden. Es ist somit notwendig, dass eine kamerabasierte Erkennung der Wartenden eine geringe Latenz aufweist und optimalerweise nahe Echtzeit oder zumindest innerhalb weniger Sekunden zu Erkennungen führt. Eine Zählung von

Personen im Rahmen der Gebäudesensorik ist deutlich geringeren Anforderungen an die Geschwindigkeit unterworfen, selbst Latenzen von mehreren Minuten sollten einen noch akzeptablen Rahmen darstellen.

Eine weitere Arbeit, welche eingebettete Systeme zur Personenzählung nutzt, besteht in der Abschlussarbeit von Simone Luetto [45]. Simone Luetto befasst sich in seiner Arbeit ebenfalls mit der Zählung von Personen in einem Vorlesungssaal, wobei er ebenfalls eine Auswahl von neuronalen Netzen zur Zählung evaluiert und eingebettete Hardware einsetzt. Simone Luetto nutzt bei seinen Untersuchungen, ähnlich wie die zuvor betrachtete Arbeit, spezielle Hardwarebeschleuniger für neuronale Netze. Auch er strebt für eine korrekte Erfassung der Personen eine Verarbeitungsgeschwindigkeit von mehr als zehn Bildern pro Sekunde [45, S. 45] an. Erneut kann als Unterschied zu der hier angestrebten Arbeit vermerkt werden, dass sowohl sehr leistungsstarke eingebettete Systeme genutzt als auch eine Verarbeitung nahe Echtzeit angestrebt wurde. Auch thematisiert Simone Luetto in seiner Arbeit eine Personenzählung, die Evaluation der Zählqualität beschränkt sich jedoch auf ein Genauigkeitsmaß, welches keine Aussage darüber vermittelt, um wie viele Personen die Zählung vom Zielwert abweicht.

Zusammenfassend bestehen somit mehrere Unterschiede zu vorherigen Arbeiten. Arbeiten, welche sich mit dem Zählen von Fußgängern befassen, unterscheiden sich wegen der seitlichen Kameraperspektive und dem fehlenden Fokus auf einer Zählung der Personen. Das Fachgebiet des Crowd Counting befasst sich hingegen vorrangig mit dem Zählen von Personen auf Bildern, welche aus einer Vogelperspektive (oder zumindest einem ähnlich erhöhten Standpunkt) erfasst wurden. Dabei werden jedoch deutlich größere Menschenmengen betrachtet, als sie jemals in einem Innenraum zu erwarten wären, zusätzlich ist der Fokus auf die Erkennung von Köpfen problematisch. Es sollen in dieser Arbeit nur Farbkameras eingesetzt werden, wodurch sich zu Arbeiten, welche auf Tiefenkameras basieren, abgegrenzt werden kann. Während eine Kombination von algorithmischen Verfahren mit Methoden des maschinellen Lernens möglich ist, soll sich diese Arbeit primär auf neuronale Netze zur Personenzählung fokussieren. Vorhandene Arbeiten, welche sich mit der Zählung von Personen befassen, tun dies in anderen Kontexten, in denen einzelne Personen unter bestimmten Kriterien erkannt werden sollen und die Zählung als Summe der einzelnen erkannten Ereignisse angesehen werden. Diese Arbeit soll Zählungen für einzelne Situationen bzw. Bilder durchführen und für jedes Bild die Anzahl der erkannten Personen bestimmen. Auch sollen andere Maße zur Bewertung der Zählqualität genutzt werden, um aufzuzeigen, wie stark die Vorhersagen von der wahren Anzahl an Personen abweichen. Zuletzt bestehen Unterschiede zu existierenden Arbeiten, welche eine Personenzählung in Innenräumen mit eingebetteten Systemen umsetzen. Bei den im Rahmen der Literaturrecherche gefundenen Arbeiten wurden zwar eingebettete Systeme eingesetzt, bei diesen handelte es sich jedoch um spezialisierte Hardware für neuronale Netze, welche kostspielig einzusetzen sind. Für schwächere, kostengünstigere Hardware bestehen besondere Herausforderungen durch

die beschränkten verfügbaren Ressourcen, welche in diesen Arbeiten nicht beachtet werden mussten. Ebenso wurde in ihnen eine Verarbeitung in oder nahe Echtzeit angestrebt, da es sich um zeitkritische Anwendungsfälle handelte. Diese Arbeit muss, um einen Mehrwert der Zählung mit Kameras und neuronalen Netzen zu beweisen, lediglich existierende Gebäudesensorik mit Latenzen von mehreren Minuten untertreffen, sodass die Anforderungen and die Laufzeit deutlich gelockert werden können.

Insgesamt soll sich diese Arbeit damit sowohl durch den Fokus auf die Nutzung von Farbbildern, den Einschränkungen durch leistungsschwache Embedded-Systeme, durch den Fokus auf der Zählqualität und durch die geringeren Anforderungen an die Geschwindigkeit von anderen Arbeiten unterscheiden.

3 GRUNDLAGEN

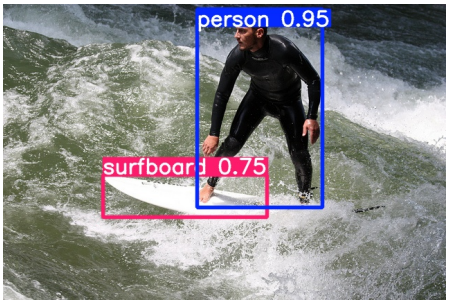
Die Umsetzung einer Personenerkennung und -zählung mit Techniken des maschinellen Lernens setzt einige Grundlagen des Gebietes der Computer Vision voraus. Wie bereits zuvor erwähnt, baut eine Personenzählung auf eine Objektdetektion auf; die Anzahl der Personen entsteht durch die Zählung der unterschiedlichen detektierten Personen. In den folgenden Abschnitten werden das Themengebiet der Objektdetektion, Convolutional Neural Networks und mit Quantisierung und Teacher-Student-Training zwei in dieser Arbeit genutzte Verfahren zur Verkleinerung neuronaler Netze eingeführt.

3.1 OBJEKTDETEKTION

Es existieren viele verschiedene Aufgaben, welche mit Verfahren des maschinellen Lernens gelöst werden können. Einfache Aufgaben wie Klassifikation und Regression sind auf vielen Fachgebieten anwendbar und nicht spezifisch für das Gebiet der Computer Vision. Die Objektdetektion ist eine fortgeschrittene Aufgabe, welche spezifisch für dieses Fachgebiet ist und mit Methoden der Computer Vision gelöst werden kann. Gegenstand der Objektdetektion ist es, alle Instanzen von Objekten aus einer oder mehreren Klassen in einem Bild zu verorten und in eine dieser Klassen einzuordnen [2, S. 875]. Diese Kombination aus Verortung und Klassifikation von Objekten resultiert in besonderen Herausforderungen, welche im Folgenden eingeführt werden sollen.

Objekte können innerhalb eines Bildes in sehr vielen möglichen Positionen und Ausrichtungen abgebildet sein. Um ein Objekt zu erkennen, muss dieses einerseits aufgefunden und von irrelevanten anderen Objekten oder

(a) Objektdetektion durch Segmentierung mit Detectron2, Bild aus dem Detectron2-Repository. [68]



(b) Objektdetektion durch Bounding Boxen mit YOLOv5, Bild aus dem COCO-Datensatz [41].

Abbildung 3.1: Vergleich von Segmentierung und Bounding Boxen als Ausgaben einer Objektdetektion.

dem Hintergrund unterschieden werden, andererseits muss das Objekt in eine der Zielklassen eingeordnet werden [2, S. 875]. Somit besteht die Ausgabe eines Verfahrens der Objektdetektion einerseits aus der Lage von Objekten als auch aus der Klassifikation dieser. Die Lage kann auf verschiedenste Arten beschrieben werden, möglich sind hierbei beispielsweise die Angabe der Koordinaten im Bild, der Koordinaten und der Größe oder einer Maske, welche die Pixel umfasst, welche dem Objekt zuzuordnen sind [2, S. 875]. Bei vielen gängigen Objektdetektoren erfolgt die Ausgabe als Bounding Box, einem zu den Bildachsen achsenparallelen Rechteck, welches das erkannte Objekt umrahmt. Abbildung 3.1 zeigt jeweils das visualisierte Ergebnis einer Objektdetektion mit Segmentierung und mit Bounding Boxen. Es ist erkennbar, dass eine Segmentierung der einzelnen Objektinstanzen auf Pixelebene deutlich aufwändiger ist, da zu jedem Objekt alle zugehörigen Pixel zugeordnet werden müssen; für eine Erkennung mit Boxen müssen nur wenige Informationen wie z. B. die Eckpunkte erzeugt werden. Da die Ergebnisse einer Segmentierung deutlich komplexer sind als Bounding Boxen und diese Komplexität für eine Personenzählung keinen Mehrwert bietet, soll sich im Folgenden auf Detektionsverfahren beschränkt werden, welche Boxen als Ausgabe erzeugen.

Aufgabe eines Objektdetektors ist in diesem Kontext also, mehrere Ausgaben zu erzeugen: eine Menge an Boxen, welche Objekte innerhalb des Bildes umschliessen und zu jeder Box eine Klassenzuordnung des umschlossenen Objektes [61, S. 302]. Dabei soll sowohl die Klassenzuordnung korrekt sein, als auch die erzeugte Box jedes Objekt möglichst genau umschließen. Die Klassenzuordnung erfolgt durch die Angabe einer Konfidenz, welche beschreibt, wie wahrscheinlich das Objekt zu einer gegebenen Klasse zugeordnet wird [61, S. 303]. Um verschiedene Verfahren zur Objektdetektion quantitativ bewerten zu können, sind Metriken notwendig, um beide Teile, die Platzierung der Boxen und die Klassifizierung, in einer Zahl zusammenzufassen. Die folgenden Abschnitte sind, sofern nicht anders vermerkt, inhaltlich aus der Arbeit von Padilla et al. [51] übernommen.

Für den Schritt der Lokalisation gilt es, die vorhergesagten Boxen des Detektors mit den wahren Boxen (Ground Truth) zu vergleichen. Um zwei Bounding-Boxen miteinander zu vergleichen, bietet es sich an, deren Flächen in Relation zueinander zu setzen. Ein einfaches Maß bildet hierbei das Intersection-over-Union-Maß (deutsch: Schnitt(-menge) über Vereinigung(-smenge)), welches in Formel 3.1 für die Mengen bzw. Flächen A und B dargestellt ist.

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

In Abbildung 3.2 ist der Inhalt der Formel grafisch für zwei Boxen dargestellt. Es ist ersichtlich, dass das Maß nur Werte zwischen Null (für disjunkte Flächen) und Eins (für zwei identische Flächen) annehmen kann. Durch einen Vergleich des Wertes von $\mathrm{IoU}(A, B)$ mit einem vorher gewählten Schwellwert t kann eine Entscheidung getroffen werden, ob A und B

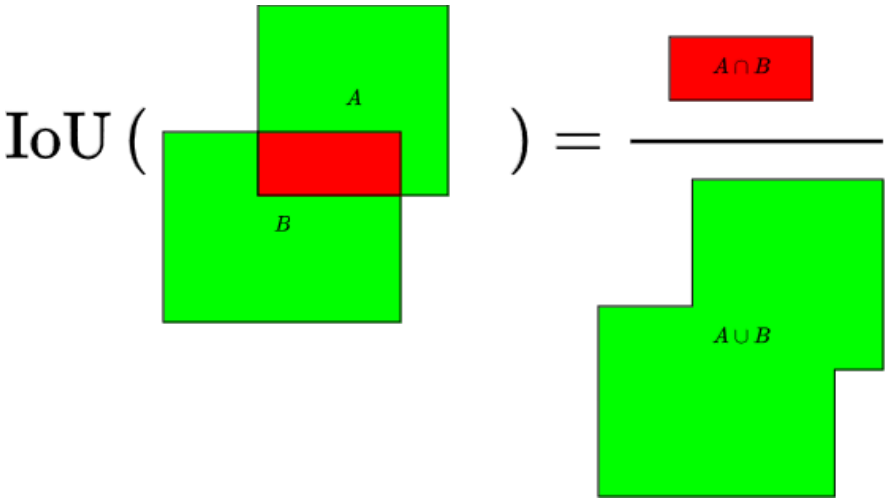


Abbildung 3.2: Grafische Darstellung der IoU-Berechnung, adaptiert aus [61, S. 302].

ausreichend ähnlich sind. Konkret bedeutet dies, dass eine vorhergesagte Box A als mit der einer Ground-Truth-Box B übereinstimmend angesehen wird, falls $\text{IoU}(A, B) \geq t$ gilt. Verschiedene Werte von t resultieren in einem mehr oder weniger strengen Bewertungsverfahren: Werte gegen Null bewerten Boxen mit nur geringer Schnittfläche als gleich, während Werte nahe Eins nur Boxen mit großer Ähnlichkeit einander zuordnen.

Während das IoU-Maß die Lokalisation bewertet, müssen die Klassifikationen separat betrachtet werden. Analog zu Klassifikationsaufgaben in anderen Kontexten, wird eine Klassifikation als korrekt angesehen, wenn die vorhergesagte Klasse der Zielklasse entspricht und die Konfidenz dieser Klassenzuordnung gleich groß oder größer eines vorher gewählten Konfidenz-Schwellwerts ist. Dieser Schwellwert ist von dem Schwellwert für das IoU-Maß abzugrenzen, es handelt sich um zwei verschiedene Werte.

Unter Kombination der Bewertung der Lokalisation durch das IoU-Maß und die Bewertung der Klassifikation lässt sich dadurch durch Vergleichen jeder Detektion mit jeder Ground-Truth-Box eine Konfusionsmatrix erstellen. Es können hierbei folgende Fälle auftreten:

- Richtig positiv / True Positive (TP): eine Vorhersage erfasst ein Objekt ausreichend gut und klassifiziert dieses korrekt.
- Falsch positiv / False Positive (FP): eine Vorhersage erfasst ein nicht existierendes Objekt, klassifiziert ein existierendes Objekt falsch oder ist ein Duplikat einer anderen richtig positiven Vorhersage, welche einen höheren IoU-Wert besitzt [51, S. 3].
- Falsch negativ / False Negative (FN): eine Ground-Truth-Box wird von keiner einzigen Box abgebildet.

Es ist zu beachten, dass das Konzept einer richtig negativen Beobachtung bei der Objektdetektion nicht verwendet wird. Dies geschieht unter dem

Gesichtspunkt, dass jeglicher Hintergrund eines Bildes (also alle Bereiche, die keine Objekte abbilden) als Negativbeispiel eingeordnet werden kann, es aber unendlich viele Boxen gibt, die den Hintergrund abbilden können.

Aus diesen Konzepten lassen sich mit Genauigkeit und Sensitivität weitere Maße ableiten. Zwecks Eindeutigkeit werden für diese beiden Maße folgend die englischen Begriffe Precision und Recall verwendet, da der Begriff der Genauigkeit im Deutschen oftmals nicht eindeutig interpretiert werden kann. Precision beschreibt die Fähigkeit eines Modells, seine positiven Vorhersagen auf nur relevante Objekte zu beschränken, während Recall bewertet, wie gut alle tatsächlich vorhandenen Objekte aufgefunden werden. Precision P und Recall R sind somit durch

$$\begin{aligned} P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \end{aligned} \tag{3.2}$$

beschrieben. Beide Maße gelten pro Klasse, das Klassifikationsproblem eines Objektdetektors wird somit als i Teilprobleme betrachtet. Ziel eines Objektdetektors ist es, sowohl Precision als auch Recall zu optimieren. Beide Maße hängen von dem gewählten Schwellwert für die Konfidenz ab, ein geringer Schwellwert führt tendenziell zu vielen ungenauen Detektionen (hoher Recall, geringe Precision), ein hoher Schwellwert eher zu wenigen, genauen Erkennungen (geringer Recall, hohe Precision). Um diese Abhängigkeit zu bewerten, wird die Fläche unter der Precision-Recall-Kurve gemessen. Durch ein Variieren des Schwellwerts können Paare aus Precision-Recall-Werten erhalten werden, welche genutzt werden können, um eine Kurve von Precision in Abhängigkeit von Recall zu interpolieren. Das Aussehen einer solchen Kurve ist in Abbildung 3.3 gegeben.

Die Fläche unter der Kurve beschreibt die durchschnittliche Precision für die betrachtete Klasse, weshalb dieser Wert als Average Precision (AP) bezeichnet wird. Die Berechnung der AP kann nun für alle Klassen des Datensatzes wiederholt werden, um die Mean Average Precision nach

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{3.3}$$

mit

N: Anzahl der Klassen im Datensatz

AP_i: Average Precision für die i-te Klasse des Datensatzes

zu berechnen. Da diese über alle Klassen mittelt, ist die Mean Average Precision im Gegensatz zur Average Precision nicht auf eine bestimmte Klasse bezogen.

Es ist zu beachten, dass das mAP-Maß für einen definierten Schwellwert von t gilt (im Gegensatz zum Schwellwert der Konfidenz, welcher zur Berechnung der Precision-Recall-Kurve variiert wird). Ein übliches Maß zur Bewertung von Objektdetektoren bildet das mAP_{50}-Maß, was den Wert von mAP bei dem arbiträren Schwellwert von $t = 0{,}5$ beschreibt. Dieses Maß

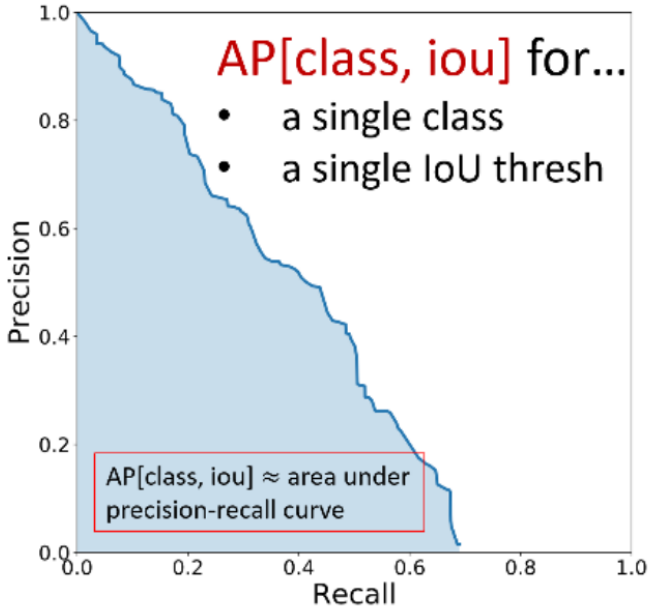


Abbildung 3.3: Beispiel einer Precision-Recall-Kurve, entnommen aus [61, S. 303]. Die in dieser Grafik genutzte Notation verdeutlicht die Abhängigkeit des AP-Wertes sowohl von der betrachteten Klasse als auch von dem gewählten IoU-Schwellwert t.

soll auch in einem Großteil dieser Arbeit genutzt werden. Um einen Detektor auf mehreren Schwellwerten zu evaluieren und diese Ergebnisse zusammenzufassen, existiert das $\text{mAP}_{50:\,5:\,95}$-Maß, bei dem alle mAP_ts-Werte für t in $[0{,}5;\ 0{,}55;\ 0{,}6;\ \ldots;\ 0{,}85;\ 0{,}9]$ gemittelt werden. Es gilt:

$$\text{mAP}_{50:\,5:\,95} = \frac{1}{10}(\text{mAP}_{50} + \text{mAP}_{55} + \text{mAP}_{60} + \ldots + \text{mAP}_{95}) \tag{3.4}$$

Dieses Maß betrachtet durch das Einbeziehen höherer Werte für t stärker, dass die Lage der vorhergesagten Bounding Boxes möglichst ähnlich zu der Ground Truth sein soll.

In dieser Arbeit wird nur eine Klasse, die der Personen, behandelt. Es gilt somit, dass $N = 1$, womit nach Formel 3.3 $\text{mAP} = \text{AP}$ und nach Formel 3.4 $\text{mAP}_{50:\,5:\,95} = \text{AP}_{50:\,5:\,95}$ gilt. Um trotzdem zu betonen, dass die Objektdetektion auf einer Klasse und mehreren Klassen unterschiedlich komplex und nicht direkt vergleichbar sind, wird in dieser Arbeit bewusst AP statt mAP angegeben, obwohl beide Maße in diesem Falle technisch gleichbedeutend sind.

3.2 CONVOLUTIONAL NEURAL NETWORKS ALS OBJEKTDETEKTOREN

Neuronale Netze bilden eine populäre Klasse an Methoden zur Lösung verschiedener Aufgaben des maschinellen Lernens und der künstlichen Intelligenz. Im Kontext der Bildverarbeitung existieren mit Convolutional Neural Networks (CNNs) besondere Arten neuronaler Netze, welche auf Bilddaten

angepasst sind. Bilder werden in diesem Kontext als gleichmäßiges, rechteckiges Raster an Pixeln dargestellt, in welchem für jede Koordinate (x, y) ein oder mehrere Werte gegeben sind, welche den Farbwert (bei einem Wert Grauwerte, bei drei Werten Rot-Grün-Blau-Farbbilder) an diesem Pixel repräsentieren.

3.2.1 *Grundlagen CNNs*

Wie von LeCun et al. ab dem Jahre 1998 popularisiert [39], sind zur Verarbeitung von Bildern Konzepte notwendig, welche diese Struktur in die neuronale Netzarchitektur einbeziehen können. "Klassische" Feedforward-Netze, in denen jedes Neuron mit jedem Neuron der vorherigen Schicht verbunden sind, können einfache Aufgaben wie die Erkennung von handschriftlichen Zeichen auf Bildern bewältigen [39, S. 5]. Sie besitzen jedoch kein Konzept der räumlichen Zusammenhänge zwischen benachbarten Pixeln und sind aus diesem Grund sehr anfällig für leichte Veränderungen der Eingabebilder (beispielsweise Rotation, Translation oder Verzerrung) [39, S. 5–6], welche die Ausgabe des Netzes stark beeinflussen können.

LeCun et al. nutzten in ihrer Arbeit [39] zur Lösung dieses Problems das Konzept der Convolutions (deutsch: Faltungen). Faltungen bilden für viele Verfahren der Signalverarbeitung eine grundlegende Operation [35, S. 222]. Eine Faltung ist ein mathematischer Operator, welcher auf zwei Funktionen g und h definiert ist und eine dritte Funktion f ergibt [35, S. 222]. Dabei stellt g die Funktion das Eingangssignals dar, welches mit dem Faltungsoperator bearbeitet werden soll [35, S. 222]. h ist die Funktion, welche den Bereich einschränkt, in welchem die Faltungsoperation wirken soll, aus diesem Grund wird sie oftmals nur in einem kleinen Bereich um den Nullpunkt als ungleich Null definiert [35, S. 222]. Für zweidimensionale, diskrete Daten, wie sie durch Bilddaten gegeben sind, gilt als Formel für die Faltung

$$f(x, y) = \sum_i \sum_j h(i, j) g(x - i, y - j) \tag{3.5}$$

an einer Koordinate (x, y) mit $x \in \mathbb{Z}$ und $y \in \mathbb{Z}$ [61, S. 96].

In Formel 3.5 ist erkennbar, dass durch eine geeignete Funktion h angegeben werden kann, welche Pixelwerte in welcher Umgebung um die Koordinaten (x, y) des Originalbildes g in die Ausgabe $f(x, y)$ miteinbezogen werden sollen. Die erhaltenen Ausgabewerte $f(x, y)$ sind maximal, wenn h und g in der Umgebung um (x, y) ähnlich zueinander sind, sodass durch die Multiplikation $h(i, j)g(x - i, y - j)$ große Werte erzeugt werden. Somit ist die Faltung ein Ähnlichkeitsmaß, welches angibt, wie stark ein durch die Funktion h dargestelltes Muster bzw. Merkmal in der Umgebung um g an der Stelle (x, y) vorhanden ist.

Unter Ausnutzung dessen, dass viele Operationen nur auf benachbarte Pixel zugreifen, welche nahe an dem Ausgabepixel an (x, y) liegen, werden die Filterfunktionen h bei CNNs nicht als Funktionen, sondern als Matrix angegeben. Für Werte, welche innerhalb der Matrix liegen, gibt der jeweili-

ge Matrixeintrag den Wert für $h(i, j)$ an, für alle außerhalb liegenden Werte wird Null angenommen. CNNs nutzen eine leicht abgewandelte Form der Faltung zur Berechnung. Formel 3.5 berechnet eine Faltung auf einem 2D-Raster an Werten, was einem Grauwertbild entspricht. Rot-Grün-Blau-Farbbilder setzen sich pro Pixel jedoch aus drei Werten zusammen, welche die Intensität des jeweiligen Farbkanals angeben. Um Zusammenhänge zwischen den einzelnen Farbwerten betrachten zu können, werden die Ergebnisse dreier unabhängiger Faltungen kombiniert und gemittelt [61, S. 232]. Das Ergebnis nach Anwendung der Faltungsoperation auf alle möglichen Wertepaare von x und y ist wieder eine Matrix, welche die Ähnlichkeit zwischen h und g in der Nachbarschaft um (x, y) angibt. Da diese Ähnlichkeit auf bestimmte Bildmerkmale (engl. Features) bezogen ist, werden diese Ausgabematrizen als "Feature Maps" bezeichnet [39, S. 6].

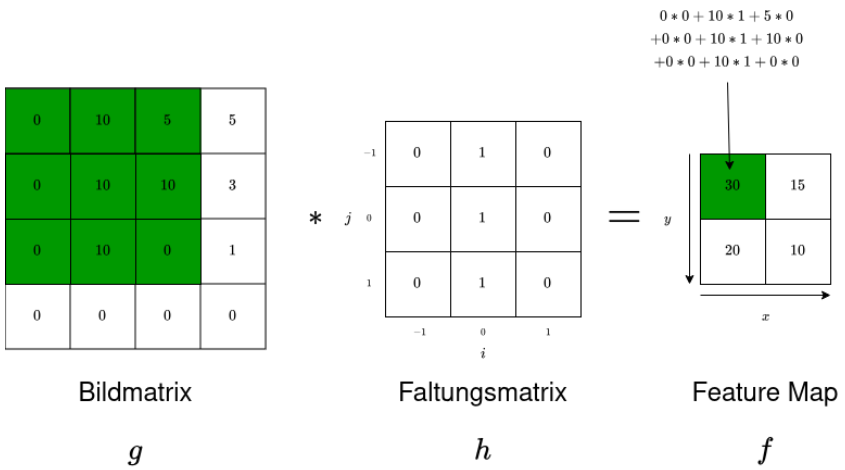


Abbildung 3.4: Rechenbeispiel für eine zweidimensionale, diskrete Faltung auf einem Bild.

Abbildung 3.4 stellt die Faltung eines Bildes mit einer zur Demonstrationszwecken gewählten Faltungsmatrix dar. Farblich hinterlegt sind die Teile des Bildes, welche zur Berechnung eines Werts innerhalb der Feature Map benötigt werden und der resultierende Eintrag in der Feature Map selbst. Es ist erkennbar, dass die Faltung selbst einer Matrixmultiplikation entspricht. Auch ist erkennbar, dass dieser Vorgang für andere Positionen wiederholt werden kann, indem die Filtermatrix entlang der Bildmatrix verschoben wird. Die resultierende Feature Map dieses Beispiels ist kleiner als das Originalbild, da die Randpixel in g keine Nachbarwerte besitzen, welche in die Berechnung einfließen könnten. $f(x, y)$ ist für $x = 0$, $y = 0$ maximal, sodass das Feature, welches durch h "gesucht" wird, in dem farblich markierten Ausschnitt des Originalbildes am stärksten vertreten ist.

Mit dem Konzept der Faltung lassen sich bereits händisch Filtermatrizen erstellen, welche aus Rasterbildern bestimmte Merkmale extrahieren. Durch die potenziell unendliche Anzahl an möglichen Filtern und Unklarheit dar-

über, welche Filter für welchen Anwendungsfall hilfreich sind, werden Filter im Kontext des maschinellen Lernens nicht manuell erstellt. Convolutional Neural Networks sehen die Filter als Parameter an, sodass diese während des Trainings des Netzes automatisch erlernt werden können [61, S. 233], indem Muster gesucht werden, welche möglichst nützlich zur Bewältigung der gegebenen Aufgabe sind. Mit dem Trainieren von Filtermatrizen für Faltungen sind neuronale Netze in der Lage, bestimmte Merkmale (engl. Features) in Bilddaten aufzufinden und zu extrahieren. Ähnlich wie in klassischen neuronalen Netzen, welche nicht auf Bilder spezialisiert sind, werden Schichten mit solchen Faltungen als Faltungsschichten (engl. Convolutional-Layer, dieser Begriff wird im Folgenden verwendet) bezeichnet. Pro Schicht wird nicht nur eine Faltung angewandt, sondern n verschiedene Filter eingesetzt, um mehrere Features in den Eingaben aufzufinden [61, S. 232–233]. Die Gesamtheit aller Filter eines solchen Convolutional-Layers erzeugt damit insgesamt n verschiedene Feature Maps. Durch Verketten von Convolutional-Layern können stückweise komplexere Merkmale extrahiert werden, indem Feature Maps vorheriger Schichten als Eingaben für spätere Schichten verwendet werden. In Abbildung 3.5 wird ein Auszug der Filter der ersten drei Schichten des InceptionV1-CNNs visualisiert. Es sind pro Schicht einige Eingabebilder für das Gesamtnetz gegeben, welche einen bestimmten Filter in der betrachteten Schicht stark aktivieren, also dem Muster entsprechen, welches von diesem Filter gesucht wird (vgl. [49, 50]). Obwohl es sich nur um einen Auszug handelt, lässt sich erkennen, dass spätere Schichten komplexere Muster wie Kurven erfassen, welche aus einfacheren Mustern wie Kanten zusammengesetzt werden. Auch ist erkennbar, dass manche Gruppen an Mustern wie Farbkontraste in mehreren Schichten mit unterschiedlicher Komplexität erfasst werden, wobei frühere Filter kleinere, einfachere und spätere Schichten komplexere, größere Strukturen erfassen.

Um die Datenmenge schrittweise zu reduzieren und das CNN somit dazu zu bewegen, nur die wichtigsten Merkmale zur Lösung der gegebenen Aufgabe zu identifizieren, werden Feature Maps schrittweise verkleinert. Möglichkeiten hierzu sind die Nutzung einer Schrittweite grösser eins pro Faltung (eine Schrittweite von n bedeutet, dass die Faltung nur für jede n-te Zeile und Spalte berechnet wird [61, S. 234]) oder unter anderem Pooling-Schichten, in denen die Werte einer Pixelnachbarschaft auf einen Wert aggregiert werden, beispielsweise durch Bilden des Maximums [61, S. 234-235]. Mit diesen Techniken kann eine Reduktion der Größe der erhaltenen Feature Maps erreicht werden, wodurch die zu verarbeitende Informationsmenge reduziert wird.

Die Verkettung von Convolutional-Layern eignet sich, wie gezeigt, zur Extraktion von schrittweise komplexeren Merkmalen aus Bildern. Diese Extraktion ist unabhängig von der Aufgabe des Gesamtnetzes und kann für verschiedene Anwendungen wiederverwendet werden, erst die Zusammenführung und Kombination der Features kann z. B. auf eine Klassifikations- oder Regressionsaufgabe zugeschnitten werden. Diese wiederverwendbaren Architekturbestandteile werden als Backbones bezeichnet [9, S. 1] [5, S. 1][18,

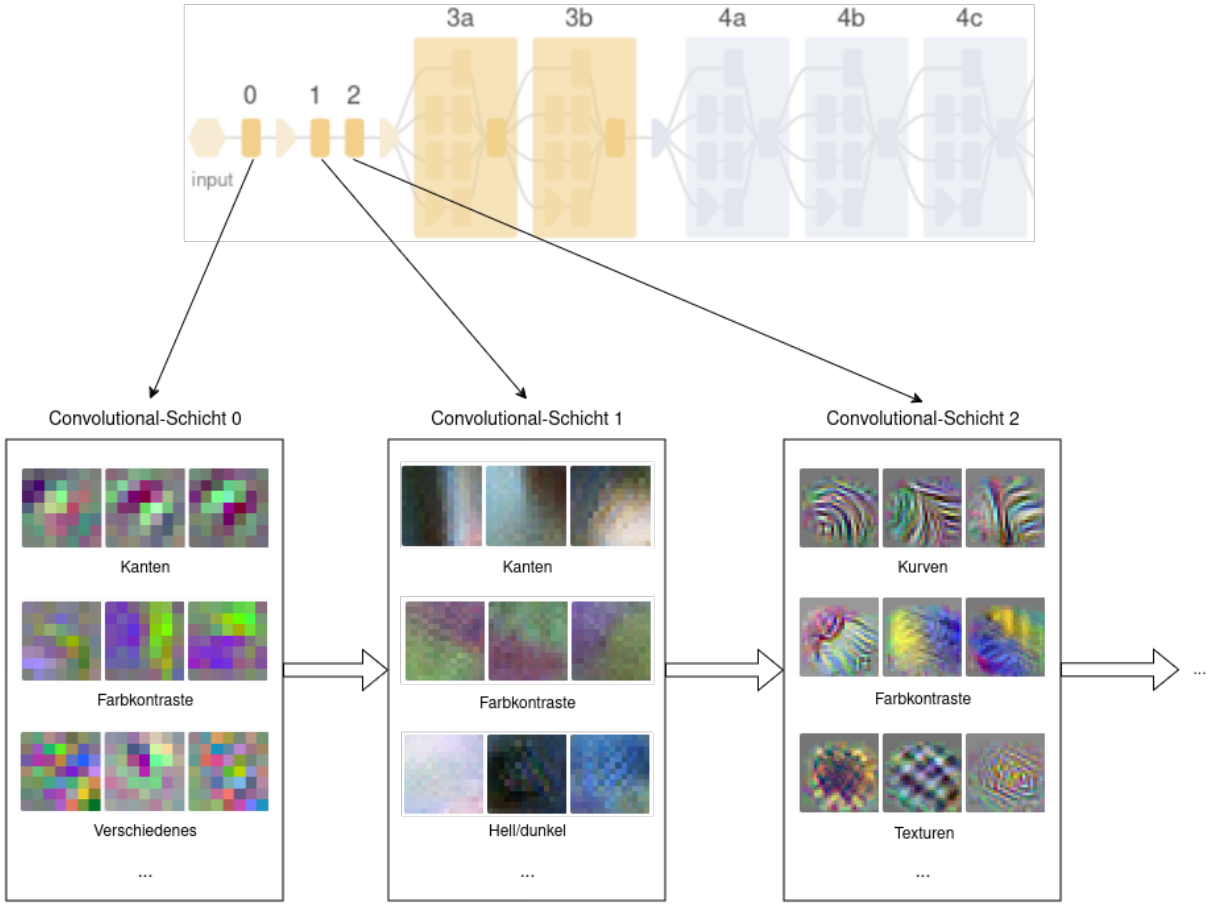


Abbildung 3.5: Visualisierung einiger Muster, welche bestimmte Convolutional-Filter aus den ersten drei Convolutional-Schichten des InceptionV1-Netzes [60] maximal aktivieren. Bildmaterial und Inhalte übernommen und adaptiert aus [49].

S. 1]. Die Fähigkeit von Backbones, Features aus Bildern zu extrahieren, ist größtenteils von der genauen Aufgabe des Netzes unabhängig. So haben sich einige populäre Backbones für eine breite Masse an Aufgaben etabliert (vgl. [70, S. 3 ff.]), welche in vielen unterschiedlichen Anwendungsfällen eingesetzt werden können. Die Teile eines CNNs, welche auf einen Anwendungsfall (beispielsweise eine Objektdetektion) zugeschnitten sind, werden als Neck und Head bezeichnet. Der Head ist zur Lösung der eigentlichen Aufgabe zuständig, im Rahmen der Objektdetektion erzeugt er die Ausgabe der Bounding Boxes und ihren Klassenzuordnungen [5, S. 1]. Ein Neck kann eingesetzt werden, um die Features von früheren Schichten des Backbones mit den extrahierten Features aus späteren Schichten zu kombinieren [5, S. 1]. Es ist zu beachten, dass das Konzept des Necks in mancher Literatur teilweise ausgelassen und als Teil des Heads angesehen wird (vgl. z. B. [9]).

3.2.2 *Aufbau moderner Objektdetektoren*

Auf neuronalen Netzen basierende Objektdetektoren stellen oftmals Convolutional Neural Networks dar oder nutzen diese zumindest in Teilen. Convolutional-Schichten werden hierbei mit anderen Bestandteilen verknüpft, um eine Objektdetektion umzusetzen. Hierbei existieren mehrere Vorgehensweisen, welche im Folgenden betrachtet werden.

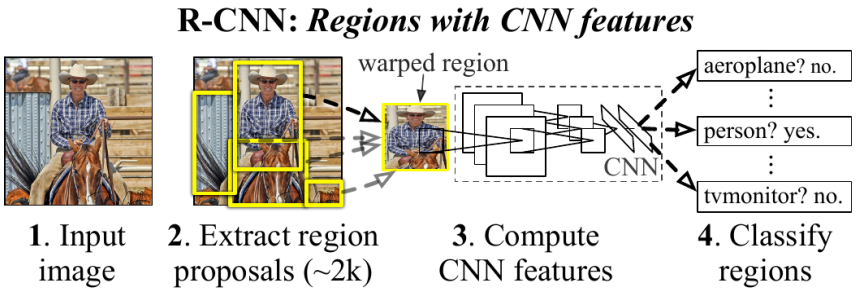


Abbildung 3.6: Vorgehen eines R-CNN-Objektdetektors, einem zweistufigen Verfahren. Entnommen aus [23, S. 1].

Objektdetektoren teilen sich in einstufige und zweistufige Verfahren auf. Eine einfache Möglichkeit zur Erkennung von Objekten besteht darin, vorhandene Modelle zur Klassifikation von Bildern für einzelne Bildausschnitte wiederzuverwenden [2, S. 878]. Ein Modell erzeugt Vorschläge für Objektregionen, welche relevante Objekte enthalten könnten, ein zweites Modell erhält diese Bildausschnitte und klassifiziert sie. Solche Verfahren werden als zweistufige Verfahren bezeichnet [61, S. 304–306]. Abbildung 3.6 stellt das Vorgehen des R-CNN-Modells von Girshick et al. [23] dar. Es ist erkennbar, dass die Vorschläge des ersten Modells die Bildausschnitte bestimmen, welche skaliert in ein zweites Modell zur Klassifikation eingegeben werden. Das Gegenstück bilden einstufige Verfahren, welche mit einem einzelnen Modell Vorhersagen für erkannte Objekte an verschiedenen Positionen erzeugen [61, S. 306]. Hierbei werden innerhalb des Modells viele verschiedene Bildausschnitte gleichzeitig betrachtet, für welche jeweils die genaue Position eines potenziellen Objektes und eine Klassifikation ausgegeben werden [61, S. 306].

Dieses Konzept soll kurz beispielhaft an der ersten Implementierung der YOLO-Architektur nach Redmon et al. [53] illustriert werden, die folgenden Beschreibungen der Schritte sind hierbei aus dem Original übernommen. Im ersten Schritt der Verarbeitung wird das Eingabebild auf eine feste Größe von 448×448 Pixeln skaliert. Dieses quadratische Bild wird im Anschluss in ein Raster mit $S \times S$ Zellen aufgeteilt, in der Originalimplementierung von Redmon et al. gilt $S = 7$. Jede Zelle ist für die Detektion von Objekten zuständig, falls der vermutete Mittelpunkt eines Objektes innerhalb dieser Zelle liegt. Pro Zelle werden B Boxen vorhergesagt (im Original $B = 2$), sodass mehrere Objekte pro Zelle erkannt werden können. Jede Box wird durch fünf Ausgabewerte dargestellt: x, y, w, h und *conf*. x und y lokalisieren den Mit-

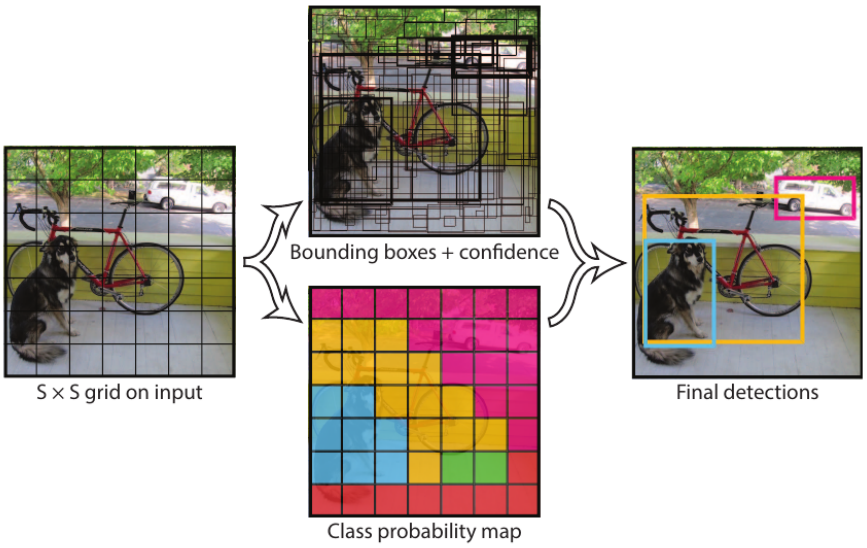


Abbildung 3.7: Vorgehen eines YOLOv1-Objektdetektors, einem einstufigen Verfahren. Entnommen aus [53, S. 2].

telpunkt der Box relativ zur Zelle, h und w beschreiben die Breite der Box im Verhältnis zum Gesamtbild. *conf* gibt die Wahrscheinlichkeit $P(\text{Object})$ an, dass überhaupt ein Objekt einer beliebigen Klasse innerhalb der Zelle vorhanden ist. Die Klassenzuordnung erfolgt pro Vorhersage durch C bedingte Wahrscheinlichkeiten $P(\text{Class}_i, \text{Object})$, welche angeben, wie wahrscheinlich das Objekt welcher Klasse zugeordnet wird. Die Ausgaben des Netzes sind somit für alle $S \times S$ Zellen jeweils eine oder mehrere Vorhersagen für Objekte und deren Bounding Boxes. Bei Objekten, welche sich über mehrere Zellen erstrecken, können mehrere Zellen verschiedene Boxen für das Objekt vorschlagen. Um doppelte Detektionen zu vermeiden, werden die vorgeschlagenen Boxen mit dem Non-Maximum-Suppression-Verfahren reduziert [53, S. 4], hierbei werden Boxen mit geringerer Konfidenz verworfen, wenn sie sich mit einer Box mit höherer Konfidenz ausreichend stark überlappen (vgl. [28, S. 3]). Abbildung 3.7 illustriert diese Schritte zusammenfassend.

Das YOLO-Netz besteht aus einem von Redmon et al. selbst erstelltem Backbone mit einem einzelnen Head, welcher die Ausgaben erzeugt. Andere Detektoren nutzen unterschiedliche Backbones, welche evtl. mit mehreren Heads verbunden sind. Während sich verschiedene einstufige Objektdetektoren im Detail unterscheiden, so sind sie in ihrem Kern konzeptuell gleich. Durch alle einstufigen Objektdetektoren wird eine Vielzahl an möglichen Positionen im Bild gleichzeitig durchsucht, um Bounding Boxen und Klassenzuordnungen für die Detektionen zu erhalten.

3.3 METHODEN ZUR VERKLEINERUNG NEURONALER NETZE

Neuronale Netze stellen komplexe statistische Modelle dar, welche viele Zusammenhänge aus Daten erlernen können. Diese große Wissenskapazität wird durch eine große Menge an Parametern innerhalb der Netze ermöglicht, welche bei Speicherung der Netze dementsprechend viel Speicherplatz benötigen. Bei den Parametern handelt es sich um reelle Zahlen, welche in Computern als Gleitkommazahlen repräsentiert werden. Dabei haben sich in der Praxis primär die nach IEEE 754 definierten Gleitkommazahlformate durchgesetzt, wobei die Formate mit 32 Bit pro Gleitkommazahl (hier als float32 bezeichnet) auf modernen Prozessoren das Standardformat darstellen. Dies bedeutet, dass jeder Parameter eines neuronalen Netzes vier Byte Speicher belegt, während eventuell nicht der gesamte Wertebereich dieser vier Byte notwendig wäre. In diesem Abschnitt werden mit Quantisierung und Teacher-Student-Training zwei Verfahren eingeführt, welche die resultierenden Netzparameter entweder in ihrer Darstellung so verändern, dass sie weniger Speicherplatz belegen oder versuchen, alternative Netzarchitekturen mit der gleichen Performance, aber weniger Gewichten zu ermöglichen.

3.3.1 *Quantisierung*

Quantisierung bezeichnet Verfahren, bei denen Werte eines kontinuierlichen Bereiches auf einen kleineren Bereich diskretisiert werden. In neuronalen Netzen finden jegliche Berechnungen mit reellen Zahlen statt, wobei die Werte dieser Zahlen aus verschiedenen Quellen kommen. Ein Teil der Werte stellt die Netzparameter wie Bias und Gewichte dar, die restlichen Werte sind die Ergebnisse vorheriger Operationen (sog. Aktivierungen) oder die Eingaben des Netzes [48, S. 2]. Illustrieren lässt sich dies an Formel 3.6 für ein einzelnes Neuron (ohne seine Aktivierungsfunktion) mit n Eingabewerten. Der Ausgabewert y ist sowohl von den Eingaben x_i als auch den Parametern b und w_i des Neurons abhängig, weitere Werte fließen nicht in die Berechnung mit ein.

$$y = b + \sum_{i=0}^{n} x_i w_i \tag{3.6}$$

mit

- y: Ausgabe des Neurons
- b: Bias des Neurons
- x_i: i-ter Eingabewert des Neurons
- w_i: i-tes Gewicht des Neurons

Eine Implementierung neuronaler Netze in Computerhardware nutzt wie erwähnt gewöhnlicherweise das float32-Format mit 32 Bits Genauigkeit. Falls nicht der gesamte Wertebereich dieses Formats ausgenutzt wird, stellen einige Bits verschwendeten Speicherplatz dar, da sie nie genutzt werden oder

immer die gleichen Werte annehmen. Durch eine Reduktion der Bits pro Zahl kann die Anzahl der ungenutzten Bits minimiert und die Grösse aller Werte reduziert werden.

Im Rahmen einer Quantisierung der Werte werden diese von einem grösseren Wertebereich (im Folgenden wird der 32-Bit-Wertebereich von Gleitkommazahlen angenommen) in einen kleineren überführt. Eine Möglichkeit besteht hierbei in der Nutzung von Gleitkommazahlformaten geringerer Präzision, beispielsweise von float16 mit 16 statt 32 Bits pro Gleitkommazahl. Für Umwandlungen zwischen Gleitkommazahlformaten, welche sich nur in der Bitanzahl unterscheiden, entspricht dieser Prozess nur einer Rundung zu dem nächsten im kleineren Format darstellbaren Wert und ist trivial, da es sich bei beiden Formaten um Gleitkommazahlformate handelt. Problematisch ist bei diesem Vorgehen, dass nur wenige weit verbreitete Gleitkommazahlformate existieren, wobei float16 hierbei das kleinste Format darstellt.

Eine Alternative zu kleineren Gleitkommazahlformaten stellt die Nutzung von Ganzzahlformaten dar. Hier existieren kleinere Datentypen, wie beispielsweise vorzeichenbehaftete Ganzzahlen mit 8 Bit (int8). Die Umwandlung von Gleitkommazahlformaten auf solche Ganzzahlformate gestaltet sich hingegen komplexer, was im Folgenden betrachtet wird.

Adaptiert von Nagel et al. [48, S. 4] ist eine uniforme Quantisierung eines Wertes $x \in \mathbb{R}$ auf eine Ganzzahl $x_{int} \in \mathbb{Z}$ durch Formel 3.7 gegeben.

$$x_{int} = \text{clamp}(\text{round}(\frac{x}{s}) + z, v_{min}, v_{max}) \tag{3.7}$$

$$\text{clamp}(x, a, b) = \left\{ \begin{array}{ll} a, & x < a \\ x, & a \leq x \leq b \\ b, & b < x \end{array} \right\} \tag{3.8}$$

mit

x:	Zu quantisierender Wert
z:	Nullpunkt des Wertebereichs
s:	Schrittweite der Quantisierung
v_{min}:	Kleinster darstellbarer Wert im Zieldatentyp
v_{max}:	Größter darstellbarer Wert im Zieldatentyp
$\text{round}(x)$:	Funktion, welche zur nächsten Ganzzahl rundet
$\text{clamp}(x, a, b)$:	Funktion, welche x auf das Intervall $[a, b]$ beschränkt, nach Formel 3.8

Es ist erkennbar, dass eine Quantisierung auf Ganzzahlen primär von den Werten von s abhängig ist. Die Schrittweite bestimmt, welche Werte auf die gleiche Ganzzahl abgebildet werden. Größere s führen dazu, dass grössere Wertebereiche auf die gleiche Ganzzahl abgebildet werden, während kleinere Werte feinere Abstufungen erhalten. Gleichzeitig schränken v_{min} und v_{max} ein, dass nicht beliebig große Werte als Ergebnis der Quantisierung erhalten können, die den darstellbaren Wertebereich des ganzzahligen Datentyps

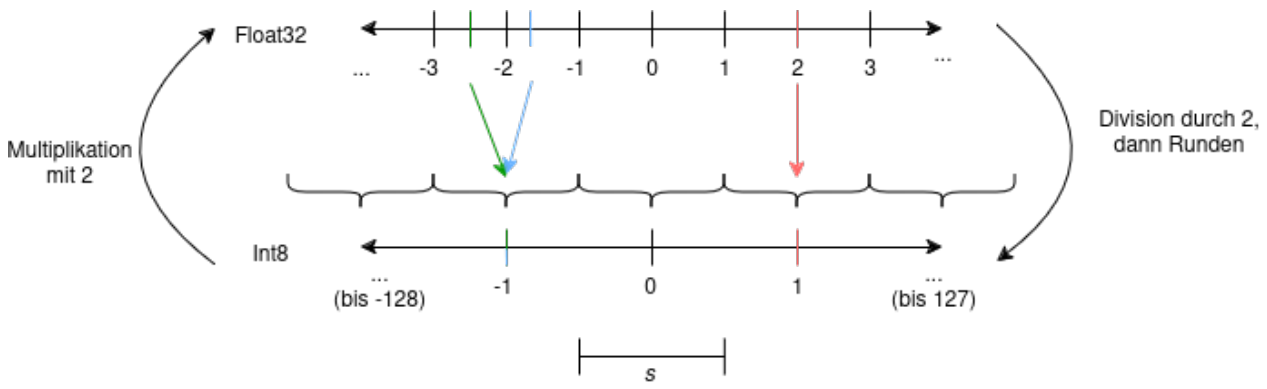


Abbildung 3.8: Beispiel füer eine Quantisierung eines float32-Wertebereichs auf einen int8-Wertebereich mit $z = 0$ und $s = 2$.

übersteigen. Für einen vorzeichenbehafteten 8-Bit-Integer (int8) sind somit $v_{min} = -128$ und $v_{max} = 127$, für einen vorzeichenlosen 8-Bit-Integer (uint8) sind $v_{min} = 0$ und $v_{max} = 255$. Der Nullpunkt z ermöglicht ein Verschieben des erlaubten Wertebereichs. Abbildung 3.8 illustriert eine Quantisierung von float32-Werten auf int8-Werte mit $z = 0$ und $s = 2$. Die farbigen Pfeile stellen dar, wie float32-Werte auf bestimmte int8-Werte abgebildet werden. Hierbei können Werte erkannt werden, welche im originalen Wertebereich unterschiedlich waren, im quantisierten Wertebereich aber auf den gleichen Wert projiziert wurden. Es ist erkennbar, dass die Quantisierung effektiv eine Skalierung des originalen Wertebereichs mit folgendem Runden auf eine Ganzzahl ist. Nicht abgebildet sind die Enden des Wertebereichs. Zumal es sich um einen int8-Wertebereich handelt, werden float32-Werte, für die $\text{round}(\frac{x}{2}) \notin [-128, 127]$ gilt, alle auf -128 oder 127 abgebildet.

Mit dem beschriebenen Schema lassen sich die Parameter eines Netzes, also Biase und Gewichte auf einen kleineren Datentypen wie int8 quantisieren (normalerweise werden jedoch nur die Gewichte quantisiert, weil diese einen grösseren Wertebereich mit höherer Genauigkeit abdecken müssen [48, S. 3]). Geschieht die Quantisierung, nachdem das Netz zuvor mit float32-Gewichten erstellt wurde, spricht man von Post-Training-Quantization (PTQ), eine Quantisierung zur Trainingszeit des Netzes (durch Simulation der Quantisierungseffekte innerhalb des Netzes) wird Quantization-Aware-Training genannt [48, S. 8-19]. In dieser Arbeit soll nur PTQ betrachtet werden.

Es ist zu beachten, dass eine Quantisierung primär eine Reduktion des Speicherbedarfs der Modellgewichte mit sich bringt, eine effizientere Berechnung muss nicht zwingend erreicht werden. Eine Quantisierung auf einen ganzzahligen Datentyp bedeutet ebenfalls, dass nicht alle Operationen, welche auf Gleitkommazahlen berechenbar sind, auch auf Ganzzahlen berechnet werden können. Für bestimmte Berechnungen müssen die Werte möglicherweise zurück zu Gleitkommazahlen dequantisiert werden, um auf diesen die Operationen auszuführen. Dies führt dazu, dass eine starke Quantisierung nicht zwingend zu einem Performancegewinn führen muss [31, S. 1]. Mit speziellen Vorgehen bei einer Quantisierung lassen sich diese Ein-

schränkungen jedoch umgehen und jegliche Operationen mit Ganzzahlarithmetik implementieren (vgl. [31] für die Quantisierung hinter TensorFlow Lite Micro, welches eine Inferenz auf Hardware ohne Gleitkommazahlunterstützung ermöglicht).

3.3.2 *Teacher-Student-Training*

Neuronale Netze können bei einer ausreichenden Anzahl an Neuronen sehr komplexe Aufgaben erlernen. Gleichzeitig sind die genauen Strukturen, welche sich innerhalb der Netze herausbilden und deren genaue Funktionsweise eine Black Box, welche höchstens in minimalem Maße von Forschenden verstanden werden kann. Es ist einfach, ein Netz zu identifizieren, welches nicht komplex genug für eine gegebene Aufgabe ist; es wird eine schlechte Performance aufweisen und unzureichende Ergebnisse erzielen. Hingegen ist es praktisch unmöglich, ein Netz als zu komplex für seine Aufgabe zu identifizieren. Ab einer bestimmten Größe laufen neuronale Netze somit Gefahr, mehr Parameter zu besitzen als zur Lösung ihrer Aufgabe notwendig, wodurch ihre Parameter zusätzlichen Speicherplatz belegen. Eine Methode, dieses Problem anzugehen, bietet das Konzept der Knowledge Distillation (zu Deutsch etwa "Wissensdestillation"), welches auch als Teacher-Student-Training bekannt ist. Folgender Absatz stellt den ursprünglichen Gedanken des Teacher-Student-Trainings vor und ist inhaltlich aus einer Arbeit von Hinton et al. [27] übertragen.

Knowledge Distillation beschreibt einen Vorgang, bei dem das Wissen eines oder mehrerer komplexer Modelle in ein kleineres, einfacheres Modell übertragen werden soll. Komplexe Modelle sind, durch ihre Größe bedingt, mit einem höheren Rechen- und Speicheraufwand verbunden. Beim überwachten Training neuronaler Netze werden die gewünschten Ausgaben vorgegeben, welche das Netz im Laufe des Trainings erlernt aus Mustern der Daten zu erzeugen. Das Konzept der Knowledge Distillation gilt für beliebige Aufgaben, wurde aber zuerst für Klassifikationsaufgaben beschrieben. Für Klassifikationsaufgaben besteht die Vorgabe aus der korrekten Klassenzuordnung jedes Trainingsbeispiels. Das Netz erlernt, eine Reihe an Klassenwahrscheinlichkeiten zu erzeugen, wobei die höchste Wahrscheinlichkeit der korrekten Klasse einer Eingabe zugeordnet werden soll. Normalerweise werden bei einer solchen Klassifikationsaufgabe selbst nach langem Training eines Netzes keine hundertprozentig eindeutigen Vorhersagen erzeugt, es entfallen immer durch leichte Unsicherheiten wenige Prozentpunkte auf andere, falsche Klassen. Grundgedanke der Knowledge Distillation ist, dass diese Unsicherheiten ausgenutzt werden können, um das Training eines neuen Netzes zu verbessern, weil die Unsicherheiten selbst einen Informationsgehalt über Muster in den Daten enthalten. Ziel des Trainings eines neuronalen Netzes ist die Extraktion solcher generalisierbarer Muster aus den Daten, sodass eine Verwendung der Ausgaben eines komplexeren Modells, des "Teachers" helfen kann, diese Muster dem kleineren "Student" zu vermitteln.

Aufgrund dieser Begriffe wird das Konzept der Knowledge Distillation auch als Teacher-Student-Training bezeichnet [67, S. 1].

Um diese Veränderung in den Trainingsprozess einfließen zu lassen, wird der Student mit einer modifizierten Verlustfunktion trainiert. Für ein Modell, dessen Ausgaben durch die Funktion $f_{student}$ berechnet werden, kann die originale Verlustfunktion L_{orig}, wie in Formel 3.9 erkennbar, in eine neue Verlustfunktion integriert werden (vgl. [27, Abs. 2]).

$$L'(x, y_{gt}) = \frac{\alpha L_{orig}(f_{student}(x), y_{gt}) + \beta L_{soft}(f_{teacher}(x), f_{student}(x))}{\alpha + \beta} \tag{3.9}$$

Analog zu $f_{student}$ erzeugt $f_{teacher}$ die Ausgabe des Teachers für eine Eingabe x. Der neue Verlustwert für ein gegebenes Ziel-Label y_{gt} setzt sich somit aus dem mit α und β gewichteten Mittel der originalen Verlustfunktion L_{orig} und der Verlustfunktion L_{soft} zur Bewertung der Differenz zwischen den Ausgaben von Teacher und Student zusammen. Für L_{orig} kann eine beliebige Verlustfunktion genutzt werden, Hinton et al. nutzen für den Beispielfall eines Klassifikationsproblems die Kreuzentropie.

Während im vorherigen Absatz eine Klassifikationsaufgabe als Beispiel genutzt wurde, kann Teacher-Student-Training auf andere Aufgaben angewandt werden, auch auf Objekterkennung. Da eine Objekterkennung neben einer Klassifikation auch die Lokalisation als Teilaufgabe beinhaltet, kann das beschriebene Vorgehen nicht direkt übertragen werden. Nach Wang et al. [67] resultiert ein Nachbilden der genauen Ausgaben eines Teacher-Objektdetektors durch seinen Student darin, dass der Student erlernt, Rauschen in den Daten zu modellieren. Dies wird von den Autoren damit begründet, dass einstufige Objektdetektoren für sehr viele mögliche Positionen gleichzeitige Vorhersagen erstellen. Viele dieser Vorhersagen enthalten keine Objekte und erfassen nur den irrelevanten Hintergrund, während ihre Klassenzuordnungen dadurch Werte beinhalten, deren Nachbildung keinen Informationswert bringt (vgl. [67, S. 2]). Die Autoren schlagen stattdessen vor, den Student die Feature Maps des Teachers nachahmen zu lassen. Während bei einer Klassifikation die Klassenwahrscheinlichkeiten darstellen, inwiefern der Teacher unsicher ist, sind bei der Lokalisation hierfür die gefundenen Features nahe eines vorhergesagtes Objektes von Bedeutung [67, S. 2]. Wang et al. schlagen somit vor, mit einer Verlustfunktion den Student dazu zu bewegen, die Features des Teachers in der Nähe der zu erkennenden Objekte nachzuahmen. Für jedes auf einem Datensatz markierte Objekt wird erst die Nachbarschaft berechnet, in welcher der Student die Feature Maps des Teachers nachahmen soll. Für diese Bereiche wird der Wert der Imitations-Verlustfunktion berechnet.

Abbildung 3.9 bildet dieses Vorgehen ab. Für das links erkennbare Eingabebild berechnen sowohl der Student als auch der Teacher eine Ausgabe. Dabei durchläuft das Bild die Convolutional-Layer, welche mit steigender Tiefe des Netzes immer komplexere Features erfassen. Um die Features von Teacher und Student vergleichen zu können, müssen diese die gleiche Brei-

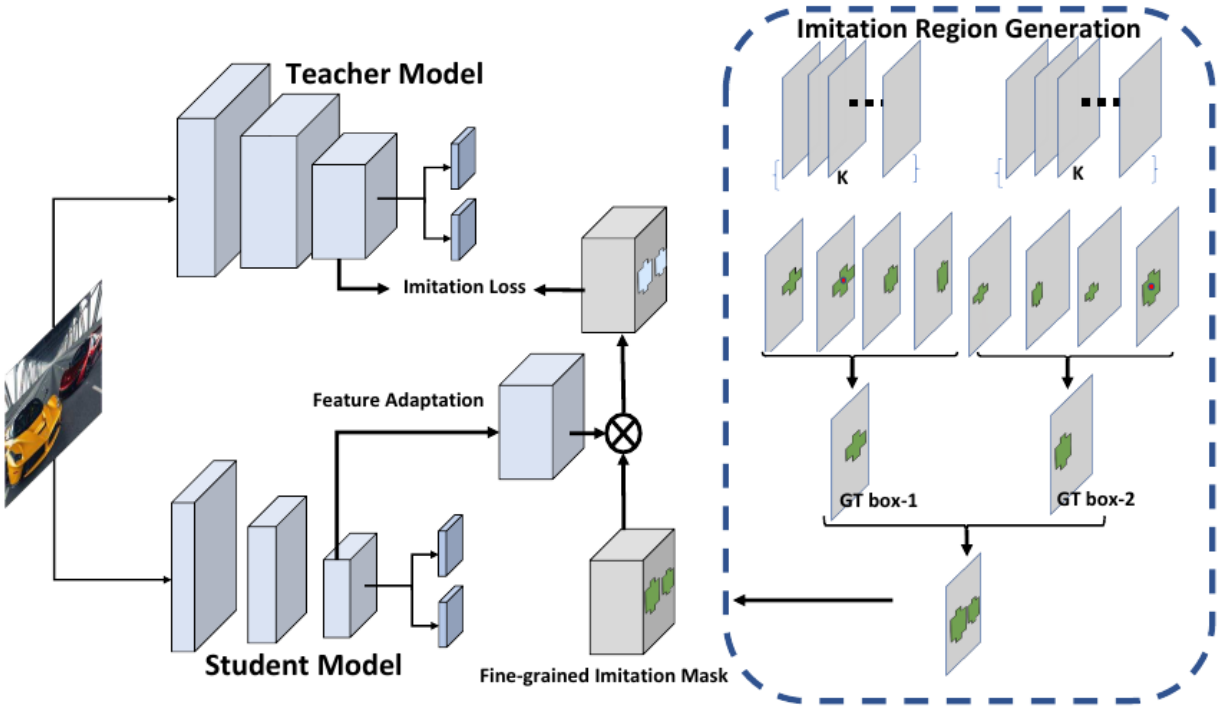


Abbildung 3.9: Schematischer Aufbau eines Teacher-Student-Trainings durch Feature Imitation nach Wang et al. [67]. Entnommen aus [67, S. 3].

te, Höhe und Anzahl an Filtern haben. Da dies nicht für alle Kombinationen aus möglichen Modellen für Teacher und Student gegeben ist, wird in den Student eine zusätzliche Convolutional-Schicht hinzugefügt, welche die letzte Feature Map des Students so transformiert, sodass ihre Dimensionalität mit der letzten Feature Map des Teachers kompatibel ist. Dies ist in Abbildung 3.9 als "Feature Adaptation" erkennbar. Da der Student den Teacher während des Trainings nur in der Nähe tatsächlich vorhandener Objekte imitieren soll, wird aus den Labels des Bildes eine Maske erzeugt. Dies ist im rechten, gestrichelt umrandeten Block erkennbar. Für jedes per Label markierte Objekt wird die Nachbarschaft bestimmt, welche zu diesem Objekt gehört. Durch Kombination der Nachbarschaften aller markierten Objekte wird eine Maske erhalten, die beschreibt, welche Stellen im Bild nahe eines zu erkennenden Objektes sind und deshalb von Bedeutung für Knowledge Distillation sind. Die Ausgaben des eingefügten Convolutional-Layers werden an den Stellen, die durch die Maske zugelassen werden, miteinander verglichen und der "Imitation Loss" berechnet. Der Imitation Loss L_{soft} kann schließlich wie in Formel 3.9 verrechnet werden und der Student erlernt, sowohl die Unsicherheiten des Teachers zu modellieren als auch durch die originale Verlustfunktion L_{orig} das generelle Verständnis der Objekte durch die Ground Truth zu erlangen.

Für die Personenerkennung wäre es von Interesse, ein möglichst kleines Netz mit wenigen Parametern zu erhalten, welches selbst auf eingebette-

ten Geräten mit wenig Speicher lauffähig ist. Mit Teacher-Student-Training könnte eventuell ein kleineres neuronales Netz erhalten werden, welches aus einem komplexeren, vorher trainiertem Netz destilliert wurde, aber trotzdem eine bessere Erkennungsperformance als ähnlich kleine, ohne Teacher trainierte Modelle aufweist.

4 METHODIK

Mit der Forschungsfrage dieser Arbeit soll untersucht werden, wie eine Personenzählung mit neuronalen Netzen auf eingebetteten Systemen umgesetzt werden kann. Durch die Teilfragen sind zur Beantwortung der Hauptfrage bestimmte Teilaspekte gegeben, welche nacheinander untersucht werden sollen. Hierbei wurde die erste Teilfrage, welche auf alternative Verfahren zur Personenerkennung und -zählung abzielt, bereits durch die Abgrenzung zu bisheriger Forschung untersucht und muss nicht mehr beantwortet werden. Die verbleibenden Teilfragen sind hingegen noch offen und sollen stückweise ausgearbeitet und beantwortet werden.

Um die Besonderheiten und Herausforderungen einer Implementierung einer Personenzählung zu beleuchten, sollen mehrere spezifische Hardwaregeräte aus dem Embedded-Bereich als Stichprobe ausgewählt werden. Hierzu soll basierend auf theoretischen Überlegungen erst eine Mindestanforderung an Hardware formuliert werden, welche zur Ausführung von Objektdetektoren notwendig ist. Seitens der Auswahl einer Stichprobe existiert kein standardisiertes Verfahren, die gewählten Geräte sollen jedoch bestimmte Klassen bzw. Eigenschaften typischer eingebetteter Geräte, welche geeignet wären, verkörpern. Gleichzeitig sollen bei der Auswahl der Geräte die besonderen Anforderungen dieser Arbeit beachtet werden, sodass die gewählten Geräte sowohl dem angestrebten Budget von weniger als 100 Euro pro Stück genügen. Als Ergebnis dieser Stichprobe können mehrere Hardwaregeräte erhalten werden. Für diese Geräte können ihre Eckdaten vorgestellt werden, um eine quantitative Angabe dazu zu erhalten, in welchen Größenordnungen sich die verfügbaren Ressourcen solcher Hardware bewegen. Mit diesen Informationen kann bestimmt werden, welche Herausforderungen auf eingebetteten Geräten durch Einschränkungen der Hardware entstehen.

Aufbauend auf die Hardwareauswahl soll eine Wahl geeigneter Netzarchitekturen zur Objekterkennung getroffen werden. Hierzu soll basierend auf einer Literaturrecherche populärer Netze gezeigt werden, welche Netzarchitekturen sich nicht für die Ausführung auf eingebetteten Systemen eignen. Somit können zu langsame oder zu ressourcenintensive Netze ausgeschlossen werden und für die weiteren Untersuchungen geeignete Architekturen gefunden werden. Vortrainierte neuronale Netze zur Objektdetektion sind üblicherweise allgemein gehalten und eignen sich zur Erkennung vieler verschiedener Objekttypen. Für diese Arbeit ist nur die Erkennung von Personen von Bedeutung, sodass die Netze auf diese Objektklasse angepasst werden müssen. Es werden Trainingsdaten benötigt, wobei geeignete Datensätze ebenfalls durch eine Literaturrecherche identifiziert werden sollen. Wie bereits im Rahmen der Abgrenzung zu bisheriger Forschung illustriert, sind passende Datensätze schwer aufzufinden, weshalb auch Datensätze mit

möglicherweise nicht vollständig auf den Anwendungsfall passenden Sachverhalten betrachtet werden sollen. Durch die Wahl geeigneter Netzarchitekturen und Datensätze können im Anschluss neuronale Netze zur Erkennung von Personen trainiert und evaluiert werden. Es sollen mehrere Varianten an Netzen trainiert werden, um die Auswirkungen der Anpassungen der Netzarchitekturen vergleichen zu können. Gleichzeitig sollen alle Modellvarianten auf allen gewählten Datensätzen trainiert werden, um Vergleiche zwischen Datensätzen mit unterschiedlichen Eigenschaften zu ermöglichen. Im Anschluss an das Training sollen die Netze auf den Datensätzen evaluiert werden. Dabei sollen mehrere Metriken zur Bewertung der Effektivität dieses Umtrainierens erfasst werden. Basierend auf Trainingsdauern kann begründet werden, ob das Training mit vertretbaren zeitlichen Aufwänden verbunden ist. Mit Metriken zur Bewertung der Objektdetektionsqualität und zur Bewertung der Zählperformance sollen die Netze bewertet werden und quantifiziert werden, ob das Erstellen neuer, auf Personen spezialisierter Netze bessere Ergebnisse erzielt als die Verwendung vortrainierter Objektdetektoren. Diese Bewertung kann ohne Nutzung der gewählten Hardware durchgeführt werden, da die Netze selbst unabhängig von Hardware bestehen. Durch die Vorbereitung und Durchführung des Trainings spezialisierter Objektdetektoren lässt sich somit die zweite Teilfrage beantworten, inwiefern das Anpassen existierender Netze für eine Personenzählung durchführbar und nützlich ist.

Mit den Verfahren der Quantisierung und des Teacher-Student-Trainings wurden zwei Methoden eingeführt, welche zur Reduktion des Speicherplatzes für die Parameter eines neuronalen Netzes oder zum Training leistungsfähiger kleiner Netze genutzt werden können. Beide Verfahren sollen auf die erstellten Objektdetektoren angewandt und ihre Auswirkungen auf die Güte der erhaltenen Netze untersucht werden. Es soll untersucht werden, ob eine Quantisierung die Performance der Modelle stark verringert und ob Teacher-Student-Training bessere Ergebnisse erzielt als ein direktes Training ohne ein Übertragen des Wissens eines größeren Modells. Mit den Ergebnissen zu den Auswirkungen von Quantisierung und Teacher-Student-Training kann dann eine Aussage erhalten werden, ob solche Techniken geeignet sind, um neuronale Netze auf die besonderen Rahmenbedingungen eingebetteter Geräte anzupassen, was Inhalt der letzten Teilfrage ist.

Mit dem beschriebenen Konzept lassen sich die verbleibenden Teilfragen beantworten. Die Erkenntnisse müssen anschliessend integriert werden, um echte Messungen auf den gewählten Hardwaregeräten zu erhalten. Hierzu soll abschliessend eine Implementierung der Netze auf den Geräten erfolgen. Durch die Implementierung auf echter Hardware können einerseits reale Performancemessungen erhalten und Erkenntnisse zur Hauptfrage hinsichtlich der verwendbaren Frameworks erhalten werden. Die Performancemessungen sollen durch Laufzeitmessungen und gegebenenfalls anderen Performancemetriken erhalten werden und Zusammenhänge mit Faktoren wie der Bildgröße dargestellt werden.

5

KONZEPT EINER EINGEBETTETEN PERSONENZÄHLUNG

In diesem Kapitel wird, dem in der Methodik entsprechendem Grundgedanken folgend, erst die Problemstellung der Ausführung neuronaler Netze auf leistungsschwachen eingebetteten Geräten durch eine Stichprobe solcher Geräte illustriert. Im Anschluss werden darauf aufbauend schrittweise Möglichkeiten zum Umgang mit diesen Herausforderungen dargestellt, sodass ein Konzept für eine Personenzählung entsteht.

5.1 HERAUSFORDERUNGEN VON COMPUTER VISION AUF EINGEBETTETEN SYSTEMEN

Die Umsetzung eines deckenmontierten Kamerasystems mit integrierter Personenzählung innerhalb des Geräts schränkt bestimmte Aspekte der verwendeten Hardware ein. Primär wird der verfügbare Platz für das Gerät eingeschränkt, was im Gegenzug auch die Rechenleistung des Systems begrenzt, da stärkere Hardware für gewöhnlich ebenfalls mehr Platz beansprucht. Diese Einschränkungen erfordern, dass spezialisierte, auf den Anwendungsfall zugeschnittene Hardware eingesetzt wird, welche die Personenzählung ermöglicht. Solche spezialisierte, als Teil eines Systems zur Erfüllung einer definierten Funktion eingesetzten Hardwarekomponenten werden als eingebettete Geräte bezeichnet [7, S. 1]. Eingebettete Systeme unterscheiden sich von allgemeiner gehaltenen Systemen damit durch ihre Spezialisierung auf eine Aufgabe, beispielsweise als Teil eines Bremssystems eines Autos [7, S. 1].

Die Ausführung neuronaler Netze stellt eine rechenintensive Aufgabe dar, was bestimmte Mindestanforderungen an Hardware mit sich bringt. Diese Anforderungen unterscheiden sich je nachdem, welche Art von neuronalem Netz in welchem Anwendungskontext eingesetzt wird. Kleine neuronale Netze zur Lösung einfacher Aufgaben benötigen nur wenige Berechnungen, größere Netze für komplexe Sachverhalte benötigen deutlich mehr Rechenschritte. Im Kontext der Bildverarbeitung besteht beispielsweise ein Unterschied zwischen einem Netz zur Erkennung von handgeschriebenen Ziffern auf kleinen Grauwertbildern und einem komplexen Netz, welches verschiedene Objekte in einem hochauflösenden Farbbild einer Kamera lokalisiert. Ersterer Fall, welcher beispielsweise von LeCun et al. im Jahre 1998 mit einem CNN gelöst wurde, nutzt ein Netz mit 60 Tausend Parametern [39, S. 7], während moderne Objektdetektoren mehrere Millionen Parameter besitzen (vgl. Tabelle 5.3). Es kann angenommen werden, dass die Objekterkennung ein generell schweres Problem darstellt, welches nur mit entsprechend komplexen Netzen zufriedenstellend gelöst werden kann, wofür leistungsstarke eingebettete Geräte vonnöten sind.

Um die Ausgaben eines neuronalen Netzes zu berechnen, werden üblicherweise Matrixmultiplikationen durchgeführt. Wie bereits im Zuge der Quantisierung dargestellt, werden hierbei Werte aus mehreren Quellen miteinander verrechnet. Ein Teil der Werte stellt Parameter des Netzes dar, welche zur Ausführungszeit des Netzes unveränderlich sind. Der Rest der Werte wird durch Eingaben in das Netz (für die erste Schicht) bzw. die Ausgaben vorheriger Schichten (für alle andere Schichten) gebildet, welche veränderlich sind und zur Ausführungszeit erzeugt werden. Beide Arten an Werten fordern unterschiedliche Aspekte der Hardware: Modellparameter müssen permanent abgespeichert werden, sodass sie aus einem ausreichend grossen Festwertspeicher geladen werden, während die Eingaben und Zwischenergebnisse im Arbeitsspeicher (engl. Random Access Memory, kurz RAM) Platz finden müssen. Während die Größe der Zwischenergebnisse maßgeblich von der Architektur des neuronalen Netzes abhängig ist, ist die Größe der Eingabebilder im Speicher nur von deren Auflösung und Farbtiefe abhängig. Beispielsweise belegt ein quadratisches RGB-Farbbild mit 256 Pixeln Kantenlänge und drei Farbkanälen mit je einem Byte Farbtiefe bereits $256 \times 256 \times 3 = 196.608$ Bytes, ein Bild mit 512 Pixeln Kantenlänge bereits 786.432 Bytes im Arbeitsspeicher. Diese Speicheranforderungen sind somit bereits sehr hoch und können nicht von kleinsten eingebetteten Geräten erfüllt werden.

Die Matrixoperationen selbst, mit welchen die Werte verrechnet werden, fordern hingegen die CPU oder andere Recheneinheiten, sodass hier eine ausreichende Rechenperformance vorhanden sein muss, um die Ausgaben der neuronalen Netze in der gewünschten Frequenz zu berechnen. Während genaue Werte von dem verwendeten Netz abhängen, so lässt sich eine Vermutung für eine Größenordnung aufstellen. Bei einem neuronalen Netz mit mehreren Millionen Parametern, wie es zuvor angenommen wurde, muss jeder Parameter in mindestens einer Multiplikations- oder Additionsoperation verwendet werden. Dies kann für beliebige neuronale Netze als untere Grenze für die Anzahl der Operationen angenommen werden, nicht nur für Convolutional Neural Networks. Im Falle von CNNs ist diese Schätzung sogar deutlich zu gering, da die Gewichte eines einzelnen Convolutional-Filters für alle Pixel seines Eingabebildes wiederverwendet werden. Hier ist zu vermuten, dass Parameter, welche Teil der Faltungsoperationen sind, mehrfach verwendet werden und hunderte bis tausende Male verwendet werden. Dabei hängt die genaue Anzahl von der Größe des Eingabebildes in die Convolutional-Schicht ab, sodass ein Faktor zwischen 100 und 1000 als eine konservative Schätzung angenommen werden kann. Somit ergibt sich die Anforderung, dass pro verarbeitetem Bild mindestens mehrere hundert Millionen bis wenige Milliarden Rechenoperationen notwendig sind. Für die Hardware bedeutet dies, dass sie ein Vielfaches dieser Operationen pro Sekunde (für eine Verarbeitung in Echtzeit) oder zumindest diese Menge an Operationen innerhalb eines vertretbaren Zeitraumes, beispielsweise weniger Sekunden, durchführen können sollte.

Basierend auf den gestellten Vermutungen hinsichtlich der Komplexität einer Personenerkennung kann erwartet werden, dass eingebettete Systeme für eine Personenerkennung ausreichend Festwertspeicher zum Speichern von Modellen mit mehreren Millionen Parametern, Arbeitsspeicher zum Halten der entstehenden Zwischenergebnisse und eine Recheneinheit für eine zeitnahe Berechnung der Ausgaben benötigen. Ausgehend von diesen Anforderungen können nun reale eingebettete Systeme identifiziert werden, welche diese abstrakten Anforderungen erfüllen und durch ihre Eckdaten quantifizierbar machen. Auf dem Markt der eingebetteten Systeme existieren hierbei sehr unterschiedliche Geräte, welche für eine Personenerkennung geeignet sein könnten. Diese lassen sich grob in die Klassen der Mikrocontroller und der Einplatinencomputer einteilen. Es ist zu beachten, dass diese Einteilung nicht eindeutig und die Abgrenzung beider Klassen unscharf ist.

Mikrocontroller stellen Geräte dar, welche alle grundlegenden Bausteine eines Computers wie CPU, Speicher und Peripherie auf einem Chip integrieren [69, S. 222]. Sie sind somit vollwertige Computer, wobei ihre Leistungsfähigkeit durch die Integration aller Bestandteile auf einem Chip möglicherweise beschränkt ist. Einplatinencomputer, engl. Single-board computer (SBCs), sind Mikrocontrollern ähnlich, integrieren alle Bestandteile jedoch fest verbaut auf einer Platine statt einem Chip, wobei sie sich von Mikrocontrollern durch eine höhere Leistung unterscheiden [34, S. 2].

Ein zusätzlicher Unterschied zwischen verschiedenen eingebetteten Geräten ist durch das Vorhandensein von Hardwarebeschleunigung vorhanden. Eine CPU ist als allgemeine Recheneinheit gebaut. Sie ermöglicht eine Vielzahl an Berechnungen, ist aber durch diese universelle Verwendbarkeit wenig spezialisiert. Hardwarebeschleuniger stellen Koprozessoren dar, welche durch eine stärkere Spezialisierung auf einen Anwendungsfall höhere Leistungen auf diesem erzielen können. Für Grafikaufgaben existieren Hardwarebeschleuniger in Form von Grafikkarten, während zur Beschleunigung der Berechnungen neuronaler Netze sogenannte "Neural Processing Units" (NPUs) existieren. Dieser zusätzliche Prozessor ist auf die typischen Berechnungen neuronaler Netze optimiert und ermöglicht für bestimmte Operationen eine parallele Berechnung mehrerer Ergebnisse [32, S. 1, S. 12]. Eine solche NPU kann sowohl in Mikrocontrollern als auch in Einplatinencomputern verbaut sein.

Basierend auf den identifizierten Anforderungen an Hardwaregeräte zur Personenzählung kann eine Auswahl mehrerer Geräte vorgenommen werden. Hierbei sollen die gewählten Geräte basierend auf ihren Eckdaten in der Lage sein, neuronale Netze in der illustrierten Komplexität auszuführen. Zusätzlich zu den identifizierten theoretischen Aspekten sollen die Gesamtkosten des Systems minimal gehalten werden. Für die im Rahmen des Forschungsprojektes dieser Arbeit angestrebte Umsetzung in ein Produkt war ein Kostenlimit von 100 Euro pro Gesamtsystem gegeben, sodass Kamera und Rechenhardware zusammen diesen Betrag nicht überschreiten sollten. Zumal die Kamerahardware in dieser Arbeit nicht betrachtet wird, sollen

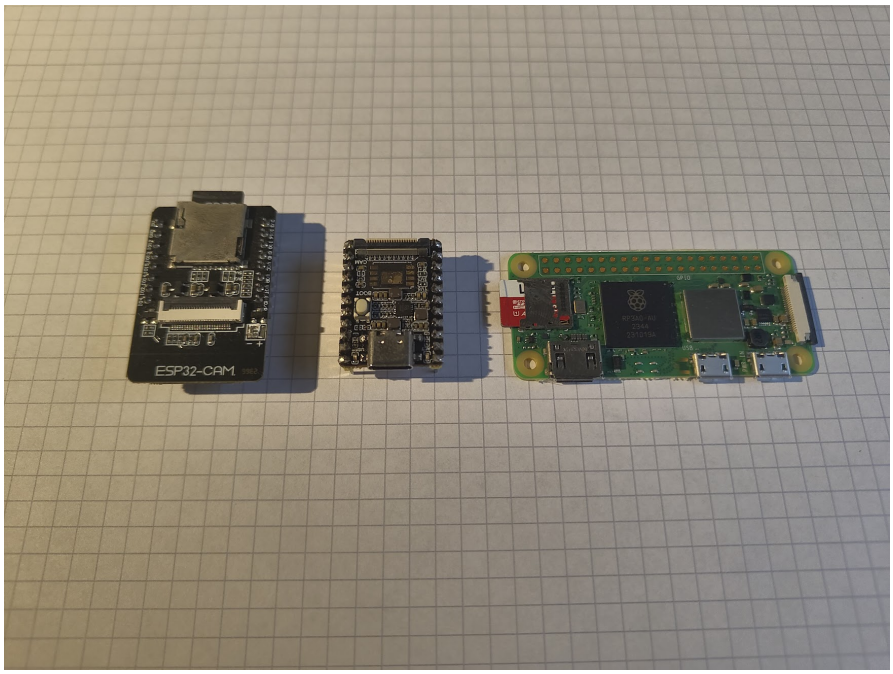


Abbildung 5.1: Die in dieser Arbeit gewählten eingebetteten Geräte. Von links: ESP32-CAM, Luckfox Pico Mini A, Raspberry Pi Zero 2 W.

die gewählten Hardwaregeräte deutlich günstiger als das gegebene Limit von 100 Euro sein.

Um die Matrixoperationen bei der Ausführung neuronaler Netze umsetzen zu können, ist ein Mikrocontroller mit ausreichend großem Arbeitsspeicher vonnöten. Der Speicher muss mindestens ausreichen, um die Matrizen zu halten, welche als Zwischenergebnisse anfallen. Die Größe dieser Matrizen ist abhängig von der Größe des zu verarbeitenden Bildes, kleinere Bilder erzeugen kleinere Zwischenergebnisse und umgekehrt. Es ist denkbar, dass eine ausreichend performante Netzarchitektur, angewandt auf Bildern mit einer geringen, aber noch erkennbaren Bildauflösung, mindestens mehrere Hundert Kilobyte bis wenige Megabyte an RAM benötigt. Mikrocontroller, welche diese Anforderungen sind selten. Ein Mikrocontroller, der die Anforderungen erfüllt, ist ein ESP32-CAM (in Abbildung 5.1 links). Er ist Teil der ESP32-Familie an Mikrocontrollern, welche vom chinesischen Unternehmen Espressif gefertigt werden. Er verfügt über eine 32-Bit-CPU mit 240 MHz Taktung und Unterstützung für Gleitkommaberechnungen und verfügt mit vier MB RAM und vier MB Flash-Speicher über ausreichend Ressourcen, um die Minimalanforderungen zur Ausführung neuronaler Netze zur Bilderkennung auszuführen.

Als ein Vertreter der Einplatinencomputer wurde sich für einen Raspberry Pi Zero 2 W entschieden. Es handelt sich um die zweite Generation der Raspberry-Pi-Zero-Reihe, welche eine Baureihe kleinerer und schwächerer Geräte, verglichen mit der populären Raspberry-Pi-Reihe darstellt. Der Raspberry Pi Zero 2 W (in Abbildung 5.1 rechts) verfügt über eine ARM-basierte 64-Bit-CPU mit vier Kernen bei einer Taktfrequenz von einem GHz

	ESP32-CAM	Raspberry Pi Zero 2 W	Luckfox Pico Mini A
Typ	Mikrocontroller	Einplatinencomputer	Einplatinencomputer
Hardware-beschleunigung	Keine	Keine	NPU mit 0,5 TOPS
CPU	ESP32-S: 32-Bit, 2 Kerne @ 160 MHz	Broadcom BCM2710A (Arm Cortex-A53): 64-Bit, 4 Kerne @ 1 GHz	RV1103 (ARM Cortex-A7): 32-Bit, 1 Kern @ 1,2 GHz
RAM	520 KB SRAM (Intern) + 4 MB PSRAM (Extern)	512 MB LPDDR2	64 MB DDR2
Festwertspeicher	4 MB Flash	microSD-Karte	microSD-Karte
Kosten	≈ 15 Euro	≈ 20 Euro	≈ 10 Euro

Tabelle 5.1: Eckdaten der Hardwaregeräte.

und 512 MB Arbeitsspeicher [43]. Der Festwertspeicher wird über eine SD-Karte realisiert [43]. Hinsichtlich Arbeits- und Festwertspeicher übertrifft der Raspberry Pi Zero 2 W (im Folgenden als "Pi Zero" abgekürzt) die vermuteten Hardwareanforderungen zur Ausführung neuronaler Netze deutlich, verfügt jedoch über keine NPU zur Hardwarebeschleunigung.

Neben dem Pi Zero wird mit dem Luckfox Pico Mini A (in Abbildung 5.1 mittig) ein weiterer Einplatinencomputer evaluiert. Neben dem Modell Pico Mini A existieren weitere Modelle der Luckfox-Pico-Familie, welche in dieser Arbeit nicht betrachtet werden. Da keine weitere Variante der Produktreihe verwendet wird, wird das Gerät im Folgenden als "Luckfox Pico" abgekürzt. Es handelt sich um einen Einplatinencomputer, welcher mit einer schwächeren 64-Bit-CPU und 64 MB RAM deutlich beschränkter ist als der Pi Zero. Zum permanenten Speichern von Daten bietet das Gerät ebenfalls einen Slot für eine SD-Karte [44] an. Der Luckfox Pico besitzt, im Gegensatz zum Pi Zero, mit einer NPU dedizierte Hardware zur Beschleunigung von ML-Aufgaben, welche bis zu 500 Milliarden Operationen pro Sekunde (= 0,5 TOPS) durchführen kann.

Tabelle 5.1 fasst die Unterschiede der drei Hardwaregeräte zusammen. Hierbei sind die Angaben aus den Datenblättern bzw. den Webseiten der Herstellern zu ESP32 [16, 17], Pi Zero [43] und Luckfox Pico [44, 55] entnommen. Es ist zu erkennen, dass der Preis der Geräte die gesetzte Kostengrenze von 100 Euro deutlich unterschreitet. Für die weiteren Untersuchungen können somit die Unterschiede zwischen Mikrocontrollern und Einplatinencomputern untersucht werden. Mit dem Pi Zero und dem Luckfox Pico sind jeweils ein Gerät mit und ohne NPU vorhanden. Somit kann untersucht werden, welche Unterschiede in der Performance durch diese Hardwarebeschleuniger entstehen.

5.2 AUSWAHL GEEIGNETER NETZARCHITEKTUREN

Neuronale Netze zur Objektdetektion existieren in einer enormen Zahl und mit großen Unterschieden zwischen den jeweiligen Netzen. Wie bereits im Rahmen der Grundlagen eingeführt, existieren verschiedene Ansätze darin, wie Objekte in Bildern lokalisiert und schließlich klassifiziert werden. Ei-

Modell	Art	Größe Eingabebild (Pixel)	Anzahl Parameter (Millionen)	Laufzeit pro Bild
R-CNN (AlexNet-Backbone) [23]	Zweistufig	227×227	≈ 60 [38]	≈ 53 s (CPU) / ≈ 15 s (GPU) [23, S. 3]
Fast R-CNN [22]	Zweistufig	224×224	≈ 18 [68]	0,1 s bis 0,32 s (GPU) [22, S. 6]
Faster R-CNN [54]	Zweistufig	224×224	≈ 34 [68]	0,2 s [54, S. 11]
YOLOv1 [53]	Einstufig	224×224	$\approx 60{,}16$ [73]	$\approx 0{,}2$s [53, S. 6]
SSD (VGG16-Backbone) [42]	Einstufig	300×300	≈ 35 [8]	$\approx 0{,}02$ s [42, S. 15]
SSD (MobileNetV3-Backbone [29]), genutzt von [58]	Einstufig	224×224	Nicht angegeben, min. $\approx 3{,}22$ für Backbone [29, S. 7]	Nicht angegeben
SSD (InceptionV2-Backbone [60]), genutzt von [45]	Einstufig	300×300	Nicht angegeben, min. $\approx 6{,}7$ [60, S. 6, Tab. 1] für Backbone	Nicht angegeben
YOLOv5 [33]	Einstufig	640×640 (vortrainiert), sonst variabel	$\approx 1{,}9$ bis $\approx 86{,}7$ (vgl. Tabelle 5.3)	0,045 s bis 0,766 s (CPU) / 6,3 ms bis 12,1 ms (GPU) [33]

Tabelle 5.2: Eckdaten für verschiedene Objektdetektoren aus anderen Arbeiten. Nicht für alle Modelle waren in den originalen Veröffentlichungen Werte für die Anzahl der Parameter gegeben, sodass diese aus Quellen wie dem Detectron2 Model Zoo [68] aus Modelldateien geschätzt werden mussten.

ne erschöpfende Auflistung von Netzarchitekturen zur Objektdetektion wäre deshalb nicht möglich. Durch die zuvor festgestellten Einschränkungen, welche durch kleine eingebettete Systeme entstehen, bilden sich besondere Anforderungen an die Netze, welche auf den Systemen überhaupt zum Einsatz kommen können. Den größten Einfluss auf die Verwendbarkeit eines gegebenen neuronalen Netzes auf eingebetteten Systemen besitzt die hinter dem Netz stehende Architektur. Eine komplexe Architektur mit vielen Schichten führt zu einem größeren Netz als eine kleinere Architektur und benötigt mehr Rechenschritte, liefert aber meist hochwertigere Ergebnisse. Aus den Eckdaten der drei in dieser Arbeit verwendeten Hardwaregeräte ist ersichtlich, dass die Netze einerseits durch ihre Parameter nicht mehr Speicherplatz beanspruchen dürfen als auf einem Gerät verfügbar ist. Andererseits darf die Speicherung der Zwischenergebnisse der durch das Netz fließenden Daten die Menge des verfügbaren Arbeitsspeichers nicht übersteigen. In diesem Abschnitt soll durch eine kurze Literaturrecherche und Vergleiche der Anforderungen mit existieren Arbeiten ausgearbeitet werden, welche Netzarchitekturen für einen Einsatz auf den ausgewählten Systemen geeignet wären.

Um einen Vergleich zwischen verschiedenen Modellen zu erhalten, bietet es sich an, zuerst etablierte, bekannte Netzarchitekturen zu untersuchen. Eine Auflistung moderner Objektdetektoren wird von Zhao et al. [72] in ihrer Arbeit vorgenommen. Sie identifizieren hierbei eine Reihe an Modellen zur Objektdetektion, welche sowohl die ersten auf neuronalen Netzen basierenden Modelle als auch neuere Verfahren beinhalten. Die ersten vier Elemente in Tabelle 5.2 stellen einen Auszug ihrer Auflistungen dar. Es handelt sich hierbei um drei Varianten des R-CNN (kurz für "Regions with CNN features" [23, S. 1]), eines der ersten Verfahren zur Objektdetektion, welches sich

in Teilen auf CNNs stützte. Das ursprüngliche R-CNN nutzt einen als Selective Search bekannten Algorithmus, welcher Bildregionen basierend auf vordefinierten Eigenschaften gruppiert, um eine Ausgabe an möglichen Regionen mit Objekten zu generieren [63]. Diese Vorschläge werden als Eingabe in ein CNN gegeben, welches eine Reihe an Features extrahiert, welche anschliessend von einer Support Vector Machine klassifiziert werden [23, S. 3]. Die ursprüngliche Version des R-CNN benötigt somit neben einem CNN weitere Verfahren, welche auf eingebetteten Geräten sehr wahrscheinlich nur schlecht oder gar nicht lauffähig wären. Zusätzlich ist die Anzahl der Parameter im eingesetzten CNN bereits sehr hoch, sodass sich dieses Modell nicht für eingebettete Geräte eignet. Die enorme Laufzeit auf leistungsstarker Hardware wie Grafikkarten impliziert eine nochmals um mehrere Größenordnungen höhere Laufzeit auf eingebetteten Systemen, sodass ein Einsatz dieses Modells ausgeschlossen werden kann. Fast R-CNN und Faster R-CNN stellen Weiterentwicklungen dieses zweistufigen Ansatzes dar, welche Optimierungen der Modellarchitektur vornehmen [22, S. 2] oder Teile der bisher nicht durch CNNs durchgeführten Berechnungen auf solche auslagern [54, S. 3]. Sie verringern durch diese Änderungen, in Tabelle 5.2 erkennbar, den Rechenbedarf, besitzen aber weiterhin eine hohe Parameteranzahl, sodass diese Verfahren nicht auf Geräten mit geringen Speicherressourcen eingesetzt werden könnten.

Eine weitere von Zhao et al. aufgeführte Netzarchitektur stellt die bereits um Grundlagenteil vorgestellte YOLO-Architektur dar. Für das ursprüngliche YOLOv1 ist ersichtlich, dass die Laufzeiten des Modells ähnlich gering sind wie die des Faster R-CNN-Modells, aber die Parameteranzahl ebenfalls zu groß für ein eingebettetes System wäre. SSD (Single Shot MultiBox Detector) stellt ein zur originalen YOLO-Architektur ähnliches Verfahren dar, welches ebenfalls mit einem Netz gleichzeitig Lokalisation und Klassifikation vornimmt. Zur YOLOv1-Architektur unterscheidet sich SSD dadurch, dass es ein VGG16-Backbone verwendet, aus welchem mehrere Convolutional-Layer zur Detektion genutzt werden, wodurch Objekte unterschiedlicher Größe besser erkannt werden sollen [42, S. 4]. Während SSD nochmals geringere Laufzeiten pro Bild erzielt, ist die Anzahl der Parameter erneut hinderlich für einen Einsatz auf eingebetteten Systemen.

Durch Betrachtung dieser aufgeführten Objektdetektoren kann erkannt werden, dass Modelle, welche nicht speziell auf eingebettete Systeme angepasst sind, nicht auf Geräten mit geringem Arbeits- und Festwertspeicher eingesetzt werden könnten. Die in der Abgrenzung dieser Arbeit erwähnten Werke nutzen Modelle, welche auf diesen Kontext angepasst sind. Die Zählung Wartender vor Aufzügen durch Shen und Chu [58] nutzt einen Single-Shot-Detector, welcher auf MobileNetV3 als Backbone aufbaut. MobileNet bildet eine bei Google entwickelte Familie an Backbones, welche auf Smartphones zugeschnitten ist und sich dadurch auf eine geringe Parameteranzahl und benötigte Rechenleistung beschränkt [29, S. 2]. Simone Luetto nutzt in seiner Zählung von Personen in einem Vorlesungssaal ebenfalls einen Single-Shot-Detector in Kombination mit InceptionV2 [60] als Back-

bone. Dieses Backbone, ebenfalls bei Google entwickelt, reduziert durch eine Kombination verschieden großer Convolutions in einem sogenannten Inception-Modul [60, S. 4-5] die Parameteranzahl auf ein ähnliches Niveau wie die vorherigen Netze. Für diese Modelle ist in Tabelle 5.2 erkennbar, dass die Anzahl der Parameter des Netzes auf wenige Millionen reduziert wird. Diese Netze könnten somit auf eingebetteten Geräten Platz finden, wobei bei Laufzeit der Netze aufgrund fehlender Angaben nicht abgeschätzt werden kann. Erkennbar ist jedoch, dass alle bisher aufgeführten Netze eine feste Bildgröße als Eingabe erwarten.

Eine Netzarchitektur, welche nicht in den aufgeführten Arbeiten eingesetzt wurde, ist die YOLOv5-Architektur. Es handelt sich um eine der YOLO-Familie zuzuordnende Netzarchitektur, welche eine Weiterentwicklung des ursprünglichen YOLO-Konzepts ist, aber von Glenn Jocher und dem Unternehmen Ultralytics veröffentlicht wurde. Hierbei ist zu beachten, dass die Nummerierung im Namen YOLOv5 keine weitere Implikation hinsichtlich der Leistung besitzt und nur zur Unterscheidung der Werke unterschiedlicher Autoren dient. YOLOv5 nähert sich in seinem Aufbau der SSD-Architektur an. Es verwendet eine weiterentwickelte Version des ursprünglichen YOLOv1-Backbones, extrahiert aus diesem aber mehrere Feature Maps verschiedener Auflösungen, um durch mehrere Heads Objekte auf unterschiedlichen Skalen erkennen zu können [64]. Gleichzeitig wird durch Nutzung eines Necks ein Zwischenschritt zwischen Extraktion von Features durch das Backbone und Interpretation dieser Features durch den Head hinzugefügt. Dieses Neck realisiert ein Spatial Pyramid Pooling, eine Schicht, mit der Feature Maps beliebiger Größe auf eine feste Größe abgebildet werden können [25]. Durch diese zusätzliche Schicht können beliebige Bildgrößen in ein YOLOv5-Modell eingegeben werden, durch die interne Reskalierung der extrahierten Feature Maps kann das gleiche Modell für Bilder variabler Auflösung verwendet werden. Es entsteht ein Vorteil gegenüber den vorgestellten SSD-Varianten, welche allesamt eine feste Bildgröße erwarten. Auch existieren mehrere unterschiedliche dimensionierte Varianten der Architektur (vgl. Tabelle 5.3), wobei die kleinste Version namens YOLOv5_n durch eine stark reduzierte Parameteranzahl für einen Einsatz auf eingebetteten Geräten geeignet sein könnte.

Es ist zu beachten, dass Tabelle 5.2 keine Angaben zum RAM-Bedarf bei Ausführung der Netze macht. Der Bedarf an Arbeitsspeicher hängt sowohl von der verwendeten Laufzeit und von der Netzarchitektur selbst ab, kann also nicht allgemeingültig aus der Parameteranzahl berechnet werden. Es kann jedoch angenommen werden, dass im schlechtesten Falle (bei einer naiven Implementierung) alle Netzparameter gleichzeitig im Arbeitsspeicher gehalten werden. In diesem Falle hängt die Parameteranzahl mit der Menge an benötigtem RAM zusammen, je nach Quantisierung wird also die gleiche Menge (int8) oder das Vierfache der Parameteranzahl (float32) als Bytes an Arbeitsspeicher benötigt. Obwohl es sich bei den vorgestellten Netzen nur um einen Auszug handelt, fällt hierbei auf, dass es sich bei den auf eingebetteten Geräten einsetzbaren Modellen in allen Fällen um einstufige

Modell	Bildgröße (Pixel)	Anzahl Parameter (Millionen)	Größe (MiB)	Gleitkommazahl-Operationen (Milliarden / Sekunde)
YOLOv5$_n$ (Nano)	640 × 640	1,9	7,6	4,5
YOLOv5$_s$ (Small)	640 × 640	7,2	28,8	16,5
YOLOv5$_m$ (Medium)	640 × 640	21,2	84,8	49,0
YOLOv5$_l$ (Large)	640 × 640	46,5	186	109,1
YOLOv5$_x$ (Extra large)	640 × 640	86,7	346,8	205,7

Tabelle 5.3: Varianten vortrainierer Versionen von YOLOv5. Adaptiert aus dem YOLOv5-GitHub-Repository [33]. Die Angaben beziehen sich auf vortrainierte Modelle; die Bildgröße bezeichnet die Maße der Trainingsbilder und definiert nicht eine zur Inferenz erwartete Größe.

Objektdetektoren handelt. Es kann an dieser Stelle vermutet werden, dass zweistufige Verfahren durch den Einsatz mehrerer CNNs oder eine Kombination eines CNNs mit anderen statistischen Modellen durch die dadurch hohen Anforderungen an Hardware sehr wahrscheinlich allgemein nicht für sehr leistungsschwache Geräte eingesetzt werden können.

Nach Betrachtung dieses Auszugs an Modellen wird die YOLOv5-Architektur (konkret die YOLOv5$_n$-Variante) als Basis der weiteren Untersuchungen ausgewählt. Begründet wird die Wahl damit, dass die YOLO- und SSD-Familie sich in modernen Iterationen sehr ähneln. Ähnliche Arbeiten nutzten SSD als Basis, variierten hierbei aber das Backbone, um eine geringe Parameteranzahl zu ermöglichen. Motiviert durch die Bedenken zur Privatsphäre, sollen bei der Umsetzung einer Personenzählung Bilder mit einer kleinstmöglichsten Auflösung genutzt werden. Die Nutzung von YOLOv5 ermöglicht es, mit jeweils einem Netz verschiedene Bildgrößen zu evaluieren, ohne dieses aufwändig für jede dieser Auflösungen neu zu trainieren. Modelle der SDD-Architektur müssten durch ihre fixe Eingabegröße für jede Auflösung neu trainiert werden, YOLOv5-Varianten könnten einmal trainiert und auf mehreren Auflösungen evaluiert werden. Ebenfalls stellt YOLOv5 neben der eigentlichen Modelle auch ein Framework [1] zum Training eigener Netze zur Verfügung, welches im Rahmen der Implementierung eigener Netzvarianten genutzt werden kann.

Insgesamt konnte mit diesem Abschnitt gezeigt werden, dass neuronale Netze zur Objektdetektion nur mit Anpassungen auf leistungsschwachen Geräten eingesetzt werden können. Netze, welche nicht speziell auf die besonders geringen Mengen an Arbeitsspeicher, Festwertspeicher und Rechenleistung dieser Hardware zugeschnitten sind, übersteigen in diesen Aspekten die Menge der verfügbaren Ressourcen. Als angepasste Netzarchitekturen konnten Versionen des Single Shot Detectors und der YOLO-Architektur dargestellt werden. Zumal die YOLO-Architektur und die SSD-Architektur sich in neueren Ausführungen annähern, wurde sich unter Betrachtung der Vorteile dieser Architektur für weitere Untersuchungen für YOLOv5 entschieden.

1 https://github.com/ultralytics/yolov5

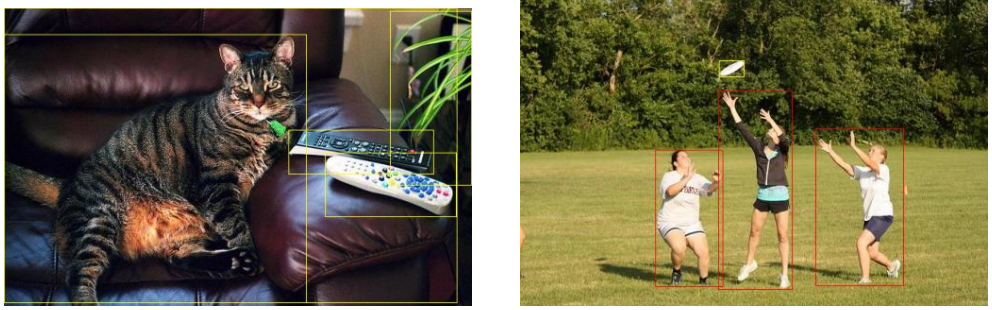
5.3 DATENSÄTZE ZUR PERSONENERKENNUNG

Zum Training eigener Netze zur Personenerkennung müssen Trainingsdaten mit ausreichendem Umfang und hoher Varianz genutzt werden, um einen Objektdetektor zu erhalten, welcher auf neue Situationen und Perspektiven verallgemeinern kann. Wie bereits im Rahmen der Abgrenzung zu existierender Forschung illustriert, existieren Datensätze auf verwandten Themengebieten, welche Personen abbilden. Sie enthalten aber keine repräsentativen Bilder für eine Zählung durch eine an der Decke eines Raumes montierte Kamera. Basierend auf einer weiteren Literaturrecherche konnten nur wenige Datensätze aufgefunden werden, welche sowohl öffentlich waren als auch ausreichend viele Bilder enthielten. Für das folgende Training der neuronalen Netze bedeutet dies, dass nicht-repräsentative Daten genutzt werden müssen. Unter diesem Gesichtspunkt wurden mehrere Datensätze ausgewählt, welche sich hinsichtlich Größe und Repräsentativität der Problemstellung unterscheiden. Nach einem Training der neuronalen Netze kann mit dieser Auswahl bestimmt werden, ob ein Training auf einem "fachfremden" Datensatz in einem Objektdetektor resultieren kann, der sich trotzdem auf die ungewöhnliche Perspektive einer deckenmontierten Kamera übertragen ließe. Im Folgenden sollen diese Datensätze kurz beschrieben werden.

Microsoft COCO

Der Microsoft-COCO-Datensatz (Microsoft Common Objects in Context, kurz MS-COCO oder nur COCO) ist ein von Lin et. al [41] für Microsoft Research zusammengestellter Datensatz, welcher Bilder verschiedener Objekte in unterschiedlichen, realistischen Umgebungen enthält. COCO existiert in verschiedenen Versionen, der in der originalen Arbeit [41] beschriebene Stand stammt aus dem Jahr 2015 und unterscheidet sich leicht von dem für diese Arbeit genutztem Stand. In der ursprünglich beschriebenen Form enthält COCO 91 verschiedene Kategorien, denen Objekte angehören können. Beispiele sind hierbei "bicycle", "toothbrush" oder "person". Spätere Versionen, auch die aus dem Jahr 2017 stammende und in dieser Ausarbeitung genutzte Version, umfassen nur noch 80 Klassen [41, S. 7].

COCO ist nicht der erste Datensatz, in welchem eine Klasse für Personen existiert. Andere Datensätze waren zum Zeitpunkt der Veröffentlichung des Datensatzes bereits verfügbar, unterschieden sich aber hinsichtlich des Kontextes der jeweiligen Bilder. Die Autoren führen als Motivation für das Erstellen des Datensatzes auf, dass andere Datensätze die abgebildeten Objekte oftmals aus typischen Blickwinkeln und vor ähnlichen Hintergründen darstellten [41, S. 1]. Ein Beispiel dafür sei, dass Objekte der Kategorie "bike" (Fahrrad) in anderen Datensätzen oftmals in einer Seitenansicht abgebildet wurden [41, S. 1]. Diese typische Darstellung entspricht der Art, wie Fahrräder üblicherweise auf Produktbildern dargestellt werden, ist aber nicht die einzige Perspektive, in welcher Fahrräder im Alltag sichtbar sind. Das Fehlen von nicht-typischen Darstellungen von Objekten wird als Mangel exis-



(a) Bild ohne Personen.

(b) Bild mit Personen.

Abbildung 5.2: Beispielbilder aus dem COCO-Datensatz. Die Bounding Boxen sind visualisiert, hierbei sind Objekte der Klasse “person” rot, andere Objektklassen gelb umrandet.

tierender Datensätze aufgeführt, von den Autoren auf Bildern des COCO-Datensatzes trainierte Modelle wiesen eine bessere Performance auf als auf vorhergegangenen Datensätzen [41, S. 1, S. 9]. COCO enthält für jedes Bild mehrere Arten an Labels, welche verschiedenen Zwecken dienen. Neben Bounding Boxes für alle relevanten Objekte sind zusätzlich pixelgenaue Masken pro Objekt vorhanden, auch fünf schriftliche Beschreibungen des Bildinhaltes sind pro Bild enthalten [41, S. 4–6]. Im Rahmen dieser Arbeit wurden nur die Bounding Boxes genutzt, da diese für eine Objekterkennung geeignet sind. Abbildung 5.2 zeigt zwei Bilder aus dem COCO-Datensatz. Es ist erkennbar, dass sowohl Bilder mit als auch ohne Personen enthalten sind.

Für das Training eines Modells zur Personenzählung kann ein Großteil der Bilder entfernt werden, welche keine Personen enthalten. Von insgesamt 118.287 Bildern enthalten 64.115 (54,2 %) mindestens eine Person, diese Bilder können für ein Training als Teil eines neuen Datensatzes beibehalten werden. Um Negativbeispiele zu erhalten, wurden 10% dieser Anzahl, also 6.411 Bilder ohne Personen als Negativbeispiele zufällig aus den verbleibenden Bildern ohne Personen in den neu entstandenen Datensatz eingefügt. Diese Untermenge des COCO-Datensatzes wurde für folgende Untersuchungen und Trainings verwendet und wird im weiteren Verlauf als “COCO-Subset” bezeichnet.

Die Verteilung der Anzahl Personen pro Bild ist für das COCO-Subset in Grafik 5.3 oben abgebildet. Der zur Evaluation genutzte Teil des Datensatzes enthält in den meisten Bildern eine Person, wobei Situationen mit mehr Personen abnehmend seltener werden. Auffällig ist, dass es mehr Bilder mit 13 Personen gibt als nach diesem Muster zu erwarten wäre. Dies lässt sich damit begründen, dass Bilder mit mehr als 10 bis 15 Instanzen einer Objektklasse ab diesem Punkt als “Crowd” markiert wurden, wobei die restlichen Instanzen nicht mehr einzeln mit Bounding Boxen, sondern mit einer Maske markiert wurden [41, S. 4]. Hierbei scheint das Limit auf dem Evaluationsanteil zufällig 13 Personen zu betragen (vgl. Anhang A.1 für den Trainingsanteil).

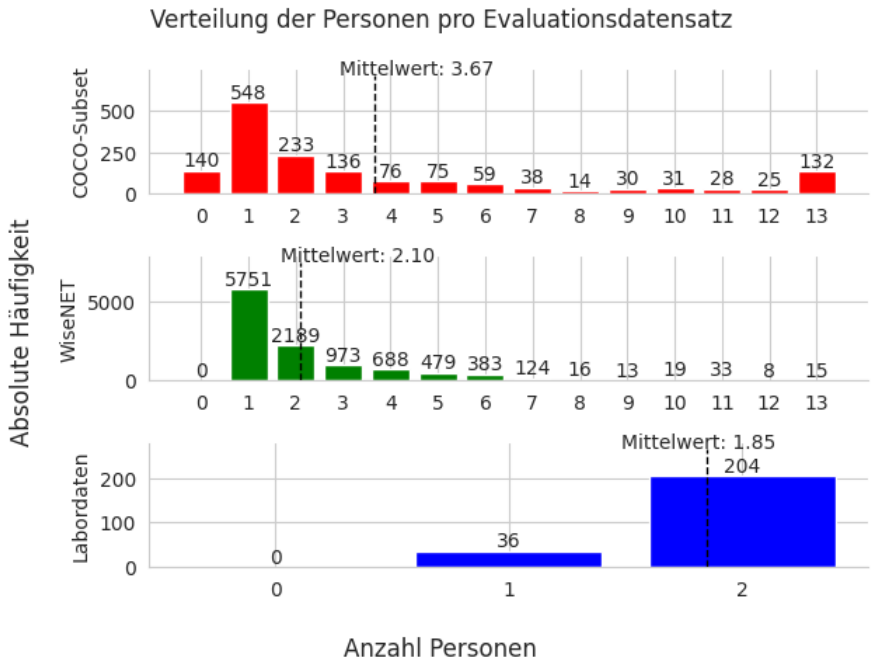


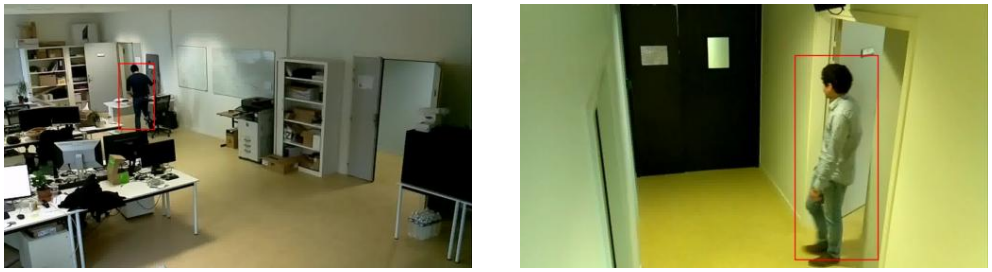
Abbildung 5.3: Verteilung der Anzahl Personen pro Bild in allen drei Evaluationsdatensätzen.

Der COCO-Datensatz und damit auch das COCO-Subset stellen einen sehr populären Benchmark-Datensatz zum Vergleich von Objektdetektionsmodellen dar. Dies ist einerseits dem enormen Umfang geschuldet, andererseits ist die beschriebene Vielfalt der abgebildeten Situationen sehr realitätsnah, wodurch der Datensatz als repräsentativ für allgemeine Objektdetektionssituationen angesehen werden kann. Im Kontext der angestrebten Personenzählung ist der Datensatz eventuell weniger geeignet, da Personen aus sehr vielen Perspektiven abgebildet werden. Eine Personenzählung mit einer deckenmontierten Kamera wird meist zu einer Vogelperspektive führen, welche innerhalb des COCO-Datensatzes selten abgebildet ist. Der COCO-Datensatz ist somit sehr umfangreich, aber möglicherweise nicht repräsentativ für die Personenerkennung und -zählung.

WiseNET-Datensatz

Ein Datensatz, welcher repräsentativere Daten enthält, ist der hier als "WiseNET-Datensatz" bezeichnete Datenbestand. Er ist Teil einer von Marroquin et al. im Jahre 2019 veröffentlichen Arbeit [47]. Es handelt sich um eine Sammlung an Videos, welche von nahe der Decke montierten Kameras aufgenommen wurden, welche eine Perspektive ähnlich wie Überwachungskameras besitzen [47, S. 3-4]. Zusätzlich zu den Bounding Boxen sind Kontextinformationen zur Lage der Kameras in dem erfassten Gebäude gegeben [47, S. 1]. Die Autoren motivieren die Sammlung der Daten damit, dass andere Datensätze diese Informationen nicht besitzen, diese aber für Aufgaben wie

Tracking der Laufwege einzelner Personen essenziell sind. Insgesamt sind Frames aus Videos mit einer Gesamtlänge von ungefähr einer Stunde in dem Datensatz vorhanden, welche mehrere Räume und Flure erfassen, wobei zwei bis 15 unterschiedliche Personen pro Video erfasst wurden [47, S. 4]. Abbildung 5.4 zeigt zwei Bilder aus den Videos mit eingezeichneten Labels.



(a) Bild aus einem Büroraum.

(b) Bild aus einem Flur.

Abbildung 5.4: Beispielbilder aus dem WiseNET-Datensatz, mit Kameras in verschiedenen Räumen. Bounding Boxen von Personen sind rot umrandet.

Der WiseNET-Datensatz besitzt eine moderate Größe, durch Extraktion der einzelnen Frames aus den Videos und Aufteilen der Bilder in einen Trainings- und Validierungsdatensatz können 42.764 bzw. 10.691 Bilder erhalten werden. Die Anzahl dieser Bilder ist vergleichbar mit dem COCO-Datensatz, wobei die Perspektive ebenfalls nicht ganz einer Vogelperspektive entspricht. Die Perspektive der Aufnahmen entspricht verglichen mit dem COCO-Datensatz eher der angestrebten Perspektive, wobei der WiseNET-Datensatz trotz einiger Variationen weniger Beleuchtungssituationen und Szenen abbildet. Die Verteilung der Personenanzahl ist in Grafik 5.3 in der mittleren Zeile zu erkennen. Im Vergleich zum COCO-Subset enthält der WiseNET-Datensatz keine Negativbeispiele und im Mittel eine geringere Anzahl von Personen pro Bild.

Im Rahmen dieser Arbeit ist der Datensatz von Interesse, da er einen auf die Personenerkennung spezialisierten Datensatz darstellt. Es kann an diesen Daten untersucht werden, inwiefern sich die auf diesem Datensatz trainierten Modelle von solchen unterscheiden, welche auf einem vielfältigeren, aber weniger repräsentativen Datensatz wie dem COCO-Subset trainiert wurden.

Laboraufnahmen

Zur Validierung der erhaltenen Modelle zur Erkennung und Zählung von Personen auf Kamerabildern wurde ein Datensatz genutzt, welcher intern im Rahmen des Forschungsprojektes dieser Arbeit erstellt wurde. Der Datensatz enthält 240 Aufnahmen eines Büros durch eine deckenmontierte Kamera, welche durch ihre Fischaugenlinse eine Verzerrung auf dem Bild erzeugt. Es handelt sich um den von Oskar Rudolf in Rahmen seiner Masterarbeit

erstellten und beschriebenen Datensatz [56, S. 61 ff.]. Ein Beispielbild ist in Abbildung 5.5 gegeben.

Abbildung 5.5: Beispielbild aus den Laboraufnahmen. Personen sind rot umrandet.

Bei den Aufnahmen handelt es sich um einen sehr kleinen Datensatz, zumal die 240 Bilder aus einem einzigen Video gewonnen wurden. Aufgrund dessen ist hinsichtlich Licht, Position der Personen und anderen Faktoren keine bzw. nur minimale Variation gegeben. Als Labels sind Bounding Boxes um die Personen gegeben. Es sind meist zwei, seltener eine Person pro Bild vorhanden (vgl. Abbildung 5.3 unten). Negativbeispiele durch Frames ohne Personen sind nicht gegeben.

Die Perspektive entspricht der Perspektive, welche im Rahmen dieser Arbeit für die Zählung angestrebt wurde, die Daten sind also repräsentativer als die der vorhergegangenen Datensätze.

Bedingt durch den geringen Umfang und die geringe Variation ist der Datensatz nicht für ein Training eines Objektdetektors nutzbar. Er kann jedoch im Rahmen der Untersuchungen als Testdatensatz genutzt werden, um zu bewerten, inwiefern sich die auf den vorher erwähnten Datensätzen erlernten Muster auf diese neuen Daten übertragen lassen. Hierbei muss jedoch beachtet werden, dass Aussagen, welche durch Messungen auf den Labordaten erhalten werden, aufgrund der geringen Größe und Varianz des Datensatzes möglicherweise nur sehr geringe Aussagekraft besitzen.

5.4 TRAINING UND EVALUATION DER NEURONALEN NETZE

Als Grundlage der folgenden Experimente wurde die kleinste Variante von YOLOv5 namens YOLOv5$_n$ genutzt. Es handelt sich um die kleinste von Ultralytics bereitgestellte Variante der Netzarchitektur, welche im Kern den gleichen Aufbau wie andere Größen des YOLOv5-Modells nutzt, aber durch eine niedrigere Anzahl Layer und geringere Parameter pro Layer nur ungefähr 1,9 Millionen Parameter besitzt (vgl Tabelle 5.3). Im Gegenzug besitzt das Modell, verglichen mit seinen größeren Varianten, eine geringere Performance auf den zum Training genutzten COCO-Daten. Ausgehend von dem ursprünglichen YOLOv5$_n$-Modell wurden mehrere Varianten erstellt,

Modell	Parameter (Millionen)	GFLOPs	Anzahl Klassen	Anpassung ggü. YOLOv5$_n$
YOLOv5$_n$	$\approx$ 1,86	$\approx$ 4,5	80	Keine.
1class$_n$	$\approx$ 1,76	$\approx$ 4,1	1	Reduktion auf nur die Objektklasse der Personen.
half$_n$	$\approx$ 0,95	$\approx$ 2,1	1	Reduktion von Filtern/Channeln in allen Convolutional-Layern im Backbone des Netzes, Head unverändert.

Tabelle 5.4: Vergleich der eingesetzten und selbst erstellten Varianten von YOLOv5$_n$.

welche mit dem vorgestellen COCO-Subset und dem WiseNET-Datensatz trainiert und evaluiert wurden.

Zur Evaluation wurden einerseits die in den Grundlagen vorgestellten Metriken zur Objektdetektion genutzt. Da das Hauptproblem der Personenzählung zwar auf die Ergebnisse einer Objektdetektion aufbaut, aber nicht zwingend mit ihr gleichzusetzen ist, wird zusätzlich die Zählqualität mit der Metrik des mittleren absoluten Fehlers (engl. Mean Absolute Error, MAE), gemessen zwischen der Anzahl der erfassten Personen und der korrekten Anzahl, bewertet.

Tabelle 5.4 listet die trainierten Varianten von YOLOv5$_n$ auf. Das aufgeführte originale, unveränderte YOLOv5$_n$ wurde als Baseline ausgewählt, um eventuelle Verbesserungen der verbleibenden Modelle durch Fokus auf die Objektkategorie der Personen in Relation zu einem nicht auf Personen spezialisiertem Netz setzen zu können. 1class$_n$ und half$_n$ wurden hingegen für die Untersuchungen jeweils auf dem COCO-Subset oder dem WiseNET-Datensatz komplett neu trainiert. 1class$_n$ erhält durch die Reduktion auf nur eine Klasse bereits eine geringere Parameteranzahl. Hierbei ist jedoch zu beachten, dass die Netzarchitektur des vortrainierten YOLOv5$_n$-Modells aus nicht bekannten Gründen von der offiziellen Architektur für diese YOLO-Variante abwich, sodass die Parameterreduktion durch 1class$_n$ möglicherweise auch durch eine ungewöhnliche Größe der Baseline zu begründen ist. Die half$_n$-Variante wurde unter dem Gesichtspunkt erstellt, dass nicht alle genutzten Datensätze gleich komplexe Sachverhalte darstellen und somit ein einfacheres Netz mit weniger Filtern auch ausreichende Ergebnisse erzielen könnte. Es ist erkennbar, dass die Reduktion der Channel das Netz gegenüber der Baseline um etwa die Hälfte der Parameter verkleinert. Auch erfordern die Varianten weniger Gleitkommazahl-Operationen, um jeweils eine Ausgabe für ein Bild zu berechnen, erkennbar an den geringeren GFLOPs (Giga Floating Point Operations Per Second). Es wurde initial ein Versuch unternommen, die Netzarchitektur von YOLOv5$_n$ mit einem alternativen SqueezeNet-Backbone [30] zu versehen, um die Parameterzahl nochmals weiter zu reduzieren, jedoch wurde dieser Versuch aufgrund mangelnder Performance schließlich verworfen und soll nicht weiter ausgeführt werden.

Für das Training wurden die voreingestellten Hyperparameter des YOLOv5-Frameworks gewählt[2]. Alle Netze wurden auf einer Eingabeauflösung

2 verfügbar unter https://github.com/ultralytics/yolov5/blob/master/data/hyps/hyp.scratch-low.yaml

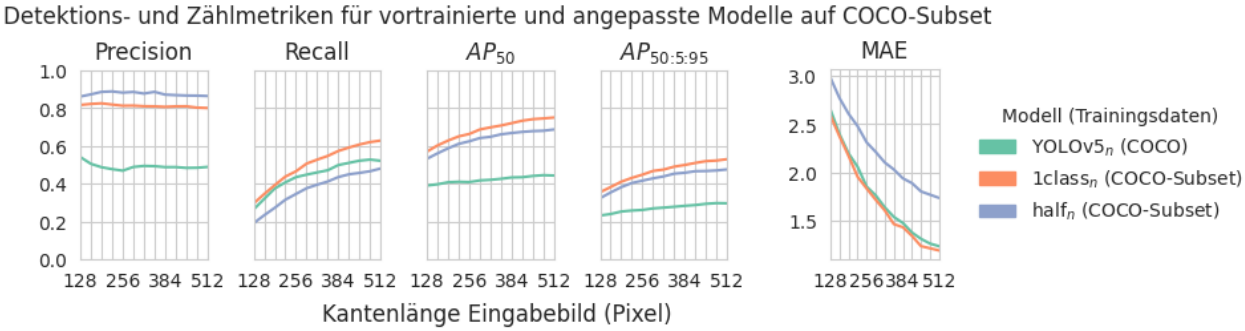


Abbildung 5.6: Vergleich der gemessenen Metriken für ein vortrainiertes YOLOv5$_n$ und die eigens trainierten, auf die Erkennung von Menschen spezialisierten Abwandlungen.

von 640 × 640 Pixeln trainiert. Das Training wurde pro Netz für 75 Epochen durchgeführt, wobei das nach AP_{50} beste Netz als finale Version genutzt wurde (sodass eventuell overfittende Trainingsstände späterer Epochen nicht berücksichtigt wurden). Als Hardware wurde ein Desktop-PC genutzt [3], auf welchem die Netze trainiert wurden. Die Trainingszeiten auf dem COCO-Datensatz schwankten für die Netze zwischen 5 und 10 Stunden, abhängig von der Größe des verwendeten Datensatzes. half$_n$-Modelle trainierten wegen ihrer geringeren Parameteranzahl und somit erforderlichen Rechenleistung schneller als die 1class$_n$-Varianten.

5.4.1 *Vergleich vortrainierter und spezialisierter Netze*

Um zu untersuchen, ob eine Spezialisierung auf die Klasse der Personen zu einer Verbesserung der Detektionsqualität führt, muss die Performance der Netze untersucht werden. Hierfür wurden erst die auf dem COCO-Datensatz trainierten Netze 1class$_n$ und half$_n$ (im Folgenden als "spezialisierte" Netze bezeichnet) mit dem vortrainierten YOLOv5$_n$ verglichen, welches nicht auf die Objektklasse der Personen zugeschnitten ist. Die YOLOv5-Architektur ermöglicht es, zur Inferenz andere Bildgrößen einzusetzen, als original zum Training verwendet wurden. Aus diesem Grund können die Netze auf verschiedenen Auflösungen ausgeführt werden, ohne dass das Netz für jede Auflösung neu trainiert werden muss. Um einen Vergleich der Auswirkungen verschiedener Bildgrößen auf die Detektions- und Zählqualität zu erhalten, wurden alle quadratischen Bildgrößen mit Kantenlänge aus $\{128, 160, 192, \ldots, 480, 512\}$ evaluiert.

Abbildung 5.6 enthält eine Darstellung der gemessenen Werte der Metriken zur Detektion und zur eigentlichen Zählung der Personen. Precision und Recall sind für den Konfidenz-Schwellwert angegeben, welcher die bestmögliche Kombination für beide Werte ergibt[4]. AP_{50} und $AP_{50:5:95}$ entsprechen der Standardberechnung. Der Grafik ist zu entnehmen, dass eine

3 AMD Ryzen 3800X mit 8 Kernen/16 Threads, 96 GB RAM, NVIDIA RTX 3080Ti 12GB
4 Vgl. https://github.com/ultralytics/yolov5/issues/9904#issuecomment-1289543788

höhere Auflösung der Bilder auf den COCO-Daten tendenziell zu höheren Metriken hinsichtlich der Objektdetektion führt, da die meisten Kurven mit höheren Werten auf der x-Achse ansteigen. Es ist erkennbar, dass YOLOv5$_n$ eine geringere Precision als die spezialisierten Modelle besitzt, also öfter Objekte falsch als Menschen klassifiziert. Der Recall ist zwischen allen Varianten ähnlich, wobei das auf eine Klasse beschränkte Modell die meisten Personen richtig erkennt. Die Werte erreichen bei einer Auflösung von 512×512 Pixeln das Maximum, wobei höhere Auflösungen wahrscheinlich weitere, aber verschwindend geringe Verbesserungen der Metrik erzielt hätten. Insgesamt befindet sich das Maximum mit einem Wert von etwa 0,5 auf einem eher niedrigen Niveau, zumal dies bedeutet, dass etwa 50% der tatsächlich abgebildeten Personen im COCO-Subset nicht gefunden werden. Bei AP_{50} und $AP_{50:5:95}$ ähneln sich die Werte der beiden spezialisierten Modelle stark, das Baseline-Modell liegt bei beiden Metriken deutlich darunter, da seine niedrigen Precision-Werte in diese kombinierten Maße mit einfliessen. In Abbildung 5.6 ist in der letzten Spalte zusätzlich zu den Metriken zur Bewertung der Objektdetektion der Zählfehler aufgeführt. Anders als für die Metriken der Objektdetektion gilt hier, dass ein kleinerer Fehler als besser anzusehen ist. Hierbei wurde das Netz mit einem IoU-Schwellwert von 0,45 und einem Konfidenz-Schwellwert von 0,25 ausgeführt und die Anzahl der erkannten Personen pro Bild gezählt. Es lässt sich erkennen, dass die Qualität der Zählung auch mit größeren Bildern ansteigt und eine genauere Zählung erhalten wird. Auffällig ist, dass nur ein geringer Unterschied zwischen dem spezialisierten 1class$_n$-Modell und dem originalen YOLOv5$_n$ besteht, während das half$_n$-Netz konstant höhere Zählfehler erzielt. Der Fehler beträgt auf der maximalen Bildgröße ungefähr 1,25, was auf dem COCO-Subset mit durchschnittlich mehr als drei Personen pro Bild ein hoher, aber eventuell akzeptabler Wert ist. Zwischen der Güte der Objektdetektion und der Güte der Zählung scheint nur ein geringer Zusammenhang zu bestehen.

Diese Untersuchungen wurden nur zwischen dem vortrainierten YOLOv5$_n$-Modell und den selbst erstellten Modellen durchgeführt. Mangels anderer vortrainierter Modelle können keine weiteren Untersuchungen dieser Art durchgeführt werden, sodass die erhaltenen Ergebnisse möglicherweise keine allgemeine Gültigkeit besitzen. Das Neutrainieren spezialisierter Netze scheint in diesem Falle nur eine geringe Auswirkung auf die Zählqualität zu besitzen, wobei die höheren Objektdetektionsmetriken der spezialisierten Modelle keine Auswirkung auf diese zu haben scheinen.

5.4.2 *Messung der Erkennungs- und Zählqualität der Modelle*

Um zu untersuchen, ob das Training auf einem anderen Datensatz als dem COCO-Subset zu einem besser für den Anwendungsfall einer deckenmontierten Kamera und ihrer ungewöhnlichen Perspektive passenden Modell führt, wurden die Netze 1class$_n$ und half$_n$ zusätzlich auf dem WiseNET-Datensatz trainiert. Die Evaluation der auf beiden Datensätzen trainierten Modelle geschah auf den Validierungsdatensätzen der erwähnten Datensät-

ze und zusätzlich auf dem Labordatensatz, welcher aufgrund seiner sehr geringen Größe nicht für das Training genutzt wurde. Im Folgenden wird jede Untersuchung auf jedem Evaluationsdatensatz gesondert betrachtet, eine Kombination aller Graphen ist im Anhang in Abbildung A.2 gegeben.

Messungen auf dem COCO-Subset

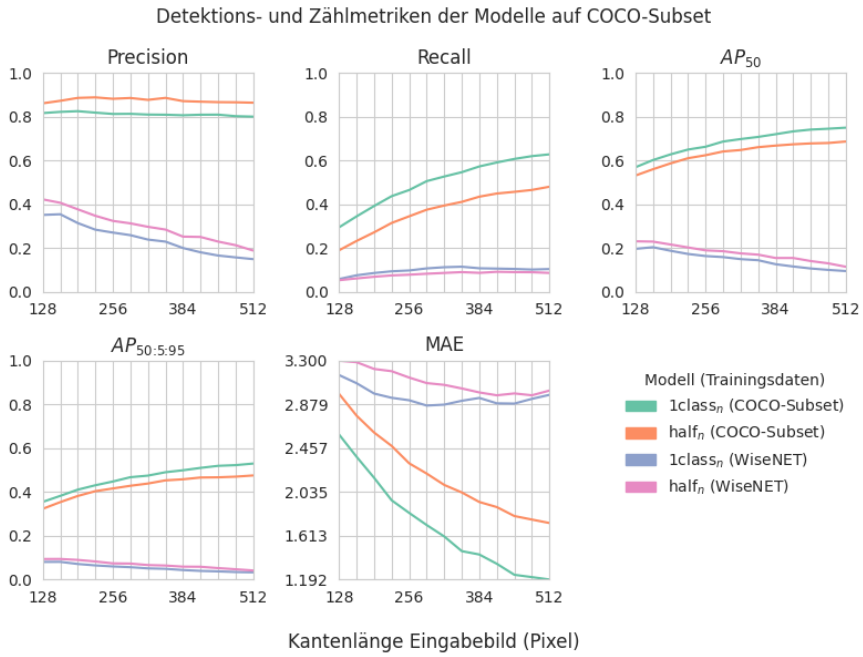


Abbildung 5.7: Metriken aller Modelle auf dem COCO-Subset.

Abbildung 5.7 bildet die Werte der Metriken zur Objektdetektion und Zählung für alle erstellten Modelle auf dem COCO-Subset ab. Jeder Graph bildet die Werte einer Metrik ab, wobei die x-Achse die variierende Bildgröße darstellt. Die Kombination aus Netzarchitektur und Trainingsdatensatz wird durch die Farbe der Linien wiedergegeben.

Es ist erkennbar, dass die auf dem COCO-Subset trainierten Modelle die beste Detektionsperformance auf dem Evaluationsanteil ihres Trainingsdatensatzes erreichen. Größere Bilder führen für manche Metriken zu Verbesserungen, andere Metriken stagnieren jedoch. Die Precision der COCO-Modelle ändert sich für größere Bilder praktisch nicht, während der Recall ansteigt und der Verlauf langsam abflacht. AP_{50} und die verschärfte Version $AP_{50:5:95}$ steigen ebenfalls mit höherer Auflösung an, wobei auf dieser Verlauf flacher wird; die strengere $AP_{50:5:95}$-Metrik erreicht hierbei konstant geringere Werte. Ebenfalls ist zu erkennen, dass ein einfacheres Modell wie half$_n$ in allen Metriken außer der Precision geringer bewertet wird als das komplexere 1class$_n$-Modell. Der Umstand, dass das einfachere Modell präziser ist, lässt sich wahrscheinlich damit erklären, dass es durch seine geringere Lernkapazität nur einfachere Muster erlernt hat, welche zur Erkennung von Personen geeignet sind. Personen, die mit diesen einfachen Mustern

erkannt werden, werden dadurch konsistent erkannt (hohe Precision), aber insgesamt werden mehr Personen nicht erkannt, weil sie nicht durch diese einfachen Muster beschrieben werden können (niedriger Recall). Die Zählperformance der auf dem COCO-Datensatz trainierten Modelle ist hingegen tendenziell mangelhaft. Eine höhere Auflösung führt auch hier zu geringer werdenden Verbesserungen, trotzdem wird im Schnitt mehr als eine Person pro Bild übersehen. Dies ist, anteilig am Durchschnitt von ungefähr 3,5 Personen pro Bild (vgl. Abbildung 5.3) ein vergleichsweiser hoher Fehler. Abbildung 5.8 visualisiert für das Modell 1class$_n$ als bestes Modell auf dem COCO-Subset für jede Anzahl an Personen pro Bild, die Verteilung der vorhergesagten Anzahl. Es ist zu erkennen, dass das Modell für eine geringe Anzahl an anwesenden Personen oftmals genau oder ungefähr die korrekte Anzahl Personen detektiert. Sobald eine größere Anzahl Personen anwesend ist, tendiert das Modell jedoch dazu, weniger Personen als tatsächlich vorhanden vorherzusagen.

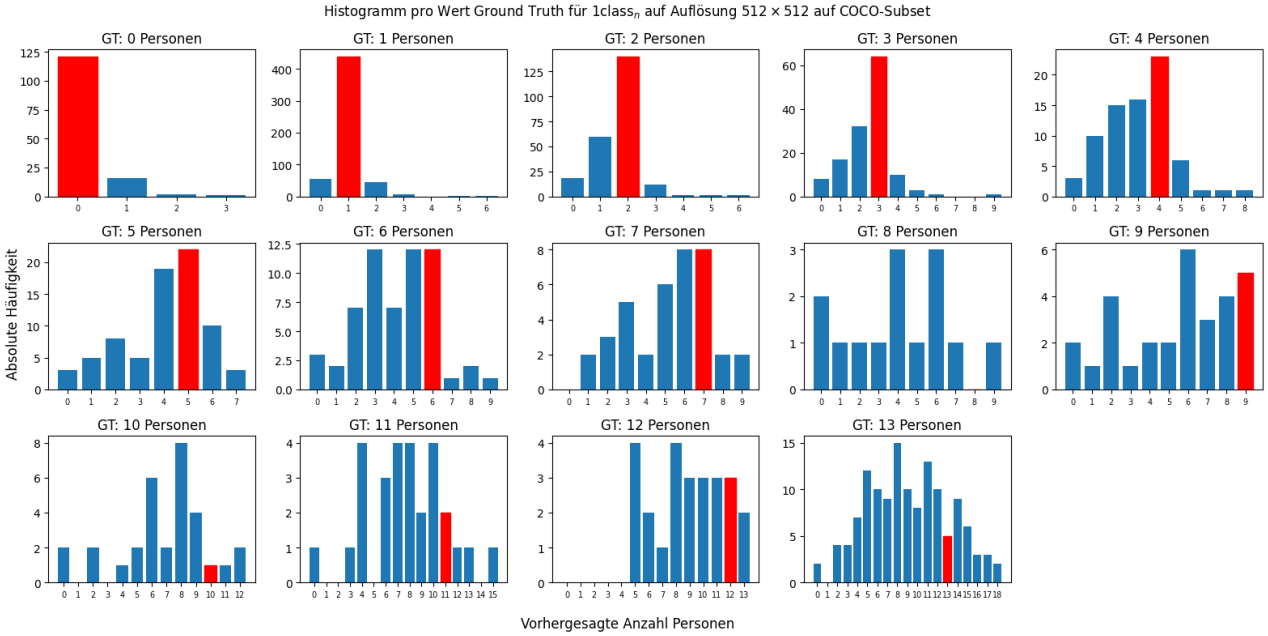


Abbildung 5.8: Histogramme zum Vergleich der vorhergesagten Anzahl von Personen pro Bild mit der echten Anzahl auf dem COCO-Subset. Abgebildet ist die Zählung durch das beste für diesen Datensatz performende Modell. Es ist zu beachten, dass die y-Achsen pro Histogramm unterschiedliche Skalen besitzen, da Bilder mit einer höheren Anzahl Personen seltener vertreten sind. Der rot hinterlegte Balken stellt die korrekte Anzahl in diesem Histogramm dar.

Die in der ersten Zeile erkennbaren Verläufe für die auf dem WiseNET-Datensatz trainierten Modelle belegen eine schlechte Übertragbarkeit des Wissens zwischen den Datensätzen. WiseNET-Modelle verbessern sich auf dem COCO-Datensatz mit höheren Auflösungen nicht, die Precision nimmt hierbei ab und der Recall verbleibt auf einem geringen Niveau. Dies wird auch durch die AP-Metriken wiedergegeben. Der Zählfehler bleibt relativ

konstant bei einer Differenz von drei Personen. Hiermit eigenen sich die auf dem WiseNET-Datensatz trainierten Modelle nicht zu einer Verallgemeinerung auf den größeren, vielfältigeren COCO-Datensatz. Dies entspricht den Erwartungen, da der WiseNET-Datensatz Bilder mit einer sehr ähnlichen Perspektive besitzt, während der COCO-Datensatz Personen in vielen verschiedenen Situationen und Perspektiven enthält.

Messungen auf dem WiseNET-Datensatz

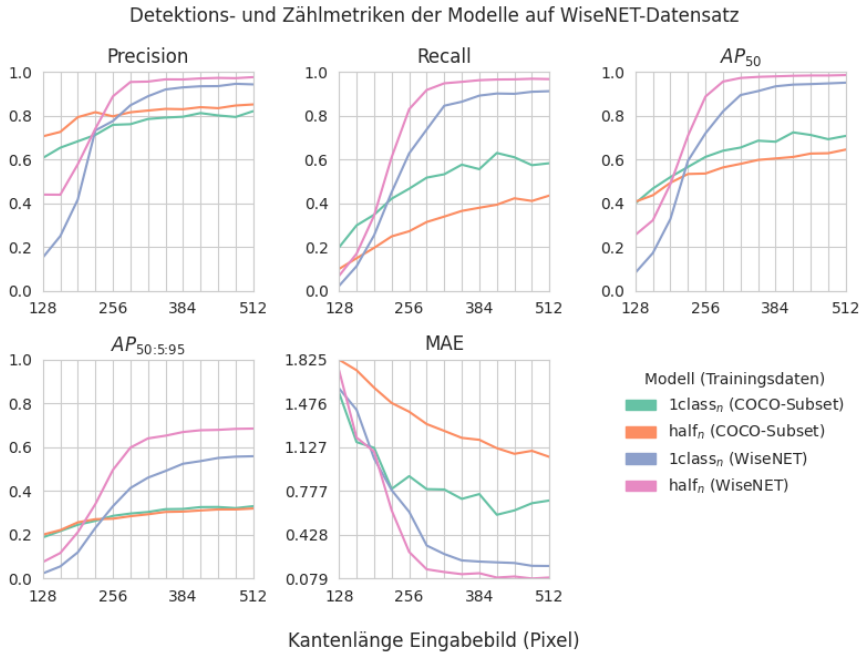


Abbildung 5.9: Metriken aller Modelle auf dem WiseNET-Datensatz.

Abbildung 5.9 betrachtet eine Evaluation auf dem WiseNET-Datensatz. Es ergibt sich ein gegenteiliges Bild, verglichen mit der Evaluation auf dem COCO-Datensatz. Die auf COCO trainierten Modelle erzielen geringere Precision- und Recall-Werte als die auf dem WiseNET-Datensatz trainierten Modelle, jedoch verbessern sie sich mit höherer Auflösung in allen Metriken leicht. Es kann vermutet werden, dass sich darin die Übertragbarkeit der auf dem COCO-Datensatz erlernten Muster auf den WiseNET-Datensatz äußert (während das Gegenteil nicht möglich war). Diese Übertragbarkeit führt auch zu einem geringeren Zählfehler, wobei auch hier ein Fehler von 0,5 bis zu einer Person auf der höchsten evaluierten Auflösung bestehen bleibt. Die auf WiseNET trainierten Modelle erzielen, wie zu erwarten, in diesem Falle sehr gute Werte. Im Gegensatz zur vorherigen Zeile ist zu erkennen, dass die Werte aller Metriken bereits ab einer geringen Auflösung von etwa 256×256 Pixeln drastisch ansteigen (für die Detektionsmetriken) bzw. absinken (für den Zählfehler). Precision und Recall nähern sich ab dieser Auflösung ihrem bestmöglichen Wert, sodass auch der AP_{50}-Wert gegen sein Optimum strebt. Hinsichtlich der genauen Lokalisation scheinen die erzeugten Boxen

nicht bestmöglich zu liegen, sodass der $AP_{50:5:95}$-Wert für die Modelle auf Werten um 50% verbleibt. Der Zählfehler läuft gegen Null, bereits ab der Hälfte der maximalen Auflösung verbessert er sich nicht mehr nennenswert. Besonders fällt auf, dass das kleinere, $half_n$-Modell in allen Belangen bessere Bewertungen erzielt als sein größeres Gegenstück. Dies kann dahingehend interpretiert werden, dass der WiseNET-Datensatz deutlich "einfachere " Daten enthält, welche bereits von einem weniger komplexen Modell erlernt werden können. Während das komplexere Modell durch seine überdimensionierten Kapazitäten zu einem Overfitting neigt, verallgemeinert das kleinere Modell besser auf die Muster des WiseNET-Datensatzes und erzielt somit bessere Ergebnisse. Abbildung 5.10 stellt für dieses Modell die Verteilung der Vorhersagen in Abhängigkeit von der korrekten Anzahl Personen dar. Es ist erkennbar, dass in den meisten Fällen die korrekte Anzahl Personen vorhergesagt wird, die Abweichungen auch bei größeren Menschenmengen geringer sind und diese Fälle auch seltener vorkommen.

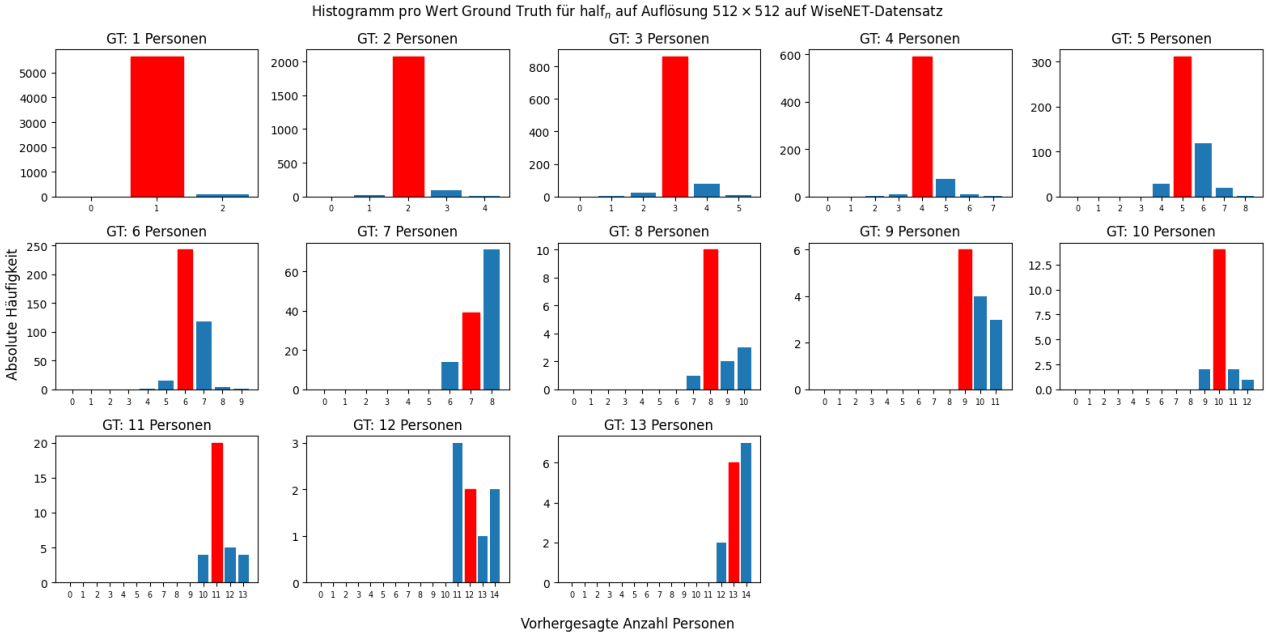


Abbildung 5.10: Histogramme zum Vergleich der vorhergesagten Anzahl von Personen pro Bild mit der echten Anzahl auf dem WiseNET-Datensatz, analog zu Abbildung 5.8. Abgebildet ist die Zählung durch das beste für diesen Datensatz performende Modell.

Durch die Evaluation der Modelle auf dem WiseNET-Datensatz können mehrere Schlüsse gezogen werden. Einerseits ist ersichtlich, dass Datensätze mit einfachen Sachverhalten, wie einer fixen Perspektive, bereits mit einfachen Netzen gut modelliert werden können. Hierbei kann durch die Verwendung eines einfacheren, kleineren Modells sogar eine höhere Performance, sowohl auf die Personenerkennung als auch -zählung bezogen, erreicht werden. Gleichzeitig scheint für einfache Sachverhalte bereits eine geringe Auflösung für die Eingabebilder ausreichend zu sein, um eine sehr gute Performance zu erreichen. Einfache Zählungen können somit mit kleinen Mo-

dellen auf niedrigauflösenden Bildern durchgeführt werden. Beide Faktoren führen zu geringeren Anforderungen an die genutzte Hardware, sodass in solchen Fällen eventuell bereits schwache Geräte wie Mikrocontroller ausreichend sein könnten. Als letzte Erkenntnis lässt sich festhalten, dass das Training auf dem COCO-Datensatz zu Wissen zu führen scheint, welches sich in Teilen auf Bilder mit anderen Perspektiven, wie hier vorhanden, übertragen lässt. Somit erscheint es, als ob die Größe und Varianz eines Datensatzes für diese Personenerkennung eine größere Rolle spielt als die Ähnlichkeit der Bildperspektiven.

Messungen auf dem Labordatensatz

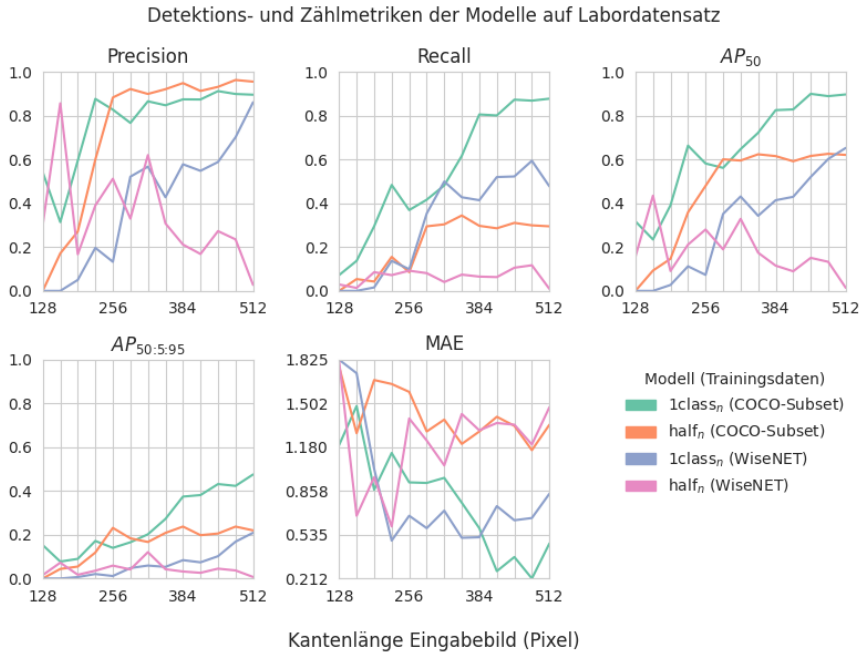


Abbildung 5.11: Metriken aller Modelle auf dem Labordatensatz.

Abbildung 5.11 untersucht die Modelle auf dem Labordatensatz. Es ist unmittelbar erkennbar, dass der Verlauf der Graphen deutlich chaotischer ausgeprägt ist als in den vorherigen Fällen. Der Verlauf der Precision zeigt, dass alle Modelle außer dem auf dem WiseNET-Datensatz trainierten half$_n$-Modell tendenziell mit steigender Auflösung besser bewertet werden und gegen eins streben. Das ausgeschlossene Modell wird mit steigender Auflösung hingegen ungenauer. Hier könnte die Einfachheit des Modells, kombiniert mit der geringen Varianz des Labordatensatzes zu einem Fall führen, in dem die erlernten Muster immer mehr Falschdetektionen erzeugen. Beim Recall verbessern sich die einfacheren Modelle, ungeachtet des Trainingsdatensatzes nur geringfügig. Die komplexeren Modelle profitieren hingegen von der steigenden Auflösung, sie scheinen hierbei auch einen Grossteil der Personen auf den Bildern korrekt zu erkennen. Das auf COCO trainierte 1class$_n$-Modell erzielt hierbei die besseren Werte. Die AP$_{50}$-Metrik weist ebenfalls

ungleichmäßige Verläufe auf, wobei die auf dem COCO-Datensatz trainierten Modelle wieder besser bewertet werden. Die Anpassung der Bounding Boxen wird durch das strengere Maß wieder konsistent schlechter bewertet. Der Zählfehler verschlechtert sich oder sinkt nur minimal für die half_n-Modelle, während die größeren Modelle Minimalwerte von ungefähr 0,25 und 0,5 erreichen. Erneut erzeugt das auf dem COCO-Datensatz trainierte Modell den besten Wert. Die Histogramme in Abbildung 5.12 zeigen einerseits die geringe Größe des Datensatzes, zumal es nur zwei mögliche Werte für die Ground Truth gibt, andererseits ist auch hier bereits eine Tendenz zum Unterschätzen der Anzahl an Personen erkennbar.

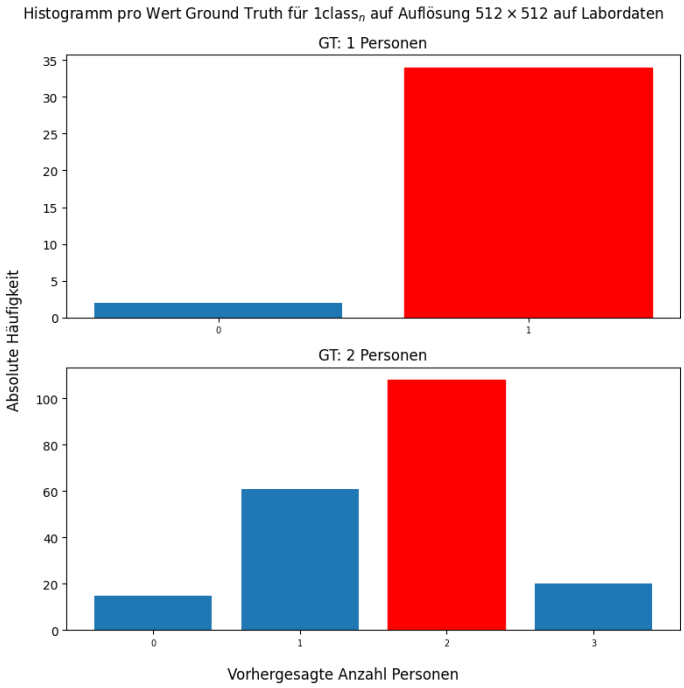


Abbildung 5.12: Histogramme zum Vergleich der vorhergesagten Anzahl von Personen pro Bild mit der echten Anzahl auf dem Labordatensatz, analog zu Abbildung 5.8 und 5.10. Abgebildet ist die Zählung durch das beste für diesen Datensatz performende Modell.

Die Interpretation der Evaluation auf dem Labordatensatz gestaltet sich schwieriger als in den vorherigen Fällen. Da der Datensatz sehr klein ist und nur die Position der Personen im Raum variiert, ist die Aussagekraft erhaltener Aussagen sehr gering. Dies spiegelt sich in den Graphen primär in den chaotischen Verläufen wieder. Die bessere Performance der auf COCO trainierten Modelle impliziert, dass das Training auf einem Datensatz mit hoher Varianz in diesem Falle besser übertragbare Modelle erzeugt. Sowohl der COCO-Datensatz als auch der WiseNET-Datensatz enthalten primär keine Vogelperspektive, wie sie auf den Labordaten auftreten. Es scheint jedoch, als ob die Varianz der COCO-Daten trotzdem zur Extraktion von Mustern innerhalb des Netzes führt, welche in Teilen auf neue bzw. selten im Trainingsdatensatz gesehene Perspektiven übertragbar sind.

Erkenntnisse aus den Messungen

Insgesamt konnten mit diesen Messungen Aussagen zur Übertragbarkeit der erlernten Muster zwischen Datensätzen und zur Qualität der betrachteten Datensätze erhalten werden. Das Training auf einem Datensatz mit vielen unterschiedlichen Bildern, aber geringer Repräsentativität scheint für den Anwendungsfall einer Personenzählung von der Decke besser übertragbar zu sein als die Nutzung eines repräsentativeren, einfacheren Datensatzes. Für einfache Anwendungsfälle wie die Detektion auf dem WiseNET-Datensatz scheinen bereits geringe Auflösungen und einfache Netze gute Ergebnisse zu erzielen, wodurch leistungsschwächere Geräte geeignet sein könnten. Zum Labordatensatz können nur sehr unsichere Aussagen getroffen werden. Die auf dem COCO-Subset trainierten Modelle erzeugen auf ihm einen akzeptabel geringen Zählfehler, ob dies für einen größeren Datensatz mit variierenden Lichtverhältnissen gelten würde, ist jedoch unklar. Indiziert durch die chaotischen Verläufe der Messwerte ist zu vermuten, dass Effekte der geringen Stichprobengröße und der mangelnden Varianz keine allgemeingültige Aussage ermöglichen.

Mit den in diesem Abschnitt durchgeführten Experimenten lässt sich eine Teilfrage der Forschungsfragen beantworten. Es konnte gezeigt werden, dass das Umtrainieren vorhandener, allgemeiner Objektdetektoren zur alleinigen Erkennung von Personen möglich ist. Das Neutrainieren war mit vertretbaren Aufwänden verbunden. Ob ein Training mit öffentlichen, nicht für den Anwendungsfall repräsentativen Datensätzen ausreichend gute Ergebnisse erzielt, konnte aufgrund des kleinen Labordatensatzes nicht beantwortet werden.

5.5 EXPERIMENTE ZUR VERKLEINERUNG NEURONALER NETZE

Mit Quantisierung und Teacher-Student-Training wurden bereits zwei ausgewählte Verfahren zur Verkleinerung neuronaler Netze vorgestellt. Es wurde gezeigt, dass Quantisierung durch die Approximation der Modellparameter durch weniger große Datentypen eine direkte Verkleinerung der Modellgewichte, während Teacher-Student-Training eine indirekte Verkleinerung durch das Verbessern kleinerer Architekturen ermöglichen kann. Beide Methoden wurden bisher nur theoretisch vorgestellt, in diesem Abschnitt sollen beide Methoden auf den erstellten neuronalen Netzen empirisch untersucht werden.

5.5.1 *Vergleich unquantisierter und quantisierter Modelle*

Zum Quantifizieren der Auswirkungen einer Quantisierung auf die genutzten Objektdetektoren wurden diese auf die Stufen float16 und int8 quantisiert. Hierzu wurden die auf dem COCO-Datensatz trainierten Modelle gewählt. Als Baseline wurde das unquantisierte Modell mit Gewichten als float32 belassen, welches den Originalzustand des Modells darstellt. Die ers-

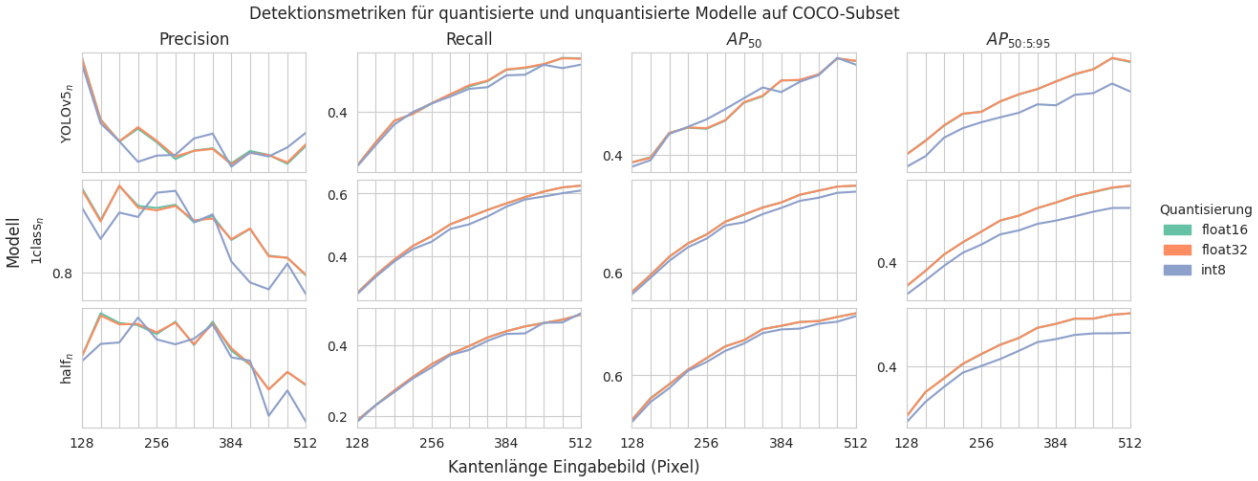


Abbildung 5.13: Vergleich der Objektdetektionsmetriken für die Modelle $YOLOv5_n$, $1class_n$ und $half_n$ auf dem COCO-Subset in verschiedenen Quantisierungsstufen. Zwischen float32 und float16 bestehen solch geringe Unterschiede, sodass die float16-Werte von den float32-Werten farblich komplett verdeckt werden.

te Quantisierung wurde auf float16 vorgenommen, wobei es sich weiterhin um Gleitkommazahlen handelt. Als zweite Quantisierungsstufe wurde eine Quantisierung auf int8 vorgenommen. Bei allen Quantisierungen wurde Post-Training-Quantization angewandt, sodass die Quantisierung nach dem Training und nicht während des Trainings durchgeführt wurde. Der Prozess der Quantisierung dauerte hierbei pro Modell nur wenige Sekunden.

Die erhaltenen quantisierten Modelle wurden auf dem Evaluationsanteil des COCO-Subsets evaluiert und die Metriken Precision, Recall, AP_{50} und $AP_{50:5:95}$ erfasst. Abbildung 5.13 zeichnet die Ergebnisse aller Objektdetektionsmetriken der drei Modelle $YOLOv5_n$, $1class_n$ und $half_n$ auf. Im Gegensatz zu vorherigen Graphen dieser Art sind die y-Achsen unabhängig pro Kästchen und teilen sich keine Skala. Im Graphen ist erkennbar, dass alle Modelle auf allen Quantisierungsstufen sehr ähnliche Werte erzielen. Die Metriken von auf float16 quantisierten Modellen unterscheiden sich nur so geringfügig von den unquantisierten Modellen, sodass sich beide Werte auf dieser Auflösung nicht mehr auseinanderhalten lassen. Zwischen den beiden float-Formaten und int8-Quantisierungen besteht ein kleiner Unterschied hinsichtlich der meisten Metriken. Dieser Unterschied ist nicht immer konstant, sodass die int8-Modelle in wenigen Fällen bessere Werte erzielen als die originalen Modelle. Alleine bezogen auf die Metrik $AP_{50:5:95}$ sind int8-quantisierte Modelle konstant minimal schlechter als float-Modelle.

Tabelle 5.5 listet für das Modell $YOLOv5_n$ (in Abbildung 5.13 oben) einen Ausschnitt der Daten auf. In tabellarischer Form ist erkennbar, dass zwischen den Werten der Quantisierungsstufen nur minimale Unterschiede bestehen. Die Differenzen wachsen mit steigender Kantenlänge des Eingabebildes an, selbst bei Bildern mit einer Größe von 512×512 Pixeln betragen die Unterschiede weniger als zwei Prozentpunkte für Recall und weniger

Größe Eingabebild (Pixel)	Quantisierung	Precision	Recall	AP_{50}	$AP_{5:5:95}$	Größe Modelldatei (Bytes)
128×128	float32	0,5454	0,2649	0,3964	0,2353	7.534.345
128×128	float16	0,5458	0,2646	0,3964	0,2353	3.791.403
128×128	int8	0,5416	0,2630	0,3943	0,2270	2.098.312
160×160	float32	0,5054	0,3233	0,3985	0,2440	7.543.417
160×160	float16	0,5060	0,3235	0,3988	0,2444	3.795.939
160×160	int8	0,5033	0,3164	0,3974	0,2340	2.098.696
...	...	...	...	...	...	...
512×512	float32	0,4900	0,5315	0,4439	0,2970	7.776.281
512×512	float16	0,4889	0,5327	0,4437	0,2966	3.912.371
512×512	int8	0,4974	0,5170	0,4420	0,2769	2.108.392

Tabelle 5.5: Auszug einiger Werte des Vergleiches in Abbildung 5.13.

als einen Prozentpunkt für die Average Precision. Ebenfalls sind die Dateigrößen der resultierenden Dateien in der Tabelle angegeben. Die in der Praxis gemessene Größe der Datei mit den Modellgewichten entspricht ungefähr den theoretischen Überlegungen. float16-Modelle sind etwa halb so groß wie die originalen float32-Modelle, während int8-Modelle ungefähr ein Viertel des Platzes belegen.

Nach Betrachtung der erfassten Metriken kann bestätigt werden, dass Quantisierung im Falle der erstellten Netze zur Objekterkennung ein wichtiges Werkzeug zum Reduzieren der Größe der Modellgewichte eines Netzes ist. Es lassen sich, bezogen auf die Metriken der eigentlichen Objekterkennung nur minimale Verschlechterungen der Modelle feststellen. Diese Unterschiede zwischen quantisierten und unquantisierten Modellen sind minimal, es handelt sich dabei um Bruchteile eines Prozentpunkts bis hin zu etwa zwei Prozentpunkten. Der Reduktion des Speicherbedarfs für die Modellgewichte entspricht nicht ganz den theoretischen Erwartungen, sodass eine Quantisierung auf float16 etwas weniger als die Hälfte der originalen Dateigröße, auf int8 etwas weniger als drei Viertel an Einsparungen mit sich bringt.

5.5.2 *Auswirkungen Teacher-Student-Training*

Eine Implementierung von Teacher-Student-Training ist, wie zuvor illustriert, für Objektdetektoren nicht trivial zu implementieren. Es muss dabei nicht nur eine Implementierung des komplizierten Verfahrens für Objektdetektoren allgemein als auch für das spezifische genutzte Modell erfolgen. Für YOLOv5 existiert eine Open-Source-Implementierung[5], welche verwendet wurde.

Um die Auswirkungen eines Teacher-Student-Trainings zu untersuchen, wurde mit 1class_m ein auf der YOLOv5_m-Variante (vgl. Tabelle 5.3) basierendes Teacher-Netz auf dem COCO-Subset trainiert. Dieses wurde auf der gleichen Hardware wie in vorherigen Versuchen für 75 Epochen bei einer Auflö-

5 https://github.com/wonbeomjang/yolov5-knowledge-distillation

Modell	Precision	Recall	AP_{50}	$AP_{50:5:95}$
$1class_n$	0,775	0,613	0,706	0,417
$1class_m$	0,821	0,686	0,79	0,533

Tabelle 5.6: Metriken des genutzten Teacher-Modells in Relation zu einem kleineren Modell.

sung von 640×640 Pixeln und mit aktiviertem Early Stopping bei 3 Epochen ohne Verbesserung trainiert. Das Training wurde dadurch nach 70 Epochen bei einer Laufzeit von 11,7 Stunden abgebrochen. Das $1class_m$-Modell erreichte in allen Bewertungsmetriken leichte Verbesserungen gegenüber dem $1class_n$-Modell auf dem COCO-Subset, die Werte sind in Tabelle 5.6 ersichtlich.

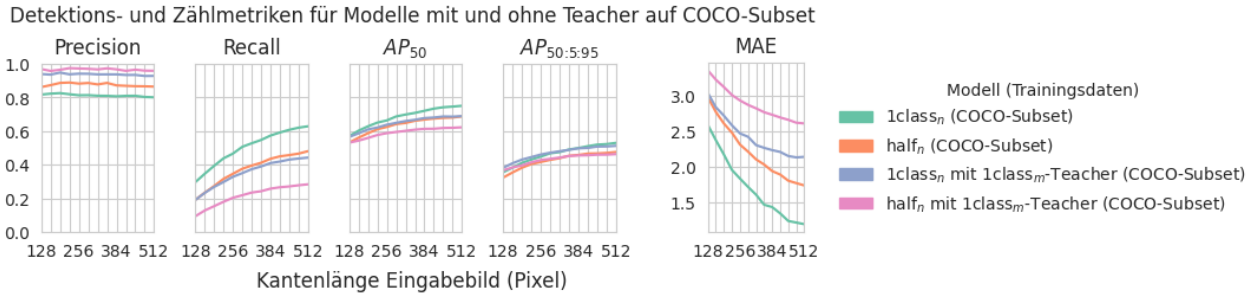


Abbildung 5.14: Metriken der auf dem COCO-Subset trainierten Modelle $1class_n$ und $half_n$ verglichen mit ihren mit einem $1class_m$ als Teacher trainierten Gegenstücken.

Mit dem $1class_m$-Modell als Teacher wurden zwei Netze als Students trainiert: ein $1class_n$-Modell und ein $half_n$-Modell. Diese wurden anschliessend, analog zu den vorher trainierten Modellen hinsichtlich ihrer Objektdetektions- und Zählungsmetriken verglichen. Das Training und die Evaluation fanden beide auf dem COCO-Subset, respektive dem Trainings- und Validations-Anteil statt. Abbildung 5.14 bildet die gemessenen Metriken der vier Modelle ab. Das Training mit einem Teacher-Modell erzeugte Modelle, welche über eine höhere Precision verfügen, aber gleichzeitig einen geringeren Recall besitzen. Es gilt weiterhin, dass das weniger komplexe $half_n$-Modell dem komplexeren $1class_n$-Modell in den meisten Fällen unterlegen ist, was sowohl für direkt trainierte als auch per Knowledge Distillation erhaltene Modelle gilt. Hinsichtlich der AP-Metriken scheint das Training in keinen bemerkbaren Unterschiede gegenüber den ohne Teacher trainierten Modellen zu resultieren. Der Zählfehler ist bei den Student-Modellen höher als bei den direkt trainierten Modellen.

Es ist somit zu vermuten, dass ein Teacher-Student-Training in dem Falle der Personenzählung nur Auswirkungen auf den Objektdetektions-Anteil besitzt. Mit Teacher trainierte Modelle besaßen in diesem Experiment eine leicht höhere Precision, wobei im Gegenzug eine deutlich geringerer Recall erzielt wurde. Die Metriken AP_{50} und $AP_{50:5:95}$ zur Bewertung der Lage der

Boxen verblieben jedoch ähnlich, sie spiegeln also anscheinend diese Veränderungen nur geringfügig wieder. Die Zählperformance wird durch die geringere Precision stark beeinflusst, sodass die per Teacher-Student-Training erzeugten Modelle im Rahmen der Zählung keine Verbesserungen gegenüber einem direkten Training ohne Teacher-Modell bieten.

5.6 ERKENNTNISSE DES KONZEPTS

Im Rahmen des Konzepts konnten mehrere Erkenntnisse erlangt werden, welche sowohl für die folgende Implementierung als auch als generelle Ergebnisse dieser Arbeit von Bedeutung sind.

Zu den Herausforderungen des Ausführens neuronaler Netze zur Personenerkennung auf eingebetteter Hardware konnten theoretische Überlegungen aufgestellt und an echter Hardware verdeutlicht werden. Es konnte festgehalten werden, dass komplexe Aufgaben der Bildverarbeitung sowohl durch das Zwischenspeichern von unkomprimierten Bildern bereits mehrere Hundert Kilobyte an Arbeitsspeicher benötigen, was bereits viele schwache eingebettete Systeme mit wenig RAM ausschließt. Da die Pixelanzahl eines Bildes als Produkt von Länge und Höhe (und einer konstanter Anzahl Farbkanäle) schnell ansteigen kann, sollten die Eingabebilder möglichst klein gehalten werden. Die Parameter eines neuronalen Netzes fordern sowohl Arbeits- als auch Festwertspeicher, sodass ein System ausreichenden Platz bieten muss, um die Parameter während der Inferenz zu halten. Die Berechnungen der Ausgaben benötigt bei einer Aufgabe wie der Objektdetektion mehrere Milliarden Rechenschritte, sodass die Hardware diese Berechnungen innerhalb einer für den Anwendungsfall angemessenen Zeitspanne bewältigen muss. Als Hardwaregeräte kommen mit diesen Rahmenbedingungen für diese Arbeit starke Mikrocontroller sowie Einplatinencomputer mit und ohne Hardwarebeschleunigung in Frage. Die Besonderheiten im Kontext eingebetteter Systeme bestehen somit durch deren geringe Speicherkapazität und Rechenleistung, welche direkt im Konflikt mit den hohen Anforderungen neuronaler Netze an beide Ressourcen stehen.

Netzarchitekturen zur Objekterkennung fokussierten sich primär nicht auf eingebettete Geräte und besitzen somit meist zu viele Parameter, um im Speicher schwacher Geräte Platz zu finden. Damit besteht eine Notwendigkeit, speziell auf eingebettete bzw. schwache Geräte angepasste Modelle zu verwenden. Als zwei solche Architekturen konnten Varianten des Single Shot Detectors (SSD) und des You-Only-Look-Once-Konzepts (YOLO) identifiziert werden. Beide Netzfamilien sind als einstufige Objektdetektoren sehr ähnlich zueinander, sodass sich für diese Arbeit aufgrund der variablen Größe der Eingabebilder für die YOLOv5-Architektur entschieden wurde.

Hinsichtlich der Datensätze zum Training eines Objektdetektors für Personen war ersichtlich, dass öffentliche Datensätze mit einer passenden Vogelperspektive und ausreichender Größe rar sind. Ausweichend mussten Datensätze mit nicht-repräsentativen Perspektiven genutzt werden, welche sich im Umfang stark unterschieden. Bedingt durch diese Abweichungen wurde die

Frage geäußert, inwiefern ein Training auf diesen Datensätzen trotzdem Objektdetektoren ergeben kann, welche Personen aus einer Vogelperspektive erkennen können.

Das Training eigener Varianten des YOLOv5-Netzes konnte mit vertretbarem zeitlichem Aufwand auf Consumer-Hardware durchgeführt werden. Die Erstellung von Netzen, welche auf Personen spezialisiert sind, ging mit erkennbaren Verbesserungen der Objektdetektionsqualität, aber nur mit minimalen Verbesserungen der Zählqualität einher. Basierend auf Vergleichen mehrerer auf unterschiedlichen Datensätzen trainierter Modelle konnte erkannt werden, dass sich die Muster, die auf verschiedenen Daten zur Erkennung von Personen genutzt werden, nicht zwingend auf neue Daten übertragen werden können. Modelle, welche auf dem Trainingsanteil eines Datensatzes trainiert wurden, erzielten auch auf seinem Evaluationsanteil die besten Ergebnisse. Für komplexe Situationen und Perspektiven wie im COCO-Datensatz, führten höhere Auflösungen bis zum gemessenen Maximum von 512×512 Pixeln stetig zu Verbesserungen. Auf dem einfacheren WiseNET-Datensatz waren bereits kleine Bilder ausreichend, um eine gute Personenzählung zu erreichen. Die Modelle tendierten bei komplexen Daten zum Unterschätzen der Anzahl der Personen auf einem Bild, wodurch ein Teil des Zählfehlers erklärt werden konnte. Beim Anwenden der auf COCO und dem WiseNET-Datensatz trainierten Modelle auf die gesammelten Labordaten ergab sich, dass die auf dem COCO-Datensatz trainierten Modelle auf diesen bessere Ergebnisse erzielten. Durch die geringe Stichprobengröße und mangelnde Varianz blieb jedoch unklar, ob dies generell bedeutet, dass ein Training eines Objektdetektors auf einem fachfremden, aber großen Datensatz ausreichend ist und das Sammeln eines repräsentativen Trainingsdatensatzes nicht nötig ist. Auch unklar blieb, ob es sich bei einer Personenzählung aus der Perspektive einer deckenmontierten Kamera um ein "einfaches" Problem wie die Zählung auf dem WiseNET-Datensatz handelt oder um ein komplexes Problem wie die Zählung auf dem COCO-Datensatz. Trotz dieser offenen Fragen konnte gezeigt werden, dass das Training eigener neuronaler Netze zur Detektion von Personen sinnvoll ist, um eine Personenzählung umzusetzen.

Im letzten Teil des Konzepts wurden Quantisierung und Teacher-Student-Training auf die erstellen Netze angewandt und ihre Auswirkungen betrachtet. Für die Quantisierung konnte auf den genutzten Modellen der Speicherbedarf der Netzparameter stark gesenkt werden, für int8-Quantisierung entsprechend der Erwartungen auf etwa ein Viertel des ursprünglichen Platzbedarfs. Die Messungen der Objektdetektionsmetriken ergab leichte Performanceverluste, welche sich jedoch auf wenige Prozent beliefen und somit keine bedeutenden Auswirkungen auf die Zählperformance haben sollten. Mit Teacher-Student-Training wurden YOLOv5$_n$-Netze mit einem größeren Teacher-Netz erstellt. Die erhaltenen Netze wiesen Unterschiede hinsichtlich ihrer Objektdetektionsmetriken auf, welche schlussendlich keine Verbesserung der Zählqualität ermöglichten. Somit konnte hier nur Quantisierung als nützliche Technik zur Anpassung dieser neuronalen Netze auf die Be-

sonderheiten eingebetteter Systeme belegt werden, Teacher-Student-Training ermöglichte es hierbei nicht, kleinere performante Objektdetektoren aus größeren Modellen zu erhalten.

Insgesamt konnten in diesem Konzept somit Antworten auf alle Teilfragen der Forschungsfrage gefunden werden. Die Implementierung auf echter Hardware und die damit verbundenen Aspekte der Forschungsfrage steht jedoch noch aus und folgt im nächsten Kapitel.

6 IMPLEMENTIERUNG DER PERSONENZÄHLUNG

Nachdem in den vorhergehenden Abschnitten ein Konzept zur Implementierung einer Personenzählung auf eingebetteten Geräten erarbeitet und Teillösungen untersucht wurden, müssen diese auf echter Hardware zusammengeführt werden. Während die Erkenntnisse des vorherigen Kapitels die aus der Hauptforschungsfrage entstandenen Teilfragen beantworten, dient dieses Kapitel der Beantwortung der Forschungsfrage an sich.

6.1 IMPLEMENTIERUNG IN VERSCHIEDENEN FRAMEWORKS

Wie bereits mehrfach illustriert, bieten Geräte wie Mikrocontroller und andere eingebettete Systeme besondere Herausforderungen an Software, welche auf ihnen laufen soll. Einerseits sind die Ressourcen dieser Systeme stark eingeschränkt, es steht nicht viel RAM oder Festwertspeicher zur Verfügung. Andererseits bieten die Geräte auch nicht alle Funktionalitäten an, welche von stärkeren Geräten erwartet werden könnte, beispielsweise können Mikrocontroller nicht immer nativ mit 32-Bit-Zahlen arbeiten. Es bedarf somit spezialisierter Software in der Form von Laufzeitumgebungen, welche neuronale Netze unter Beachtung dieser Besonderheiten ausführen können. In den kommenden Abschnitten sollen die erstellten neuronalen Netze auf den drei Hardwaregeräten ausgeführt werden. Hierfür werden mehrere, unterschiedliche Laufzeitumgebungen genutzt, welche in diesem Abschnitt vorgestellt werden. Mit dieser Auswahl an Laufzeitumgebungen lässt sich ein Teil der Forschungsfrage beantworten, indem illustriert wird, welche Frameworks eine Ausführung neuronaler Netze auf eingebetteten Geräten ermöglichen.

Für den ESP32 existiert kein Framework, welches speziell auf dieses Gerät zugeschnitten ist. Stattdessen gibt es für eine weite Reihe an Mikrocontrollern die Laufzeitumgebung TensorFlow Lite Micro (TFLM). TensorFlow Lite Micro ist eine Variante von TensorFlow Lite (TFL), was seinerseits ein Framework zum Ausführen von Machine-Learning-Modellen wie neuronalen Netzen auf mobilen Geräten ist. Es ist zu beachten, dass sowohl TensorFlow Lite als auch TensorFlow Lite Micro im Laufe des Erstellens dieser Arbeit zu "LiteRT" und "LiteRT for Microcontrollers" umbenannt wurden [74]. In dieser Arbeit werden beide Frameworks weiterhin als "TensorFlow Lite" und "TensorFlow Lite Micro" bezeichnet, zumal die referenzierte Literatur diese Namen verwendet. TFL und TFLM verwenden Modelle, welche in dem für diese Frameworks geschaffenen "FlatBuffer"-Format vorliegen [10, S. 6]. Die Modelle werden von einem Interpreter eingelesen, welcher die einzelnen Schritte (z. B. die Convolutional-Layer eines CNNs) hintereinander ausführt, um die Ausgaben des Modells zu erhalten. TFLM unterscheidet sich

von TFL darin, dass es eine nochmals im Umfang reduzierte Version ist, welche auf Mikrocontroller abzielt. Ein Unterschied zwischen TensorFlow Lite und TensorFlow Lite Micro besteht unter anderem darin, dass TensorFlow Lite Micro nur unquantisierte oder auf int8 quantisierte Modelle unterstützt, zumal Microcontroller üblicherweise keine native Unterstützung für Rechnungen mit Gleitkommazahlen besitzen[1]. Ein weiterer Bereich, in dem Anpassungen gegenüber TensorFlow Lite erfolgt sind, stellt die Speicherverwaltung dar, welche auf eingebetteten Systemen mit wenig RAM besonders kritisch ist. TFLM passt sich insofern auf diese Besonderheiten an, indem die Laufzeit selbst keinen neuen Speicher für die Zwischenergebnisse eines Modells anfordert [10, S. 6]. Stattdessen müssen Entwickler einen Puffer (als "Arena" bezeichnet) übergeben, welcher von TensorFlow Lite Micro zur Verwaltung der Ergebnisse genutzt wird. Auch optimiert das Framework, wann ein Zwischenergebnis mit einem neuen überschrieben werden kann, indem die Wiederverwendung von Speicherbereichen optimiert wird.

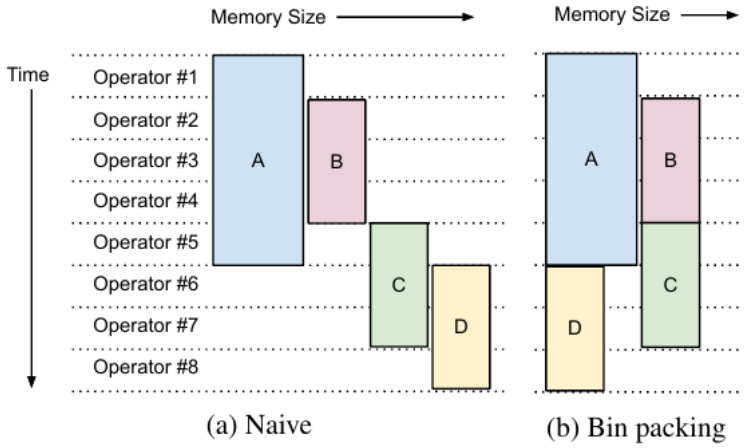


Abbildung 6.1: Schema zur Minimierung des genutzten Arbeitsspeichers für Zwischenergebnisse durch TensorFlow Lite Micro. Entnommen aus [10, S. 7].

Abbildung 6.1 zeigt den Graphen einer solchen Optimierung. Es sind vier Operationen A bis D erkennbar, diese stellen jegliche Berechnungen dar, welche im Modell definiert sind, um seine Ausgaben zu berechnen. Die x-Achse stellt den Speicherbedarf dar, während die nach unten zeigende y-Achse die Zeit darstellt. In beiden Fällen benötigt Operation C die Ergebnisse von Operation B und Operation D die Ergebnisse von Operation A. Im naiven Fall besitzen alle Zwischenergebnisse disjunkte Speicherbereiche, sodass der Gesamtspeicherbedarf die Summe aller dieser Bereiche umfasst. Zwischen den Speicherbereichen bestehen Abhängigkeiten, welche ausgenutzt werden können, um den Gesamtspeicherbedarf zu reduzieren. Der optimierte mit "Bin packing" bezeichnete Fall nutzt dies aus. Die Ergebnisse von A werden von D genutzt, dabei schreibt Operation D ihre Ergebnisse in den Speicherbereich der Ergebnisse von A und löscht diese damit effektiv. Dies ist erlaubt,

1 https://github.com/tensorflow/tflite-micro/issues/2587

da die Ergebnisse von A von keiner weiteren Operation benötigt werden. Gleiches geschieht mit den Ergebnissen von B durch C, sodass insgesamt ein geringerer Speicherbedarf, erkennbar durch die geringere Breite des von allen Operationen eingenommenen Bereichs, entsteht. Hierdurch wird nur die Speicherbelegung optimiert, eine geringere Laufzeit entsteht (im Graphen durch die gleichbleibende Höhe erkennbar) generell nicht. Die Minimierung des Speicherbedarfs ist in dem geschilderten Fall nur ein einfaches Beispiel, in realen Modellen können komplexere Abhängigkeiten zwischen mehreren Operationen bestehen, welche schwerer zu optimieren sind. Solche Optimierungen ermöglichen es TFLM, die Menge an genutzten Arbeitsspeicher zu reduzieren, was auf eingebetteten Systemen besonders hohen Stellenwert besitzt.

Vorteilhaft an TensorFlow Lite Micro ist, dass es sich um eine hardwareagnostische Laufzeit handelt. Jegliche Operationen sind in C++ implementiert und somit auf mutmaßlich allen modernen eingebetteten Geräten einsetzbar. Dies bedeutet jedoch auch, dass TensorFlow Lite Micro ohne Anpassungen keine Hardwarebeschleunigung ausnutzen kann, jegliche Berechnungen also auf der CPU ausgeführt werden. Auf Geräten mit spezieller Hardware wie einer NPU kann somit kein Performancegewinn durch diese entstehen.

Für den Luckfox Pico wäre TensorFlow Lite Micro ebenso einsetzbar, doch bietet der Hersteller des Geräts ein spezielles Framework für ML-Modelle an. Um die NPU des Geräts zu nutzen, werden ein spezielles Format und eine spezielle Laufzeit eingesetzt, welches geeignete Operationen eines Modells auf die NPU auslagert. Um Modelle aus anderen Frameworks zu importieren, bietet Luckfox durch ein Toolkit [52] die Möglichkeit, mit PyTorch, TensorFlow oder weiteren Frameworks erstellte Modelle zu importieren. Es werden nicht alle möglichen Rechenoperationen eines Modells unterstützt, sodass nicht beliebige Modelle lauffähig sind oder die nicht unterstützten Operationen durch Alternativen ersetzt werden müssen [52, Abs. 4.6].

Mit der RKNN-Laufzeit existiert für den Luckfox Pico und eine Reihe anderer Geräte mit Rockchip-Prozessoren eine angepasste Laufzeit, welche von der NPU der Geräte Gebrauch machen kann. Dies macht die Laufzeit spezifisch für die Geräte des Herstellers Rockchip, bietet aber auf diesen Geräten die bestmögliche Performance unter Nutzung der NPU. Eine weitere Möglichkeit, welche nicht weiter illustriert werden soll, wäre der Einsatz einer Python-Laufzeit, welche ein Deep-Learning-Framework wie PyTorch oder TensorFlow direkt ausführen könnte. Der Luckfox Pico besitzt ein speziell auf den Embedded-Bereich zugeschnittenes Linux-Betriebssystem, auf welchem Python und die von einem Framework benötigten Bibliotheken direkt in das Betriebssystem inkludiert werden müssten. Da dies einen sehr viel höheren Aufwand darstellt als die Nutzung der RKNN-Laufzeit, wird diese Möglichkeit ausgelassen.

Der Pi Zero verfügt, bezogen auf die Menge an Arbeitsspeicher und Prozessorleistung, über die größten Ressourcen der ausgewählten Geräte. Er besitzt keine Hardwarebeschleunigung durch eine NPU, sodass keine spezielle Laufzeit für eine NPU in Frage kommt. Als Betriebssystem kann Raspber-

ry Pi OS genutzt werden, was eine Variante der Linux-Distribution Debian darstellt. Bedingt durch die größeren verfügbaren Ressourcen ist dieses Betriebssystem allgemeiner gehalten, es erlaubt unter anderem die Installation beliebiger Python-Bibliotheken. Hierdurch können auf dem Pi Zero mehrere Ansätze zum Ausführen neuronaler Netze realisiert werden. Es können Laufzeiten wie TensorFlow Lite genutzt werden, wobei Modelle in das spezielle TFL-Dateiformat konvertiert werden müssen. Alternativ können die Modelle ohne Konvertierung belassen werden und direkt in dem gleichen Framework ausgeführt werden, in welchem sie trainiert wurden. Dies bietet einen geringeren Aufwand hinsichtlich Konvertieren des Modells, bietet aber möglicherweise durch den Overhead von Python als interpretierter Programmiersprache eine geringere Ausführungsgeschwindigkeit.

Nach Betrachtung der drei Geräte und der auf ihnen einsetzbaren Laufzeiten für neuronale Netze ist erkennbar, dass zur Ausführung neuronaler Netze auf eingebetteten Geräten eine Vielzahl an Frameworks und Laufzeiten existieren. Bereits bei exemplarischer Betrachtung der hier gewählten Geräte sind bei drei Geräten mehr als drei Frameworks verwendbar. Es ist jedoch auch erkennbar, dass diese Vielfalt nicht zwingend bedeutet, dass jedes Gerät jede Laufzeit nutzen kann oder sollte. Vielmehr kommen für schwache Geräte wie Mikrocontroller nur generelle Laufzeiten wie TFLM in Frage, da ihre Ressourcen zu beschränkt für andere Frameworks sind oder keine Alternativen für diese Hardware existieren. Geräte mit Beschleunigern wie NPUs benötigen auf diese Hardware angepasste Laufzeiten, welche die Hardware effektiv nutzen können, aber gleichzeitig einen höheren Entwicklungsaufwand zum Umwandeln und Anpassen der Modelle mit sich bringen. Stärkere Einplatinencomputer besitzen ausreichende Ressourcen, um ein komplettes Betriebssystem mitsamt Python-Laufzeit auszuführen, sodass hier sogar die zum Training genutzten Frameworks direkt ohne Anpassungen ausgeführt werden können.

Auch ist zu betonen, dass die hier betrachteten Frameworks nur ein stichprobenartiger Auszug sind. Beispielsweise existiert seitens PyTorch die Laufzeit ExecuTorch[2], welche sich zum Zeitpunkt dieser Arbeit jedoch noch in einer Beta-Phase befindet.

Insgesamt kann gezeigt werden, dass für eingebettete Systeme sehr viele Laufzeiten für Machine-Learning-Modelle wie neuronale Netze vorhanden sind, für ein spezifisches Gerät im konkreten Falle jedoch nur wenige oder nur eine Laufzeit anwendbar sind. Für leistungsstärkere Geräte besteht eine größere Auswahl, wobei die Auswahl auch weniger spezifisch für die Hardware des Systems ist und die Modelle eventuell langsamer ausgeführt werden als mit spezifischeren Laufzeiten.

2 https://github.com/pytorch/executorch

6.2 PERFORMANCEMESSUNGEN

Um die Leistungsfähigkeit der drei Hardwaregeräte zu quantifizieren, wurden einige der vorher beschriebenen neuronalen Netze auf den Geräten implementiert. Sie wurden auf Testbildern ausgeführt und dabei Performancemetriken wie die Laufzeit erfasst, wobei nach Möglichkeit auch weitere relevanten Metriken erfasst wurden. Um die Auswirkungen der Bildgröße zu untersuchen, wurden unterschiedlich große, quadratische Bilder mit einer Kantenlänge zwischen 128 und 512 Pixeln mit einer Schrittweite von 32 erstellt. Bei den Testbildern handelte es sich um zehn zufällig gewählte Bilder aus dem COCO-Subset.

Messungen auf dem ESP32

Der ESP32 stellt die leistungsschwächste Hardware der getroffenen Auswahl dar, wobei er mit vier Megabyte einen kleinen Arbeitsspeicher besitzt, welcher zwischen ausführendem Programm und dem Netz geteilt wird und dadurch stark ausgelastet sein könnte. Unter dem Gesichtspunkt, dass der ESP32 vor allem durch die geringe Menge an Arbeitsspeicher eingeschränkt wird, wurde zusätzlich die Auslastung des Arbeitsspeichers erfasst.

Die Implementierung erfolgte mit dem bereits vorgestellten TensorFlow-Lite-Micro-Framework. Hierzu musste das originale YOLOv5-Modell mit dem gleichnamigen Framework in das TFLite-Dateiformat konvertiert werden. Um die Dateigrößen der exportierten Modelldateien gering zu halten, damit diese in den vier Megabyte umfassenden Flashspeicher des ESP32 geladen werden konnten, mussten die Modelle auf int8 quantisiert werden. Zusätzlich musste konfiguriert werden, dass dabei jegliche Netzoperatoren mit int8-Werten berechnet werden. Als Programmiersprache wurde, wie oftmals für eingebettete Geräte möglich, C++ genutzt. Es existieren Optimierungen von TFLM zur beschleunigten Ausführung auf ESP32-Geräten[3], diese wurden bei diesen Untersuchungen nicht eingebunden. Alle zehn Testbilder wurden jeweils von der angeschlossenen microSD-Karte geladen, dekomprimiert und in das neuronale Netz eingegeben. Als Netze wurden die Modelle YOLOv5_n, 1class_n und half_n eingesetzt. Für jedes Bild wurde die Laufzeit erfasst, während der Speicherbedarf der Arena konstant blieb und für alle Bilder einer Größe gilt. Diese Messungen wurden für die beschriebenen Bildgrößen wiederholt.

Abbildung 6.2 zeigt den Verlauf der gemessenen Inferenzzeiten in Abhängigkeit von der Auflösung des Eingabebildes. Es ist erkennbar, dass die Laufzeiten der Modelle einem quadratischen Verlauf in Abhängigkeit von der Bildgröße folgen. Dies ist zu erwarten, da die Größe der Feature Maps und somit der benötigten Filteroperationen von der Anzahl der Pixel des Eingabebildes abhängt und diese Anzahl bei einem quadratischen Bild auch quadratisch ansteigt. Ebenso kann erkannt werden, dass zwischen den Mo-

3 https://github.com/espressif/esp-tflite-micro

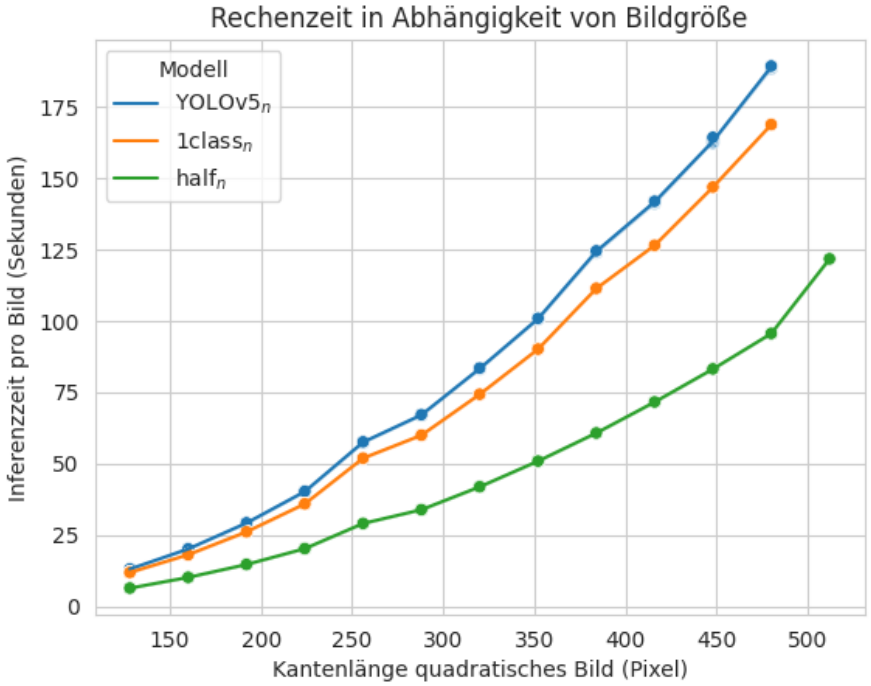


Abbildung 6.2: Inferenzzeiten auf dem ESP32. Punkte sind einzelne Messungen, Linien verlaufen durch den Median der Werte der jeweiligen Auflösung. Farblich hinterlegt ist der Interquartilsabstand (in diesem Falle durch geringe Varianz nicht erkennbar).

dellen YOLOv5$_n$ und 1class$_n$ nur ein geringer Unterschied in der Laufzeit besteht. Die Reduktion von 80 auf eine Klasse scheint hierbei nur zu minimalen Verringerungen zu führen. Diese Modelle benötigen für die kleinsten getesteten Bilder mit 128×128 Pixeln ungefähr zehn Sekunden, während die größten Bilder maximal ungefähr 180 Sekunden Laufzeit benötigen.

Ebenso können leichte Unregelmäßigkeiten in diesem Muster erkannt werden. So ist beispielsweise bei einer Auflösung von 256×256 Pixeln eine leicht höhere Laufzeit vorhanden als für einen quadratischen Verlauf zu erwarten wäre. Diese leichte Unregelmäßigkeit kann nicht direkt mit den Modellen erklärt werden. Wahrscheinlich treten hierbei hardwarespezifische Effekte auf, welche zu leicht veränderten Latenzen führen. Vorstellbar wäre beispielsweise, dass ab einer bestimmten Auflösung nicht mehr alle Ergebnisse einer Matrixmultiplikation innerhalb des Prozessor-Caches gehalten werden können, wodurch die folgenden Cache-Misses zu einer leicht erhöhten Zugriffszeit führen. Eine genaue Bestimmung dieser Gründe konnte nicht durchgeführt werden, es können somit nur Vermutungen aufgestellt werden. Auch erkennbar ist die Stabilität der Laufzeiten für Bilder gleicher Kantenlänge. Zwischen den Laufzeiten der Bilder existieren Differenzen von wenigen Millisekunden bis wenigen Hundert Millisekunden. Es kann dadurch angenommen werden, dass der Inhalt eines Bildes praktisch keinen Einfluss auf die Laufzeit hat.

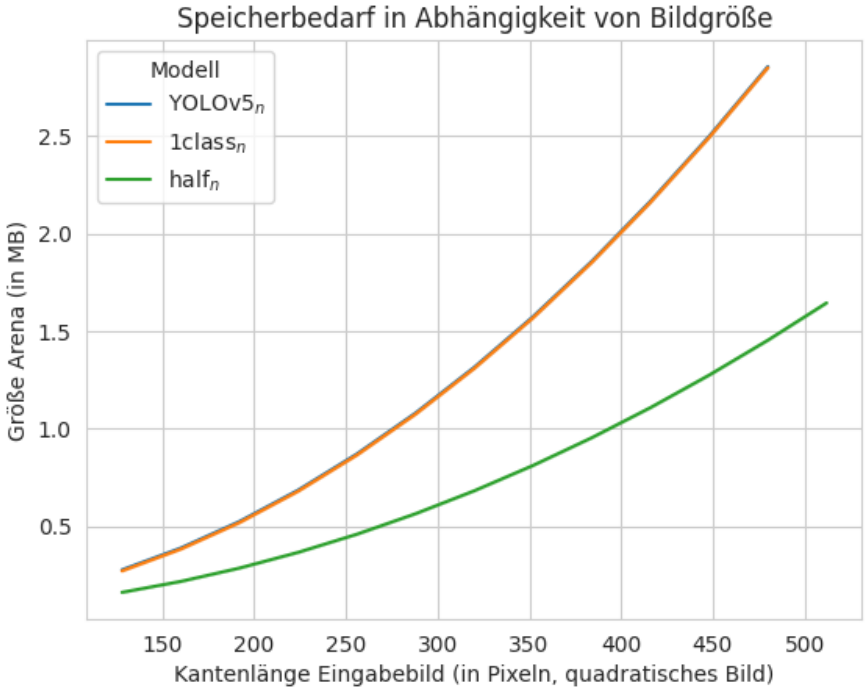


Abbildung 6.3: Speicherbedarf der durch TensorFlow Lite Micro genutzten Arena auf dem ESP32. Da die Arenagröße nicht von den Eingaben in das Netz abhängig ist, ist nur eine Messung pro Auflösung gegeben.

Das Modell half$_n$ mit einer reduzierten Anzahl an Filtern weist durchgehend geringere Laufzeiten als die anderen Modelle auf. Auch steigen seine Laufzeiten langsamer an, ein Bild mit 512×512 Pixeln wird in etwa 120 Sekunden verarbeitet. Trotz der geringeren Laufzeiten ist erkennbar, dass die Kurve ebenfalls quadratische Komplexität beschreibt und die Laufzeit des Modells verglichen mit den anderen Modellen langsamer, aber ebenfalls annähernd quadratisch ansteigt. Es ist zu beachten, dass für die Modelle YOLOv5$_n$ und 1class$_n$ in den Graphen keine Messungen für eine Auflösung von 512×512 Pixeln vorhanden sind. Bilder mit dieser Größe benötigten für diese Modelle bereits mehr Arbeitsspeicher, als die drei Megabyte, welche maximal für die Arena zur Verfügung gestellt werden konnten, sodass das Programm für Bilder dieser Größe nicht lauffähig war.

Die Laufzeiten befinden sich allesamt in einer Größenordnung von wenigen Sekunden bis hin zu einigen Minuten. Der ESP32 ermöglicht selbst auf der kleinsten Bildgröße keine Verarbeitung in Echtzeit, auf großen Bildern müssen pro Bild mehrere Minuten Verarbeitungszeit veranschlagt werden.

Neben den Laufzeiten der Modelle ist auch der Speicherbedarf für die Zwischenergebnisse der Berechnungen von Interesse, dieser ist in Abbildung 6.3 dargestellt. Der Speicherbedarf beginnt bei wenigen hundert Kilobyte und steigt mit der Bildgröße quadratisch bis auf fast drei Megabyte an. Auch hier kann ein quadratischer Verlauf basierend auf den Messpunkten vermutet werden. Die Größe der Arena richtet sich nach dem Speicherbedarf der Zwischenergebnisse während der Berechnung der einzelnen Layer. Zu-

mal diese, wie zuvor beschrieben, mit steigender Kantenlänge quadratisch anwachsen, wächst auch ihr Speicherbedarf quadratisch.

Hinsichtlich des Verlaufs des Speicherbedarfs ist erkennbar, dass die Modelle YOLOv5$_n$ und 1class$_n$, welche in ihrer Laufzeit leichte Unterschiede aufwiesen, hinsichtlich des Speicherbedarfs praktisch gleich sind; die Verlaufslinien der Modelle überlappen sich in der Darstellung komplett. Das parameterreduzierte half$_n$-Modell weist, analog zur Laufzeit, einen geringeren Speicherbedarf und ein langsameres Ansteigen dessen auf.

Messungen auf dem Pi Zero

Der Raspberry Pi Zero 2 W unterscheidet sich von der anderen Geräten in der Hinsicht, dass er ein leistungsstarker, aber allgemeinerer Einplatinencomputer ist. Er bietet deutlich mehr Leistung als ein Mikrocontroller wie der ESP32, verglichen mit dem Luckfox Pico verfügt er über mehr Arbeitsspeicher und mehr Prozessorleistung.

Für den ESP32 und den Luckfox Pico musste wegen Einschränkungen der eingesetzten Frameworks bzw. des Prozessors auf quantisierte Modelle zurückgegriffen werden. Auf dem Pi Zero besteht hingegen die Möglichkeit, eine Python-Laufzeit zu nutzen, auf welcher übliche Deep-Learning-Frameworks lauffähig sind. Somit konnte das YOLOv5-Framework hier zur Inferenz und Messung der Laufzeiten auf den Testbildern genutzt werden. Zumal dieses Framework im Gegensatz zu TFLM und der RKNN-Laufzeit keine Einschränkungen hinsichtlich der verwendeten Quantisierung vorgibt, konnten hierbei die Quantisierungsstufen int8, float16 und float32 verglichen werden.

Abbildung 6.4 enthält die gemessenen Laufzeiten für die drei int8-quantisierten Modelle. Wie auch auf dem ESP32 ist ein quadratischer Verlauf erkennbar, welcher etwas langsamer ansteigt. Einzelne Punkte stellen tatsächlich gemessene Werte dar, wobei Ausreißer nach oben erkennbar sind (für x-Werte jenseits 250 liegen diese oberhalb des abgebildeten Ausschnitts). Diese Ausreisser traten ausschliesslich bei der ersten Messung jedes Modells auf einer Auflösung auf, sodass die Werte als Messfehler ausgeschlossen werden können. Es handelt sich wahrscheinlich um Verzögerungen durch Laden von Programmbibliotheken vor der ersten Messung. Die Laufzeiten beginnen bei unter 50 Millisekunden für die kleinsten Bilder und steigen bis auf ungefähr 750 Millisekunden an. Zumal diese wenigen Ausreißer nur einmalig pro Modell auftraten, besitzen sie keinen Einfluss auf den dargestellten Median und den Interquartilsabstand.

Abbildung 6.5 enthält den Graphen der Laufzeiten des YOLOv5$_n$-Modells auf dem Pi Zero. Hierbei stellt die Farbe der Punkte die Quantisierung des genutzten Modells dar. Während die Laufzeiten aller Quantisierungsstufen anfangs ähnlich sind, so bewegen sich die Verläufe mit steigender Bildgröße immer weiter auseinander. Ausreißer sind wieder wie in Abbildung 6.4 vorhanden. Die verbleibenden Laufzeiten verhielten sich sehr stabil, wie durch die geringe Größe des Interquartilsabstands erkennbar ist.

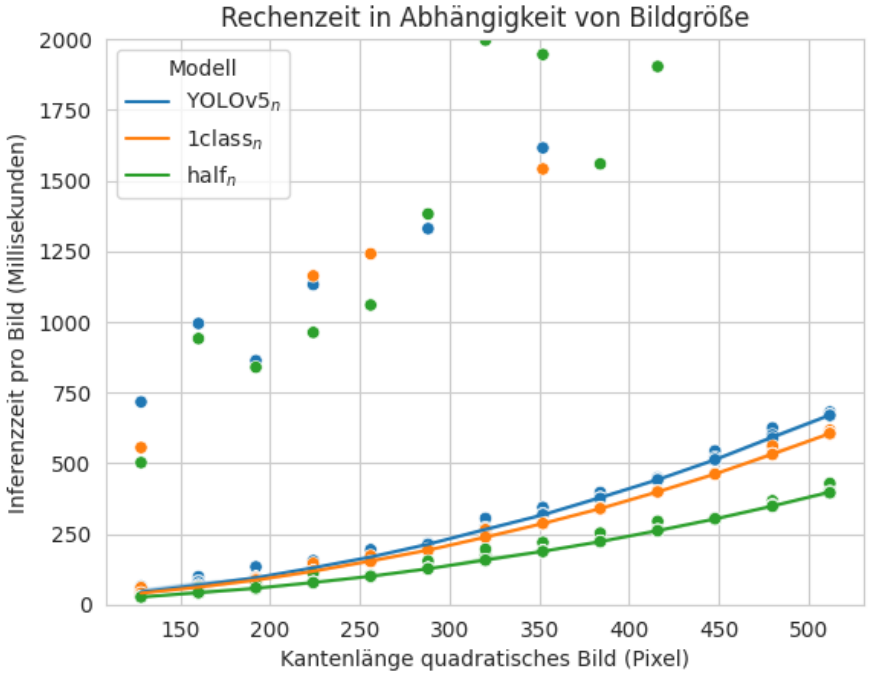


Abbildung 6.4: Erfasste Inferenzzeiten für int8-quantisierte Modelle auf dem Raspberry Pi Zero 2 W. Punkte sind einzelne Messungen, Linien verlaufen durch den Median bei der jeweiligen Auflösung. Farblich hinterlegt ist der Interquartilsabstand (25-Prozent- bis 75-Prozent-Quantil).

Auffälligerweise widersprechen diese Messwerte den ersten Erwartungen hinsichtlich der Auswirkungen einer Quantisierung. Auf float16 quantisierte Modelle besitzen ähnliche Laufzeiten wie auf int8 quantisierte Varianten, welche beiderseits höhere Latenzen als float32-Modelle besitzen. Es wäre einfach anzunehmen, dass die Nutzung eines "einfacheren" Datentyps auch direkt zu einer höheren Ausführungsgeschwindigkeit führt, zumal kleinere Datentypen oftmals einfacher zu verarbeiten sind. Die erhaltenen Messwerte zeigen jedoch, dass dies in der Praxis von weiteren Faktoren abhängig ist und sich nicht verallgemeinern lässt. Der float16-Datentyp spielt oftmals eher im Kontext von Grafikkarten eine Rolle und wird von ihnen nativ unterstützt. CPUs bieten für diesen Datentyp eher selten Unterstützung an, sodass jegliche Berechnungen mit diesem Datentyp mit anderen, unterstützten Datentypen emuliert werden müssten. Es wäre hingegen zu erwarten, dass der Datentyp int8 von einer CPU deutlich schneller verarbeitet werden könnte, selbst unter Nutzung emulierter Berechnungen. Alleine mit dem Overhead der Emulation der Befehle lässt sich nicht erklären, dass diese Modelle schlechter performen als das unquantisierte Modell. Als Ursache wurde identifiziert, dass die Inferenz mit TensorFlow Lite Micro standardmäßig keine Bibliotheken zur Beschleunigung von int8-Operationen auf dem Pi Zero einsetzt. Nach Einbinden der spezifischen Bibliotheken (ARM NN) konnte die Laufzeit der int8-quantisierten Modelle erhöht werden, wie in Abbildung 6.5 durch die roten Werte erkennbar. Es ist ersichtlich, dass un-

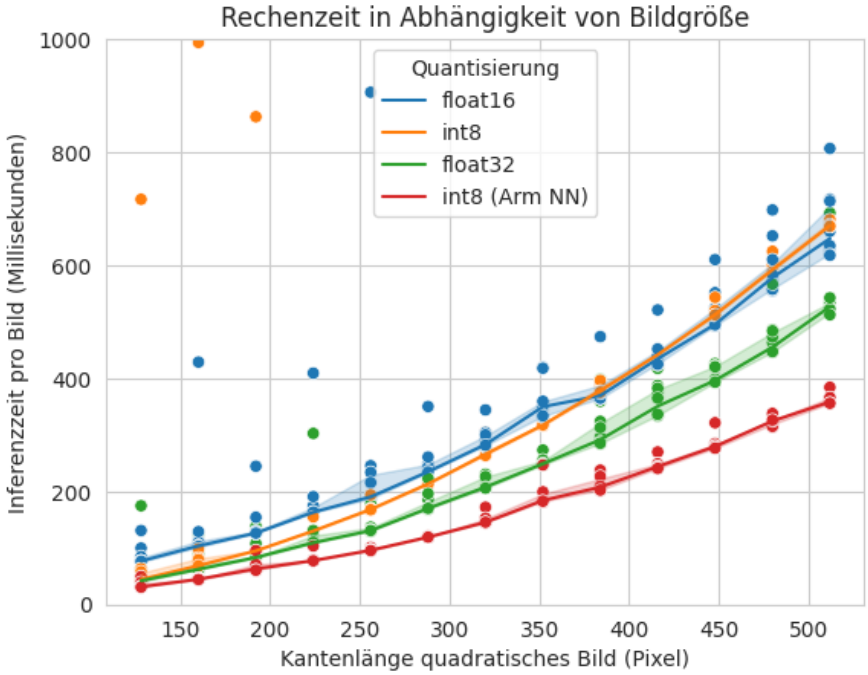


Abbildung 6.5: Erfasste Inferenzzeiten des Modells YOLOv5$_n$ in Abhängigkeit von der Bildgröße, nach Quantisierung gegliedert. Punkte sind einzelne Messungen, Linien verlaufen durch den Median bei der jeweiligen Auflösung. Farblich hinterlegt ist der Interquartilsabstand.

ter Nutzung der Hardwarebeschleunigung für int8 die quantisierten Modelle schneller als ihre unquantisierten Gegenstücke sein können. Die Geschwindigkeitszuwächse bewegen sich jedoch dabei in der gleichen Größenordnung, die Ausführungszeit wird in diesem Falle zwischen float32 und int8 um etwa 100 Millisekunden gesenkt, was bei einer Laufzeit von 500 Millisekunden nur eine geringe Reduktion ausmacht.

Es ist erkennbar, dass eine Quantisierung von neuronalen Netzen nicht zwingend mit einer Verbesserung der Ausführungsgeschwindigkeit einhergeht. Ein Leistungszuwachs ist in der Praxis auch davon abhängig, ob der Datentyp, auf den quantisiert wird, von der Hardware unterstützt wird. So zeigte sich in dem Falle des Pi Zero, dass eine Quantisierung auf float16 zu höheren Inferenzzeiten führt, da dieser Datentyp nicht direkt unterstützt wird und somit mit einem Overhead emuliert werden muss. Während int8-quantisierte Modelle auf dem Pi Zero erst ebenfalls zu keinem Geschwindigkeitszuwachs in der erwarteten Höhe führten, konnte unter Nutzung der korrekten Bibliotheken die Performance erhöht werden. Trotz dessen ergab sich, dass die Performance nicht um ein Vielfaches anstieg, sondern nur etwa verdoppelt werden konnte. Insgesamt lässt sich am Beispiel des Pi Zero zeigen, dass Quantisierung sehr abhängig von der genutzten Hardware ist und nur unter passenden Bedingungen zu einer schnelleren Inferenz der neuronalen Netze führt. In Fällen, in denen der Datentyp, auf den quantisiert

wurde, nicht nativ von der Hardware unterstützt wird und die Berechnungen emuliert werden müssen, sinkt die Performance mitunter.

Messungen auf dem Luckfox Pico

Für den Luckfox Pico wurden ebenfalls Messungen hinsichtlich der Laufzeit für alle drei Modelle durchgeführt. Messungen des Speicherbedarfs der Laufzeit konnten nicht durchgeführt werden, da hierfür keine Programmierschnittstellen in der Software vorhanden sind.

Zur Messung der Laufzeiten wurden erneut alle Netze auf den zehn Bildern ausgeführt. Hierzu mussten die Netze erst in das Luckfox-spezifische RKNN-Format konvertiert werden. Bei der Umwandlung in das RKNN-Format kann eine Quantisierung vorgenommen werden. Hierbei kann das Modell als int8 quantisiert werden oder in seiner ursprünglichen Version in float32 belassen werden. Für Prozessoren der Baureihe RV1103, wie sie beim Luckfox Pico Mini A verbaut sind, wird nur ein quantisiertes Modell unterstützt, sodass die Modelle bei der Konversion in das RKNN-Format zu int8 quantisiert wurden. Zur Ausführung wurde das Netz in ein von Luckfox bereitgestelltes C++-Beispielprogramm aus dem RKNN Model Zoo[4] eingebunden. Dieses Programm musste zur Ausführung der Modelle 1class_n und half_n modifiziert werden, damit die Reduktion auf eine Klasse korrekt beachtet wurde.

Abbildung 6.6 stellt die gemessenen Laufzeiten auf dem Luckfox Pico dar. Hierbei wurde wieder die reine Inferenzzeit des Netzes innerhalb des C++-Programms gemessen, Schritte wie die Vorverarbeitung der Bilder wurden nicht mitgemessen. Zu beachten ist, dass die y-Achse im Unterschied zu den vorherigen Messungen des ESP32 die Zeiten in Millisekunden darstellt. Erneut kann die quadratische Laufzeit erkannt werden, größere Bilder führen zu zunehmend stärker ansteigenden Laufzeiten. Im Gegensatz zu den Laufzeiten des ESP32 fällt auf, dass das Modell 1class_n geringere Laufzeiten aufweist als das Modell YOLOv5_n. Hierbei könnte der Unterschied zwischen einer und 80 verschiedenen Objektklassen eine Rolle spielen, ein genauer Grund kann jedoch erneut nicht festgestellt werden. Das Modell half_n besitzt wieder geringere Laufzeiten, was sich mit den Erkenntnissen aus den vorherigen Messungen deckt. Die Messungen verhalten sich auch in diesem Falle sehr stabil, zwischen den zehn verschiedenen Eingabebildern besteht nur eine geringe Varianz bezüglich der entstandenen Laufzeit; der eingezeichnete Interquartilsabstand ist praktisch nicht erkennbar.

Auffällig ist der enorme Geschwindigkeitszuwachs gegenüber dem ESP32. Diese entsteht sehr wahrscheinlich durch die Nutzung der NPU, welche für int8-Daten bis zu 0,5 TOPS ($0{,}5 \times 10^{12}$ Operationen pro Sekunde, vgl. Tabelle 5.1) durchführen kann. Mit Inferenzzeiten von maximal 40 Millisekunden wäre eine Verarbeitung von Bildern mit etwa 25 Bildern pro Sekunde möglich, was eine Personenzählung in Echtzeit ermöglichen würde.

4 https://github.com/airockchip/rknn_model_zoo

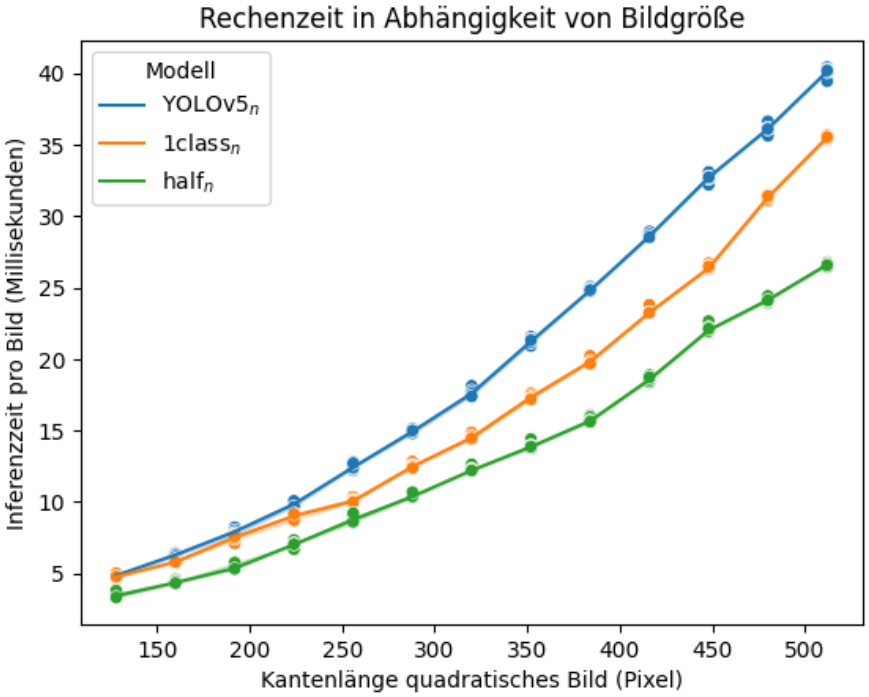


Abbildung 6.6: Erfasste Inferenzzeiten auf dem Luckfox Pico. Punkte sind einzelne Messungen, Linien verlaufen durch den Median bei der jeweiligen Auflösung. Farblich hinterlegt ist der Interquartilsabstand.

Auf dem Luckfox Pico konnte somit gezeigt werden, dass dieses Gerät einen enormen Leistungszuwachs gegenüber den anderen Geräten verzeichnet und eine Verarbeitung von Videodaten in Echtzeit ermöglichen könnte. Dieser Zuwachs lässt sich nicht einzig durch stärkere CPU begründen, sondern ist primär auf die spezialisierte NPU zurückzuführen.

6.3 ERKENNTNISSE DER IMPLEMENTIERUNG

Die Implementierung der vorher erstellten neuronalen Netze zur Personenzählung führte zu weiteren Ergebnissen, welche zur Beantwortung der Forschungsfrage genutzt werden können.

Es existiert eine enorme Vielzahl an Laufzeitumgebungen und Frameworks zur Ausführung neuronaler Netze auf verschiedener Hardware. Dies konnte damit illustriert werden, dass für die drei genutzten Geräte dieser Arbeit bereits mehr als drei Laufzeitumgebungen eingesetzt werden konnten, wobei nur für starke, allgemeinere Computersysteme wie den Pi Zero eine Auswahl bestand. Für schwache Mikrocontroller oder Geräte mit proprietären Hardwarebeschleunigern bestand in diesem Falle keine Wahl, für den ESP32 musste TensorFlow Lite Micro und für den Luckfox Pico die RKNN-Laufzeit eingesetzt werden. Aufwändig war das Umwandeln der trainierten Modelle in die Dateiformate der jeweiligen Frameworks, wobei teilweise

mehrere Umwandlungen verkettet werden mussten und in Teilen auch manuelle Intervention notwendig war.

Im Rahmen der Implementierung mehrerer Netze auf der Hardware wurden Messungen der Laufzeit und teilweise anderer Metriken vorgenommen. Der ESP32 war hinsichtlich Rechenleistung und Speicherbedarf stark ausgelastet, sodass auf ihm nicht alle Modelle auf allen Bildauflösungen lauffähig waren. Laufzeiten und Speicherbedarf folgten bei den genutzten Bildern einem quadratischen Verlauf, was mit dem quadratisch steigenden Flächeninhalt der genutzten Bilder begründet werden konnte. Ein kleineres Modell wie das half_n-Modell sparte sowohl Laufzeit als auch RAM ein. Der ESP32 konnte Bilder nur in einer Größenordnung von Sekunden bis Minuten verarbeiten, er wäre für einen realen Einsatz somit zu langsam. Da der ESP32 ein vergleichsweise starker Mikrocontroller ist, gilt wahrscheinlich allgemein, dass Mikrocontroller für diesen Anwendungsfall nicht geeignet sind.

Der Raspberry Pi Zero erzielte bessere Laufzeiten als der ESP32 und konnte zum Vergleich der verschiedenen Quantisierungsstufen genutzt werden. Als Ergebnis wurde erhalten, dass eine Quantisierung der Netzparameter neben einer Reduktion des Speicherbedarfs nicht zwingend auch zu geringeren Laufzeiten führt; dies ist stattdessen von der genutzten Hardware und den von ihr unterstützten Datentypen abhängig.

Der Luckfox Pico ermöglichte eine Verarbeitung der Bilder in Echtzeit, da er durch seine integrierte NPU nur wenige Millisekunden pro Bild zur Inferenz eines neuronalen Netzes benötigte. Basierend auf seinen gemessenen Laufzeiten konnte illustriert werden, dass für die Ausführung neuronaler Netze in Echtzeit auf eingebetteten Systemen ein Hardwarebeschleuniger wie die verbaute NPU des Luckfox Pico unabdingbar ist.

Mit der Betrachtung von Laufzeitumgebungen und Frameworks und der Implementierung unter Nutzung derer konnten die verbleibenden Aspekte der Forschungsfrage beantwortet werden. Es ist ersichtlich, dass zur Implementierung viele unterschiedliche Laufzeitumgebungen und Frameworks zu nutzen sind, welche sich je nach verwendeter Hardware stark unterscheiden und aufwändige Umwandlungen zwischen Frameworks und Dateiformaten mit sich ziehen. Mit der tatsächlichen Implementierung konnte gezeigt werden, dass Einplatinencomputer mit Hardwarebeschleunigung für neuronale Netze die beste Wahl für eine Personenzählung in Echtzeit darstellen. Abseits einer Verarbeitung in Echtzeit sind Einplatinencomputer ohne Beschleuniger nutzbar, auch auf starken Mikrocontrollern ist eine Zählung implementierbar. Auf solch schwachen Geräten ist jedoch wenig Spielraum für eventuelle Erweiterungen gegeben und die Netze benötigen mehrere Minuten für die Verarbeitung eines einzelnen Bildes.

7 DISKUSSION UND AUSBLICK

Nachdem in den vorherigen Abschnitten die besonderen Herausforderungen der Ausführung neuronaler Netze auf eingebetteten Geräten zur Personenzählung untersucht und Lösungsansätze für diese Herausforderungen dargelegt wurden, wurden diese ebenfalls durch empirische Messungen in ihrer Effektivität belegt. Obwohl gezeigt wurde, dass die gewählten Netzarchitekturen zu einer Personenerkennung in der Lage sind, so können durch die Messungen alleine nicht alle Aspekte der Umsetzung betrachtet werden. In diesem Abschnitt werden die erhaltenen Ergebnisse diskutiert und weitere, nicht aus den Messungen ersichtliche Erkenntnisse erläutert. Basierend auf dieser Diskussion wird ein Ausblick sowohl auf zukünftige Forschung als auch auf die zukünftige Richtung des Forschungsprojekts zur Personenzählung gegeben, welche beide an die Ergebnisse dieser Arbeit anknüpfen könnten.

7.1 VERGLEICHE MIT ANDEREN ARBEITEN

Wie bereits sowohl in Kapitel 2 als auch in den verbleibenden Teilen mehrfach erwähnt, existieren bereits viele andere Arbeiten zu Personenzählungen. Obwohl sich diese in ihren genauen Anwendungsfällen und Definitionen unterscheiden, so lassen sich ihre Erkenntnisse und die weiterer Arbeiten trotzdem mit dieser Arbeit vergleichen.

Die Zählung von wartenden Personen vor einem Aufzug von Shen und Chu [58] nutzt im Gegensatz zu dieser Arbeit einen Single Shot Detector zur Zählung der Personen. Sie nutzen als eingebettete Hardware einen Hardwarebeschleuniger von NVIDIA, welcher alleine bereits die in dieser Arbeit angestrebte Kostengrenze von 100 Euro übersteigt. Ihr System wäre somit nicht kostengünstig mit hohen Stückzahlen umzusetzen und nicht für den Anwendungsfall einer Personenzählung für Gebäudesensorik geeignet. Von Interesse ist, dass sich die Autoren auch explizit mit den Auswirkungen von Verdeckungen auf die Zählqualität befassen. In ihren Versuchen belegen Shen und Chu, dass ihr Detektor mit steigender Anzahl Personen und mehr Verdeckungen geringere Genauigkeiten erzielt. In ihren Bildern kommen Fälle mit maximal acht Personen vor, wobei Verdeckungen bereits bei drei Personen zu Ungenauigkeiten führen. Zur Bewertung geben sie die Genauigkeit an, also den Anteil der Situationen mit korrekt erkannter Personenanzahl, relativ zur Gesamtanzahl.

In dieser Arbeit wurde kein Genauigkeitsmaß zur Bewertung der Zählqualität genutzt, sondern durch den mittleren absoluten Fehler in der Einheit “Personen” angegeben. Aus diesem Grund sind die Erkenntnisse von Shen und Chu nicht direkt mit diesem Fehlermaß zu vergleichen. Dennoch

erscheint sich bei ihrer Zählung eine ähnliche Tendenz zu ergeben. Für größere Menschenmengen ergibt sich in ihrer Arbeit eine steigende Wahrscheinlichkeit für eine falsche Zählung, wobei unklar bleibt, ob zu wenige oder zu viele Personen erfasst werden. Auch bestärken ihre Ergebnisse den Ansatz der Zählung durch eine an der Decke montierten Kamera, da es hierbei deutlich seltener zu Verdeckungen zwischen Personen kommen kann als bei der Perspektive, die durch Shen und Chu genutzt wird.

Die Personenzählung von Simone Luetto [45] nutzt ebenfalls ein SSD-Modell, welches auf Aufnahmen eines Vorlesungssaals angewandt wird. Auch er gibt zur Bewertung der Zählung nur eine Genauigkeit dafür an, wie groß der Anteil der erkannten Personen pro Bild ist. Auch er nutzt deutlich stärkere Hardware und anspruchsvollere Vorverarbeitungsverfahren, welche nicht mit den Geräten dieser Arbeit möglich wären. Sein Modell erreicht eine Genauigkeit von bis zu 95 Prozent, wobei die genutzten Netze ebenfalls mit Teilen einer Untermenge des COCO-Datensatzes trainiert wurden. In der Diskussion seiner Ergebnisse vermutet der Autor jedoch, dass die geringe Größe seines Datensatzes und mangelnde Unterschiede zwischen Trainings- und Testdaten wahrscheinlich zu einer Überschätzung der Genauigkeit des Systems unter realen Bedingungen führt [45, S. 107].

Die Ergebnisse von Simone Luettos Arbeit decken sich in Teilen mit dieser Arbeit. Während erneut stärkere Hardware als in dieser Arbeit verwendet wurde, so decken sich die identifizierten Probleme mit den hier erlangten Erkenntnissen. Auch Simone Luetto warnt vor einer eventuell geringen Aussagekraft seiner Ergebnisse, da nur wenige eigene Trainingsdaten zu öffentlichen Daten wie dem COCO-Datensatz hinzugefügt wurden. Die auf öffentlichen Datensätzen vortrainierten Modelle erzielen eine genaue Zählung, da seine Bilder vermutlich aus der gleichen Perspektive wie übliche COCO-Daten aufgenommen wurden. Es lässt sich vermuten, dass dies als Hinweis gedeutet werden kann, dass für Zählungen auf typischen Perspektiven, wie beispielsweise durch Überwachungskameras, ein auf dem COCO-Datensatz trainiertes Netz generell ausreichend sein könnte. Für Anwendungsfälle wie den einer Zählung aus der Vogelperspektive, könnten repräsentative Trainingsdaten deutlich wichtiger sein.

Die vorliegende Arbeit knüpft an die Masterarbeit von Oskar Rudolf [56] an. In seiner Arbeit befasst sich Oskar Rudolf mit dem Vergleich verschiedener neuronaler Netzarchitekturen, welche für eine Personenzählung durch eine deckenmontierte Kamera einsetzbar wären. Seine Untersuchungen befinden sich auf einer abstrakteren Ebene und betrachten eingebettete Systeme nicht als Kern der Arbeit. Er kommt zu dem Schluss, das Modelle der YOLO-Familie schneller als R-CNN-Ableger sind, wobei andere Modellfamilien schwerer hinsichtlich ihrer Eignung für den Anwendungsfall einzuordnen sind. Auch erwähnt er, dass die Unterschiede zwischen Modellfamilien stärker ausgeprägt sind als innerhalb einer Familie zwischen unterschiedlich großen Modellen. Oskar Rudolf verwendet zur Bewertung der Modelle primär die Maße zur Bewertung der Objektdetektion, also Precision, Recall und die Varianten der Average Precision.

Diese Erkenntnisse wurden teilweise in dieser Arbeit bestätigt, in anderen Fällen konnten jedoch keine aufbauenden Erkenntnisse erlangt werden. Ein Problem, welches auch in dieser Arbeit fortbestand, ist durch die enormen Aufwände zur Evaluation verschiedener Netze gegeben. Während für vortrainierte Netze eine Vielzahl an Quellen wie der Detectron2 Modell Zoo [68] besteht, gestaltet sich die Evaluation deutlich schwieriger, sobald die Modelle mit eigenen Daten neu trainiert werden müssen. Besonders erschwert wird dies durch die Vielzahl an unterschiedlichen Frameworks und Laufzeiten, welche im folgenden Abschnitt separat thematisiert wird. Bestätigt werden konnte die Eignung von YOLOv5 als Netzarchitektur, welche auch auf kleinsten eingebetteten Geräten eingesetzt werden kann. Es konnten in dieser Arbeit mit kleinen Netzen akzeptable Zählergebnisse erzielt werden; dies deckt sich mit den Ergebnissen aus der erwähnten Vorgängerarbeit, wonach größere Modelle auf dieser Aufgabe nicht zwingend bessere Ergebnisse erzielen (vgl. [56, S. 65]). Die Vermutung, dass zweistufige Verfahren wie R-CNN durch ihre erhöhte Parameteranzahl und Rechenkomplexität nicht auf kleinsten eingebetteten Systemen eingesetzt werden können, wurde von Oskar Rudolf nicht geäußert. In dieser Arbeit konnten mehrere von Oskar Rudolf als Teil seines Ausblicks erwähnte Untersuchungen durchgeführt werden, vorrangig die Implementierung auf echter eingebetteter Hardware und die Bewertung von Techniken wie Teacher-Student-Training. Eine Vergrößerung des Labordatensatzes, wie sie von ihm ebenfalls als zukünftige Erweiterung angesprochen wird, steht zum Zeitpunkt des Verfassens dieser Arbeit weiterhin aus.

Sehr ähnlich ist die Arbeit von Saffari et al. [57]. Sie beschäftigen sich auch mit der Nutzung schwacher eingebetteter Geräte zur Erkennung von Personen, ebenfalls mit der Begründung der Minimierung von Kosten. Sie nutzen zum Aufnehmen der Bilder einen kostengünstigen Bildsensor, welcher von einem Mikrocontroller angesteuert wird. Der Controller überträgt die Bilder drahtlos an einen Raspberry Pi 4, welcher die Bilder mit einem YOLOv5_s-Modell nach Personen durchsucht. Die Bilder besitzen eine sehr geringe Auflösung von nur 120×120 Pixeln [57, S. 2]. Die resultierende Erkennung benötigt pro Bild eine Laufzeit von etwa 100 Millisekunden, wobei mehr als 90 Prozent der Personen korrekt detektiert werden [57, S. 1].

Die Arbeit von Saffari et al. ist zu dieser Arbeit somit sehr ähnlich. Es wird ein Fokus auf die Nutzung schwacher eingebetteter Systeme gelegt, auch wird YOLOv5 als Modell verwendet. Auch die erhaltenen Ergebnisse sind ähnlich: die Laufzeiten scheinen für einen Einplatinencomputer wie den Raspberry Pi realistisch, der Raspberry Pi Zero 2 dieser Arbeit erzeugte (obwohl er leistungstechnisch schwächer ist als der Raspberry Pi 4) vergleichbare Laufzeiten. Im Gegensatz zu dieser Arbeit nutzen Saffari et al. keine Verarbeitung direkt auf dem Mikrocontroller, die neuronalen Netze werden auf dem Raspberry Pi ausgeführt. Es liegt somit keine derart "aggressive" Verwendung eingebetteter Systeme vor, wie sie in dieser Arbeit geschah. Die erhaltene Detektionsqualität der Autoren ist höher, da sie selbst gesammelte

Daten als Trainingsdatensatz verwenden, wobei der Umfang mit etwa 4.300 Bildern größer als der Labordatensatz dieser Arbeit ist.

Eine letzte ähnliche Arbeit, mit der ein Vergleich gezogen werden soll, ist eine Untersuchung verschiedener Objektdetektoren zur Personenerkennung auf eingebetteten Geräten durch Kim et al. [37]. In ihrer Arbeit untersuchen sie verschiedene Objektdetektoren, darunter auch Varianten von R-CNN, YOLO und SSD. Zur Evaluation nutzen sie einen Datensatz mit Bildern von Überwachungskameras, welche teilweise aus der Vogelperspektive, wie in dieser Arbeit angestrebt, aufgenommen wurden. Ihr Datensatz ist jedoch ebenfalls nicht öffentlich verfügbar. Bei ihrer Bewertung der Objektdetektoren werden Average Precision und Laufzeit der Modelle angegeben. Wie in dieser Arbeit wurden Berechnungen zur Detektionsqualität auf Desktop-Hardware, die Messungen zu Laufzeiten auf der eingebetteten Hardware durchgeführt [37, S. 5]. Als Ergebnis ihrer Evaluation halten die Autoren fest, dass einstufige Detektoren wie YOLO oder SSD signifikant geringere Laufzeiten erzielen als die als zweistufige Verfahren evaluierten R-CNN-Modelle. Die Erkennungsqualität der einstufigen Modelle schwankt jedoch stark, nur ein YOLOv3-Modell ergibt nach Auffassung der Autoren eine vergleichsweise gute Balance zwischen guten Erkennungen und geringer Laufzeit. Als Hardware nutzen Kim et al. auch vergleichsweise starke Geräte von NVIDIA, weshalb ihre Laufzeiten um mehrere Größenordnungen geringer sind als solche dieser Arbeit (vgl. [37, S. 5–6]).

Die Aussagen von Kim et al. zeichnen ein ähnliches Bild wie diese Arbeit. Auch sie betrachten ein Modell der YOLO-Familie als für eine Personenerkennung geeignet, können aber durch einen größeren Datensatz mit repräsentativen Daten bessere Aussagen erhalten. Leider ist ihr Datensatz nicht öffentlich, sodass ihre Versuche nicht reproduziert werden können. Im Gegensatz zu dieser Arbeit untersuchen die Autoren auch R-CNN-basierte Verfahren; in dieser Arbeit wurden diese wegen ihrer enormen Modellgröße und hohen Laufzeit direkt ausgeschlossen. Diese Entscheidung scheint durch Kim et al. bestätigt zu werden, da R-CNN-Modelle bei ihren Untersuchungen zwar sehr genau, aber auch sehr langsam sind. R-CNN-Modelle sind in ihren Versuchen um einen Faktor zehn langsamer [37, S. 6], auf der Hardware dieser Arbeit hätte dies in Teilen zu unannehmbaren Laufzeiten geführt.

Die betrachteten Arbeiten bestätigen die Erkenntnisse dieser Arbeit, dass eine Personenzählung in dem Rahmen dieser Arbeit umsetzbar und sinnvoll ist. Andere Arbeiten bekräftigen einerseits die Vermutung, dass nur einstufige Verfahren für den Einsatz auf eingebetteten Systemen mit geringen Ressourcen geeignet sind. Andererseits lassen sich Messungen aus vergleichbaren Arbeiten nicht ohne Weiteres mit den Untersuchungen dieser Arbeit vergleichen, da in allen gesichteten Fällen deutlich stärkere eingebettete Hardware eingesetzt wurde als in dieser Arbeit verfügbar war.

7.2 FRAGMENTIERUNG DURCH VIELZAHL AN ML-FRAMEWORKS

Bei den trainierten Modellen wurden verschiedenste Frameworks zum Ausführen auf unterschiedlicher Hardware verwendet. Hierbei fiel auf, dass die in dieser Arbeit genutzten Geräte allesamt unterschiedliche Frameworks zur Ausführung benötigten.

Für Mikrocontroller wie den ESP32 musste TensorFlow Lite Micro eingesetzt werden, da kein ähnliches, auf besonders schwache Geräte wie Mikrocontroller zugeschnittenes Framework existiert. Der Luckfox Pico erfordert zur Einbindung seiner Neural Processing Unit ein proprietäres Format, welches zwar gut unterstützt wird, aber trotzdem einige manuelle Schritte, wie die Konversion der Modelle in ein Zwischenformat oder das Entfernen nicht unterstützter Operationen (vgl. [52, Abschnitt 4.6]), erforderte. Hierbei war ebenfalls kein alternatives Framework gegeben, welches die Hardware des Gerätes gleich effektiv hätte nutzen können. Alleinig für den Pi Zero konnte eine "einfache" Python-Laufzeit genutzt werden, sodass die Modelle mit dem gleichen Framework ausgeführt werden konnten, in welchem sie auch erstellt wurden.

Im Rahmen von Recherchen fiel auf, dass existierende Modelle zur Objektdetektion und zur Personenzählung in sehr vielen verschiedenen Formaten vorliegen und für eine Ausführung auf eingebetteter Hardware in sehr unterschiedliche Formate exportiert werden müssen. Einerseits spiegelt sich darin wider, dass das Fachgebiet der Computer Vision mit Methoden des Machine Learnings stark zwischen mehreren Frameworks aufgeteilt ist. Basierend auf Statistiken zu Veröffentlichungen auf der Plattform "Papers with Code" machten Veröffentlichungen, welche entweder PyTorch oder TensorFlow as Framework nutzten, im Jahre 2022 zusammen etwa drei Viertel aller Paper aus, während das restliche Viertel auf verschiedenste, weniger populäre Frameworks verteilt wird [36, S. 9, Abb. 4]. Modelle, die in einem dieser Frameworks erstellt wurden, sind nicht ohne Maßnahmen wie eine Nachimplementierung der Netzstruktur in dem jeweils anderen Framework nutzbar. Beispielsweise ermöglicht das YOLOv5-Framework, welches auf PyTorch basiert, einen Export in das TFL-Format nur, indem eine Parallelimplementierung der gesamten Netzarchitektur in TensorFlow umgesetzt wurde. Zum Export werden die PyTorch-Gewichte in die äquivalente TensorFlow-Repräsentation umgewandelt und das TensorFlow-Modell für TensorFlow Lite exportiert[1]. Somit muss bereits beim Training eines Modells mit einem Deep-Learning-Framework darauf geachtet werden, ein Framework zu nutzen, welches für die angestrebte Hardware kompatible Modelle exportieren kann.

Die Popularität der Deep-Learning-Frameworks erklärt jedoch nicht alleine die große Menge an Laufzeiten für eingebettete Systeme. Ausschlaggebend hierfür sind wahrscheinlich die Unterschiede der Hardware eingebetteter Systeme. Besonders im Embedded-Bereich liegt ein besonders großes

1 Vgl. Implementierung unter https://github.com/ultralytics/yolov5/blob/master/models/tf.py

Augenmerk darauf, möglichst effiziente und auf die Hardware angepasste Berechnungen durchzuführen. Dies kann mit speziellen, angepassten Laufzeitumgebungen und Dateiformaten für neuronale Netze erreicht werden. In Fällen, in denen sich im Rahmen der Planung bereits für ein bestimmtes Hardwaregerät entschieden wurde, stellt dies kein Problem dar. In anderen Fällen, in denen mehrere Plattformen verglichen werden sollen, entsteht hierdurch ein enormer Mehraufwand in der Entwicklung. Durch die Vielzahl an Laufzeitumgebungen und Dateiformaten entsteht nach dem Training der Netze, welches üblicherweise einen sehr zeitaufwändigen Schritt darstellt, nochmals ein hoher Zeitaufwand durch den Export der Modelle auf die verwendete Hardware.

Diese Fragmentierung führt zu einer schlechten Usability für Entwickler. Viele Frameworks vernachlässigen es, möglichst viele dieser Formate zumindest rudimentär zu unterstützen, beziehungsweise Anleitungen zur Konversion zur Verfügung zu stellen. Es obliegt damit dem Entwickler, die nötigen Tools zur Umwandlung auf- und einzusetzen. Dieser Mangel an Usability stellt nach Auffassung des Autors ein Problem für das Fachgebiet des Machine Learning dar. Entwickler müssen eine Vielzahl an proprietären Tools installieren und zusammenführen, um ein neuronales Netz auf einem einzigen Gerät implementieren zu können. Tools, welche diese Schritte bündeln und abstrahieren, sind selten und möglicherweise ihrerseits mit Problemen belastet. Als Beispiel kann das in dieser Arbeit genutzte Framework zu YOLOv5 erwähnt werden. Dieses wurde unter anderem ausgewählt, da es ein einfaches Modelltraining ermöglicht und den Export trainierter Modelle in viele verschiedene Ausgabeformate integriert. So konnte der Export in das TFLite-Format direkt und ohne die Installation zusätzlicher Software erfolgen. Das YOLOv5-Framework ist in dieser Hinsicht benutzerfreundlich, da es die Schritte des Exports und des Trainings einfach kombiniert. Gleichzeitig sind die von den YOLOv5-Autoren beanspruchten Performanceverbesserungen ihrer Implementierung wissenschaftlich umstritten. Bis zum Zeitpunkt des Verfassens dieser Arbeit existiert zu YOLOv5 kein wissenschaftliches peer-reviewed Paper, welches die Verbesserungen gegenüber anderen Ablegern der YOLO-Familie darstellt. Dies besitzt keinen negativen Einfluss auf die Ergebnisse dieser Arbeit, da die Leistungsunterschiede zwischen YOLOv5 und anderen YOLO-Varianten keine vorrangige Rolle spielen.

Zusammenfassend besteht auf dem Markt für Deep-Learning-Frameworks und noch stärker im Falle von Laufzeitumgebungen für neuronale Netze auf eingebetteten Geräten eine enorme Fragmentierung in viele, untereinander inkompatible Softwarelösungen. In der Praxis bedeutet dies, dass ein Vergleich unterschiedlicher Hardware sehr zeitaufwändig ist, da zwischen mehreren Dateiformaten, Laufzeitumgebungen und mitunter Programmiersprachen umgewandelt werden muss. Der zusätzliche Aufwand muss bei Implementierungen beachtet werden, da er neben dem Training der Netze einen zusätzlichen großen Aufwand darstellt. Diese Einschätzung deckt sich mit den Beobachtungen der Autoren von TensorFlow Lite Micro, welche in ihrer Arbeit bemängeln, dass zum Zeitpunkt ihrer Arbeit kein einheitliches

Framework für eine Ausführung neuronaler Netze auf eingebetteter Hardware existierte [10, Abs. 2.2]. Dieser Entwicklung sollte mit der Entwicklung von TFLM entgegengesteuert werden, um ein einheitliches Framework bzw. eine Laufzeit für verschiedenste Geräte zur Verfügung zu stellen. Dieses Ziel scheint nicht komplett erreicht worden zu sein. Dabei scheint die Entwicklung von TFLM einerseits zu spät erfolgt zu sein, sodass Hersteller wie Luckfox bereits proprietäre Laufzeiten und Dateiformate entwickelt hatten. Andererseits löst TFLM nicht das Problem, dass abseits des Embedded-Kontexts auch eine Fragmentierung, primär in die Frameworks TensorFlow und PyTorch existiert. Da TFLM nur TensorFlow-Modelle unterstützt, besteht weiterhin das Problem, dass Modelle, welche in PyTorch erstellt wurden, nicht ohne Zwischenschritte in das TFL-Format umgewandelt werden können.

Auswirkung dieser Problematik auf diese Arbeit war, dass mit YOLOv5 nur eine Familie an Objektdetektoren experimentell untersucht werden konnte. Andere Architekturen wie die aufgeführten Varianten des Single Shot Detectors wurden in Erwägung gezogen, aber verworfen, weil zum Export der erhaltenen Netze kein vorhandenes Softwarepaket wie das YOLOv5-Framework gegeben wäre. Um die erhaltenen Modelle beispielsweise in das TensorFlow-Lite-Format exportieren zu können, hätte eine Parallelimplementierung in TensorFlow vorgenommen werden müssen (das YOLOv5-Framework basiert auf PyTorch) oder es hätten (mitunter fehleranfällige) Programme zur Konversion zwischen beiden Frameworks eingesetzt werden müssen. Da diese Probleme sich theoretisch bei jeder Embedded-Laufzeitumgebung wiederholen könnten, wurde in dieser Arbeit nur eine Netzarchitektur, für welche die Exporte auf mehrere Laufzeitumgebungen gut dokumentiert sind, eingesetzt.

7.3 LIZENZIERUNG DER NEURONALEN NETZE

Die Untersuchungen und Implementierungen der neuronalen Netze dieser Arbeit wurden in einem theoretischen Forschungskontext vorgenommen. Bestimmte Aspekte, welche in der Forschung keine oder nur eine untergeordnete Rolle spielen, wurden somit nicht betrachtet. Ein wichtiger Aspekt zur Weiterverwendung von Software besteht in der Lizenzierung dieser.

Das zum Training der neuronalen Netze genutzte YOLOv5-Framework ist unter einer proprietären Lizenz und der GNU Affero General Public License (AGPL) [21] lizenziert, welche eine Erweiterung der GNU General Public License v3 (GPLv3) [62] ist. Die GPLv3 bietet Nutzern die üblichen Rechte zum Ausführen, Ändern und Weiterverbreiten der Software. Diese Rechte stehen unter der Bedingung, dass Nutzenden der Software Zugang zum Quellcode der Software gegeben werden muss. Gleichzeitig stellt die GPL eine Copyleft-Lizenz dar, was vorschreibt, dass Software, welche GPL-lizenzierte Teile beinhaltet, ebenfalls GPL-lizenziert werden muss. Die AGPL erweitert die GPLv3 zusätzlich um den Punkt, dass die Bedingungen der GPL auch gelten, wenn Software nur über ein Netzwerk in ein Gesamtsystem integriert wird [19]. Dies bedeutet, dass beispielsweise Apps, welche mit

einem Server interagieren, welcher AGPL-lizenzierten Code ausführt, auch auch mit der AGPL lizenziert werden müssen.

Während die AGPL dazu dient, Open-Source-Software weiterhin offen zuhalten, so stellt sie mitunter für kommerzielle Produkte ein Problem dar. Solange beliebiger Code GPL-lizenziert ist, muss der Quellcode des Gesamtsystems ebenfalls GPL-lizenziert sein und veröffentlicht werden. Eine Veröffentlichung des Quellcodes einer Software steht im Konflikt mit den Wettbewerbsinteressen einer Firma. Das YOLOv5-Framework bietet neben der AGPL ebenfalls eine kommerzielle Lizenz an, welche keine Veröffentlichung des Quellcodes erfordert[2]. Unternehmen können unter Zahlung eines (nicht fest ausgeschriebenen) jährlichen Geldbetrags das Framework lizenzieren und für kommerzielle Zwecke einsetzen. Anzumerken ist, dass nicht alle vorhergegangenen YOLO-Varianten anderer Autoren eine Copyleft-Lizenz besaßen. Beispielsweise verwendet YOLOv4 [4] eine eigene Lizenz, welche Nutzern jegliche Rechte, ohne Bedingungen übergibt[3]. Andere Veröffentlichungen wie etwa YOLOR [66] nutzen wie YOLOv5 die GPLv3, stellen aber keine alternativen, kommerziellen Lizenzen zur Verfügung[4].

Hinter YOLOv5 steht mit Ultralytics ein Unternehmen, welches Interesse am Vermarkten einer kommerziellen Lizenz besitzt. Kritisch anzusehen ist, dass Ultralytics nach eigener Interpretation der verwendeten AGPL auch Anspruch auf die trainierten Modelle erhebt, unabhängig davon, ob diese auf vortrainierten Gewichten basieren oder komplett neu trainiert wurden[5]. Dies widerspricht der üblichen Auffassung der GPL, nach derer Ausgaben eines Programms nicht den Lizenzbedingungen der erzeugenden Software unterliegen [20]. Die Interpretation seitens Ultralytics steht somit im Konflikt mit als allgemein angenommenen Prinzipien. Durch diese unterschiedlichen Interpretationen wären kommerzielle Produkte, welche YOLOv5-Netze enthalten, eventuell rechtlich angreifbar, falls keine kommerzielle Lizenz von Ultralytics zur Anwendung kam. Ein solcher Rechtsstreit könnte zwar dazu führen, dass die Interpretation von Ultralytics für ungültig erklärt werden würde, hätte aber höchstwahrscheinlich durch die benötigten zeitlichen und personellen Aufwände eines Rechtsstreits eine abschreckende Wirkung für viele Unternehmen. Somit kann vorgeworfen werden, dass Ultralytics mit dieser Strategie eine Art "erpresserische" Nutzung der AGPL vornimmt: jegliche kommerzielle Nutzung von YOLOv5 wird durch eine aggressive Interpretation der Lizenzbedingungen an eine Veröffentlichung geknüpft, durch Erwerb einer kommerziellen Lizenz kann dies verhindert werden. Der Grundgedanke, dass Open-Source-Software durch die GPL geschützt werden soll, wird hierbei zweckentfremdet.

Bei zukünftigen Arbeiten im Rahmen der angestrebten Personenzählung sollte von der Verwendung des YOLOv5-Frameworks und anderen von Ultralytics veröffentlichten Projekten abgesehen werden. Während es nicht ab-

2 https://www.ultralytics.com/license
3 https://github.com/AlexeyAB/darknet/blob/master/LICENSE
4 https://github.com/WongKinYiu/yolor/blob/main/LICENSE
5 https://github.com/ultralytics/ultralytics/issues/2129#issuecomment-1516026065

gestritten werden kann, dass das YOLOv5-Framework ein einfaches Training von State-of-the-Art Netzen zur Personenzählung ermöglicht, so existieren sowohl von wissenschaftlicher als auch kommerzieller Seite Bedenken. Wie bereits zuvor im Rahmen der Diskussion der verwendeten Frameworks erwähnt, können die von Ultralytics für YOLOv5 beworbenen Verbesserungen nicht wissenschaftlich diskutiert werden, zumal keine eigenen Veröffentlichungen des Autors Glenn Jocher zu seiner YOLO-Variante vorliegen. Hinsichtlich der Integration der mit dem YOLOv5-Framework trainierten neuronalen Netze in ein Produkt ist unklar, welche Interpretation der AGPL und GPLv3 rechtlich Bestand besitzt. Obwohl es als wahrscheinlich anzusehen ist, dass Ultralytics Interpretation der AGPL ungültig ist, so kann dies nur vermutet werden. Alleine diese Unsicherheit könnte es zukünftig notwendig machen, die Netze mit einer Softwareumgebung zu erzeugen, welche eine lizenzrechtlich sicherere Grundlage bietet.

7.4 AKZEPTANZ DER PERSONENZÄHLUNG DURCH KAMERAS

Unabhängig der technischen Herausforderungen einer kameragestützten Personenzählung existieren soziale und rechtliche Bedenken, welche einem tatsächlichen Einsatz der Technik im Weg stehen könnten. Mangelnde Akzeptanz des Systems könnte den Einsatz komplett verhindern, sodass diese Probleme besonders kritisch untersucht werden müssen. Zumal beide Bereiche nicht auf dem technischen Gebiet dieser Arbeit liegen, soll nur ein kurzer Überblick gegeben werden und keine ausführliche Untersuchung der Bedenken erfolgen.

Nach Held et al. [26, S. 1] führt eine bewusste Videoüberwachung unter anderem zu einer erhöhten Selbstwahrnehmung, was als unangenehm wahrgenommen wird. Dieser Effekt führt zu einem erhöhten Stressniveau, bildet sich jedoch in den von den Autoren durchgeführten Versuchen nach einer Weile zurück [26, S. 1]. Eine weitere Untersuchung von Büscher et al. [6] führt unter anderem eine Untersuchung der Auswirkungen einer echten oder erwarteten Kameraüberwachung auf die Wahrnehmung einer Interview-Situation durch Versuchspersonen durch [6, S. 35]. Hierbei wurden die Versuchspersonen gebeten, einen Text vorzulesen; dabei wurden die Beobachtungssituation durch An- und Abwesenheit anderer Personen und Aufstellen einer mehr oder weniger eindeutig als laufend erkennbaren Kamera variiert [6, S. 31-32]. Basierend auf ihren empirischen Ergebnissen argumentieren die Autoren, dass bestätigt werden kann, dass eine wahrgenommene, aber nicht tatsächlich stattfindende Überwachung per Kamera ähnliche Gefühle verursachen kann wie eine tatsächliche Überwachung [6, S. 54].

Basierend auf diesen Forschungsergebnissen ist denkbar, dass eine Personenzählung durch im Raum fest verbaute Kamera eventuell negative Auswirkungen auf die Personen in einem Raum haben könnte, falls diese die Kamera als diese erkennen (oder es zumindest vermuten). Geschuldet dessen, dass bereits eine wahrgenommene Überwachung zu den genannten Effekten führen kann, wäre es somit denkbar, dass sich die erfassten Personen

selbst nach Versicherung darüber, dass keine Kamerabilder gespeichert werden, in Räumen mit dem System unwohl und beobachtet fühlen. Ob es bei einem langfristigem Einsatz eines Kamerasystems zu Gewöhnungseffekten, wie von Held et al. beschrieben, kommt, lässt sich nicht bestimmen und könnte Gegenstand zukünftiger Forschung sein.

Neben eventuellen Problemen der Akzeptanz einer kameragestützten Erfassung durch die erfassten Personen besteht in diesem Kontext auch eine rechtliche Problematik, welche die Umsetzung der Personenzählung beeinflussen könnte. Eine Erfassung von Personen über Kameras stellt nach Art. 4 Abs. 1 Datenschutzgrundverordnung (DSGVO) eine Erfassung personenbezogener Daten dar. Kamerabilder bilden das Erscheinungsbild einer Person ab und beziehen sich somit auf identifizierbare Informationen einer natürlichen, physischen Person. Die Zählung mit einer Kamera stellt ebenso eine nach Art. 4 Abs. 2 DSGVO gültige Verarbeitung solcher personenbezogener Daten dar. Rechtmäßig wäre die Verarbeitung von Kamerabildern zur Personenzählung nach Art. 6 Abs. 1 f DSGVO, falls die aufgenommenen Personen der Aufnahme zum Zwecke der Zählung zustimmen oder eine rechtliche Verpflichtung bzw. eine Aufgabe im öffentlichen Interesse wahrgenommen werden muss oder ein berechtigtes Interesse verfolgt wird. Es könnte argumentiert werden, dass diese Anforderungen erfüllt sind, da die effiziente Klimatisierung von Gebäuden und die damit verbundene Wirtschaftlichkeit für private Gebäude ein berechtigtes Interesse darstellt und zusätzlich der Klimaschutz eine Aufgabe im öffentlichen Interesse ist. Gleichzeitig ist das Recht am eigenen Bild nach Art. 2 Abs. 1 Grundgesetz ein sehr wertvolles Recht, welches besonderem Schutze unterliegt. Da sowohl Interessen der abgebildeten als auch der die Bildaufnahmen verarbeitenden Parteien vorhanden sind, müsste im Falle eines Rechtsstreits eine Interessenabwägung durchgeführt werden, welche aufgrund der erwähnten Standpunkte möglicherweise zugunsten der Aufgenommenen bewertet werden könnte. Diese Möglichkeit birgt immenses Risiko und alleine die eventuell aufwändig zu erstreitende rechtliche Lage könnte einen Einsatz des Systems behindern. Mit anderen Technologien, welche Personen durch andere, weniger stark identifizierenden Erfassungsmethoden aufnehmen, wäre dies eventuell nicht notwendig.

Insgesamt ist die rechtliche Situation hinsichtlich einer kameragestützten Personenzählung komplex, da im Zweifelsfall hochsensible persönliche Daten (trotz mangelnder Speicherung) verarbeitet werden müssen. Obwohl die Idee der kameragestützten Zählung nicht neu ist und in der Praxis bereits von verschiedenen Unternehmen umgesetzt wird, könnte es rechtlich unsicher sein, welche Maßnahmen (Informationsschilder vor Eingängen, etc.) von Anwendern des Zählsystems durchgeführt werden müssen, um einen Rechtsstreit zu vermeiden. In diesem Abschnitt aufgeführte Argumente sollen lediglich einen Eindruck darüber vermitteln, dass sich Kameraaufnahmen in einem besonders kritischem Kontext befinden, dessen rechtliche Auslegung mitunter sehr komplex werden kann. Eine genaue Untersuchung der

rechtlichen Lage ist in dieser Arbeit nicht möglich und sollte, wie eben erwähnt, mit diesem Abschnitt auch nicht verfolgt werden.

7.5 QUALITÄT UND QUANTITÄT DER DATEN

In dieser Arbeit wurden mehrere Personen enthaltende Datensätze zum Training der Objektdetektoren und zur Evaluation verwendet. Hierbei unterschieden sich die Datensätze in ihrer Eignung für die Problemstellung und in ihrer Güte und Umfang enorm.

Es fiel auf, dass für das Problem der Personenzählung nur wenige öffentlich verfügbare Datensätze existieren. Dies ist überraschend, da andere verwandte Themengebiete wie Crowd Counting und Pedestrian Detection (siehe Abgrenzung zu bisheriger Forschung) über einige wenige, dafür aber umfassende Datensätze verfügen, welche frei verfügbar sind. Für die Personenerkennung und -zählung ließen sich deutlich weniger Daten finden, welche nicht nur als Teil existierender Arbeiten erwähnt, sondern auch veröffentlicht wurden. Dies ist durch mehrere Umstände zu erklären.

Einerseits sind Crowd Counting und Pedestrian Detection Problemstellungen, welche meist in öffentlichen Räumen betrachtet werden. Fußgänger bewegen sich im Straßenverkehr, im Kontext dessen Kameraerfassungen wie durch selbstfahrende Autos als akzeptabel angesehen werden. Auch können Aufnahmen von Fußgängern gegebenenfalls in Teilen anonymisiert werden, indem beispielsweise Gesichter verdeckt werden. Ein System zur Erkennung von Fußgängern sollte aus Sicherheitsgründen nicht komplett auf erkennbare Gesichter angewiesen sein, weshalb eine solche Anonymisierung der Datensätze sogar verargumentiert werden könnte. Menschenmengen, wie sie beim Crowd Counting betrachtet werden, treten ebenfalls tendenziell eher in öffentlichen Räumen auf. Es können zwar auch bei Veranstaltungen in geschlossenen Räumen oder in privaten Kontexten große Menschenmengen entstehen, trotzdem ist anzunehmen, dass die Datensätze für Crowd Counting in Situationen aufgenommen wurden, in denen die Beteiligten eine Bildaufnahme erwarten konnten. Daten für eine Personenzählung in geschlossenen Räumen, wie sie für diese Arbeit benötigt wurden, unterscheiden sich von diesen beiden Fachgebieten. In geschlossenen Räumen und in alltäglichen Situationen wird von den Beteiligten meist eine Einhaltung der Privatsphäre angenommen. Aufnahmen, die solche Situationen abbilden, sind somit schwerer zu erhalten, da von allen Abgebildeten eine Einwilligung erhalten werden muss. Andererseits können Datensätze für eine Personenzählung eventuell nicht veröffentlicht werden, obwohl sie einen großen Umfang und eine gute Qualität besitzen, da die Aufgenommenen der Veröffentlichung nicht zugestimmt haben. Basierend auf diesen Faktoren lässt sich bis zu einem gewissen Grad erklären, weshalb Datensätze zur Personenzählung in Innenräumen schwer aufzufinden waren.

Für die gewählten Datensätze konnte festgestellt werden, dass ein Training auf einem allgemeinen Datensatz zur Personenerkennung in Netzen

resultiert, welche ausreichend gut auf Bilder mit einer ungewöhnlichen Perspektive übertragen werden können. Diese Aussagen basieren auf Messungen der Modellqualität auf den Labordaten, welche nur 240 Einzelbilder einer einzelnen Videoaufnahme darstellen. Somit besteht eine geringe Aussagekraft der auf diesem Datensatz erhaltenen Aussagen. Basierend auf den Messungen war außerdem ersichtlich, dass die auf dem WiseNET-Datensatz trainierten Modelle bereits mit geringer Komplexität und niedrigen Auflösungen der Eingabebilder sehr gute Ergebnisse erzielten. Es kann angenommen werden, dass auf Datensätzen mit einfachen Mustern bereits kleine Objektdetektoren für eine Personenzählung ausreichen könnten. Somit bleibt die Frage offen, ob Daten, wie sie von einer zentral an der Decke montierten Kamera aufgenommen werden, auch als solche "einfache" Daten gelten. Die erhaltenen Ergebnisse konnten keine eindeutige Antwort auf diese Frage geben. Zwar erzielten die auf dem allgemeineren COCO-Datensatz trainierten Modelle bessere Ergebnisse auf den Labordaten als solche, die auf dem einfacheren, kleineren WiseNET-Datensatz trainiert wurden. Dies belegt jedoch nur, dass beide Datensätze unterschiedlich gut auf die neuartigen Labordaten verallgemeinern. Hierbei scheinen die vielfältigen Muster, welche durch ein Training auf COCO erlernt werden, hilfreicher zu sein als Muster, welche nur für eine spezifische Perspektive wie im WiseNET-Datensatz gelten. Möglicherweise fokussieren sich die COCO-Modelle auch stärker auf das Erkennen von teilweise verdeckten Personen, was die Erkennung aus einer Vogelperspektive verbessert. Wegen der geringen Menge an Labordaten wurde kein neuronales Netz auf diesen trainiert. Es ist somit unklar, ob Bilder, die von einer deckenmontierten Kamera erzeugt werden, einfach zu erlernende Muster wie im WiseNET-Datensatz oder eher komplexe Muster wie im COCO-Datensatz enthalten.

Zukünftige Forschung könnte sich mit einer Erweiterung des Labordatensatzes befassen. Ziel sollte hierbei sein, mehr Varianz, also Videoaufnahmen mit verschiedenen Lichtverhältnissen, unterschiedlicher Anzahl der Personen und Kameraeinstellungen in den Datensatz einzubringen. Optimalerweise könnte die Menge an Daten so weit erhöht werden, dass ein Trainings- und Testdatensatz gebildet werden können. Dann könnten die Versuche dieser Arbeit wiederholt werden: mit auf dem COCO-Datensatz und auf den Labordaten trainierten Modellen könnte genau bestimmt werden, ob und wie sich beide Situationen unterscheiden. Selbstverständlich wäre die Aussagekraft weiterhin an die Qualität der Labordaten gebunden, durch eine Verbesserung der Datenqualität und -quantität wären die erhaltenen Aussagen jedoch wertvoller. Es könnte untersucht werden, ob, wie auf dem WiseNET-Datensatz, bereits kleine, einfache Modelle zur Personenzählung ausreichend sind, oder ob das Erlernen komplexer Muster zur Personenerkennung auf dem COCO-Datensatz weiterhin die besten Ergebnisse ermöglicht.

7.6 ERWEITERUNGEN DIESER ARBEIT

Wie bereits in den vorherigen Abschnitten dargestellt, konnten in dieser Arbeit nur einige wenige neuronale Netzarchitekturen zur Objektdetektion und somit zur Personenzählung betrachtet werden, die Untersuchungen selbst wurden nur mit einer Netzarchitektur, YOLOv5, durchgeführt. Dies begründet sich mit der bereits erwähnten enormen Fragmentierung der Frameworks und Laufzeiten sowohl auf dem Gebiet des Deep Learning als auch der Implementierung neuronaler Netze auf eingebetteter Hardware. Auch wurden mehrere Hardwaregeräte untersucht, um die Leistungsunterschiede zwischen den Geräteklassen der Mikrocontroller und der Einplatinencomputer, inklusive der Verwendung von Hardwarebeschleunigern, zu untersuchen. Aufbauend auf diese Untersuchungen kann das Forschungsprojekt, in diesem diese Arbeit angesiedelt ist, eine Auswahl eines passenden Hardwaregerätes für weitere Forschungen vornehmen. Hierbei ist es wahrscheinlich, dass die Wahl auf einen Einplatinencomputer fallen wird, zumal sich Mikrocontroller wie der ESP32 als lauffähig, aber unzulänglich langsam erwiesen haben. Obwohl Latenzen mehrerer Minuten im Kontext der Gebäudesensorik ausreichend sind, um schnellere Antwortzeiten als vorhandene klassische Sensorsysteme zu garantieren, so wäre ein Mikrocontroller hardwareseitig dauerhaft am Limit seiner Leistung angelangt; ein späteres Upgrade durch leistungsfähigere neuronale Netze wäre wahrscheinlich unmöglich. Zumal auf Einplatinencomputern, wie sie in dieser Arbeit evaluiert wurden, deutlich mehr Leistung verfügbar wäre als für die Ausführung eines YOLOv5$_n$-Netzes oder einer der erstellten Varianten, könnten weitere Untersuchungen mit nochmals leistungsstärkeren Netzen durchgeführt werden. Im Falle der YOLOv5-Architektur wäre es möglich, dass die nächstgrößere Variante YOLOv5$_s$ bessere Zählergebnisse erzielen und die Leistung eines Einplatinencomputers noch nicht übersteigen könnte. Diese Untersuchungen wären jedoch von den Erkenntnissen durch Sammeln eines größeren Trainingsdatensatzes, wie zuvor beschrieben, abhängig. Sollte sich ergeben, dass die Personenzählung mit einem kleinen Netz gute Ergebnisse erzielt und ein größeres Netz nur marginal bessere Ergebnisse erzielt, wären Experimente mit solchen Netzen überflüssig.

7.7 PERSONENZÄHLUNGEN MIT ANDEREN VERFAHREN

In dieser Arbeit wurde sich primär auf die Personenerkennung auf Farbbildern mit neuronalen Netzen fokussiert. Wie bereits als Teil der Abgrenzung zu bisheriger Forschung dargestellt, sind neuronale Netze nicht die einzigen Verfahren zur Personenzählung. Es existieren eine Vielzahl an Verfahren, welche anstelle eines neuronalen Netzes eingesetzt werden oder in Kombination das Verfahren erweitern könnten. Ebenfalls stellen Farbbilder nicht die einzige Datenbasis dar, welche für eine Personenzählung genutzt werden kann. Gleichzeitig gibt es einige Bereiche, in denen diese Arbeit verbessert und weiter ausgeführt werden könnte.

Eine bereits im Kapitel zu bisheriger Forschung erwähnte Alternative stellen Tiefenkameras dar, welche in ihren Pixelwerten keine Farben, sondern die Distanz eines Objektes zur Kamera kodieren. Tiefenkameras sind jedoch deutlich kostspieliger als klassische Farbkameras, weshalb sie bei einer Integration in ein fertiges Produkt zu einem zu hohen Stückpreis führen würden. Ein Vorteil von Tiefenkameras wäre hingegen eine höhere Verträglichkeit auf sozialer und rechtlicher Ebene. Personen können auf Tiefenbildern weiterhin identifiziert werden, eine Identifizierung ist dabei aber schwerer als auf Farbbildern. Es wäre vorstellbar, dass die vom Personenzählungssystem erfassten Personen unter diesem Gesichtspunkt eine Erfassung mit Tiefenbildern gegenüber Farbbildern bevorzugen könnten und das System eher akzeptieren würden.

Eine Möglichkeit, die Vorteile akzeptabler Hardwarekosten mit den Privatsphäre wahrenden Eigenschaften alternativer Sensortypen zu kombinieren, bestünde in der Nutzung von Wärmebildsensorik. Wärmebildkameras sind üblicherweise nochmals deutlich teurer als Tiefenkameras und Farbkameras. Solche Wärmebildkameras besitzen eine hohe Auflösung, welche für das Problem der Personenzählung überdimensioniert sein könnte. Es existieren jedoch deutlich kostengünstigere, aber dadurch bedingt auch in ihrer Auflösung sehr eingeschränkte Wärmebildsensoren, welche für eine rudimentäre Personenerkennung und -zählung eingesetzt werden könnten. Sirmacek und Riveiro nutzen in ihrer Arbeit [59] einen AMG88xx-Wärmebildsensor mit 8×8 Pixeln Auflösung, um die Anzahl der Personen in einem Meetingraum abzuschätzen. Die Autorinnen werten den deckenmontierten Sensor sowohl mit einem algorithmischen Ansatz der klassischen Computer Vision als auch mit einem Ensemble-Verfahren des Machine Learning aus. Hierbei ist zu beachten, dass der Sensor in den Experimenten ein vergleichsweise geringes Sichtfeld besitzt und maximal vier Personen gleichzeitig im Raum anwesend sind. Die Zählung von Sirmacek und Riveiro erreicht in diesem Umfeld eine korrekte Zählung in ungefähr 95% der aufgenommenen Videoframes [59, S. 13]. Ob sich diese Performance verallgemeinern und auf größere Räume übertragen lässt, ist unklar.

Trotzdem könnte die Nutzung eines kostengünstigen Wärmebildsensors eine mögliche Alternative zu Farbkameras für das angestrebte System zur Personenzählung sein. Für diese Arbeit konnte mangels repräsentativer Datensätze nicht abgeschätzt werden, wie viele Personen in normalen Situationen in Innenräumen zu erwarten wären. Datensätze wie der COCO-Datensatz, welcher zum Training verwendet wurde, enthalten dadurch wahrscheinlich oftmals deutlich mehr Personen pro Bild, als jemals von einer deckenmontierten Kamera erfasst werden könnten. Sollte sich nach Sammeln eines repräsentativen Datensatzes ergeben, dass pro Bild maximal fünf Personen abgebildet werden oder es ab einer solch hohen Raumauslastung für das Raumklima irrelevant ist, was die genaue Anzahl an Personen im Raum ist, könnte die Zählung auf einem niedrigauflösenden Wärmebild bereits ausreichend sein. Diese Überlegungen bedürfen weiterer Forschung und

können ohne neue Daten und tiefergehendes Wissen bezüglich Gebäudeklimatisierung nicht beantwortet werden.

Falls weiter an der Nutzung von Farbbildern als Datengrundlage festgehalten wird, könnten die bisher vernachlässigten temporalen Zusammenhänge zwischen aufeinanderfolgenden Bildern mit beachtet werden. Während eine naive Personenerkennung jedes Bild gleich verarbeiten würde, könnte ein intelligentes Verfahren die Bilder vorher filtern. Beispielsweise könnten Differenzbilder zwischen aufeinanderfolgenden Bildern gebildet werden und darauf ein Kriterium definiert werden, um das neuronale Netz nur erneut auszuführen, wenn zwischen den Bildern ein signifikanter Unterschied besteht. Ein solches Verfahren könnte die Systemauslastung reduzieren, indem die rechenintensive Inferenz des neuronalen Netzes seltener durchgeführt wird. Eine weitere Möglichkeit wäre es, basierend auf den Differenzen zwischen Bildern die Regionen einzuschränken, welche von dem neuronalen Netz auf Personen durchsucht werden. So könnten Bereiche, in denen sich die Pixelwerte gar nicht oder nur minimal verändert haben, ignoriert werden, während Regionen mit Veränderungen durch mutmaßliche Bewegungen von Objekten in den Objektdetektor eingegeben werden. Ein Filtern basierend auf veränderten Pixelwerten würde neue Herausforderungen mit sich bringen, beispielsweise müssen Ereignisse wie Einschalten eines Lichts, wobei sich alle Pixel im Bild massiv verändern würden, von Bewegungen, wodurch nur lokale Veränderungen im Bild stattfinden, unterschieden werden. Dies könnte von Interesse für zukünftige Forschung sein, bietet aber in der Praxis nur für Geräte einen Mehrwert, welche das Bild nicht ausreichend schnell verarbeiten können. Geräte wie der in dieser Arbeit untersuchte Luckfox Pico würden von solch einem Verfahren nicht profitieren, weil es für sie durch die Hardwarebeschleunigung schneller ist, das neuronale Netz auf jedem Bild auszuführen anstatt vor jedem Bild ein kostspieliges Filtern von Bildregionen durchzuführen.

Es ist ersichtlich, dass noch viele Möglichkeiten für aufbauende Forschung existieren. Besonders durch die Kombination von algorithmischen, weniger rechenintensiven Vorverarbeitungsschritten der klassischen Computer Vision könnte die Rechenlast durch die neuronalen Netze gesenkt werden.

8

ZUSAMMENFASSUNG UND FAZIT

In dieser Arbeit wurde eine Personenzählung basierend auf neuronalen Netzen untersucht, welche auf einem eingebetteten System innerhalb einer kleinen deckenmontierten Kamera laufen könnte. Um die Problemstellung einer Personenzählung auf leistungsschwachen eingebetteten Geräten auszuarbeiten, wurde die Leitfrage in mehrere Teilfragen unterteilt.

Motiviert wurde diese Arbeit durch ein Forschungsprojekt, im Rahmen dessen eine kamerabasierte Personenzählung existierende Gebäudesensorik ersetzen oder ergänzen sollte. Um den Forschungsgegenstand abzugrenzen, konnte gezeigt werden, dass existierende Arbeiten oftmals entweder eine Echtzeitverarbeitung der Bilder benötigen und dadurch deutlich leistungsstärkere Hardware verwenden müssen oder andere Ziele mit der Personenzählung verfolgen. Seitens der genutzten Hardware konnte sich diese Arbeit durch Untersuchung kostengünstiger, schwacher Geräte von anderen Arbeiten differenzieren. Ein weiterer Unterschied war in der Natur der Zählung gegeben: vorhandene Forschung bezog sich primär auf die Zählung von Personen in Bewegung und sah die Zählung als eine Erfassung bestimmter Ereignisse, beispielsweise das Übertreten einer Linie, an. In dieser Arbeit konnten jedoch Untersuchungen hinsichtlich des Zählfehlers durchgeführt werden, um zu quantisieren, um wie viele Personen sich die genutzten Verfahren verzählen. Es konnte gezeigt werden, dass verwandte Themengebiete wie Pedestrian Detection und Crowd Counting sich durch eine andere Perspektive typischer Daten oder durch Situationen mit einer deutlich höheren Anzahl an Personen, als in einem Innenraum vorstellbar wäre, von dem angestrebten Anwendungsfall unterscheiden. Eine Nutzung alternativer Sensordaten wie Tiefenbildern anstelle von Farbbildern konnte als zu kostspielig identifiziert werden. Mit diesen Vergleichen konnte diese Arbeit von existierender Literatur abgegrenzt und eine Antwort auf die erste Teilfrage gegeben werden.

Um ein Lösungskonzept auszuarbeiten, wurden mehrere aufeinanderfolgende Untersuchungen durchgeführt. Zuerst wurde eine Mindestanforderung an die Hardware einer Personenzählung anhand theoretischer Überlegungen gestellt. Hierbei konnte vermutet werden, dass moderne neuronale Netze zur Objektdetektion mit mehreren Millionen Parametern einhergehen, welche einen Festwertspeicher von mehreren Megabyte und einen Arbeitsspeicher von mindestens etwa einem Megabyte notwendig machen. Für die Rechenleistung konnte vermutet werden, dass ein Vielfaches der Parameteranzahl des Netzes an Rechenoperationen notwendig sind, um ein Bild zu verarbeiten, wobei die genaue Größenordnung in Relation zur Bildfläche steht. Basierend auf diesen Vermutungen wurden drei Hardwaregeräte ausgewählt, welche diesen Anforderungen genügen. Hierbei wurden mit einem

ESP32-Mikrocontroller, einem Luckfox Pico Mini A und einem Raspberry Pi Zero 2 W möglichst unterschiedliche Geräte ausgewählt, um sowohl die Leistungsdifferenzen zwischen der Klasse der Mikrocontroller und der Einplatinencomputer, als auch zwischen Geräten mit und ohne Hardwarebeschleunigung untersuchen zu können. Es konnte gezeigt werden, dass die Ausführung neuronaler Netze eine große Menge an Hardwareressourcen erfordert. Mit diesen Betrachtungen konnte eine Antwort auf die zweite Teilfrage erhalten werden und die Herausforderungen des Ausführens neuronaler Netze auf eingebetteten Systemen illustriert werden.

Anschließend wurde sich der Beantwortung der dritten Teilfrage zugewandt. Hierbei wurde die Sinnhaftigkeit eines Neutrainierens und Anpassens vortrainierter, allgemeiner Objektdetektoren thematisiert. Um geeignete Netzarchitekturen zu identifizieren, welche mit den besonders geringen Ressourcen eingebetteter Systeme lauffähig wären, wurde eine Literaturrecherche durchgeführt. Hierbei konnte gezeigt werden, dass ältere, zweistufige Verfahren wie R-CNN durch ihre enorme Parameteranzahl und ihre hohen Laufzeiten bereits auf nicht-eingebetteten Systemen nicht für die Geräte infrage kommen. Es konnte illustriert werden, dass neuere einstufige Verfahren zur Objektdetektion nicht generell mit einer geringeren Parameteranzahl einhergehen, die Laufzeiten jedoch deutlich verringert werden konnten. Basierend auf Weiterentwicklungen einstufiger Objektdetektoren konnten einige Netzarchitekturen identifiziert werden, welche durch besondere Anpassungen besonders wenige Parameter und nötige Berechnungen besitzen und somit auf eingebetteten Geräten eingesetzt werden können. Als zwei solcher Architekturen wurden verkleinerte Varianten des Single-Shot-Detectors (SSD) und mit YOLOv5 eine Weiterentwicklung des "You Only Look Once"-Detektors vorgestellt werden. Unter dem Gesichtspunkt, dass die Konzepte beider Netze sehr ähnlich sind, YOLOv5 aber den Einsatz des gleichen Netzes auf mehreren Bildauflösungen ohne Neutrainieren des Netzes ermöglicht und somit der Einfluss von Bildgrößen auf die Detektionsqualität untersucht werden konnte, wurde sich für die YOLOv5-Architektur entschieden.

Um ein Training verschiedener YOLOv5-Varianten auf Datensätzen mit Personen zu ermöglichen, wurden solche Datensätze recherchiert. Es konnte gezeigt werden, dass der COCO-Datensatz, welcher oft als Benchmark zum Evaluieren von Objektdetektionsmodellen genutzt wird, auf die Klasse der Personen gefiltert werden kann und weiterhin eine angemessene Größe für das Training neuer Netze besitzt. Hierbei konnte vermutet werden, dass die Bilder des COCO-Datensatzes nicht repräsentativ für die zu erwartende Perspektive einer deckenmontierten Kamera wären. Mit dem WiseNET-Datensatz konnte ein Datensatz geringeren Umfangs, aber repräsentativerer Bilder vorgestellt werden. Er enthält weniger Bilder als der COCO-Datensatz und besitzt generell weniger Varianz. Um eine Evaluation auf repräsentativen Daten zu ermöglichen, wurde ein Datensatz aus einer vorhergehenden Arbeit im Rahmen des Forschungsprojektes eingeführt, welcher nur einen sehr geringen Umfang besitzt.

Unter Nutzung der erhaltenen Datensätze wurden mehrere Netze trainiert und verglichen. Es konnte gezeigt werden, dass das Training auf der Untermenge des COCO-Datensatzes, welche nur Personen enthält, zu einer Verbesserung der genutzten Objektdetektionsmetriken führt. Hinsichtlich des Fehlers der Zählung konnte gezeigt werden, dass diese Verbesserung nicht direkt übertragbar ist und sich durch Nutzung eines neu trainierten Netzes nur eine minimale Verbesserung der Zählung ergibt. Der zeitliche Aufwand zum Training eines spezialisierten Netzes konnte als vertretbar angesehen werden. Gleichzeitig konnte gezeigt werden, dass eine Verkleinerung der YOLOv5$_n$-Architektur, beispielsweise durch Reduktion der Parameteranzahl, zu einer starken Verschlechterung der Detektion und der Zählung führt, wodurch weitere Verkleinerungen ohne Verschlechterung des Gesamtsystems als unwahrscheinlich angesehen werden können. Mit Modellen, welche auf der gleichen Architektur basieren, aber auf unterschiedlichen Netzen trainiert wurden, konnte die Übertragbarkeit der erlernten Muster zwischen den Datensätzen untersucht werden. Die Ergebnisse legen nahe, dass ein Training auf dem COCO-Datensatz und dem WiseNET-Datensatz sehr unterschiedliche Muster extrahiert, sodass sich auf einem Datensatz trainierte Modelle nicht gut auf den jeweils anderen Datenbestand übertragen lassen. Auffällig war, dass auf dem WiseNET-Datensatz trainierte und evaluierte Modelle bereits mit kleinen Bildgrößen sehr genaue Zählungen erzielen konnten, während für den COCO-Datensatz größere Bilder tendenziell zu immer besseren Metriken führte. Es konnte vermutet werden, dass dies bedeutet, dass die Bildauflösung für einfachere Situationen, wie sie bei einer Personenzählung von der Decke vorstellbar sind, gering gehalten werden kann, ohne dass die Performance der Zählung stark verringert wird. Beim Übertragen der Modelle vom COCO- und WiseNET-Datensatz auf die anwendungsfallgetreuen Aufnahmen fiel auf, dass die erlernten Muster nur schwer übertragbar sind und selbst im beim besten Modell ein Zählfehler von durchschnittlich etwa 0,5 Personen pro Bild (bei einem Mittelwert von 1,85 tatsächlich vorhandenen Personen) erzielt wird.

Durch diese Untersuchungen konnte belegt werden, dass ein Neutrainieren von auf Personen spezialisierten Objektdetektionsmodellen eine mit vertretbarem Aufwand verbundene Aufgabe vorliegt. Mangels ausreichend repräsentativen oder umfangreichen Datensätzen konnte nur vermutet werden, dass diese Netze, sollten sie auf einem geeigneten Datensatz trainiert werden, eine deutlich genauere Zählung gegenüber vortrainierten, allgemeinen Netzen ermöglichen könnten. Somit konnte eine Vermutung zur dritten Teilfrage gestellt werden; eine aussagekräftige Antwort darauf, ob ein Neutrainieren der Netze tatsächlich hilfreich ist, konnte wegen der erwähnten Datenprobleme jedoch nicht gegeben werden.

Nach Training der neuronalen Netze wurden die Konzepte der Quantisierung und des Teacher-Student-Trainings auf einigen Netzen evaluiert, inwiefern sie zu einer Verkleinerung der Netze, entweder durch Reduktion des Speicherbedarfs oder durch Erstellung eines kleineren, aber genaueren Netzes, in diesem Fall geeignet sind. Die Quantisierung von float32 auf int8

konnte als ein geeignetes Werkzeug dargestellt werden, bei den quantisierten Netzen verringerten sich die Werte der Metriken zur Objektdetektion jeweils nur um den Bruchteil eines Prozentpunktes. Es konnte somit belegt werden, dass Quantisierung auch im Falle von Objektdetektoren eingesetzt werden kann, um den Bedarf an Festspeicher für die Gewichte eines neuronalen Netzes stark zu verringern. Mit Teacher-Student-Training erhaltene Netze erzielten in den Untersuchungen keine besseren Ergebnisse als direkt ohne Teacher trainierte Modelle. Sie erzielten eine höhere Präzision, welche gleichzeitig mit einem geringeren Recall einherging und insgesamt zu einer ungenaueren Zählung führte. Mit der Untersuchung beider Methoden konnte zumindest die Quantisierung als wertvolle Technik zur Anpassung neuronaler Netze auf die Besonderheiten eingebetteter Systeme identifiziert werden, wodurch die letzte Teilfrage beantwortet werden konnte.

Um die Erkenntnisse zu Teilaspekten der Personenzählung der vorhergegangenen Kapitel zusammenzuführen, wurde anschliessend eine Implementierung der Zählung auf den drei ausgewählten Hardwaregeräten vorgenommen, indem die Netze auf ihnen ausgeführt wurden. Zur Ausführung auf dem ESP32 wurde TensorFlow Lite Micro eingesetzt, wobei mehrere Konversionen des neuronalen Netzes zwischen verschiedenen Frameworks vorgenommen werden mussten. Der Mikrocontroller konnte Bilder auf der kleinsten Bildgröße von 128×128 Pixeln in etwa 10 Sekunden, für eine Bildgröße jenseits 400×400 Pixeln nur innerhalb mehrerer Minuten verarbeiten. Hierbei konnte der theoretisch zu erwartende quadratische Zusammenhang zwischen Kantenlänge des (quadratischen) Eingabebildes und der Laufzeit empirisch bewiesen werden. Messungen des Speicherbedarfes belegten, dass der Arbeitsspeicher des ESP32 stark ausgelastet und ab einer bestimmten Bildgröße nicht mehr ausreichend war, um die Zwischenergebnisse der Berechnungen vorzuhalten. Es konnte somit belegt werden, dass selbst ein Mikrocontroller mit vergleichsweise hoher Leistung für diesen Anwendungsfall nicht geeignet ist, da er Bilder nur mit Rechenzeiten von mehreren Minuten pro Bild verarbeiten kann und nur wenige Ressourcen des Systems für andere Berechnungen neben der Netzinferenz verbleiben.

Für die Implementierung auf dem Pi Zero konnte eine Python-Laufzeit eingesetzt werden, was eine Konversion zwischen Frameworks und Formaten für die Netzgewichte ersparte. Der Pi Zero konnte durch seine höhere Leistung bereits mehrere Bilder pro Sekunde verarbeiten, wobei eine Quantisierung auf kleinere Datentypen teilweise, aber nicht konsistent zu geringeren Inferenzzeiten führte. Die Messungen belegten, dass Einplatinencomputer wie der Pi Zero ausreichende Leistung für eine Zählung nahe Echtzeit besitzen. Auf dem Luckfox Pico wurde eine Implementierung mit der proprietären Laufzeit des Herstellers des Geräts umgesetzt. Messungen der Laufzeiten ergaben, dass der Luckfox Pico als Einplatinencomputer mit einem Hardwarebeschleuniger für neuronale Netze problemlos in der Lage ist, die Personenerkennung in Echtzeit mit Latenzen von wenigen Millisekunden pro Bild durchzuführen. Mit der Implementierung auf echter Hardware konnte die Forschungsfrage abschliessend beantwortet werden, indem

die Laufzeiten und Frameworks zur Implementierung auf den Geräten illustriert wurden.

Als Fazit dieser Arbeit kann ausgesagt werden, dass eine Personenzählung auf eingebetteten Systemen unter Beachtung deren Einschränkungen umsetzbar ist. Falls besondere Netzarchitekturen mit einer geringen Anzahl an Parametern eingesetzt werden, welche mit Verfahren wie einer Quantisierung verkleinert werden, können sogar Mikrocontroller zum Einsatz kommen. Die Verwendung dieser ist jedoch durch ihre Laufzeiten von mehreren Minuten pro Bild (obwohl die angestrebte Zählung nicht in Echtzeit erfolgen muss) nicht zu empfehlen. Einplatinencomputer können, durch ihre größere Leistung deutlich mehr Bilder in weniger Zeit verarbeiten und sind somit die Geräteklasse, welche für eine praktische Umsetzung genutzt werden sollte. Einplatinencomputer wie der genutzte Luckfox Pico ermöglichen es durch Hardwarebeschleunigung, Videoaufnahmen einer Kamera in Echtzeit zu verarbeiten, während sie gleichzeitig sehr kostengünstig sind und durch ihren kleinen Platzverbrauch einfach in existierende Hardwaredesigns integriert werden können. Ob die Zählung mit kleinen, auf eingebetteten Systemen laufenden Netzen ausreichend genau ist, ist unklar. Auf den getesteten Datensätzen traten Zählfehler auf, welche, relativ zur mittleren Anzahl an Personen pro Bild gemessen, als eher hoch anzusehen sind. Dabei neigten die Netze dazu, die Anzahl an Personen auf Bildern mit größeren Menschenmengen zu unterschätzen. Mangels repräsentativer Datensätze ist jedoch unklar, ob diese Ungenauigkeiten auch nach Training auf einem zukünftig zu erstellenden, repräsentativem Datensatz in ähnlich starker Form fortbestehen würden. Im Kontext der Gebäudesensorik ist anzunehmen, dass eine exakte Zählung der anwesenden Personen ohnehin nicht vonnöten ist; es muss nur zwischen keiner, leichter oder hoher Belegung eines Raumes unterschieden werden. Unter solchen Rahmenbedingungen ist es denkbar, dass eine Umsetzung einer Personenzählung auf eingebetteten Systemen angemessene Ergebnisse erzielen könnte.

Teil II

APPENDIX

A

ANHANG

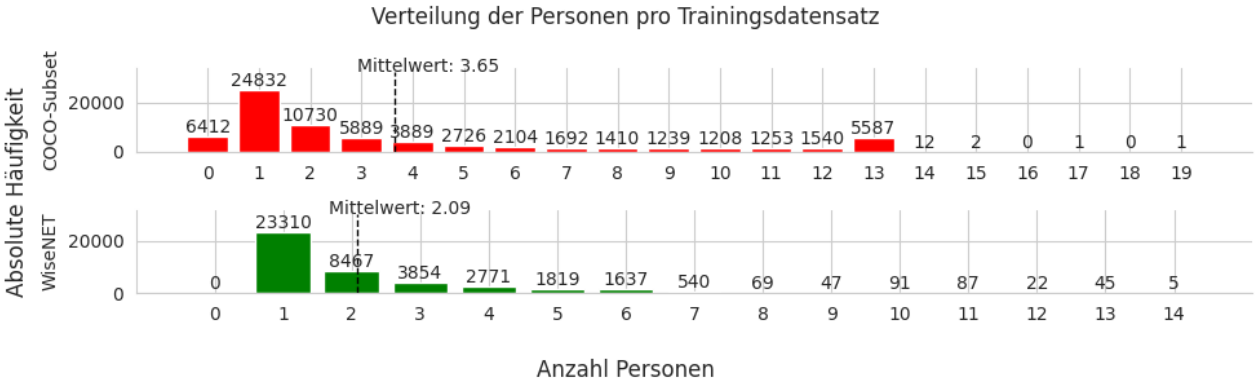


Abbildung A.1: Verteilung der Anzahl Personen pro Bild in den Trainingsdatensätzen. Im Gegensatz zu Grafik 5.3 ist der Labordatensatz nicht enthalten, da dieser aufgrund seiner Größe nicht zum Training verwendet wurde.

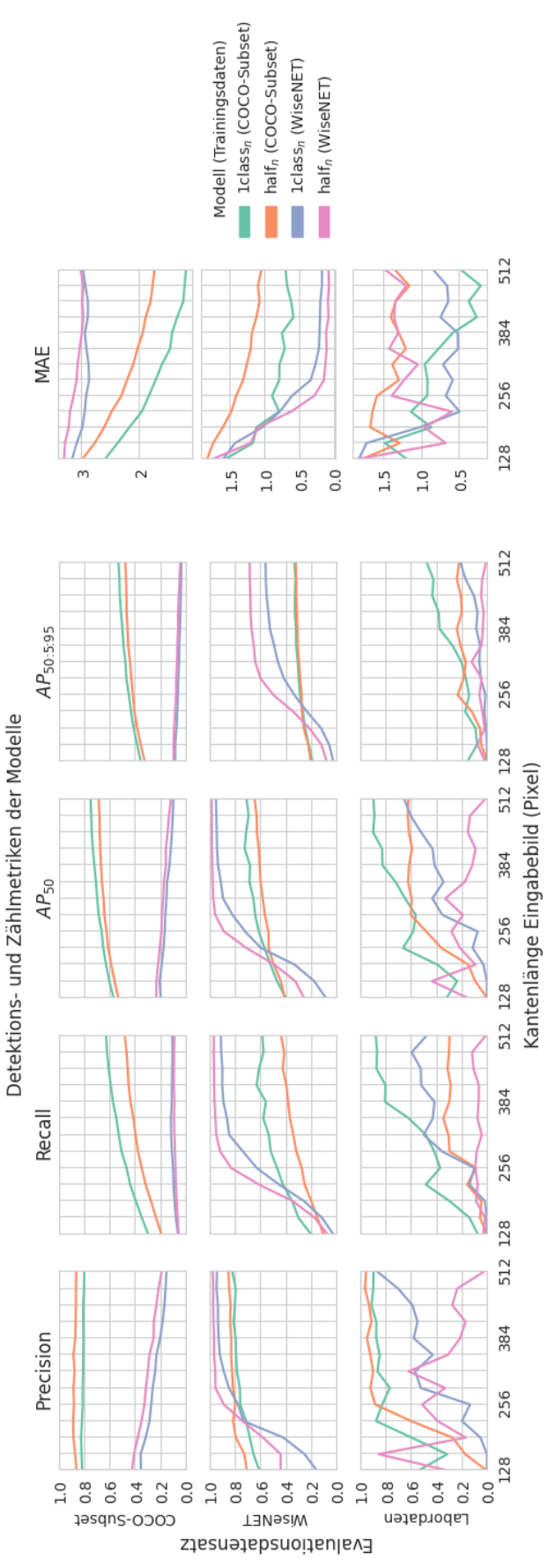


Abbildung A.2: Metriken aller auf allen Datensätzen trainierter Modelle. Es sind sowohl die Abhängigkeiten der Werte von der Bildauflösung als auch die Unterschiede zwischen den Auswirkungen der Trainingsdatensätze erkennbar.

LITERATUR

[1] Misbah Ahmad, Imran Ahmed, Kaleem Ullah, Iqbal khan, Ayesha Khattak und Awais Adnan. "Person Detection from Overhead View: A Survey". In: *(IJACSA) International Journal of Advanced Computer Science and Applications*, 2019.

[2] Yali Amit, Pedro Felzenszwalb und Ross Girshick. "Object Detection". In: *Computer Vision: A Reference Guide*. Hrsg. von Katsushi Ikeuchi. Cham: Springer International Publishing, 2021, S. 875–883. ISBN: 978-3-030-63416-2. DOI: 10.1007/978-3-030-63416-2_660. URL: https://doi.org/10.1007/978-3-030-63416-2_660.

[3] L. N. Andrianaivo, R. D'Autilia und V. Palma. "ARCHITECTURE RECOGNITION BY MEANS OF CONVOLUTIONAL NEURAL NETWORKS". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W15 (2019), S. 77–84. DOI: 10.5194/isprs-archives-XLII-2-W15-77-2019. URL: https://isprs-archives.copernicus.org/articles/XLII-2-W15/77/2019/.

[4] Alexey Bochkovskiy, Chien-Yao Wang und Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *CoRR* abs/2004.10934 (2020). arXiv: 2004.10934. URL: https://arxiv.org/abs/2004.10934.

[5] Sara Bouraya und Abdessamad Belangour. "Deep Learning based Neck Models for Object Detection: A Review and a Benchmarking Study". In: *International Journal of Advanced Computer Science and Applications* 12.11 (2021). DOI: 10.14569/IJACSA.2021.0121119. URL: http://dx.doi.org/10.14569/IJACSA.2021.0121119.

[6] Maximilian Büscher, Amina Gutjahr, Gerrit Hornung, Stephan Schindler, Annika Selzer, Indra Spiecker, Sarah Stummer, Thomas Wilmer und Paul Zurawski. *Verhaltensbeeinflussung durch Beobachtung? Eine explorative Nutzerbefragung zu den Auswirkungen verschiedener Beobachtungssituationen*. Techn. Ber. Darmstadt: Nationales Forschungszentrum für angewandte Cybersicherheit ATHENE, 2022.

[7] Raul Camposano und Jörg Wilberg. *Embedded system design*. Bd. 1. 1. 1996, S. 5–50. DOI: 10.1007/BF00134682. URL: https://doi.org/10.1007/BF00134682.

[8] Torch Contributors. *Models and pre-trained weights*. https://pytorch.org/vision/main/models.html. Online: letzter Zugriff am 18.11.2024. 2024.

[9] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan und Lei Zhang. "Dynamic Head: Unifying Object Detection Heads with Attentions". In: *CoRR* abs/2106.08322 (2021). arXiv: 2106.08322. URL: https://arxiv.org/abs/2106.08322.

[10] Robert David u. a. "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems". In: *Proceedings of Machine Learning and Systems*. Hrsg. von A. Smola, A. Dimakis und I. Stoica. Bd. 3. 2021, S. 800–811. URL: https://proceedings.mlsys.org/paper_files/paper/2021/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf.

[11] Luca Del Pizzo, Pasquale Foggia, Antonio Greco, Gennaro Percannella und Mario Vento. "Counting people by RGB or depth overhead cameras". In: *Pattern Recogn. Lett.* 81.C (Okt. 2016), 41–50. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2016.05.033. URL: https://doi.org/10.1016/j.patrec.2016.05.033.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li und Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, S. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[13] Piotr Dollar, Christian Wojek, Bernt Schiele und Pietro Perona. "Pedestrian detection: A benchmark". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, S. 304–311. DOI: 10.1109/CVPR.2009.5206631.

[14] Piotr Dollár, Ron Appel, Serge Belongie und Pietro Perona. "Fast Feature Pyramids for Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8 (2014), S. 1532–1545. DOI: 10.1109/TPAMI.2014.2300479.

[15] Piotr Dollár, Christian Wojek, Bernt Schiele und Pietro Perona. "Pedestrian Detection: An Evaluation of the State of the Art". In: *PAMI* 34 (2012).

[16] *ESP32-CAM Development Board*. https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf. Online; letzter Zugriff am 25.10.2024. 2024.

[17] *ESP32-S Wi-Fi+BT SoC Module*. https://fcc.report/FCC-ID/2AHMR-ESP32S/4169856.pdf. Online; letzter Zugriff am 25.10.2024. 2017.

[18] Omar Elharrouss, Younes Akbari, Noor Almadeed und Somaya Al-Maadeed. "Backbones-review: Feature extractor networks for deep learning and deep reinforcement learning approaches in computer vision". In: *Comput. Sci. Rev.* 53.C (Nov. 2024). ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2024.100645. URL: https://doi.org/10.1016/j.cosrev.2024.100645.

[19] Free Software Foundation. *Why the Affero GPL*. https://www.gnu.org/licenses/why-affero-gpl.html. Online; letzter Zugriff am 16.11.2024. 2022.

[20] *Frequently Asked Questions about the GNU Licenses - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/gpl-faq.html#GPLOutput. Online; letzter Zugriff am 07.10.2024. 2024.

[21] *GNU Affero General Public License - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/agpl-3.0.html.en. Online; letzter Zugriff am 07.10.2024. 2007.

[22] Ross B. Girshick. "Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell und Jitendra Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, S. 580–587. DOI: 10.1109/CVPR.2014.81.

[24] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu und Jason Yosinski. "Faster Neural Networks Straight from JPEG". In: *Advances in Neural Information Processing Systems*. Hrsg. von S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi und R. Garnett. Bd. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/7af6266cc52234b5aa339b16695f7fc4-Paper.pdf.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren und Jian Sun. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. Techn. Ber. structure. 2015.

[26] Cornelius Held, Julia Krumm, Petra Markel und Ralf P Schenke. "Intelligent video surveillance". In: *Computer* 45.3 (2012), S. 83–84.

[27] Geoffrey Hinton, Oriol Vinyals und Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML]. URL: https://arxiv.org/abs/1503.02531.

[28] Jan Hosang, Rodrigo Benenson und Bernt Schiele. " Learning Non-maximum Suppression ". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Juli 2017, S. 6469–6477. DOI: 10.1109/CVPR.2017.685. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.685.

[29] Andrew Howard u. a. " Searching for MobileNetV3 ". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2019, S. 1314–1324. DOI: 10.1109/ICCV.2019.00140. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00140.

[30] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally und Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size". In: *CoRR* abs/1602.07360 (2016). arXiv: 1602.07360. URL: http://arxiv.org/abs/1602.07360.

[31] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam und Dmitry Kalenichenko. "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". In: *2018 IEEE/CVF Conference on Computer*

Vision and Pattern Recognition. 2018, S. 2704–2713. DOI: 10.1109/CVPR.2018.00286.

[32] Jun-Woo Jang u. a. "Sparsity-Aware and Re-configurable NPU Architecture for Samsung Flagship Mobile SoC". In: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA) |*. IEEE, 2021.

[33] Glenn Jocher u. a. *ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation*. Version v7.0. Nov. 2022. DOI: 10.5281/zenodo.7347926. URL: https://doi.org/10.5281/zenodo.7347926.

[34] Steven J. Johnston, Philip J. Basford, Colin S. Perkins, Herry Herry, Fung Po Tso, Dimitrios Pezaros, Robert D. Mullins, Eiko Yoneki, Simon J. Cox und Jeremy Singer. "Commodity single board computer clusters and their applications". In: *Future Generation Computer Systems* 89 (2018), S. 201–212. ISSN: 0167-739X. DOI: 10.1016/j.future.2018.06.048. URL: https://www.sciencedirect.com/science/article/pii/S0167739X18301833.

[35] Bernd Jähne. *Digitale Bildverarbeitung. Und Bildgewinnung*. 8. Auflage 2024. Berlin, Heidelberg: Springer Berlin Heidelberg, 2024. 1707 S. ISBN: 9783662595107.

[36] Donghyun Kang, TaeYoung Kang und Junkyu Jang. "Papers with code or without code? Impact of GitHub repository usability on the diffusion of machine learning research". In: *Information Processing and Management* 60.6 (Nov. 2023), S. 103477. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2023.103477.

[37] Chloe Eunhyang Kim, Mahdi Maktab Dar Oghaz, Jiri Fajtl, Vasileios Argyriou und Paolo Remagnino. "A Comparison of Embedded Deep Learning Methods for Person Detection". In: *CoRR* abs/1812.03451 (2019). arXiv: 1812.03451. URL: http://arxiv.org/abs/1812.03451.

[38] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Hrsg. von F. Pereira, C.J. Burges, L. Bottou und K.Q. Weinberger. Bd. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[39] Y. LeCun, L. Bottou, Y. Bengio und P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. DOI: 10.1109/5.726791.

[40] Bo Li, Hongbo Huang, Ang Zhang, Pei-Chi Liu und Chengfeng Liu. "Approaches on crowd counting and density estimation: a review". In: *Pattern Analysis and Applications* 24 (2021), S. 853 –874. DOI: 10.1007/s10044-021-00959-z. URL: https://api.semanticscholar.org/CorpusID:233942407.

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár und C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Hrsg. von David Fleet, Tomas Pajdla, Bernt Schiele und Tinne Tuytelaars. Cham: Springer International Publishing, 2014, S. 740–755.

[42] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu und Alexander C. Berg. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Hrsg. von Bastian Leibe, Jiri Matas, Nicu Sebe und Max Welling. Cham: Springer International Publishing, 2016, S. 21–37. ISBN: 978-3-319-46448-0.

[43] Raspberry Pi Ltd. *Raspberry Pi Zero 2 W*. https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf. Online; letzter Zugriff am 13.09.2024. 2024.

[44] *Luckfox Pico Mini A RV1103 Linux Micro Development Board, Integrates ARM Cortex-A7/RISC-V MCU/NPU/ISP Processors, onboard 64MB memory*. https://www.luckfox.com/Luckfox-Pico-Mini-A. Online; letzter Zugriff am 13.09.2024. 2024.

[45] Simone Luetto. "People counting using detection networks and self calibrating cameras on edge computing". Masterarbeit. Politecnico di Torino, 2019.

[46] Filip Malawski. "Top-view people counting in public transportation using Kinect". In: *Challenges of Modern Technology* 5 (2014). URL: https://api.semanticscholar.org/CorpusID:38535489.

[47] Roberto Marroquin, Julien Dubois und Christophe Nicolle. "WiseNET: An indoor multi-camera multi-space dataset with contextual information and annotations for people detection and tracking". In: *Data in Brief* 27 (Dez. 2019), S. 104654. ISSN: 2352-3409. DOI: 10.1016/j.dib.2019.104654.

[48] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen und Tijmen Blankevoort. "A White Paper on Neural Network Quantization". In: *CoRR* abs/2106.08295 (2021). arXiv: 2106.08295. URL: https://arxiv.org/abs/2106.08295.

[49] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov und Shan Carter. "An Overview of Early Vision in InceptionV1". In: *Distill* (2020). https://distill.pub/2020/circuits/early-vision. DOI: 10.23915/distill.00024.002.

[50] Chris Olah, Alexander Mordvintsev und Ludwig Schubert. "Feature Visualization". In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

[51] Rafael Padilla, Sergio L. Netto und Eduardo A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, S. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[52] *RKNN Inference Test | LUCKFOX WIKI*. https://wiki.luckfox.com/Luckfox-Pico/Luckfox-Pico-RKNN-Test/. Online; letzter Zugriff am 30.09.2024. 2024.

[53] Joseph Redmon, Santosh Divvala, Ross Girshick und Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, S. 779–788. DOI: 10.1109/CVPR.2016.91.

[54] Shaoqing Ren, Kaiming He, Ross B. Girshick und Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[55] Ltd. Rockchip Electronics Co. *Rockchip RV1103 Datasheet*. https://files.luckfox.com/wiki/Luckfox-Pico/PDF/Rockchip%20RV1103%20Datasheet%20V1.1-20220427.pdf. Online; letzter Zugriff am 13.09.2024. 2024.

[56] Oskar Rudolf. "Evaluation vortrainierter neuronaler Netzwerke zur Anwendung auf die autonome Zählung von Personen mit Objektdetektion". Masterarbeit. Hochschule Darmstadt, Mai 2024.

[57] Ali Saffari, Sin Yong Tan, Mohamad Katanbaf, Homagni Saha, Joshua R. Smith und Soumik Sarkar. "Battery-Free Camera Occupancy Detection System". In: *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*. MobiSys '21. ACM, Juni 2021. DOI: 10.1145/3469116.3470013.

[58] Tsu-Chuan Shen und Edward T.-H. Chu. "Edge-Computing-Based People-Counting System for Elevators Using MobileNet-Single-Stage Object Detection". In: *Future Internet* 15.10 (2023), S. 337.

[59] Beril Sirmacek und Maria Riveiro. "Occupancy Prediction Using Low-Cost and Low-Resolution Heat Sensors for Smart Offices". In: *Sensors* 20.19 (Sep. 2020), S. 5497. ISSN: 1424-8220. DOI: 10.3390/s20195497.

[60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke und Andrew Rabinovich. " Going deeper with convolutions ". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Juni 2015, S. 1–9. DOI: 10.1109/CVPR.2015.7298594. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594.

[61] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[62] *The GNU General Public License v3.0 - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/gpl-3.0.html.en. Online; letzter Zugriff am 07.10.2024. 2007.

[63] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers und A. W. M. Smeulders. "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104.2 (Apr. 2013), S. 154–171. ISSN: 1573-1405. DOI: 10.1007/s11263-013-0620-5.

[64] *Ultralytics YOLOv5 Architecture*. https://docs.ultralytics.com/yolov5/tutorials/architecture_description. Online; letzter Zugriff am 21.11.2024.

[65] Umweltbundesamt. *Indikator: Energieverbrauch für Gebäude*. https://www.umweltbundesamt.de/daten/umweltindikatoren/indikator-energieverbrauch-fuer-gebaeude. Online; letzter Zugriff am 25.11.2024. 2024.

[66] Chien-Yao Wang, I-Hau Yeh und Hong-Yuan Mark Liao. "You Only Learn One Representation: Unified Network for Multiple Tasks". In: *CoRR* abs/2105.04206 (2021). arXiv: 2105.04206. URL: https://arxiv.org/abs/2105.04206.

[67] Tao Wang, Li Yuan, Xiaopeng Zhang und Jiashi Feng. "Distilling Object Detectors With Fine-Grained Feature Imitation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, S. 4928–4937. DOI: 10.1109/CVPR.2019.00507.

[68] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo und Ross Girshick. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[69] Klaus Wüst. *Mikroprozessortechnik*. Vieweg+Teubner Verlag Wiesbaden, 2006. ISBN: 978-3-8348-9084-9. DOI: 10.1007/978-3-8348-9084-9.

[70] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar und Brian Lee. "A survey of modern deep learning based object detection models". In: *Digital Signal Processing* 126 (2022), S. 103514. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2022.103514. URL: https://www.sciencedirect.com/science/article/pii/S1051200422001312.

[71] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao und Yi Ma. "Single-Image Crowd Counting via Multi-Column Convolutional Neural Network". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, S. 589–597. DOI: 10.1109/CVPR.2016.70.

[72] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu und Xindong Wu. "Object Detection with Deep Learning: A Review". In: *CoRR* abs/1807.05511 (2018). arXiv: 1807.05511. URL: http://arxiv.org/abs/1807.05511.

[73] *pjreddie/darknet: Convolutional Neural Networks*. https://github.com/pjreddie/darknet. Online; letzter Zugriff am 18.11.2024.

[74] Google AI Edge team. *TensorFlow Lite is now LiteRT*. https://developers.googleblog.com/en/tensorflow-lite-is-now-litert/. Online; letzter Zugriff am 04.11.2024. 2024.