HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES

Erkennung und Klassifizierung elektronischer Mobilgeräte sowie der darin enthaltenen Batterien auf X-Ray-Aufnahmen mittels Transfer Learning

Bachelorarbeit
im Modul Neuroinformatik
Studiengang Wirtschaftsinformatik
der Hochschule Ruhr West

David Rohrschneider
100012344

Erstprüfer:	Prof. Dr. Uwe Handmann
Zweitprüferin:	Nermeen Abou Baker
Betreuerin:	Nermeen Abou Baker

Bottrop, Juli 2022

Kurzfassung

Mit dem wachsenden Konsum elektronischer Mobilgeräte steigen auch die Gefahrenpotenziale, welche aus den in ihnen enthaltenen Lithium-Ionen-Batterien resultieren. Ob bei der Sortierung von Batterien in Recyclinghöfen, oder bei Sicherheitskontrollen an Flughäfen: Die Nachfrage der autonomen Erkennung von Batterien in elektronischem Müll oder Gepäck der Passagiere steigt.

In der vorliegenden Bachelorarbeit wird deshalb der aktuelle Stand der Technik folgender Thematik dargelegt: Erkennung und Klassifizierung elektronischer Mobilgeräte sowie der darin enthaltenen Batterien auf X-Ray-Aufnahmen mittels Transfer Learning. Zunächst wird dabei auf die aktuellen Möglichkeiten im Bereich Machine Learning, sowie letzte Veröffentlichungen bzgl. ähnlicher Thematik eingegangen. Um den Stand der Technik zu verbessern, werden daraufhin mehrere Versuche mit dem aktuell präzisesten Machine-Learning-Modell zur Echtzeit-Objekterkennung „YOLOv5“ und einem umfassenden Datensatz namens „HiXray“ durchgeführt. Der Gebrauch vom Konzept Transfer Learning und dessen Effekt auf die Versuchsreihe wird im Laufe der Arbeit immer wieder angeschnitten. Die Ergebnisse des Experiments zeigen: Mit YOLOv5 ist zwar noch keine vollständig autonome Erkennung elektronischer Mobilgeräte und derer Batterien auf X-Ray-Aufnahmen möglich, jedoch konnte unter Nutzung von Transfer Learning der Stand der Technik verbessert werden. Weitere Forschung in diesem Bereich könnten diese aber bereits in naher Zukunft ermöglichen, wodurch Sicherheitsrisiken minimiert und diverse Prozesse an Sicherheitskontrollen oder Recyclinghöfen automatisiert werden könnten.

Schlagwörter: Objekterkennung, X-Ray, Mobilgeräte, Lithium-Ionen-Batterie, Transfer Learning, Recycling, Sicherheit, YOLOv5

Abstract

With the increasing need for mobile devices, the autonomous detection of Lithium-Ion batteries gets more important with respect to recycling processes or airport security.

The following work deals with the current state of the art of the topic: Detection and classification of electronic devices and their batteries on X-Ray images using Transfer Learning. After exposing current opportunities and papers about this topic, the goal is to improve the state of the art using the object detection model “YOLOv5” and a relevant dataset called “HiXray”. The results show that the state of the art can be surpassed while taking advantage of the benefits of Transfer Learning. Upcoming work may lead to an autonomous detection of electronic devices and Lithium-Ion batteries on X-Ray images.

Keywords: object detection, X-Ray, mobile devices, battery, Transfer Learning

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

LIB	Lithium-Ionen-Batterie
EMG	Elektronisches Mobilgerät
NN	Neuronales Netzwerk
SOTA	State-Of-The-Art
AP	Average Precision
MAP	Mean Average Precision
TP	True-Positive
TN	True-Negative
FP	False-Positive
FN	False-Negative
IOU	Intersection Over Union
GCP	Google Colaboratory Pro

Vorwort

Diese Bachelorarbeit dient als Abschlussarbeit meines Studiums im Fach Wirtschaftsinformatik an der Hochschule Ruhr West und befasst sich mit dem Stand der Technik des folgenden Bereiches:

Erkennung und Klassifizierung elektronischer Mobilgeräte sowie der darin enthaltenen Batterien auf X-Ray-Aufnahmen mittels Transfer Learning.

Darin soll zu ähnlichen Veröffentlichungen Bezug genommen werden und mit der Ausführung eines Experiments neue Erkenntnisse, Problematiken und zukünftige Aussichten dieses Forschungsgebietes geschildert werden.

Durch die Seminare im Wahlmodul Neuroinformatik entwickelte ich ein Eigeninteresse für den Bereich Machine-Learning, sowie Objekterkennung mittels neuronaler Netze und in Zusammenarbeit mit meiner Betreuerin Nermeen Abou Baker traf ich die Entscheidung, mich mit o.g. Thematik auseinanderzusetzen.

Das beschriebene Experiment nahm aufgrund des Planungs- und Durchführungsaufwandes viel Zeit in Anspruch, jedoch förderte es nicht nur meine Kenntnisse bezüglich Machine Learning, sondern auch mein generelles Interesse an dessen aktueller Forschung und Möglichkeiten.

Im Zuge dieser Arbeit haben Nermeen Abou Baker und ich eine wissenschaftliche Abhandlung über selbiges Thema in englischer Sprache verfasst (Baker et al., 2022), welche am 16.05.2022 bei der ESANN 2022 (d-side conferences, o. D.) eingereicht wurde.

ESANN steht für „European Symposium on Artificial Neural Networks“ und ist eine europäische Konferenz, in welcher aktuelle Publikationen aus dem Bereich Machine Learning vorgestellt und diskutiert werden können. Dieses Jahr findet sie vom 05.10 bis zum 07.10. sowohl in Belgiens Stadt Brügge als auch online statt.

Unsere Abhandlung wurde vom Komitee für die Präsentation eines Posters zugelassen, welches bei der Veranstaltung in Brügge ausgehängt wird.

Daher bedanke ich mich an dieser Stelle bei meiner Begleiterin für eine ständige und zuverlässige Anleitung und Unterstützung in den Monaten der Bearbeitung. Zusammen mit ihr und Prof. Dr.-Ing. Uwe Handmann konnte ich nicht nur neue Erkenntnisse im Rahmen o.g. Thematik erzielen, sondern darüber hinaus eine wissenschaftliche Abhandlung im Namen der Hochschule veröffentlichen, welche vom Komitee einer internationalen Konferenz anerkannt wurde.

David Rohrschneider

Bottrop, 27. Juni 2022

1 Einleitung

1.1 Problemstellung

Das aktuelle Zeitalter wird auch als das Zeitalter der Technik bezeichnet. Der Begriff findet seinen Ursprung in der Tatsache, dass die Digitalisierung einen Punkt erreicht hat, an dem sich der Einsatz moderner Technologien prägend auf den Alltag der Menschheit auswirkt.

Ein zentraler Meilenstein der technologischen Entwicklung war die Erfindung einer wiederaufladbaren Lithium-Ionen-Batterie (LIB), welche 1985 erstmals von der Firma Moli Energy, heute bekannt unter dem Namen E-One Moli Energy Corporation, entwickelt wurde. Doch aufgrund mehrerer Sicherheitsproblematiken wurde die Herstellung dieser Batterien eingestellt, bis das Unternehmen Sony 1991 eine verbesserte, sowie stabilere Variante der LIB auf den Markt brachte. Nach dieser Veröffentlichung vervielfältigte sich die Herstellung von LIB in mehreren Gebieten und sie fanden Anklang im alltäglichen Gebrauch von elektronischen Mobilgeräten (EMG) (Leuthner et al., 2013).

Heutzutage ist das Mitführen mobiler Endgeräte aufgrund ihrer vielfältigen Nutzbarkeit zur Normalität geworden, was sich beispielsweise durch die Verzehnfachung an verkauften Mobiltelefonen im Zeitraum 2007 bis 2014 auszeichnen lässt (Statista, 2022).

Auf der Kehrseite bergen die inzwischen hochentwickelten Geräte trotz ihres Ansehens stets einige Sicherheitsrisiken in sich, die meist auf das Vorhandensein einer oder mehrerer LIB zurückzuführen sind. Hierbei kann es laut Leuthner et al. (2013) durch verschiedene Auslöser, bspw. dem elektrischen oder thermischen Fehlgebrauch, sowie metallischer Verunreinigungen oder anderer Schädigungen an der Batterie, zum Anstieg der Zelltemperatur kommen. Durchaus können weitere darauffolgende Reaktionen zu weiterer Hitzeentwicklung und schließlich zu einem Zellbrand führen.

1.1.1 Problematik im Flugverkehr

Das Risiko einer unkontrollierten Erhitzung und ggf. eines Brandes in der LIB wird vor Allem an Orten zu einem Problem, an denen sich das Gerät nicht immer in unmittelbarer Nähe des Besitzers befindet und dieser somit nicht umgehend reagieren kann. Eine beispielhafte Situation ist das Vorhandensein eines derartigen mobilen Gerätes in einem Koffer am Flughafen oder sogar an Bord eines Flugzeuges.

Laut der Federal Aviation Administration (2022) ereigneten sich im April diesen Jahres 7 Unfälle, in denen LIB involviert waren und zu einem erhöhten Sicherheitsrisiko führten. Zwei dieser Vorfälle wurden an Bord von Frachtflügen festgestellt, die anderen fünf jeweils in einem Passagierflugzeug.

Das Gefahrenpotenzial einer LIB ist durchaus aktuell, weshalb unter anderem die European Union Aviation Safety Agency (EASA) explizit vor dem Mitführen mobiler Geräte in Flugzeugen warnt, da diese meistens derartige Batterien enthalten. Die EASA weist darauf hin, mobile Endgeräte ausschließlich im Handgepäck zu transportieren (siehe Abb. 1) (European Union Aviation Safety Agency, o. D.):

Abbildung 1: Lithiumbatterien Infoblatt EASA

Quelle: European Union Aviation Safety Agency, o. D.

Um die Sicherheit der Passagiere während eines Fluges zu gewährleisten, werden vor der Abreise die einzelnen Gepäckstücke kontrolliert. Üblicherweise werden dafür Röntgengeräte verwendet, wodurch Inhalte der Gepäckstücke auf Basis ihrer chemischen Dichte identifiziert werden können.

An dieser Stelle spielt die Separierung der EMG vom restlichen Handgepäck eine Rolle, denn die in diesen enthaltenen LIB besitzen aufgrund bestimmter Metalle eine hohe chemische Dichte. Die meisten LIB haben einen Anteil von 5%-20% Kobalt, 5%-10% Nickel, 5%-7% Lithium, sowie 5%-10% anderer Metalle wie Kupfer, Aluminium oder Eisen auf (Zheng et al., 2018). Das erhöhte Vorkommen solcher Stoffe macht sich in Röntgenaufnahmen durch eine dunklere Verfärbung im Bereich der Batterie bemerkbar, was dazu führt, dass darunter oder darüber platzierte Objekte ggf. nicht erkennbar für das zuständige Sicherheitspersonal sind.

Daraus folgt ein Sicherheitsproblem, weshalb größere EMG bspw. an deutschen Gepäckkontrollen laut der Bundespolizei in separate Gepäckwannen gelegt werden müssen (Bundespolizei, o.D.).

Falls aber doch ein Passagier seine EMG nicht separiert hat, muss der Sicherheitsbeauftragte diese manuell auf dem Bildschirm erkennen und ggf. das Gepäckstück erneut scannen, nachdem die EMG entfernt wurden.

1.1.2 Recycling elektronischer Abfälle

Recycling ist von großer Bedeutung für die globale Umwelt, was sich auch auf die Verwertung gebrauchter EMG und darin enthaltener LIB bezieht. Aus dem Fakt, dass sich die Nutzung EMG in den letzten Jahren deutlich vermehrt hat, lässt sich schließen, dass sich ebenso die Anzahl der Entsorgungen solcher Geräte vergrößert. Laut dem Statistischem Bundesamt (2022) wurden 2020 insgesamt 899.300 Tonnen Elektro- und Elektronikaltgeräte, 11,2% mehr als im Vorjahr, recycelt.

Um weiterhin die Wertschöpfung aus den wachsenden Abfällen gewährleisten zu können, entwarf die Europäische Kommission einen Aktionsplan für eine Kreislaufwirtschaft. In diesem wird mitunter „ein neuer Rechtsrahmen für Batterien zur Verbesserung der Nachhaltigkeit und zur Stärkung des Kreislaufpotenzials von Batterien“ (Europäische Kommission, 2020) angegeben.

Laut dem Europäischen Parlament muss Elektro- und Elektronikabfall recycelt werden, da sie potenziell schädliche Materialien enthalten. Diese stellen sowohl ein Risiko für die Umwelt als auch für Menschen, die am Recycling von Elektronikabfall beteiligt sind, dar (Europäisches Parlament, 2022).

Die Batterien werden in der Regel vor der Behandlung in Recyclinganlagen sortiert, was trotz des Risikos bspw. auf dem Recyclinghof Sortbat in Belgien zunächst händisch durchgeführt wird. Dabei werden Alkalin-, Nickel-Cadmium-, Nickelmetallhydrid-, sowie LIB manuell vom Förderband genommen, da diese einer separaten Recyclingprozedur unterliegen (siehe Abb. 2) (Sortbat, o.D.).

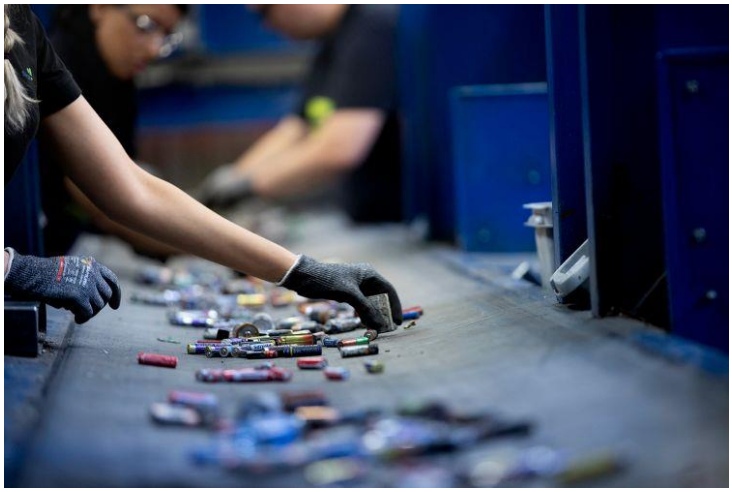

Abbildung 2: Händisches Sortieren von Batterien auf dem Recyclinghof Sortbat

Quelle: Sortbat, o.D.

In der Forschung von Sterkens et al. (2021) wird berichtet, dass vor dem Sortieren von Batterien bei jedem elektronischen Gerät zunächst klassifiziert werden muss, ob es Batterien enthält. Bei diesem Prozess gibt es einen Anteil an Geräten, bei denen sich das Entfernen von Batterien schwieriger gestaltet, da sie bspw. verklebt oder verschweißt worden sind. Um trotzdem eine Separierung sicherzustellen, werden teilweise grobe Methodiken wie dem Öffnen des Geräts mittels einer rotierenden Trommel oder eines Prallbrechers angewandt.

Daraus resultiert ein erheblicher manueller Zeit-Kosten-Aufwand, bei dem außerdem z.B. durch beschädigte LIB ein erhöhtes Gesundheitsrisiko für das Personal entstehen kann.

1.2 Ziel der Arbeit und Vorgehensweise

Die Erkennung von EMG ist also sowohl in aufgegebenem Gepäck zum Erhalt der Kontrolle über etwaige Sicherheitsrisiken, bspw. der starken Erhitzung durch beschädigte Teile der LIB, als auch im Handgepäck zur Prävention von darunter platzierten, potenziell gefährlichen Gegenständen unumgänglich und damit ein wichtiger Aspekt für die Gewährleistung der Sicherheit an Flughäfen.

Eine vollautomatisierte Separierung von LIB könnte auch beim Recycling nicht nur Risikofaktoren und dessen Folgekosten, sondern auch sämtliche Personalkosten einsparen. Darüber hinaus wäre sie im Sinne der EU-Kommission ein sinnvoller Beitrag zur Stärkung der Kreislaufwirtschaft und zur Vermeidung von Umwelt- und Personenschäden.

In dieser Arbeit wird die automatisierte Erkennung von EMG und der darin enthaltenen LIB auf X-Ray-Aufnahmen mittels Transfer Learning erforscht und getestet, sodass in Zukunft die Risiken in Bezug auf LIB vermindert werden können. Um den State-Of-The-Art (SOTA) zu verbessern, wird im Rahmen der Methodik ein aktueller Datensatz und eine der präzisesten, für Objekterkennung ausgelegten Deep Learning Architekturen verwendet.

Mit dieser technischen Grundlage gilt es, für ein zufälliges Bild des Datensatzes die folgenden drei zentralen Fragen zu beantworten:

1 Ist auf dem Bild ein EMG zu sehen?

2 Wenn ja, wo befindet sich die LIB?

3 Zu welcher der drei Klassen gehört die LIB (prismatische, flache oder zylindrische LIB)?

Im Laufe der Methodik wird an zwei Stellen vom Konzept Transfer Learning Gebrauch gemacht: Für die Erkennung fünf verschiedener EMG werden Initialgewichte verwen-

det, welche bereits mit einem größeren öffentlichen Datensatz vortrainiert worden sind. Dadurch starten die Gewichte des neuronalen Netzes (NN) nicht mit einer zufälligen Wertzuweisung, sondern die zuvor von einem größeren Datensatz erlernten Features können direkt auf die spezifischere Aufgabe der Erkennung von fünf EMG-Klassen auf X-Ray-Aufnahmen transferiert werden. Dadurch können ggf. auch bei einer geringen Anzahl an Trainingsepochen Resultate erzielt werden, die der Verbesserung des SOTA beitragen.

Nachdem mit diesen Initialgewichten das Training des NN zur Erkennung der fünf EMG-Klassen abgeschlossen ist, werden die daraus erhaltenen Gewichte wiederum als Initialgewichte für das Training zur Erkennung von drei verschiedenen LIB-Klassen genutzt. Auf diese Weise soll das gewonnene Wissen auf eine spezifischere Aufgabe übertragen und somit ein redundantes Neuerlernen allgemeiner Features aus X-Ray-Aufnahmen vermieden werden, wodurch sich die Anzahl der benötigten Trainingsepochen verringern kann.

Das NN soll zum Schluss in der Lage sein, die Batterien zu lokalisieren und einer von drei verschiedenen LIB-Klassen zuzuordnen. Um die bestmögliche Genauigkeit des NN zu erreichen, wird das Training in sieben verschiedenen Variationen durchgeführt, bei denen jeweils die Hyperparameter oder der Datensatz verändert werden. Nach jedem abgeschlossenen Training werden die erhaltenen Gewichte des NN durch eine Inferenz mit neuen Bildern des Testsatzes validiert und mithilfe gängiger Machine-Learning-Metriken, wie z.B. Accuracy oder Precision, verglichen.

Außerdem werden die besten erzielten Ergebnisse vor dem Hintergrund vergleichbarer Abhandlungen evaluiert, sodass festgestellt werden kann, ob der SOTA durch diese Arbeit verbessert werden konnte.

1.3 Aufbau der Arbeit

Der erste Teil der Arbeit beschäftigt sich mit dem Stand der Technik. Darin wird ein Überblick über generelle Ansätze zur Erkennung von Objekten mittels Deep Learning, relevante Open-Source-Modelle zur Echtzeit-Objekterkennung, öffentliche X-Ray-Datensätze, sowie aktuelle Veröffentlichungen im Rahmen der Erkennung von EMG und Batterien auf X-Ray-Bildern verschafft.

Das Experiment ist der Hauptbestandteil dieser Arbeit. In der Vorbereitung und Methodik wird erläutert, wie die Auswahl des Datensatzes, der Entwicklungsumgebung und des Deep-Learning-Modells zustande kam. Auf dessen Basis wird dann die Vorbereitung der Datensätze, sowie die Methodik des Experiments und der Nutzung von Transfer Learning schrittweise beschrieben.

Die dabei erzielten Ergebnisse werden unter Verwendung von Grafiken und Tabellen evaluiert und im Kontext des SOTA diskutiert. Um die Arbeit in den Stand der Technik eingliedern zu können, werden die Ergebnisse von zwei aktuellen Veröffentlichungen ähnlicher Thematik aufgegriffen.

Zum Abschluss werden die Ergebnisse des Experiments, sowie dessen Bedeutung für den Stand der Technik und die o.g. Anwendungsbeispiele, zusammengefasst. Darüber hinaus erfolgt ein kurzer Ausblick auf mögliche Strategien zur Verbesserung dieser Forschung.

Während Methodik, Diskussion und Zusammenfassung des Experiments werden jeweils immer wieder Integration und Einfluss von Transfer Learning aufgegriffen, um dessen Vorteile bei der Erkennung von EMG und LIB auf X-Ray-Aufnahmen zu konkretisieren.

2 Stand der Technik

2.1 Deep-Learning-Architekturen zur Echtzeit-Objekterkennung

Die Funktionsweise der bis heute veröffentlichten Modelle zur Objekterkennung auf Bildern variiert von Modell zu Modell, wobei die grundsätzliche Vorgehensweise gleichbleibt: Das Eingabebild wird in mehrere Bereiche unterteilt, welche jeweils von verschiedenen Klassifikatoren auf das Vorhandensein einer bestimmten Objektklasse bewertet werden. Auf diese Weise kann dann auf Basis der resultierenden Wahrscheinlichkeiten für die erkannten Klassen in jedem Bereich entschieden werden, um welche Objektklasse es sich am wahrscheinlichsten handelt, oder ob keine der gesuchten Objektklassen vorzufinden sind (Liu et al., 2019).

Die SOTA-Modelle verwenden verschiedene Architekturen, wobei für die Echtzeit-Objekterkennung bestimmte Faktoren abzuwägen sind.

Dazu gehört bspw. die Geschwindigkeit der verarbeiteten Bilder, oftmals gemessen in Bildern pro Sekunde (Englisch: frames per second oder fps). Es existiert keine allgemeine Grenze, ab der die Geschwindigkeit des Modells als Echtzeit-Objekterkennung eingestuft werden kann, in manchen wissenschaftlichen Beiträgen wie Liu et al. (2019) wird jedoch ein Minimum von 30fps verwendet.

Außerdem ist die Genauigkeit des Modells ein entscheidender Faktor bei der Abwägung, welche unter Anderem anhand der sogenannten Average Precision (AP) oder Mean Average Precision (MAP) bemessen werden kann. Diese Metrik wird zu Vergleichszwecken anhand von öffentlichen Datensätzen, wie dem MS COCO Datensatz (Lin et al., 2014), ermittelt. Laut W. Liu et al. (2016) sind durchschnittlich im MS COCO Datensatz kleinere Objekte zu identifizieren, was durchaus bei der Beurteilung von Modellen zur Objekterkennung zu berücksichtigen ist.

2.1.1 Zweistufige/Einstufige Modelle zur Objekterkennung

Generell kann aktuell zwischen zwei Oberklassen von Modellen zur Objekterkennung unterschieden werden. In der zweistufigen Objekterkennung werden zunächst Vorschläge für auffällige Regionen des Eingabebildes generiert und dann mittels Regression klassifiziert (Lu et al., 2020). Zu dieser Art gehören bspw. R-CNN und dessen Nachfolgermodelle, sowie Feature Pyramid Networks (FPN).

Einstufige Modelle führen die Regression und Klassifikation direkt auf dem Eingabebild aus und lassen den ersten Schritt der Separierung auffälliger Regionen weg. Beispiele dieser Art sind die Modelle You Only Look Once v5 (YOLOv5) (Ultralytics, o. D.), Single Shot MultiBox Detector (SSD) (W. Liu et al., 2016), oder EfficientDet (Tan et al., 2020).

Laut der Forschung von Lu et al. (2020) erhöht das Separieren auffälliger Regionen des Bildes bei zweistufigen Modellen die Genauigkeit der Objekterkennung, sie sind aber strukturell komplexer und benötigen mehr Zeit zur Berechnung. Daher eignen sich einstufige Modelle aufgrund ihrer kompakten Struktur und Geschwindigkeit eher zur Echtzeit-Objektorientierung.

In der nachfolgenden Tabelle sind die fünf Modelle Faster R-CNN (Ren et al., 2017), Mask R-CNN (He et al., 2017), SSD512, EfficientDet-D3 und YOLOv5m basierend auf Ihren Ergebnissen auf dem MS COCO Datensatz zum Vergleich aufgelistet.

Tabelle 1: Vergleich fünf ausgewählter SOTA-Modelle zur Objekterkennung

	Faster R-CNN	**Mask R-CNN**	**SSD512**	**Efficient-Det-D3**	**YOLOv5m**
MAP	42,70%	62,30%	48,50%	65,90%	64,10%
Geschwindigkeit	5fps	5fps	22fps	36fps	121fps
Erscheinungs-jahr	2016	2018	2016	2020	2021

Tabelle 1 ist zu entnehmen, dass EfficientDet-D3 die höchste MAP und YOLOv5m die höchste Geschwindigkeit der ausgewählten Modelle aufweist. Da sich YOLOv5m im Vergleich der Geschwindigkeiten deutlich von den anderen abhebt und die MAP lediglich 1,8% niedriger als bei EfficientDet-D3 ist, verfügt es unter den ausgewählten SOTA-Modellen über die beste Balance zwischen Präzision und Geschwindigkeit.

2.1.2 YOLOv5

Die neueste von Ultralytics (o.D) geschaffene SOTA-Architektur gehört zu der Gruppe der Einstufigen Modelle zur Objekterkennung, wodurch es an Komplexität und Rechenkapazität spart. Sie besteht aus den folgenden drei Hauptkomponenten:

1 Backbone

YOLOv5 nutzt als Backbone-Architektur das Cross Stage Partial Network (CSPNet), wodurch laut Wang et al. (2020) ein Rechenaufwand von bis zu 20% gegenüber anderen und somit auch in älteren Versionen von YOLO vorkommenden Backbone-Strukturen erzielt werden können. Die Backbone-Komponente ist größtenteils zur Extraktion von Features aus dem Eingabebild da (Ultralytics, 2022).

2 Neck

In der mittleren Komponente der Architektur werden sogenannte Feature-Pyramiden generiert, welche dem Modell dabei helfen Objekte aus verschiedenen Perspektiven zu erkennen. In YOLO5 wurde in dieser Ebene das Spatial Pyramid Pooling (He et al., 2014) und die Struktur des Path Aggregation Networks (S. Liu et al., 2018) aus den Vorgängerversionen von YOLO verschnellert (Ultralytics, 2022).

3 Head

Der letzte Teil der Architektur generiert die jeweiligen Ausgabevektoren mit den Wahrscheinlichkeiten für jede Klasse und den zugehörigen Bounding-Boxen (siehe Glossar). Er besitzt dieselbe Struktur wie die bereits in YOLOv3 verbaute Head-Komponente (J. Redmon et al., 2018).

Ultralytics (2022) brachte mit der Veröffentlichung von YOLOv5 fünf verschiedene Größen des Modells heraus, welche in Komplexität, Schnelligkeit und Präzision variieren (siehe Abb. 3).

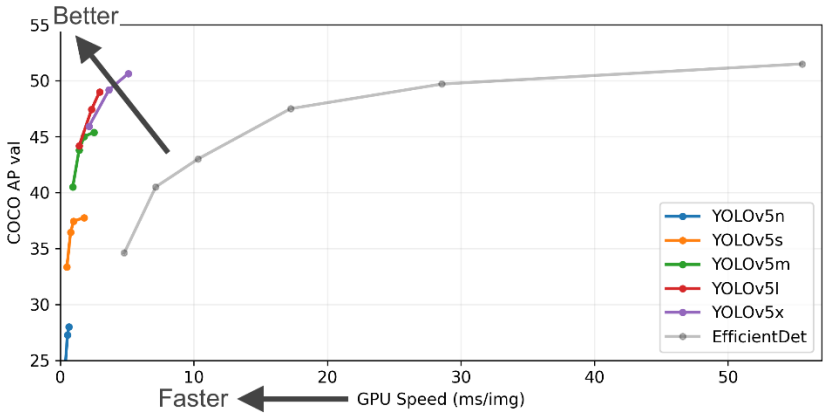

Abbildung 3: Vergleich der fünf YOLOv5-Modelle und EfficientDet

Quelle: Ultralytics (2022)

Der in Abbildung 3 gezeigte Graph beschreibt die durchschnittliche Präzision der Modelle in Prozent, getestet auf dem Validation-Set des COCO-Datensatzes, in Abhängigkeit von der Geschwindigkeit in Millisekunden pro Bild (ms/img), gemessen auf einer Amazon EC2 P3 GPU-Instanz. Je höher der Wert der Präzision auf der Y-Achse und je niedriger der Wert der Geschwindigkeit auf der X-Achse ist, desto besser ist der jeweilige Wert.

Hieraus wird ersichtlich, dass sich die drei Modelle YOLOv5m, YOLOv5l und YOLOv5x in einer ähnlichen Region befinden. YOLOv5x erreicht bei einer Geschwindigkeit von ca. 5 ms/img die höchste durchschnittliche Präzision der YOLO-Modelle mit ca. 51%. Im Bereich von 44% bis 47% weisen die drei o.g. Modelle Überschneidungen auf, obwohl der Komplexitätsgrad der Modelle unterschiedlich ist. Vor der Wahl des geeigneten Modells für eine Aufgabe sollte daher zunächst die verfügbare Rechenkapazität der verwendeten Maschine mit den Ansprüchen bzgl. Präzision und Geschwindigkeit abgewogen werden, da sich die Resultate möglicherweise nur marginal unterscheiden.

2.2 Öffentliche X-Ray-Datensätze mit elektronischen Geräten

In einem Benchmark über aktuell veröffentlichte X-Ray-Datensätze in Bezug auf Objekterkennung wurden die Datensätze namens SIXray (Miao et al., 2019), OPIXray (Wei et al., 2020), PIDray (B. Wang et al., 2021), HiXray (Tao et al., 2021) und GDXray Baggage (Mery et al., 2015) identifiziert. Der PIDray-Datensatz beinhaltet X-Ray-Aufnahmen mit künstlich eingefügten Gegenständen, die anderen ausschließlich originale X-Ray-Aufnahmen von Sicherheitskontrollen an Flughäfen oder Bahnhöfen.

Die Erkennung von EMG und LIB auf X-Ray-Aufnahmen mit bspw. Gepäckstücken aus einer Sicherheitskontrolle mittels Deep Learning setzt voraus, dass ein Datensatz mit passenden Objektklassen existiert, welche auf das Problem der Erkennung von EMG und LIB anwendbar ist. Außerdem sollte er über eine ausreichend große Anzahl an Daten verfügen, da die in EMG enthaltenen Batterien in Relation zu umliegenden Objekten kleiner erscheinen und daher von Deep-Learning-Modellen schwerer zu identifi-

zieren sind. Um einen Überblick über die o.g. X-Ray-Datensätze zur Erkennung von Objekten zu schaffen, zeigt die folgende Tabelle deren Anzahl an Bildern und darin enthaltene Klassen im Vergleich:

Tabelle 2: Öffentliche X-Ray-Datensätze (EMG-Klassen gelb markiert)

	Anzahl Bilder	**Klassen**
SIXray	1.059.231	Gun, Knife, Wrench, Pliers, Scissors, Hammer
OPIXray	8.885	Folding Knife, Straight Knife, Scissors, Utility Knife, Multi-tool Knife
PIDray	47.677	Gun, Bullet, Knife, Wrench, Pliers, Power-bank, Baton, Lighter, Sprayer, Hammer, Scissors, Handcuffs
HiXray	102.928	Portable_Charger_1, Portable_Charger_2, Mobile_Phone, Laptop, Tablet, Cosmetic, Water, Nonmetallic_Lighter
GDXray Baggage	8.150	Handgun, Razor Blade, Shuriken, Pen Case, Clip, Spring, Door Key, Knife

Wie Tabelle 2 zu entnehmen ist, beinhalten nur zwei der Datensätze mindestens eine EMG-Klasse: PIDray und HiXray. Da sowohl die Anzahl der Bilder als auch die Anzahl der EMG-Klassen im HiXray-Datensatz herausstechen, ist dieser für die Erkennung von EMG mittels Deep Learning als SOTA zu betrachten.

2.3 Objekterkennung auf X-Ray-Bildern

Die Forschung, speziell bezogen auf die Erkennung von EMG oder Batterien auf X-Ray-Aufnahmen, ist bisher in zwei Veröffentlichungen behandelt worden.

2.3.1 Erkennung von EMG

In der Abhandlung von Tao et al. (2021) wird zum einen der HiXray-Datensatz und zum anderen das von ihnen entwickelte Lateral Inhibition Module vorgestellt.

Darin wird erwähnt, dass die Annotation der Bilder durch professionelles und geschultes Flughafenpersonal im Stil der Pascal-VOC-Schreibweise (Everingham et al., 2009) erfolgte, wodurch die Konstanz der Beschriftungen sichergestellt wurde.

HiXray umfasst 8 verschiedene Objektklassen: „Portable Charger 1“, „Portable Charger 2“ (PO1 & PO2), „Water“ (WA), „Laptop“ (LA), „Mobile Phone“ (MP), „Tablet“ (TA), „Cosmetic“ (CO) und „Non-metallic Lighter“ (NL).

Zusätzlich zur Beschreibung des Datensatzes werden die enthaltenen Objekte in drei Stufen der Überlagerung eingeordnet:

1 Keine oder wenig Überlagerung

Die Objekte sind sehr deutlich sichtbar und werden nicht von anderen bedeckt.

2 Partielle Überlagerung

Die Objekte sind zum Teil von anderen Objekten überlagert, aber trotzdem für das menschliche Auge abzugrenzen.

3 Schwere oder volle Überlagerung

Die Objekte sind vollständig bedeckt und sind nur schwer zu erkennen bzw. von anderen abzugrenzen.

Zur Erkennung von EMG auf den Bildern wird das Lateral-Inhibition-Modul in bestehende Architekturen integriert. Dieses soll vor dem Hintergrund der Funktionsweise des menschlichen Gehirns den Fokus auf ein betrachtetes Objekt setzen, indem rauschende Informationen in dessen unmittelbarer Nachbarumgebung unterdrückt werden. Dieses Filtern von Rauschen funktioniert mittels bidirektionaler Propagation und die resultierenden Feature-Maps werden durch die sogenannte Boundary-Activation verfeinert.

Im dortigen Experiment wird das Lateral-Inhibition-Modul den drei Modellen SSD, FCOS (Fully Convolutional One-Stage) und YOLOv5s hinzugefügt und mit dem HiXray-Datensatz getestet. Die Ergebnisse zeigen, dass durch das Lateral Inhibition Module die MAP jedes der bestehenden Modelle verbessert wird. In Kombination mit YOLOv5s wurde die beste AP für jede Objektklasse erzielt, welche in der folgenden Tabelle in % dargestellt sind:

Tabelle 3: Ergebnisse der Forschung von Tao et al. (2021) auf dem HiXray-Datensatz

	PO1	**PO2**	**WA**	**LA**	**MP**	**TA**	**CO**	**NL**	**Durchschnitt**
YOLOv5s	95,5	94,5	92,8	97,9	98,0	94,9	63,7	16,3	81,7
YOLOv5s + Lateral Inhibition Module	96,1	95,1	93,9	98,2	98,3	96,4	65,8	21,3	83,2

Aus Tabelle 3 lässt sich schlussfolgern, dass die durchschnittliche AP durch die Einbindung des Lateral Inhibition Module um 1,5% angehoben wurde und die Präzision der beiden Objektklassen CO und NL deutlich unter den restlichen Klassen liegt. Lässt

man diese und WA weg, bleiben lediglich die Objektklassen PO1, PO2, LA, MP und TA, welche alle als EMG zu identifizieren sind.

Die durchschnittliche AP dieser fünf Klassen beträgt 96,8%, was den aktuellen Stand der Technik zur Erkennung von EMG auf X-Ray-Bildern mittels Deep-Learning-Architekturen darstellt.

2.3.2 Erkennung und Klassifizierung von Batterien

Die Abhandlung von Sterkens et al. (2021) beschäftigt sich mit der Erkennung und Klassifizierung von Batterien mittels Deep Learning im Rahmen der Automatisierung des Recyclings elektronischer Abfälle.

Für das Experiment wurden 532 Objekte aus elektronischen Abfällen mit einem CT-Scanner aufgenommen, um unabhängig von äußeren Beschmutzungen oder Beschädigungen des Objektes die im Inneren befindlichen Batterien sichtbar zu machen. Nach der Durchleuchtung wurden 943 Batterien identifiziert und jeweils einer von sechs Batteriesorten zugewiesen. Die Verteilung der Batterietypen ist der folgenden Grafik zu entnehmen:

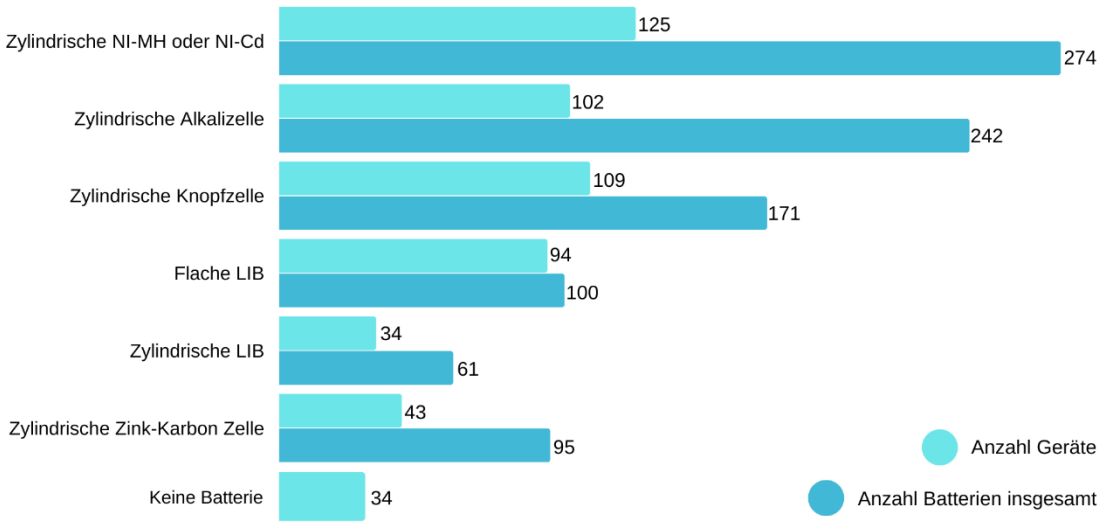

Abbildung 4: Verteilung der Batterien im Datensatz aus Sterkens et al. (2021)

Quelle: Sterkens et al. (2021), eigene Darstellung

Die Geräte wurden dann einzeln aus den einzelnen Bildern ausgeschnitten, um jeweils nur das Gerät selbst ohne Hintergrundrauschen abzubilden. Zusätzlich wurde die Variation und Anzahl der Bilder mittels Augmentation erhöht.

Um den daraus resultierenden Datensatz zur Erkennung der sechs Batteriearten zu trainieren und zu testen wurde das Modell YOLOv2 mit LightNet-Framework als Backbone verwendet, eines der Vorgänger von YOLOv5.

Alle Bilder wurden jeweils einmal mit einer niedrigen und einmal mit einer höheren Strahlungseinstellung des CT-Scanners aufgenommen. Beide Datensätze wurden unabhängig voneinander in zwei verschiedenen Versuchen für das Training verwendet, um im Nachhinein die Resultate vergleichen zu können.

Mit den aus dem Training erhaltenen Gewichten sollte dann erstens eine Klassifikation stattfinden, ob das aktuell betrachtete Bild ein elektronisches Gerät zeigt oder nicht. Dafür wurden die auf einem Bild erkannten Batterien gezählt und festgelegt, dass ein elektronisches Gerät zu sehen ist, wenn mindestens eine Batterie von YOLOv2 erkannt wurde. Zweitens sollte das Modell Batterien lokalisieren, mit Bounding-Boxen kennzeichnen und drittens diese einer der sechs o.g. Batterieklassen zuordnen.

Die Ergebnisse wurden jeweils unter der Angabe der Precision, dem Recall und dem aus der Verknüpfung dieser beiden Werte entstehendem F1-Score bewertet. Zur Vereinfachung sind in Abbildung 5 nur die Ergebnisse mit der höheren Strahlungseinstellung des CT-Scanners veranschaulicht, da sie die Ergebnisse der niedrigeren Strahlungseinstellung in jedem Punkt übertreffen.

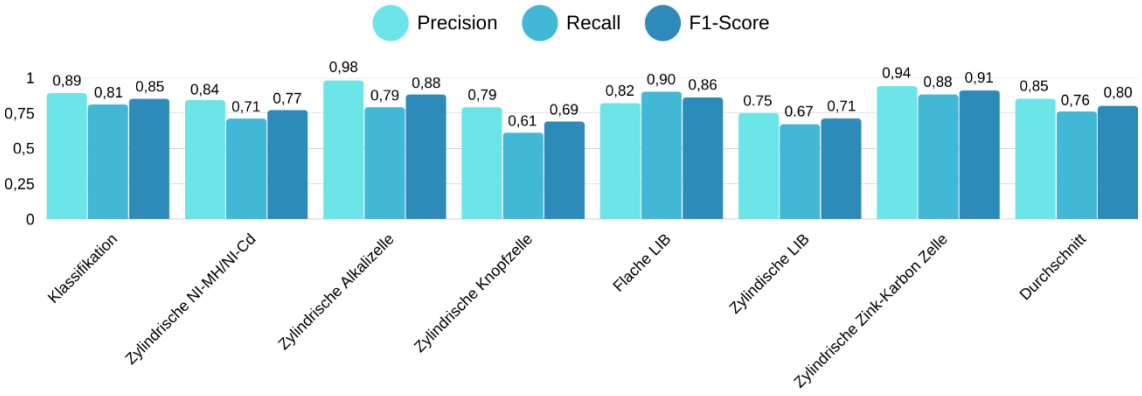

Abbildung 5: Ergebnisse aus Sterkens et al. (2021)

Quelle: Sterkens et al. (2021), eigene Darstellung

Aus Abbildung 5 ist abzuleiten, dass die SOTA-Präzision der reinen Klassifikation, ob ein elektrisches Gerät auf einer X-Ray-Aufnahme zu sehen ist oder nicht, bei 89% liegt.

Darüber hinaus liegt die SOTA-Präzision bezüglich der Erkennung von darin enthaltenen Batterien für die Klasse der flachen LIB bei 82% und für die Klasse der zylindrischen LIB bei 75%.

3 Vorbereitung und Methodik des Experiments

3.1 Auswahl des Datensatzes

Für die Erkennung technischer Geräte und der darin enthaltenen Batterien auf X-Ray-Aufnahmen wurden zunächst grundsätzliche Anforderungen an den Datensatz festgelegt:

1 Thematische Übereinstimmung

Der Datensatz beinhaltet X-Ray-Aufnahmen mit verschiedenen Objekten. Unter diesen Objekten sind verschiedene EMG und darin enthaltene Batterien zu identifizieren.

2 Ausreichender Umfang

Der Datensatz umfasst mindestens 10.000 verschiedene Bilder mit EMG-Klassen, sodass dem Modell beim ersten Training grundlegende Features von EMG auf X-Ray-Aufnahmen beigebracht werden können, um diese dann auf die Erkennung von LIB zu transferieren. Außerdem ist die Varietät der EMG und LIB sichergestellt, sie sind also in verschiedenen Anordnungen, Größen und Perspektiven ausgerichtet, um bei einer Inferenz auf neuen Aufnahmen die Klassen besser identifizieren zu können.

3 Vorhandene Objektklassen

Im Datensatz sind mindestens 3 verschiedene Klassen von EMG zu finden. Die Bounding-Boxen der Annotationen umrahmen präzise und konstant die jeweiligen Objekte. Die darin enthaltenen LIB sind bei der Betrachtung erkennbar und zwei oder mehr verschiedenen LIB-Klassen zuzuordnen.

4 Realer Kontext

Die Aufnahmen entstammen einem realen Kontext und sind nicht synthetisch generiert.

Unter Berücksichtigung dieser Anforderungen wurden die in 2.2 erwähnten Datensätze verglichen. Da der HiXray-Datensatz alle 4 Punkte als einziger erfüllt, wird dieser für die weitere Methodik verwendet.

3.1.1 HiXray-Datensatz

Der in der Forschung von Tao et al. (2021) vorgestellte Datensatz ist nicht direkt öffentlich, kann aber für akademische Zwecke angefordert werden. Er enthält insgesamt

45.364 Bilder, welche in Zusammenarbeit mit professionell geschultem Flughafenpersonal erstellt wurden. Sie sind im JPG-Format abgespeichert und weisen eine durchschnittliche Auflösung von 1200px * 900px auf, welche aber von Bild zu Bild variiert.

Die Bilder sind auf zwei Sätze aufgeteilt, wobei 36.295 dem Trainings- und 9.069 dem Testsatz zugewiesen wurden. Das entspricht einem Verhältnis von etwa 4:1.

Im Datensatz sind die bereits in 2.3.1 erwähnten acht verschiedenen Objektklassen mittels Bounding-Boxen für jedes Bild annotiert. Die Annotationen dieser Bounding-Boxen sind im selben Format wie die des Pascal VOC-Datensatzes in separaten Text-Dateien beigefügt, welche über eindeutige IDs auf die zugehörigen Bilder verweisen. Es sind also 45.364 Textdateien mit insgesamt 102.928 Annotationen der acht Klassen vorhanden. Das Vorkommen dieser Objektklassen auf den Bildern des Trainings- und Testsatzes ist in der folgenden Abbildung dargestellt:

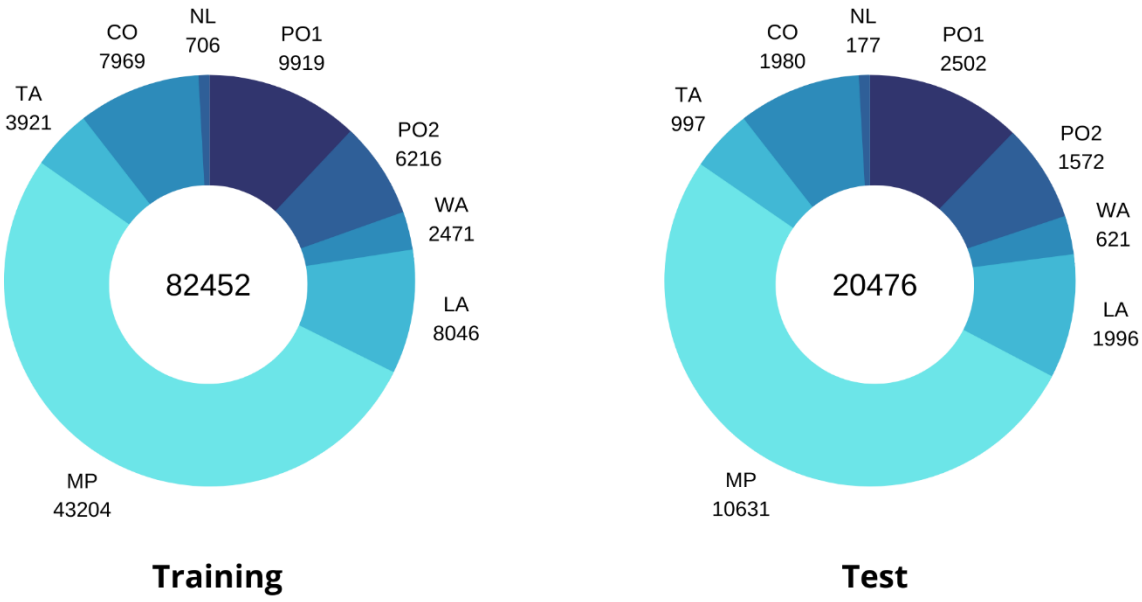

Abbildung 6: Vorkommen der Objektklassen im HiXray-Datensatz

Quelle: Tao et al. (2021), eigene Darstellung

In der Arbeit von Tao et al. (2021) wird außerdem die Anzahl der Klassen pro Bild erwähnt, welche im Durchschnitt bei 2,27 liegt. Durchschnittlich sind also auf einem Bild mehr als 2 Objekte zu sehen, welche jeweils einer der acht Objektklassen zuzuordnen sind.

Anzahl der Bilder, der vorkommenden EMG-Klassen und der durchschnittlich auf einem Bild befindlichen Klassen sind für die Nutzung von Transfer Learning von Vorteil. Denn trainiert man im ersten Schritt das Modell mit einer großen Datengrundlage, so könnte das Folgetraining durch den Transfer des dabei erhaltenen Wissens auf die Erkennung von LIB-Klassen auch mit einem kleineren Datensatz zu einer hohen Präzi-

sion führen. Demnach könnte dann der Aufwand, der durch manuelle Annotation der LIB-Klassen auf allen Bildern des HiXray-Datensatzes anfällt, verkleinert werden.

Im weiteren Verlauf wird eine Textdatei mit den jeweiligen Annotationen der Bounding-Boxen als Label-Datei und die Kombination eines Bildes und seiner zugehörigen Label-Datei als Sample bezeichnet.

3.2 Vorbereitung des Experiments

3.2.1 Entwicklungsumgebung und Machine-Learning-Modell

Das Trainieren von Deep-Learning-Architekturen erfordert hohe Rechenkapazitäten und dauert je nach Trainingsparametern unterschiedlich lange, weshalb zur Ausführung dieser Prozesse meist eine sogenannte Graphical Processing Unit (GPU) zur Beschleunigung verwendet wird. Um die folgenden Versuche unter Nutzung einer solchen GPU durchführen zu können, wird der von Google LLC angebotene Cloud-Service Google Colaboratory Pro (GCP), genutzt.

Mittels GCP kann Python-Code zu bspw. Machine-Learning-Zwecken in einem sogenannten Notebook ausgeführt werden, welches mit einer der virtuellen Maschinen von Google verbunden ist. Die für diese Arbeit verwendeten technischen Voraussetzungen stützen sich also auf die in der GCP-Laufzeit verfügbaren Ressourcen, welche der folgenden Tabelle zu entnehmen sind:

Tabelle 4: Technische Spezifikationen der genutzten GCP-Laufzeit

CPU	**GPU**	**RAM**
Intel(R) Xeon(R) CPU @ 2.20GHz	Tesla P100-PCIE-16GB	27,3 Gigabyte

Basierend auf der Recherche aus 2.1.1 wird für das Experiment YOLOv5 als Deep-Learning-Modell genutzt. Von den in 2.1.2 vorgestellten Varianten wird das Modell YOLOv5m verwendet, da es bezüglich der AP für den MS-COCO-Datensatz teils Überschneidungen mit den komplexeren Modellen YOLOv5l und YOLOv5x aufweist und somit einen guten Ausgleich zwischen Komplexität, Geschwindigkeit und Präzision schafft. Durch die kompaktere Größe ist eine schnellere Verarbeitung im Hinblick auf die Echtzeit-Objekterkennung auf Bildern möglich, was zum Beispiel im automatischen Recyclingprozess zu Zeit- und Kosteneinsparungen führen kann. Die Einrichtung von YOLOv5 in GCP ist im Anhang hinterlegt (siehe Anhang 3.1).

Darüber hinaus existiert beim Training von YOLOv5 die Möglichkeit, die unter der Nutzung des MS-COCO-Datensatzes trainierten Gewichte als Initialgewichte für ein neues, unabhängiges Training zu verwenden. Das hat den Vorteil, dass grundlegende Features bei der Erkennung von Objekten auf Bildern nicht erneut erlernt werden müs-

sen und die Gewichte für einen neuen Kontext mit einem kleineren Datensatz lediglich verfeinert werden.

3.2.2 Vorbearbeitung des HiXray-Datensatzes

In der Beschreibung von YOLOv5 (Ultralytics, 2022) wird angegeben, dass die Modelle für das Training von eigens erstellten Datensätzen eine spezielle Formatierung der Label-Dateien erfordern, in welcher die Objektklassen und die Positionsangaben ihrer Bounding-Boxen relativ zur Gesamtgröße des Bildes angegeben sind. Da die Label-Dateien im HiXray-Datensatz aber in einer anderen Formatierung bestehen, müssen diese für das Training in YOLOv5-Modellen zunächst angepasst werden (siehe Anhang 1.1).

Mittels Python wird für diesen Zweck eine Funktion zur Konvertierung aller Label-Dateien des Datensatzes geschrieben und ausgeführt (siehe Anhang 3.2).

Wie bereits in Tabelle 2 aus dem Benchmark zu aktuellen X-Ray-Datensätzen zu sehen war, enthält der HiXray-Datensatz 5 EMG-Klassen und 3 Klassen, welche für das Experiment keine Rolle spielen: „Water“, „Cosmetic“ und „Non-metallic Lighter“. Da in dieser Arbeit der Fokus ausschließlich auf der Erkennung von EMG und Batterien liegt und das Beibehalten dieser Klassen möglicherweise einen negativen Einfluss auf die Resultate hat, werden sie aus den Label-Dateien des Datensatzes entfernt. Mittels Python wird für diesen Zweck eine Funktion geschrieben, die über alle Label-Dateien iteriert und jeweils die Annotationen der 3 irrelevanten Klassen entfernt (siehe Anhang 3.3).

Der daraus entstandene Datensatz enthält nun zum einen Samples, welche mindestens eine der fünf EMG-Klassen enthalten (siehe Anhang 1.2) und zum anderen Samples ohne jegliche Klassenannotationen (siehe Anhang 1.4).

Um eine klare Separierung zwischen diesen zu schaffen, werden alle Samples ohne Annotationen in einen separaten Datensatz ausgelagert. Der Datensatz mit den EMG-Klassen wird im Folgenden als True-Datensatz und der ohne jegliche Klassen als False-Datensatz bezeichnet. Im folgenden Diagramm sind diese beiden Datensätze mit der jeweiligen Anzahl der Samples zur Verdeutlichung dargestellt:

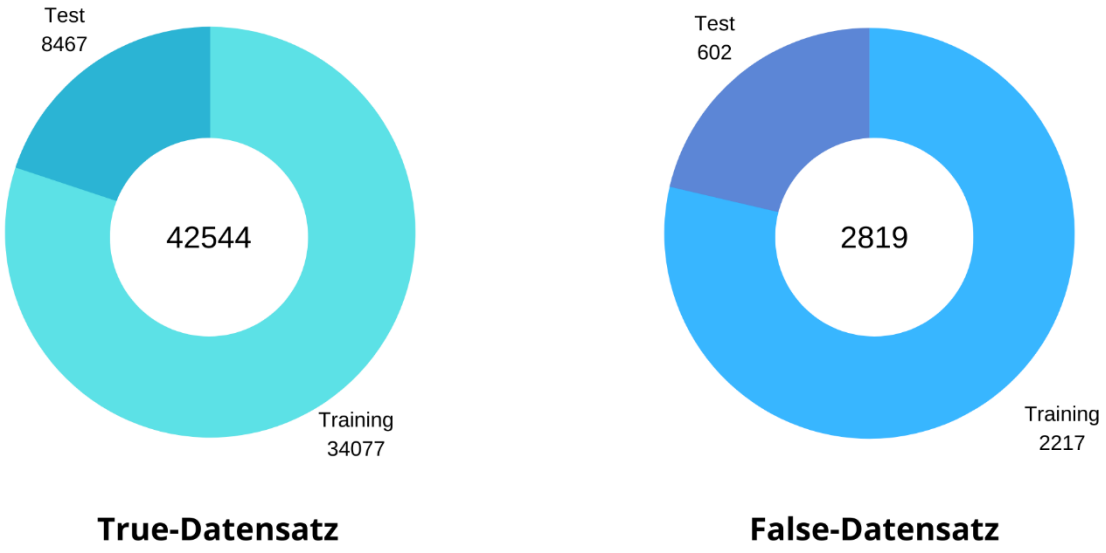

Abbildung 7: Anzahl der Samples im True- und False-Datensatz

Quelle: Eigene Darstellung

3.2.3 Erstellung 1. Datensatz zur Erkennung von fünf EMG-Klassen

In der ersten Phase des Experiments geht es darum, Yolov5m mit einem möglichst großen Anteil der Samples aus dem HiXray-Datensatz zu trainieren. Das hat den Grund, dass mit dem trainierten Modell zum einen die fünf EMG-Klassen auf nicht im Trainingsdatensatz enthaltenen Bildern erkannt und ggf. im Vergleich zum Stand der Technik bessere Ergebnisse erzielt werden sollen.

Zum anderen sollen die resultierenden Gewichte für den Kontext der Erkennung von LIB genutzt werden. Dafür ist es wichtig, dem mit einer großen Datengrundlage im ersten Training möglichst viele wiederverwendbare Features der X-Ray-Aufnahmen beizubringen.

Zuerst wird dafür ein möglichst großer Teil des True-Datensatzes für die Erkennung der 5 verschiedenen EMG abgespalten, welcher aber während des Trainings nicht die Rechenkapazität von der GCP-Laufzeit übersteigt. Um zu testen, wie viele Bilder des Datensatzes maximal in den RAM geladen werden können, werden zunächst mehrere Trainings mit verschiedenen Sample-Anzahlen durchgeführt.

Die Anzahl der Training-Samples wird dabei zunächst auf 5000 festgelegt und bei jeder Probe um 5000 erhöht. Die des Testsatzes beträgt jeweils ein Viertel des Trainingssatzes (siehe Anhang 2). Aus den Ergebnissen erschließt sich, dass der Trainingslauf bei einem Trainingssatz von 25000 Samples nicht beendet werden kann, da GCP bei einer zu hohen RAM-Auslastung über einen Sicherheitsmechanismus das Programm beendet. Der festgestellte Höchstwert an Trainings-Samples beträgt also 20000. Verglichen mit den Ergebnissen des nächstniedrigeren Satzes, also 15000, konnten die Werte

Precision, Recall und AP aber nur um 0 bis 1% durch die Erhöhung der Sample-Anzahl verbessert werden, obwohl der Trainingsprozess von größerer Dauer war.

Außerdem sollen auch Samples ohne ein EMG berücksichtigt werden, damit das Modell ebenso erkennt, dass sich keine der EMG-Klassen auf einem Bild befindet. Unter Berücksichtigung dieser Fakten wird die Anzahl der Samples im ersten Datensatz wie folgt festgelegt:

Tabelle 5: Anzahl der Samples im ersten Datensatz

	Training	**Test**	**Insgesamt**
Samples mit min. einem EMG	12000	3000	15000
Samples ohne EMG	2000	500	2500
Samples insgesamt	14000	3500	17500

Der Trainingssatz soll also 12000 Samples mit mindestens einer und 2000 ohne jegliche EMG-Klasse enthalten. Der Testsatz beinhaltet jeweils ein Viertel der Sample-Anzahl.

Da die resultierenden Gewichte des ersten Trainings für den Zweck der Erkennung von Batterien weiterverwendet werden, sollen nicht die einzelnen EMG-Klassen an sich einen möglichst gleichen Anteil am Datensatz haben, sondern die darin vorkommenden Batterie-Klassen, welche zunächst aber noch nicht in den Label-Dateien annotiert sind.

Unter diesem Vorwand werden die EMG auf den Bildern des Datensatzes manuell auf die in ihnen enthaltene Batterie untersucht. Es wurden die drei verschiedenen LIB-Klassen „Prismatische LIB“, „Zylindrische LIB“ und „Flache LIB“ festgestellt (siehe Anhang 1.3), welche wie folgt in den einzelnen EMG-Klassen vorkommen:

Tabelle 6: Vorkommen der LIB-Klassen in den in HiXray enthaltenen EMG

	PO1	PO2	Mobile Phone	Laptop	Tablet
Prismatische LIB	X				
Zylindrische LIB		X		X	
Flache LIB			X	X	X

In Tabelle 6 wird ersichtlich, dass die „Flache LIB“ in den drei EMG-Klassen „Mobile Phone“, „Laptop“ und „Tablet“ vorkommt. In Abbildung 6 ist zu erkennen, dass die Klasse „Mobile Phone“ den größten Anteil der EMG-Klassen in den Samples von HiXray ausmacht. Übertragen auf die Verteilung der LIB-Klassen in den Samples bedeutet das, dass die „Flache LIB“ generell deutlich öfter vorkommt als die anderen bei-

den. Auf den meisten Bildern, auf denen die EMG-Klassen „PO1“ oder „PO2“ zu sehen sind, ist auch mindestens eine der anderen drei EMG-Klassen zu sehen, was eine exakt gleiche Verteilung erschwert. Deshalb werden für die Erstellung des ersten Datensatzes keine willkürlichen Bilder aus dem True-Datensatz entnommen, sondern mittels einer Python-Funktion herausgefiltert. Diese iteriert über den gesamten True-Datensatz und übernimmt das aktuell betrachtete Sample nur, wenn darin mindestens einmal die Klasse „PO1“ oder „PO2“ annotiert ist. Der Prozess wird unterbrochen, wenn die gewünschte Anzahl an Samples erreicht ist, sprich 12000 für den Trainings- und 3000 für den Testsatz (siehe Anhang 3.4).

Danach werden aus dem False-Datensatz 2000 Samples für den Trainings- und 500 für den Testsatz hinzugefügt. Somit ist der erste Datensatz komplett und auf der folgenden Abbildung zusammengefasst:

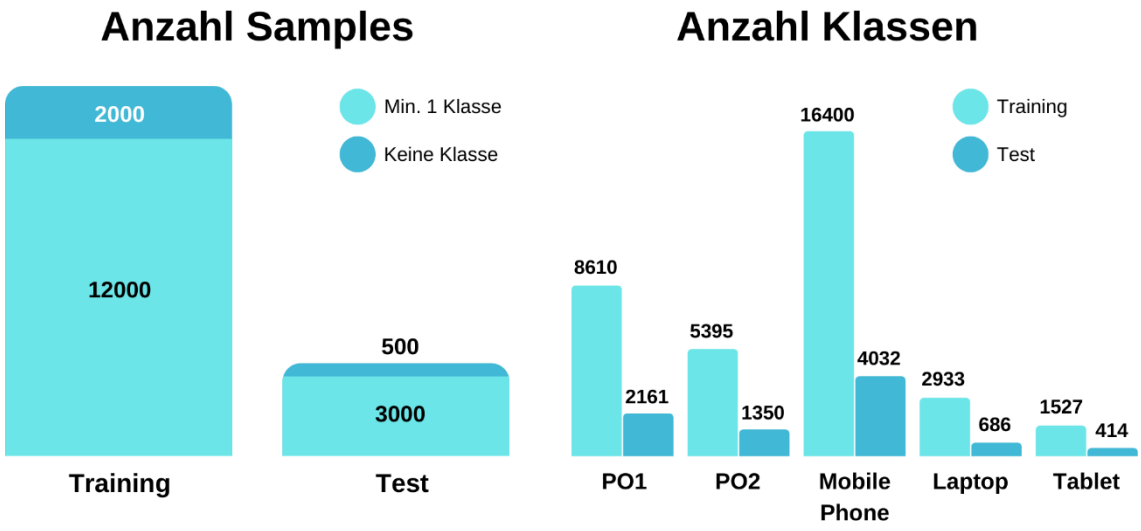

Abbildung 8: Übersicht Samples und Klassen aus dem ersten Datensatz

Quelle: Eigene Darstellung

Aus Abbildung 8 wird deutlich, dass durch die Python-Funktion die Verteilung der EMG-Klassen nicht gleichmäßig aufgeteilt werden konnte und vor Allem die Anzahl der „Mobile Phone“ Klasse heraussticht. Der Grund dafür ist, dass sie fast auf jedem der Samples neben den anderen EMG-Klassen mindestens einmal vorkommt und somit eine gleichmäßige Aufteilung der EMG-Klassen unmöglich ist.

Im Hinblick auf die enthaltenen LIB-Klassen im Trainingssatz ergibt sich unter Berücksichtigung der Tabelle 6 für EMG mit prismatischen LIB ein Anteil von 24,7%, für EMG mit zylindrischen LIB 19,7% und für EMG mit flachen LIB 55,6%.

3.2.4 Erstellung 2. Datensatz zur Erkennung und Klassifizierung von LIB

Die Erstellung des 2. Datensatzes erfordert mehr manuelle Arbeit und somit auch mehr Zeit, da die drei in Tabelle 6 identifizierten Batterieklassen nicht in den Annotationen

des HiXray-Datensatzes enthalten sind und auf jedem Bild zunächst manuell gekennzeichnet werden müssen.

Da aus zeittechnischen Gründen die manuelle Kennzeichnung der drei Batterieklassen auf allen Samples des HiXray-Datensatzes nicht vollständig durchgeführt werden kann, wird ein Konzept des 2. Datensatzes mit weniger Samples entworfen. Außerdem soll durch die Anwendung von Transfer Learning einer niedrigen Präzision aufgrund der kleinen Datenmenge entgegengewirkt werden.

Zunächst wird die Anzahl der Samples in Trainings- und Testsatz festgelegt. Diese sollen einen Anteil von 80-85% an Samples aus dem o.g. False-Datensatz enthalten, da auch Bilder ohne jegliche Klassen miteinbezogen werden sollen. In Anbetracht der verfügbaren Zeit wurde die folgende Anzahl an Samples für den 2. Datensatz festgelegt:

Tabelle 7: Anzahl der Samples im 2. Datensatz

	Training	**Test**	**Insgesamt**
Samples mit min. einer Klasse	1600	375	1975
Samples ohne Klasse	300	75	375
Samples insgesamt	1900	450	2350

Das Konzept soll sich am Kontext der in den Geräten enthaltenen Batterien orientieren, also wird darauf geachtet, dass die gleiche Anzahl an Bildern mit den verschiedenen EMG-Klassen, welche dieselbe Art von Batterie aufweisen, ausgewählt wird. Dabei wird sich an Tabelle 6 orientiert, um die Verteilung der Batterieklassen vor der manuellen Annotation auf eine Ebene zu bringen.

Unter den in Tabelle 7 erwähnten 1600 Samples mit mindestens einer Klasse sollen sich 300 Samples befinden, in welchen nur die EMG-Klasse „PO2“ vorkommt, da diese in allen Fällen eine zylindrische LIB beinhaltet.

300 Samples sollen nur die EMG-Klasse „Mobile Phone“, weitere 300 nur die EMG-Klasse „Laptop“ und 300 nur die EMG-Klasse „Tablet“ enthalten. In jedem von den drei EMG-Klassen sind flache LIB vorzufinden, welche allerdings unterschiedlich hoch und breit sind. Aufgrund dieser unterschiedlichen Maße werden die drei EMG-Klassen separat berücksichtigt, damit das Modell auch verschiedene Ausführungen eine flachen LIB wiedererkennt. Zusätzlich ist es wichtig, die EMG-Klasse „Laptop“ zu separieren, da sie in manchen Fällen eine zylindrische LIB aufweist.

Die zylindrische LIB und die flache LIB kommen zumeist mehr als einmal in den jeweiligen EMG vor, die prismatische LIB aber jeweils nur einmal pro „PO1“-Klasse. Aus diesem Grund sollen 400 Samples enthalten sein, auf welchen jeweils nur die Klasse „PO1“ vorkommt.

Die 375 Samples mit mindestens einem EMG im Testdatensatz setzen sich auf einer ähnlichen Art und Weise zusammen. Darin sollen ebenso die Samples so aufgeteilt sein, dass die EMG-Klassen „PO1“, „PO2“, „Mobile Phone“, „Laptop“, „Tablet“ jeweils in 75 Samples vorkommen.

Die Aufteilung der EMG-Klassen und derer zugehörigen LIB-Klassen im Trainingssatz ist mit jeweils einem zugeschnittenen Beispielbild in der folgenden Abbildung dargestellt:

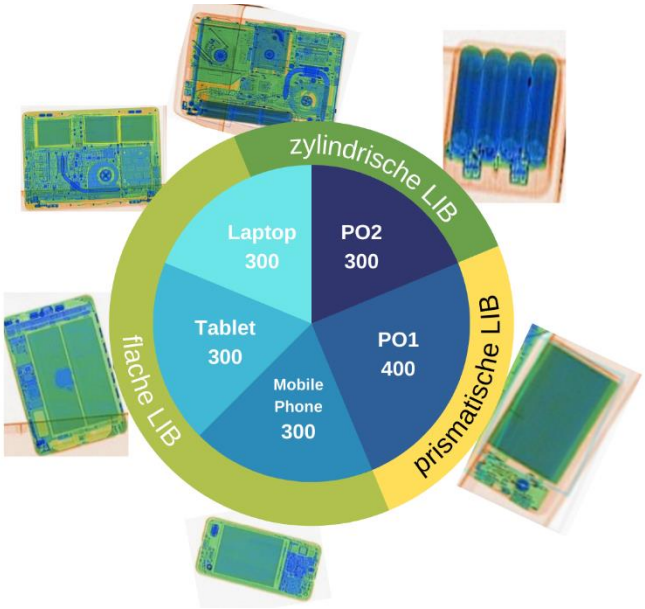

Abbildung 9: Aufteilung der EMG-Klassen mit zugehörigen LIB-Klassen im Trainingssatz

Quelle: Eigene Darstellung

Das Konzept des Datensatzes soll nun umgesetzt werden, indem zunächst für jede LIB-Kasse ein Datenpool an Samples erstellt wird. Für diesen Zweck wird der True-Datensatz dreimal kopiert. Dann wird eine weitere Python-Funktion geschrieben, welche über alle Samples der aktuellen Kopie iteriert und nur jene nicht löscht, in denen nur die angegebene Klasse vorkommt. Dieser Vorgang wird für jede der LIB-Klassen und ebenfalls für den Testsatz wiederholt (siehe Anhang 3.5).

Aus diesen Datenpools wird dann jeweils die zuvor festgelegte Anzahl an Samples zum neuen Datensatz hinzugefügt. Anschließend werden dem Trainingssatz 300, und dem Testsatz 75 Samples aus dem False-Datensatz beigefügt.

Nach diesem Prozess ist die Zuteilung der Samples des 2. Datensatzes abgeschlossen und nun werden alle LIB einzeln manuell annotiert. Für die Annotation der Samples wird die Webapplikation Roboflow von Roboflow, Inc. (o.D.) verwendet. In dieser wird der vorbereitete Datensatz zunächst in ein privates Projekt hochgeladen, wobei die bestehenden Annotationen der EMG-Klassen im Datensatz erhalten bleiben. Als nächstes wird auf jedem Sample manuell eine Bounding-Box um jede ersichtliche LIB-Klasse gezeichnet. Insgesamt wurden auf den 2350 Samples, also Trainings- und

Testsatz zusammen, 5175 LIB identifiziert und der jeweiligen LIB-Klasse zugeordnet. Zieht man die 375 Bilder ohne jegliche EMG davon ab, so existieren im Durchschnitt 2,62 LIB pro Sample. Die Aufteilung aller Klassen des 2. Datensatzes setzt sich also wie folgt zusammen:

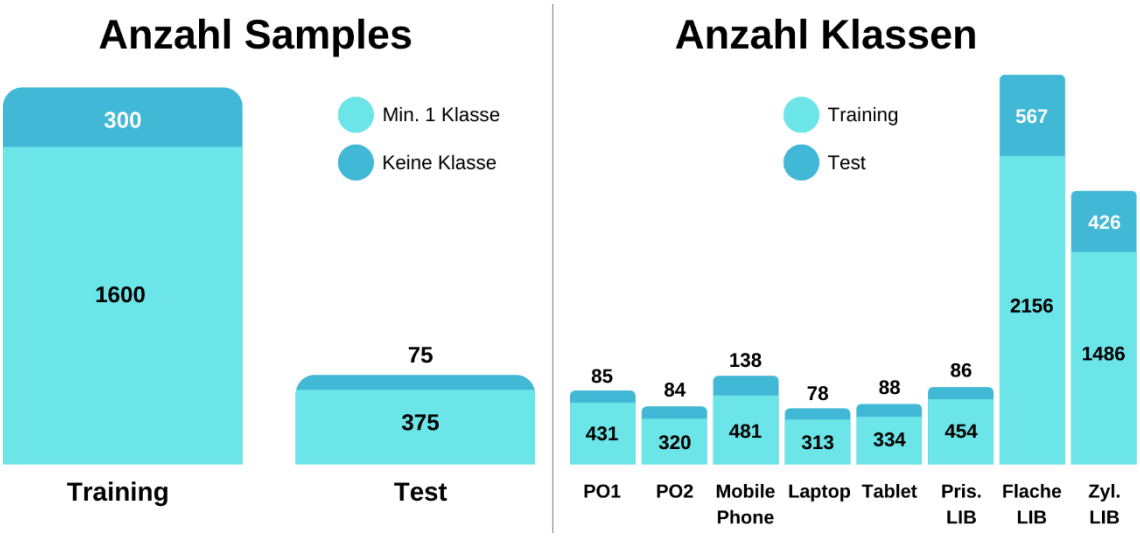

Abbildung 10: Übersicht Samples und Klassen aus dem 2. Datensatz

Quelle: Eigene Darstellung

Wie auf Abbildung 10 zu erkennen ist, stimmt die Anzahl der prismatischen LIB fast mit der Anzahl der „PO1“-Klasse überein. In der Regel beinhaltet ein „PO1“ nur eine prismatische LIB, in manchen Fällen sind es allerdings mehrere. Die herausstechenden Anzahlen der flachen und zylindrischen LIB finden Ihren Ursprung in der Tatsache, dass ein EMG in den meisten Fällen mehrere LIB zugleich beinhaltet. So haben also Mobiltelefone, Tablets und Laptops zumeist nicht nur eine LIB-Zelle, sondern zwei oder auch mehr. Es stellt sich außerdem heraus, dass die Laptops mit zylindrischen LIB ein Batteriepack aus 6 zylindrischen LIB beinhalten. Da jede LIB-Zelle einzeln annotiert wurde ist die Anzahl der flachen und zylindrischen LIB also wesentlich höher als die der jeweiligen EMG.

3.2.4.1 Komplikationen während der Annotation der drei LIB-Klassen

Bereits in der Abhandlung von Tao et al. (2021) wurde erwähnt, dass der HiXray-Datensatz Samples enthält, auf denen sich Objekte der EMG-Klassen Überschneiden. Diese Überschneidungen der Objekte können zu einer Okklusion bestimmter Bereiche führen, was eine visuelle Abgrenzung der sich überschneidenden Objekte erschwert. Die bereits in 2.3.1 erwähnten Stufen der Überlagerung von Objekten auf den Samples macht sich auch bei der Annotation der Batterien bemerkbar, da die LIB in den EMG in manchen Fällen schwer von einer anderen abzugrenzen ist. Während in Tao et al. (2021) zwar konkret auf die verschiedenen Stufen der Überlagerung eingegangen wird, spielen diese in dieser Arbeit allerdings keine weitere Rolle. Stattdessen soll hierbei

lediglich zur Kenntnis genommen werden, dass sich die Überlagerungen negativ auf die Erkennung der einzelnen LIB auswirken können.

Hinzu kommt, dass die Stellung der EMG in den Aufnahmen variiert und zur Verzerrung der LIB führen kann. Wurde also bspw. ein EMG seitlich und flach platziert, so überdecken in manchen Fällen die eigenen Teile des EMG die darin enthaltene LIB. Der Grund dafür ist, dass sich die Durchlässigkeit der X-Ray-Strahlen verschlechtert, wenn sich dichtere Substanzen wie bspw. Metalle überlagern. Auf der Aufnahme ist an diesen Stellen dann eine dunkelblau bis schwarze Verfärbung zu erkennen, innerhalb welcher die LIB schlecht zu erkennen ist.

Der dritte Problemfaktor während der Annotation sind fehlerhafte Aufnahmen. Hierunter fallen Aufnahmen, in welchen es durch Softwarefehler des X-Ray-Gerätes zu Verzerrungen in bestimmten Bereichen des Bildes gekommen ist. Liegt das EMG etwa genau in diesem Bereich, so ist auch die darin enthaltene LIB vollständig verzerrt und kann als solche nicht mehr identifiziert werden.

Darüber hinaus sind den Urhebern des HiXray-Datensatzes in manchen Samples grundlegende Fehler bei der Annotation von EMG unterlaufen. So kommt es also vor, dass Bounding-Boxen nicht das gesamte EMG umfassen, der falschen Klasse zugewiesen wurden, oder gar nicht gekennzeichnet wurden. Diese Fehler werden während der manuellen Annotation der LIB behoben, indem bestehende Bounding-Boxen vergrößert, hinzugefügt oder der richtigen Klasse zugewiesen werden.

Die Problematiken sind zwar nicht in allen Samples zu finden, können aber Auswirkungen auf die Erkennung der betroffenen Objektklassen mittels YOLOv5 haben. In der folgenden Abbildung sind die vier Problematiken des Datensatzes anhand von Beispielen aus dem Datensatz veranschaulicht:

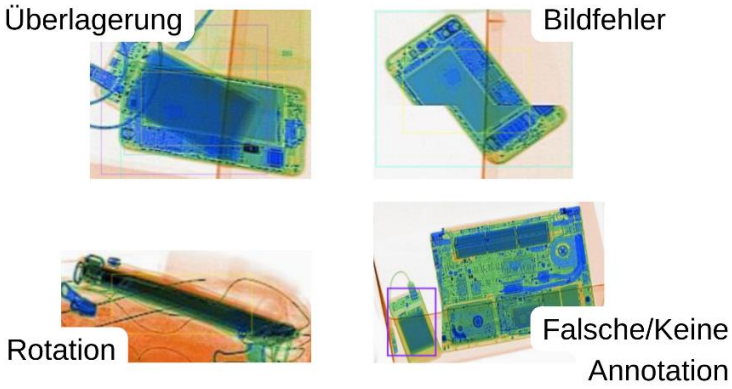

Abbildung 11: Problematiken des HiXray-Datensatzes

Quelle: Eigene Darstellung

3.2.5 Erstellung 3. Datensatz zur Klassifikation: Präsenz eines EMG

Für die Klassifikation, ob auf einer X-Ray-Aufnahme ein Bild zu sehen ist oder nicht, werden in der Durchführung des Experiments zwei Methoden angewandt. Die erste basiert auf dem mit dem 2. Datensatz trainierten YOLOv5-Modell und in der zweiten wird ein ebenfalls von Ultralytics bereitgestellter YOLOv5-Klassifikator neu trainiert und getestet. Für die Klassifikation wird ein von den ersten beiden losgelöster Datensatz erstellt, welcher nicht im 2. Datensatz enthaltene Bilder des True- und False-Datensatzes enthält. Damit wird sichergestellt, dass bei der ersten Methode der Klassifikation auch neue und nicht nur bereits trainierte Bilder klassifiziert werden können.

Zur Erstellung des Datensatzes wird zunächst wieder das Konzept festgelegt. Da die reine Klassifikation keine einzelnen Objektklassen benötigt, können die Label-Dateien in diesem Fall ignoriert werden. Alle Samples, die mindestens eine EMG-Klasse enthalten, werden zu einer Klasse „EMG vorhanden“ generalisiert. Die Samples ohne jegliche EMG-Klassen gehören dementsprechend zur Klasse „Kein EMG vorhanden“.

Die Ausführung des YOLOv5-Klassifikators in der ersten Methode setzt allerdings zum einen eine exakte Ordnerstruktur des Datensatzes und zum anderen einheitliche Maße für jedes der verwendeten Bilder voraus. Aus diesem Grund wird zuerst die Ordnerstruktur wie folgt festgelegt:

- 3. Datensatz
  - Training
    - EMG vorhanden
    - Kein EMG vorhanden
  - Test
    - EMG vorhanden
    - Kein EMG vorhanden

Als nächstes wird die Anzahl der Bilder für Trainings- und Testsatz bestimmt. Da in der ersten Methode für die Klassifikation die bereits trainierten Gewichte verwendet werden, wird für diese kein Trainingssatz benötigt. Für die zweite Methode setzt sich der Trainingssatz aus 2100 Bildern des True-Datensatzes für die Klasse „EMG vorhanden“ und 2000 Bildern des False-Datensatzes für die Klasse „Kein EMG vorhanden“ zusammen. Die Klasse „EMG vorhanden“ enthält 100 Bilder mehr, um sie beim Training etwas mehr in den Fokus zu rücken.

Für den Testsatz werden bei beiden Methoden dieselben Bilder verwendet. Dieser setzt sich aus jeweils 400 Bildern für die Klassen „EMG vorhanden“ und „Kein EMG vorhanden“ zusammen, welche nicht bereits für den 2. Datensatz verwendet wurden.

Da die Ausführung des YOLOv5-Klassifikator-Modelles nur mit Bildern einheitlicher Abmessung möglich ist, wird der 3. Datensatz in zwei Ausführungen gespeichert. In einer Ausführung werden mittels einer Python Funktion alle Bilder auf eine einheitliche

Bildgröße von 600 mal 600 Pixeln gebracht (siehe Anhang 3.7). Dabei werden sie nicht zugeschnitten, sondern gestaucht, um einen Verlust von wichtigen Informationen auf den Bildern zu vermeiden. In der anderen Ausführung werden die Originale der Bilder beibehalten und der Trainingssatz wird entfernt, da bei der Klassifikation mittels des bereits trainierten Modelles kein erneutes Training erforderlich ist.

Der 3. Datensatz setzt sich also wie folgt zusammen:

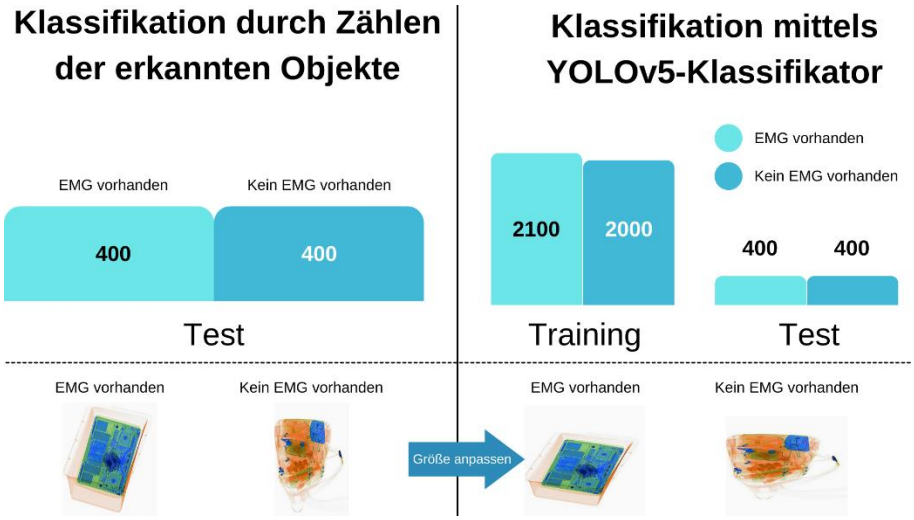

Abbildung 12: Übersicht 3. Datensatz

Quelle: Eigene Darstellung

3.3 Durchführung des Experiments

3.3.1 Training zur Erkennung von fünf EMG-Klassen

Nachdem alle Datensätze vorbereitet wurden und GCP erfolgreich eingerichtet wurde, können nun die Experimente durchgeführt werden. Zur Erkennung der fünf EMG-Klassen wird zunächst der 1. Datensatz in die GCP-Laufzeit hochgeladen. Als nächstes wird das offizielle GitHub-Repository von YOLOv5 importiert, sodass dessen Dateien innerhalb der Session bereitstehen. Nachdem die dafür benötigten Python-Bibliotheken installiert und die von Ultralytics (2022) bereitgestellten vortrainierten Gewichte des YOLOv5m-Modells importiert wurden, muss eine .yaml-Datei erstellt werden, welche für die Ausführung von YOLOv5 relevant ist. Diese enthält Informationen über den Pfad des Trainings- und Testsatzes, die Anzahl der darin enthaltenen Objektklassen, sowie dessen jeweilige Namen.

Unter Angabe der .yaml-Datei und der Datei mit den YOLOv5m-Gewichten wird dann mit dem folgenden Befehl das Training zur Erkennung der fünf EMG-Klassen „PortableCharger1“, „PortableCharger2“, „MobilePhone“, „Laptop“ und „Tablet“ gestartet:

```
!python train.py --img 640 --batch 25 --epochs 20 --data dataset1.yaml
--weights yolov5m.pt
```

Durch diesen Befehl wird die Python-Datei „train.py“ aus dem importierten YOLOv5-Repository mit den dahinter angegebenen Parametern aufgerufen.

Der Parameter „img“ beschreibt die Größe der Eingabebilder in Pixel, denn für Training und Inferenz des Deep-Learning-Modells müssen alle Bilder derselben Größe entsprechen. Bevor die einzelnen Bilder verarbeitet werden, werden sie also zunächst verkleinert oder vergrößert, wobei die angegebene Zahl für die Breite und Höhe des Zielbildes steht. Diese wird in diesem Fall mit dem von Ultralytics vorgegebenem Standardwert 640 belegt.

Der Parameter „batch“ gibt an, nach wie vielen vorwärts propagierten Samples eine Aktualisierung der Gewichte durchgeführt wird. In diesem Trainingslauf wird dafür ein Wert von 25 angegeben.

„epochs“ steht für die Anzahl der Male, wie oft der gesamte Trainingssatz des angegebenen Datensatzes trainiert wird. Um ein Overfitting der Trainingsdaten aufgrund einer zu hohen Wiederholungsrate zu vermeiden, soll der Trainingsprozess 20-mal durchgeführt werden.

Der „data“-Parameter verweist auf die o.g. .yaml-Datei, in welcher die grundlegenden Informationen des Datensatzes festgehalten sind.

Zuletzt wird noch der „weights“-Parameter angegeben. Dieser bestimmt, welche Startgewichte für das Training verwendet werden. Gibt man dort keinen Wert an, so würden die Gewichte unter Angabe eines weiteren Parameters, welcher die jeweilige Größenordnung des Modelles angibt, zufällig initialisiert werden. Daher wird an dieser Stelle das erste Mal Transfer Learning angewandt, indem für diesen Parameter die o.g. Datei mit den vortrainierten YOLOv5m-Gewichten angegeben wird. Laut Ultralytics (2022) wurden diese Gewichte aus einem Training des öffentlichen MS-COCO-Datensatzes gewonnen. Dabei verwendeten sie die Standard-Hyperparameter von YOLOv5 (siehe Anhang 3.6) und trainierten den Datensatz über 300 Epochen. Die Startgewichte wurden also bereits auf die Erkennung von Objektklassen des MS-COCO-Datensatzes trainiert und werden beim erneuten Training unter Nutzung des 1. Datensatzes aus 3.2.3 für die Erkennung der fünf EMG-Klassen auf X-Ray-Aufnahmen spezialisiert.

Die o.g. Hyperparameter sind auch bei diesem Training nicht verändert worden und wiesen somit die von Ultralytics (2022) festgelegten Standard-Parameter auf. Sie beinhalten unter anderem die initiale und finale Learning-Rate, Weight-Decay, Warmup-Epochs, Warmup-Bias-Learning-Rate, Box-Loss-Gain, Class-Loss-Gain, Grenzwert der Intersection-Over-Union (IOU) (siehe Glossar) und einige Parameter zur Augmentation der Eingabebilder während des Trainings (siehe Anhang 3.6).

Das Training läuft ca. 2,5 Std. und YOLOv5 führt nach jeder Epoche automatisch eine Validierung auf dem in der .yaml-Datei angegebenen Testsatz durch, um ggf. bei einem Overfitting frühzeitig abbrechen zu können. Nachdem alle Epochen abgeschlossen sind, werden zum einen die Gewichte nach der letzten Epoche und die Gewichte nach der besten Epoche automatisch gespeichert. Ultralytics (2022) definiert als beste Epoche diejenige mit dem höchsten Fitness-Wert. Dieser stellt eine gewichtete Kombi-

nation der nach jeder Epoche gemessenen Metriken Precision, Recall, MAP@0,5 und MAP@0,5:0,95 dar. Die Ziffern nach dem @-Symbol legen den Grenzwert der IOU fest, ab dem eine Voraussage des Modells als richtig interpretiert wird.

Außerdem wird im Anschluss daran nochmal mit den Gewichten der besten Epoche eine Validierung auf dem Testsatz durchgeführt, dessen Ergebnisse ebenfalls automatisch abgespeichert werden. Unter diesen Ergebnissen befinden sich Confusion-Matrix, F1-Kurve, Precision-Kurve, Recall-Kurve, mehrere Grafiken zu Statistiken der Labels, sowie Bilder des Testsatzes mit den vorausgesagten Bounding-Boxen der Objektklassen und dem jeweiligen Confidence-Wert.

3.3.2 Transfer Learning zur Erkennung und Klassifizierung von LIB

Nachdem das erste Training abgeschlossen wurde, können nun die daraus erhaltenen besten Gewichte als Initialgewichte für das Training des 2. Datensatzes verwendet werden. Um die Ergebnisse zu verbessern, werden im Folgenden sieben verschiedene Ansätze für das Training der Erkennung und Klassifizierung von LIB ausprobiert. Bei diesen werden jeweils vor dem Training bestimmte Parameter oder der Datensatz variiert.

3.3.2.1 Erster Versuch – Datensatz mit EMG- und LIB-Klassen

Analog zur Vorgehensweise des ersten Trainings wird der 2. Datensatz in die GCP-Laufzeit hochgeladen. Dazu wird wieder eine .yaml-Datei erstellt, in welcher die bereits o.g. Informationen zum Datensatz angegeben werden.

Das bereits importierte YOLO5-GitHub-Repository wird nun wiederverwendet, indem die Python-Datei „train.py“ erneut zum Start des Trainings wie folgt aufgerufen wird:

```
!python train.py --img 640 --batch 25 --epochs 40 --data dataset2.yaml
--weights training1-best-weights.pt
```

Die Anzahl der Trainingsepochen wird zunächst auf 40 festgelegt und während des Trainings wird das Verhalten auf ein ggf. auftretendes Overfitting begutachtet.

Für das „data“-Attribut wird nun die .yaml-Datei des 2. Datensatzes angegeben, damit dieser nun für das Training verwendet wird.

Um die aus dem vorherigen Training erzielten Gewichte nun im Sinne des Transfer Learning zu verwenden, werden diese beim „data“-Attribut angegeben.

Das Training benötigt nun insgesamt ca. 3,75 Std., die Ergebnisse der Validierung auf dem Testsatz und die durch das Training erzielten Gewichte werden analog zum ersten Training automatisch abgespeichert.

3.3.2.2 Zweiter Versuch – Datensatz mit nur LIB-Klassen

Da sich die im Datensatz enthaltenen LIB-Klassen jeweils innerhalb der EMG befinden, kann es bspw. bei der Erkennung der EMG-Klasse „PortableCharger1“ zur Verwechs-

lung mit der in ihr enthaltenen prismatischen LIB kommen, wenn die LIB einen großen Teil des EMG einnehmen (siehe Abb. 13).

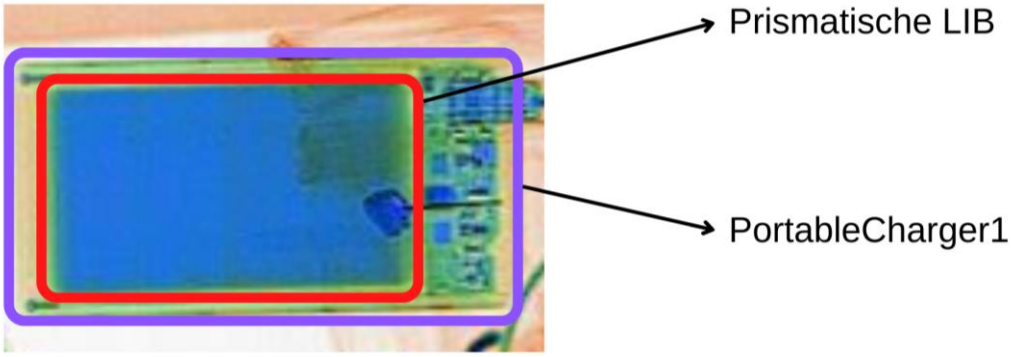

Abbildung 13: PortableCharger1 mit darin enthaltener prismatischen LIB

Quelle: Eigene Darstellung

Die Objektklassen sind zudem in Relation zu anderen auf den Bildern vorkommenden Objekten kleiner, was eine Unterscheidung der beiden Klassen aus Abbildung 13 ebenfalls erschweren könnte.

Aus diesem Grund wird für den zweiten Versuch der Datensatz so abgeändert, dass alle EMG-Klassen daraus entfernt werden und nur noch die LIB-Klassen auf den Samples annotiert sind, da die reine Erkennung der fünf EMG-Klassen bereits mit dem ersten Training abgedeckt wurde und der Fokus somit auf die reine Erkennung der LIB-Klassen gelegt werden kann. Das Entfernen der fünf EMG-Klassen erfolgt über das Online-Tool Roboflow, mit dessen Nutzung der in 3.2.4 vorgestellte Datensatz erstellt wurde.

Dieser abgeänderte Datensatz wird dann zunächst wieder in die GCP-Laufzeit hochgeladen und nach der Erstellung der zugehörigen .yaml-Datei wird das Training mit dem folgenden Befehl gestartet:

```
!python train.py --img 640 --batch 32 --epochs 40 --data
dataset2v2.yaml --weights training1-best-weights.pt
```

Wie der Codezeile zu entnehmen ist, wird nun der „batch“-Parameter auf 32 erhöht. Dies hat den Hintergrund, dass der Batch-Size für eine performantere Ausführung des Trainings eine Zweierpotenz vergeben werden kann. Dadurch wird die Anzahl der verfügbaren physischen Prozessoren berücksichtigt, welche i.d.R. ebenfalls einer Zweierpotenz entspricht.

Für das „data“-Attribut wird nun die neue .yaml-Datei angegeben, welche auf den abgeänderten Datensatz verweist.

Nach 1,88 Std. ist das Training abgeschlossen, wonach die Gewichte und Ergebnisse der Validierung auf dem Testsatz wieder automatisch gespeichert werden.

3.3.2.3 Dritter Versuch – Einfrieren der letzten Ebene

Der dritte Versuch verwendet denselben Datensatz wie der zweite, nur dass hierbei die letzte Ebene des Modells eingefroren werden soll. Dadurch werden die Gewichte der letzten Ebene des YOLOv5m-Modells, welche der Zuweisung der Ausgabevektoren und den entsprechenden Bounding-Boxen dient, während des Trainings nicht aktualisiert. YOLOv5 bietet einen dafür anwendbaren „freeze“-Parameter, welcher bei der Ausführung der „train.py“-Datei optional angegeben werden kann.

Die Struktur des YOLOv5m-Modells weist 25 Ebenen auf, welche von 0 bis 24 nummeriert sind, daher soll in diesem Fall der „freeze“-Parameter den Wert 24 erhalten. Da in YOLOv5 dieser Parameter allerdings so interpretiert wird, dass bspw. bei einem Wert von 24 alle Ebenen von 0 bis 24 eingefroren werden, muss dafür zunächst der zugehörige Code in der „train.py“-Datei so angepasst werden, dass nur die angegebene Ebene eingefroren wird.

Danach wird das Training mit dem folgenden Befehl gestartet:

```
!python train.py --img 640 --batch 32 --epochs 50 --data
dataset2v2.yaml --weights training1-best-weights.pt --freeze [24]
```

Zusätzlich zum „freeze“-Parameter wird die Anzahl der Epochen diesmal auf 50 erhöht, da nun durch das Einfrieren der letzten Ebene 32.328 Gewichte während des Trainings nicht aktualisiert werden müssen und die Berechnungen schneller ausgeführt werden können. Das Training ist nach 2,1 Std. abgeschlossen, ca. 20 Minuten mehr als beim vorherigen.

3.3.2.4 Vierter Versuch – Augmentation des Datensatzes

In den nächsten Versuchen wird der „freeze“-Parameter wieder entfernt, damit auch die Gewichte der letzten Ebene des Modells aktualisiert werden. Hierbei wird nun versucht, die Ergebnisse unter Verwendung von mehr Samples im Trainingssatz zu verbessern. Da die manuelle Annotation weiterer Samples einen vermehrten Zeitaufwand durch Konzeption und Durchführung verursachen würde, wird eine Augmentation des bestehenden Datensatzes mittels Roboflow durchgeführt.

Dafür wird der bereits in Roboflow abgelegte Datensatz mit ausschließlich den Annotationen der LIB-Klassen verwendet. In Roboflow können beim Export des Datensatzes verschiedene Augmentationen angewandt werden, um die Anzahl der Samples künstlich zu erhöhen. In diesem Fall werden 3 verschiedene Rotationen auf die Samples des Trainingssatzes angewandt:

1 Rotation um einen Wert zwischen -45° und +45°

2 Rotation um 90° gegen den Uhrzeigersinn

3 Rotation um 90° im Uhrzeigersinn

Aus einem Sample entstehen also jeweils 3 neue, dessen Label-Dateien automatisch an die neue Position der Bounding-Boxen angepasst werden (siehe Abb. 14).

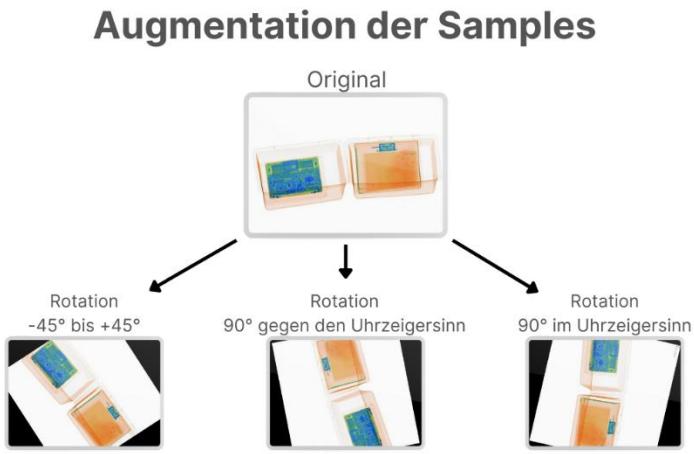

Abbildung 14: Augmentation der Samples

Quelle: Eigene Darstellung

Ursprünglich enthielt der Datensatz 1900 Samples im Trainingssatz, nach der Augmentation durch Roboflow sind es dann 5678, was nahezu die dreifache Anzahl an Samples ist. Der Datensatz wird dann wieder in die GCP-Laufzeit geladen und nach dem Erstellen der neuen .yaml-Datei wird das Training mit dem folgenden Befehl gestartet:

```
!python train.py --img 640 --batch 32 --epochs 50 --data
dataset2v3.yaml --weights training1-best-weights.pt
```

Der Trainingsprozess ist nach ca. 3,7 Std. beendet und die Ergebnisse werden analog zu den o.g. Versuchen automatisch abgespeichert.

3.3.2.5 Fünfter Versuch – Training über 150 Epochen

Im fünften Ansatz der Versuchsreihe soll der augmentierte Datensatz unter Verwendung einer höheren Epochenanzahl trainiert werden, um zu prüfen, ob dabei bessere Ergebnisse erreicht werden können, oder diese sich ggf. durch Overfitting verschlechtern. Da der augmentierte Datensatz bereits in der GCP-Laufzeit besteht, kann das Training mit dem folgenden Befehl gestartet werden:

```
!python train.py --img 640 --batch 32 --epochs 150 --data
dataset2v3.yaml --weights training1-best-weights.pt
```

Wie auch in den vorherigen Versuchen werden weiterhin die Gewichte des ersten Trainings verwendet, um das Konzept des Transfer Learning beizubehalten.

Die Anzahl der Epochen erhält hierbei den Wert 150, sodass der Datensatz dreimal so oft wie im vorherigen Versuch trainiert wird.

Das Training ist nach ca. 10,8 Std. abgeschlossen, die Resultate werden wieder automatisch abgespeichert.

3.3.2.6 Sechster Versuch – Training über 200 Epochen

Da sich der F1-Wert im fünften Versuch verbessert hat und kein Overfitting festzustellen war, wird die Anzahl der Epochen weiterhin gesteigert. Dieses Mal soll das Training über 200 Epochen laufen, also 50 Epochen mehr als im vorherigen Versuch.

Die restlichen Parameter werden wieder beibehalten, um einen optimalen Vergleich der Resultate zu gewährleisten. Das Training wird dann mit dem folgenden Befehl gestartet:

```
!python train.py --img 640 --batch 32 --epochs 200 --data
dataset2v3.yaml --weights training1-best-weights.pt
```

Nach ca. 14,8 Std. ist das Training abgeschlossen, Resultate und Gewichte werden wieder automatisch gespeichert.

3.3.2.7 Siebter Versuch – Ohne Transfer Learning

Da das Training in allen Versuchen auf den Gewichten des Trainings zur Erkennung der fünf EMG-Klassen aufbaut, soll nun nochmal ein direkter Vergleich zu einem Versuch hergestellt werden, in welchem die Standardgewichte von YOLOv5m als Initialgewichte für das Training verwendet werden. Diese sind wie in 3.3.1 bereits beschrieben unter Nutzung des MS-COCO-Datensatzes vortrainiert.

Im siebten Versuch wird also nicht vollständig von Transfer Learning abgesehen, sondern es wird lediglich das Training zur Erkennung der fünf EMG-Klassen übersprungen. Dadurch kann im Anschluss evaluiert werden, ob die Einbindung eines zweiten Transfer-Learnings einen positiven oder negativen Effekt auf die Erkennung und Klassifizierung von LIB hat. Mit dem folgenden Befehl wird das Training innerhalb von GCP gestartet:

```
!python train.py --img 640 --batch 32 --epochs 200 --data
dataset2v3.yaml --weights yolov5m.pt
```

Es werden demnach dieselben Parameter und derselbe Datensatz wie im sechsten Versuch verwendet, nur dass diesmal „yolov5m.pt“ und nicht die aus 3.2.3 gewonnene „training1-best-weights.pt“ als Parameter für die Initialgewichte verwendet wird.

Auch hier ist das Training nach ca. 14,8 Std. abgeschlossen, die Ergebnisse und Gewichte werden wieder automatisch gespeichert.

Zur Zusammenfassung stellt die folgende Tabelle eine grobe Übersicht der sieben Versuche und den darin veränderten Eigenschaften dar:

Tabelle 8: Ablationstabelle der sieben Versuche

Versuch	1	2	3	4	5	6	7
EMG- und LIB-Klassen	✓						
Nur LIB-Klassen		✓	✓	✓	✓	✓	✓
Einfrieren der letzten Ebene			✓				
Augmentation des Datensatzes				✓	✓	✓	✓
40 Epochen	✓	✓					
50 Epochen			✓	✓			
150 Epochen					✓		
200 Epochen						✓	✓
Ohne Transfer Learning							✓
Dauer des Trainings (Std.)	3,75	1,88	1,5	3,7	10,8	14,8	14,8

3.3.3 Klassifikation: Präsenz eines EMG

Zur Klassifikation, ob ein EMG auf einem Bild zu sehen ist, werden zwei unterschiedliche Methoden durchgeführt und miteinander verglichen. Dafür wird ein lokales Python-Projekt erstellt, wobei zunächst der in 3.2.5 erstellte Datensatz in das Projekt eingebunden wird.

3.3.3.1 Klassifikation durch Abzählen

In der ersten Methode zur Klassifikation wird das bereits bestehende YOLOv5m-Modell mit den in 3.3.2.6 erzielten Gewichten so umfunktioniert, dass mittels der Anzahl der auf einem Eingabebild erkannten LIB-Klassen eine Aussage über die Präsenz von EMG getroffen wird.

Dafür wird eine neue Python-Klasse erstellt, in der zunächst YOLOv5m mit den Gewichten der letzten Trainingsepoche aus 3.3.2.6 importiert wird.

Nun wird eine Python-Funktion erstellt, die das Unterverzeichnis „EMG vorhanden“ aus dem Testsatz durchläuft und jedes Bild mittels des o.g. Modells auf vorkommende LIB-Klassen analysiert. Nach jedem Bild werden in der Ausgabematrix die erkannten LIB-Klassen gezählt.

Befindet sich mindestens eine Reihe in der Matrix, so wird das Ergebnis als True-Positive (TP) gewertet, da die Klassifikation richtig ist. Andernfalls wird das Ergebnis als False-Negative (FN) gewertet. Ist die Schleife beendet, so erfolgt eine Konsolenausgabe mit den Werten TP, FN und Accuracy, welche den Anteil der TP an der Gesamtanzahl der analysierten Bilder darstellt.

Analog zu dieser Vorgehensweise wird eine zweite Python-Funktion erstellt, welche über das Unterverzeichnis „Kein EMG vorhanden“ des Testsatzes iteriert. Hierbei werden allerdings leere Ausgabematrizen als True-Negative (TN) bewertet und Matrizen mit mindestens einer Reihe als False-Positive (FP). Nach Abschluss der Schleife erfolgt ebenfalls eine Konsolenausgabe mit den Werten TN, FP und Accuracy, welche hierbei den Gesamtanteil der TN angibt (siehe Anhang 3.8).

3.3.3.2 Klassifikation durch YOLOv5-Klassifikator

Das GitHub-Repository wird nun zunächst unter der Angabe des Branches „Classifier“ in die GCP-Laufzeit geklont. In diesem werden unter Angabe der jeweiligen Größe des Modells, in diesem Fall YOLOv5m, die letzte Ebene der Struktur durch eine Klassifikator-Ebene ersetzt (Ultralytics, 2022). Durch diese Änderung werden nicht mehr die einzelnen Objektklassen erkannt, sondern das Bild an sich einer Klasse zugewiesen.

Danach wird der bereits auf eine Bildgröße von 600px mal 600px vereinheitlichte Klassifikator-Datensatz aus 3.2.5 importiert.

Unter Verwendung dieses Datensatzes wird dann das Training mit dem folgenden Befehl gestartet:

```
!python classifier.py --img 600 --batch 25 --epochs 20 --data
„/content/ClassificationDataset“ --model yolov5m
```

Das Training soll 20 Epochen andauern und der „data“-Parameter verweist auf das Datensatz-Verzeichnis. Zuletzt wird wieder YOLOv5m als Modell ausgewählt, um einen Vergleich zur ersten Methode ziehen zu können.

Nach dem erfolgreich abgeschlossenen Training werden analog zu den vorherigen Trainings wieder die Gewichte der besten und der letzten Epoche gespeichert.

Nun sollen die Ergebnisse für den Testsatz auf dieselbe Weise wie in der ersten Methode ermittelt werden. Dafür wird im lokalen Projekt eine Instanz des Modells mit den besten Gewichten des Trainings erstellt, mittels welcher neue Bilder analysiert werden können. Das Ergebnis einer Inferenz dieses Modells wird als numerischen Wert zurückgegeben, welcher den Index der jeweiligen Klasse enthält. So wird also in diesem Beispiel eine 0 zurückgeliefert, wenn das jeweilige Bild der Klasse „EMG enthalten“ zugesprochen wurde und eine 1, wenn die Voraussage „Kein EMG enthalten“ war.

Die Klassifikation der Bilder erfolgt wieder über zwei Python-Funktionen derselben Struktur wie in der ersten Methode, nur dass diesmal nicht die erkannten LIB-Klassen gezählt werden, sondern die Richtigkeit der Klassifikation geprüft wird (siehe Anhang 3.9).

4 Resultate und Diskussion

Nach dem Abschluss der Methodik sollen die erhaltenen Testergebnisse nun unter Nutzung der im Folgenden beschriebenen Metriken evaluiert und mit dem aktuellen Stand der Technik verglichen werden.

4.1 Bewertungsschema

4.1.1 Precision und Recall

Für die zentrale Bewertung der Erkennung von Objekten, sprich Lokalisierung mittels Bounding-Box und Zuweisung der richtigen Objektklasse, werden die Metriken Precision und Recall genutzt.

Precision stützt sich auf den Anteil der korrekt erkannten Objekte an der Gesamtzahl der erkannten Objekte und berechnet sich folgendermaßen:

$$Precision = \frac{\Sigma\, TP}{\Sigma\, TP + \Sigma\, FP}$$

Recall bezieht sich auf den Anteil der korrekt erkannten Objekte an allen möglichen korrekten Erkennungen:

$$Recall = \frac{\Sigma\, TP}{\Sigma\, TP + \Sigma\, FN}$$

Eine Erkennung wird als TP gewertet, wenn die IOU mindestens bei 0,5 liegt. Liegt diese unter dem Grenzwert, so resultiert daraus ein FP und ein FN.

4.1.2 AP und MAP

neben Precision und Recall die AP und die MAP verwendet.

Der Wert der AP gibt die durchschnittliche Precision über allen ermittelten Recall-Werten einer Klasse an. Die MAP stellt den Durchschnitt der AP über alle im jeweiligen Training relevanten Objektklassen dar und kann aus der von YOLOv5 automatisch erstellten PR-Kurve, sowie aus den in der Konsole ausgegebenen Werten abgeleitet werden.

Die automatisch von YOLOv5 kalkulierten Werte der MAP liegen jeweils einem IOU-Grenzwert von 0,5 zugrunde.

4.1.3 F1-Wert

Der F1-Wert stellt einen direkten Zusammenhang zwischen Precision und Recall her und findet seine Anwendung beim Vergleich zur SOTA-Veröffentlichung bezüglich der Erkennung verschiedener Batterieklassen von Sterkens et al. (2021). Er setzt sich wie folgt zusammen:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4.1.4 Accuracy

Bei der Klassifikation, ob ein EMG auf einem Bild zu sehen ist oder nicht, wird mittels der Accuracy der Anteil der korrekten Klassifikationen ermittelt. 0 stellt dabei den schlechtesten und 1 den besten zu erreichenden Wert dar. Er setzt sich außerdem wie folgt zusammen:

$$\frac{TP + TN}{TP + FN + TN + FP}$$

4.2 Erkennung von fünf EMG-Klassen

Nachdem das Training von YOLOv5m über 20 Epochen beendet und der Testsatz mit den daraus erhaltenen Gewichten getestet wurde, werden nun die Testergebnisse ausgewertet und mit dem Stand der Technik verglichen. In der folgenden Tabelle sind jeweils Precision, Recall, F1-Wert und AP für jede Klasse, sowie der Durchschnitt aller Klassen aus dem Test aufgeführt:

Tabelle 9: Ergebnisse der Erkennung von fünf EMG-Klassen

	PO1	PO2	Mobile Phone	Laptop	Tablet	Durchschnitt
Precision	0,966	0,96	0,939	0,899	0,936	0,94
Recall	0,945	0,921	0,965	0,997	0,867	0,939
F1-Wert	0,955	0,94	0,952	0,945	0,9	0,938
MAP	0,972	0,964	0,971	0,972	0,96	0,968

Aus Tabelle 9 wird ersichtlich, dass für jede der Klassen eine MAP von mindestens 0,96 erzielt wurde. Der Durchschnitt aller Klassen beträgt 0,968, also 96,8%. Vergleicht man diesen Wert mit den bisherigen Ergebnissen von Tao et al. (2021) aus Tabelle 3, so ist keine Verbesserung oder Verschlechterung der Resultate festzustellen.

In der Forschung von Tao et al. (2021) wurde ebenfalls YOLOv5 verwendet, wobei sich dabei für die weniger komplexe Variante YOLOv5s mit einem zusätzlichen von ihnen entworfenen Modul entschieden wurde. Überträgt man die dortigen Erkenntnisse auf die in dieser Arbeit mittels YOLOv5m erzielten Resultate, so könnte sich die MAP durch das Hinzufügen ihres Moduls erhöhen.

Dabei sind auch die Unterschiede des Datensatzes zu beachten, der in dieser Arbeit aufgrund von begrenztem Arbeitsspeicher in GCP vor dem Training verkleinert und präpariert wurde. Trainiert man also den gesamten Datensatz über eine höhere Anzahl an Epochen, so könnte die durchschnittliche MAP auch auf diese Weise angehoben werden.

Betrachtet man die Precision- und Recall-Werte der Klassen aus Tabelle 8, so fällt zum einen auf, dass die Precision bei der Klasse „Laptop“ den geringsten Wert aufweist, der Recall-Wert aber am höchsten ist. Daraus kann geschlussfolgert werden, dass es einen hohen Anteil an FP gibt, der z.B. durch die Verwechslung anderer Objekte oder Hintergrundrauschen mit der Klasse „Laptop“ hervorgerufen wurde.

Zum anderen wird bei der Klasse „Tablet“ ein dazu komplementäres Verhalten der Werte Precision und Recall auffällig. Das Defizit des Recall-Wertes könnte vor dem Hintergrund des Precision-Defizits bei der Klasse „Laptop“ bedeuten, dass einige Objekte der Klasse „Tablet“ fälschlicherweise als „Laptop“ klassifiziert wurden. Eine mögliche Ursache dieses Verhaltens wären Ähnlichkeiten in Struktur und Größe der beiden EMG-Klassen auf den X-Ray-Aufnahmen. Hinzu kommt, dass die Klasse „Tablet“ im Trainingssatz des ersten Datensatzes aus 3.2.3 den geringsten Anteil der vorkommenden Klassen ausmacht.

4.3 Erkennung und Klassifizierung von LIB

Unter Anwendung des 2. Datensatzes wurden sieben Versuche zur Erkennung und Klassifikation von LIB auf X-Ray-Aufnahmen durchgeführt.

4.3.1 Erster Versuch – Datensatz mit EMG- und LIB-Klassen

Zunächst wurde das Modell darauf trainiert, mit dem Transfer der Gewichte aus dem vorherigen Training sowohl die fünf EMG-Klassen als auch die darin befindlichen drei LIB-Klassen zu erkennen und der jeweiligen Klasse zuzuordnen.

Tabelle 10: Ergebnisse der Erkennung von EMG- und LIB-Klassen – erster Versuch

	Precision	Recall	MAP	F1-Wert
PO1	0,984	0,723	0,874	0,834
PO2	0,933	0,798	0,876	0,86
Mobile Phone	0,959	0,67	0,839	0,789
Laptop	0,99	0,974	0,988	0,982
Tablet	0,939	0,909	0,927	0,924
Prismatische LIB	0,971	0,86	0,918	0,912
Flache LIB	0,956	0,881	0,947	0,917
Zylindrische LIB	0,902	0,819	0,884	0,858
Durchschnitt	0,954	0,829	0,907	0,887
Durchschnitt (nur LIB-Klassen)	0,943	0,853	0,916	0,896

Nach dem Hinzufügen der LIB-Klassen zum Datensatz und dem Weiterverwenden der Gewichte des ersten Trainings wird aus Tabelle 10 ersichtlich, dass sich die Precision der meisten EMG-Klassen durch ein weiteres Training verbessert hat.

Im Gegensatz dazu ist der Recall-Wert dieser Klassen deutlich gesunken, was auf eine erhöhte Anzahl an FN schließen lässt. Die Ursache dafür könnten Verwechslungen mit den in ihnen enthaltenen LIB-Klassen sein, da diese bei den drei EMG-Klassen „PO1“, „PO2“ und „Mobile Phone“ einen Großteil des Gerätes selbst ausmachen. Je kleiner das Gerät einer der EMG-Klassen ist, desto größer ist also der Anteil der darin enthaltenen LIB am Gerät selbst, was vor Allem bei gleicher Form von EMG und LIB zu einer Verwechslung führen kann. Beispiele für die relativen Größen der LIB in EMG sind in der folgenden Abbildung veranschaulicht:

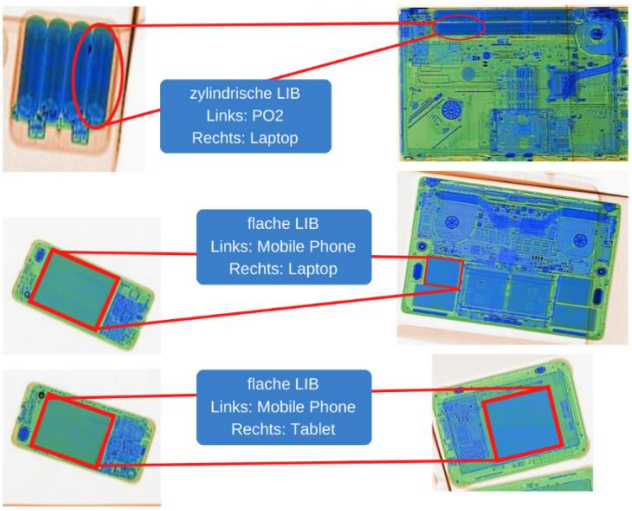

Abbildung 15: Relative Größe der LIB innerhalb von verschiedenen EMG

Quelle: Eigene Darstellung

Darin ist zu erkennen, dass eine flache oder zylindrische LIB innerhalb eines Mobiltelefons oder einer Powerbank einen größeren Anteil am Gerät selbst ausmacht als in einem Laptop oder Tablet. Die Wahrscheinlichkeit einer Verwechslung zwischen EMG- und LIB-Klasse bei den kleineren Geräten ist also höher.

Wird dann bspw. das gesamte Mobiltelefon als flache LIB klassifiziert und nicht die darin befindliche flache LIB selbst, so entstünde ein FP der Klasse „flache LIB“ und ein FN der Klasse „Mobile Phone“. Dieses Phänomen kann außerdem auch in umgekehrter Weise auftreten und durch den Anstieg der FN den Recall-Wert negativ beeinflussen.

In der folgenden Abbildung sind zwei Beispiele von Verwechslungen der EMG- und LIB-Klassen zu sehen:

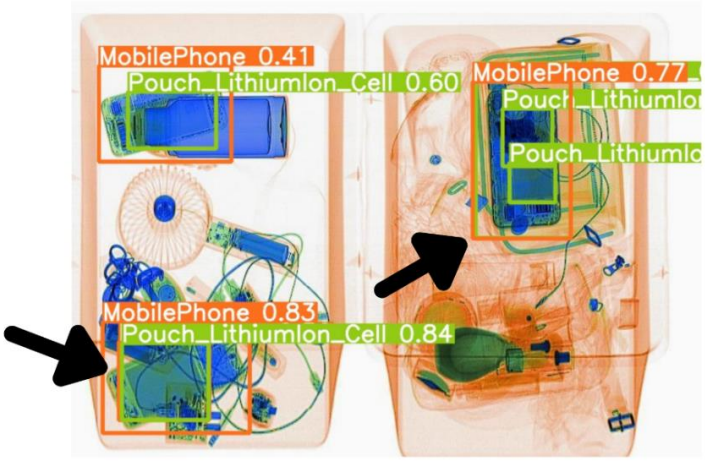

Abbildung 16: Verwechslung der EMG- und LIB-Klassen

Quelle: Eigene Darstellung

Die Pfeile zeigen jeweils auf die Stellen, bei denen die Überschneidungen der Bounding-Boxen von „Mobile Phone“ und einer flachen LIB deutlich werden. Links unten wurde eine flache LIB zusätzlich als „Mobile Phone“ klassifiziert, was dementsprechend als FP zu werten ist. Rechts oben ist die Überschneidung schwieriger zu erkennen, jedoch wurden dort nicht nur die flachen LIB, sondern auch das gesamte Mobiltelefon als flache LIB erkannt, was ebenso zu einem FP führt. Erhöht sich die Anzahl der FP, so sinkt der Wert der Precision.

Aus Abbildung 16 wird also deutlich, dass das Beibehalten von EMG- und LIB-Klassen tatsächlich zu Verwechslungen mit negativem Einfluss auf die Precision führt. Jedoch werden die Klassen zusätzlich auch richtig erkannt, wodurch der Recall durch dieses Verhalten unberührt bleibt.

Das Defizit der Recall-Werte muss also einen anderen Ursprung haben, welcher im Laufe der nächsten Versuche behandelt wird. Weil der Fokus bei dieser Versuchsreihe sowieso auf der Erkennung und Klassifizierung von LIB lag, wurden in den darauffolgenden Versuchen die EMG-Klassen aus dem Datensatz entfernt. Dies sollte zur Folge haben, dass zunächst die Verwechslungen der betroffenen Klassen beseitigt werden, um die dadurch entstandenen FP zu vermeiden.

4.3.2 Zweiter Versuch – Datensatz mit nur LIB-Klassen

Zur Überprüfung, ob durch das Entfernen der EMG-Klassen bessere Ergebnisse für die LIB-Klassen erzielt werden, wurde im zweiten Versuch das Training mit dem dafür präparierten Datensatz ebenfalls über 40 Epochen durchgeführt. Die Testergebnisse sind in der folgenden Tabelle aufgeführt:

Tabelle 11: Ergebnisse der Erkennung von LIB-Klassen – Zweiter Versuch

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,971	0,969	0,916	0,952
Recall	0,821	0,905	0,798	0,841
MAP	0,903	0,963	0,895	0,92
F1-Wert	0,89	0,936	0,853	0,893

Wie in Tabelle 11 zu sehen ist, steigen sowohl durchschnittliche Precision als auch MAP der LIB-Klassen durch die Entfernung der EMG-Klassen an.

Auf der anderen Seite sinkt der durchschnittliche F1-Wert, denn die Recall-Werte haben sich weiter verschlechtert. Wie erwartet steigt also durch das Verhindern von Verwechslungen zwischen LIB und EMG die Precision, der Recall bleibt auf der anderen Seite unberührt.

Weil der durchschnittliche Recall bei der Erkennung der fünf EMG-Klassen aus 4.2 deutlich höher ist, wurde im darauffolgenden Versuch die letzte Ebene des Modells während des Trainings eingefroren. Dadurch sollten die Features im neuen Kontext, also den LIB-Klassen, spezialisiert und die finale Klassifikation mit den alten, ggf. besser geeigneten Gewichten durchgeführt werden.

4.3.3 Dritter Versuch – Einfrieren der letzten Ebene

Nachdem das Modell dann über 50 Epochen trainiert wurde, erzielte es für den Testsatz die folgenden Resultate:

Tabelle 12: Ergebnisse der Erkennung von LIB-Klassen – Dritter Versuch

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,918	0,957	0,906	0,927
Recall	0,849	0,908	0,816	0,858
MAP	0,902	0,957	0,883	0,914
F1-Wert	0,882	0,932	0,859	0,891

Wie in Tabelle 12 festzustellen ist, sind die Precision-Werte gesunken. Dahingegen haben sich die Recall-Werte marginal verbessert, was aber nicht zur Erhöhung des F1- oder MAP-Wertes genügt.

Aus dem Sachverhalt lässt sich schließen, dass durch das Einfrieren der letzten Ebene wieder ein erhöhtes Aufkommen an FP zustande kommt, da generell mehr LIB vom Modell identifiziert wurden. Dadurch nehmen auf der anderen Seite die FN geringfügig ab, womit sich die Recall-Werte wiederum verbessern.

Da zur Evaluierung vor Allem MAP und F1-Wert betrachtet werden und diese hierbei gesunken sind, wird in den folgenden Versuchen das Einfrieren der letzten Schicht keine weitere Anwendung finden.

Eine weitere mögliche Erklärung für das immer noch vorhandene Defizit der Recall-Werte wäre der Einfluss der in 3.2.4.1 beschriebenen Überlagerung mehrerer Objekte. Es könnte also sein, dass LIB überhaupt nicht erkannt werden, wenn sie durch andere Objekte überlagert werden oder bei seitlicher Perspektive bspw. mehrere LIB übereinander liegen. Um diesem Problem entgegenzuwirken, sollte im vierten Versuch durch die Augmentation des Datensatzes eine höhere Vielfalt und Anzahl an Samples im Trainingssatz geschaffen werden, ohne dabei weitere manuelle Annotationen machen zu müssen. Auf diese Weise sollte das Modell während des Trainings mehr Perspektiven der LIB kennenlernen und somit bei einer Inferenz neuer Bilder die LIB-Klassen deutlicher identifizieren können.

4.3.4 Vierter Versuch – Augmentation des Datensatzes

Nach der Augmentation des Datensatzes, was die Anzahl an Samples im Trainingssatz etwa verdreifachte, wurde das Training wieder über 50 Epochen durchgeführt.

Die Ergebnisse für den Testsatz sind in der folgenden Tabelle aufgeführt:

Tabelle 13: Ergebnisse der Erkennung von LIB-Klassen – Vierter Versuch

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,902	0,937	0,871	0,903
Recall	0,86	0,914	0,78	0,851
MAP	0,926	0,964	0,888	0,926
F1-Wert	0,88	0,925	0,823	0,876

Im Vergleich zu den Ergebnissen ohne Augmentation aus Tabelle 13 ist der durchschnittliche Recall-Wert gestiegen, während die Precision-Werte deutlich gesunken sind. Analog zum dritten Versuch lässt sich daraus schließen, dass generell mehr LIB erkannt worden sind und sich die Anzahl der FP erhöht.

Eine Ursache dafür könnten zu wenige Trainingsepochen sein, da im Vergleich zum zweiten Versuch der Trainingssatz verdreifacht, die Anzahl der Epochen aber lediglich um 10 erhöht wurde. Unter der Annahme, dass die Augmentation des Datensatzes bei einer weiteren Erhöhung der Epochen eine bessere Wirkung zeigt, wurde das Modell im fünften Versuch über 150 und im sechsten Versuch über 200 Epochen trainiert.

4.3.5 Fünfter Versuch – Training über 150 Epochen

Das Erhöhen der Epochen könnte ein Overfitting des Trainingssatzes zur Folge haben. Aus diesem Grund wurde im fünften Versuch die Anzahl verdreifacht und während des Trainings auf den MAP-Wert jeder Epoche geachtet. Um dessen Entwicklung zu visualisieren, wurde das folgende Diagramm erstellt:

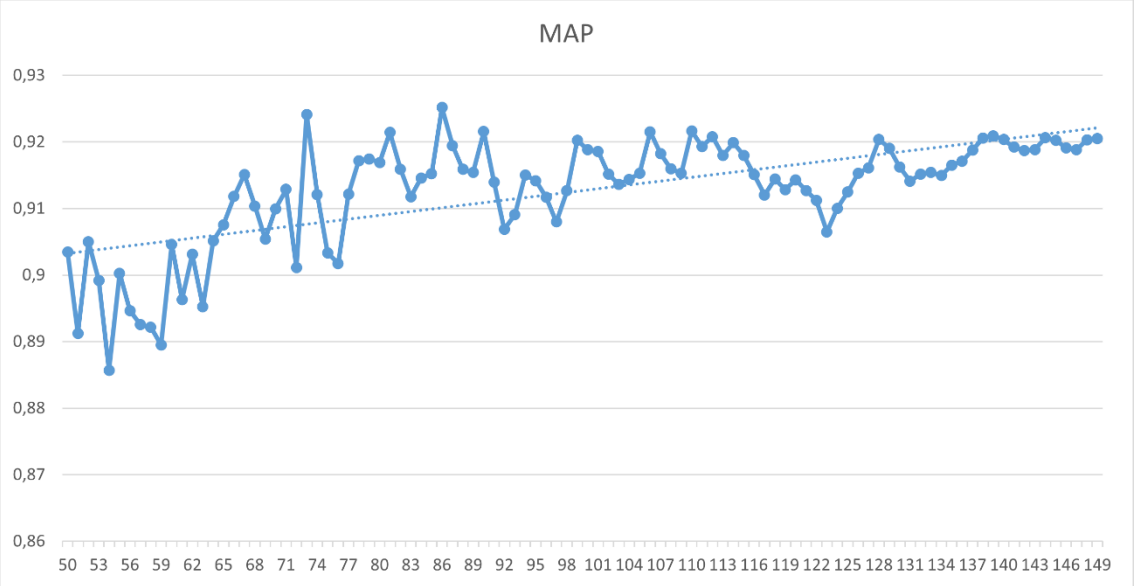

Abbildung 17: Verlauf des MAP-Wertes über 150 Epochen

Quelle: Eigene Darstellung

Abbildung 17 zeigt den Verlauf der MAP-Werte aus den Epochen 50 bis 150 und die gepunktete Linie beschreibt den Trend des Graphen. Dabei wird ersichtlich, dass die MAP während des Trainings Schwankungen im Bereich von Epoche 50 bis ca. 130 aufweist. Im Bereich von 131 bis 150 stagniert der Graph dann, steigt aber bis zuletzt durchschnittlich weiter an. Zwischen den schwankenden Werten wurde bei Epoche 86 mit 0,925 der Höchstwert der MAP erreicht, welcher jedoch danach wieder rapide abfällt.

Anhand des Trends ist zu erkennen, dass das Training über weitere 100 Epochen nach den 50 bereits absolvierten den Wert der MAP von 0,903 auf 0,921 anhebt.

Die Ergebnisse für den Testsatz unter Verwendung der Gewichte der letzten Epoche sind in der folgenden Tabelle zusammengefasst:

Tabelle 14: Ergebnisse der Erkennung von LIB-Klassen – Fünfter Versuch

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,919	0,958	0,891	0,923 (+2,2%)
Recall	0,791	0,915	0,871	0,859 (+0,9%)
MAP	0,894	0,964	0,904	0,921 (-0,6%)
F1-Wert	0,85	0,936	0,881	0,89 (+1,5%)

Im Vergleich zum Training über 50 Epochen aus 4.3.4 haben sich hierbei nicht nur alle Precision-Werte, sondern auch der durchschnittliche Recall verbessert.

Insgesamt übersteigen die durchschnittliche Precision-, Recall- und F1-Werte die des vorherigen Versuchs und die durchschnittliche MAP fällt um 0,6%. Da sich die meisten Werte verbessert haben und auf Abbildung 17 kein deutliches Anzeichen von Overfitting zu erkennen ist, wird im folgenden Versuch die Anzahl der Epochen weiter erhöht.

4.3.6 Sechster Versuch – Training über 200 Epochen

Nach dem Abschluss der 200 Epochen wurde analog zum vorherigen Versuch ein Diagramm mit den MAP-Werten der Epochen 50 bis 200 erstellt (Abbildung 18).

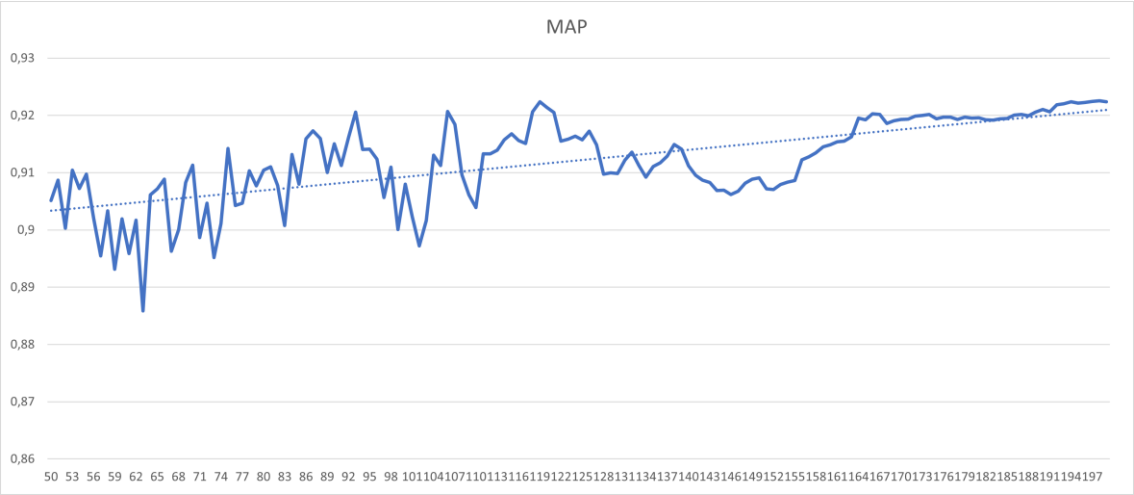

Abbildung 18: Verlauf des MAP-Wertes über 200 Epochen

Quelle: Eigene Darstellung

In Abbildung 18 schwankt der Graph zunächst im Bereich von Epoche 50 bis 120, fällt dann durchschnittlich bis zu Epoche 155 und steigt schließlich bis zu Epoche 200 wie-

der an. Außerdem erreicht die MAP im Gegensatz zu Abbildung 17 ihren Höchstwert von 0,933 nicht im Bereich der Schwankungen, sondern erst bei der vorletzten Epoche.

Aus diesem Grund ist auch bei einem Training über 200 Epochen kein Overfitting festzustellen, was ebenfalls durch den steigenden Trend deutlich wird.

Die Ergebnisse nach der letzten Epoche für den Testsatz sind in der folgenden Tabelle zusammengefasst:

Tabelle 15: Ergebnisse der Erkennung von LIB-Klassen – sechster Versuch

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,92	0,961	0,921	0,934 (+1,2%)
Recall	0,869	0,909	0,859	0,879 (+2,3%)
MAP	0,904	0,963	0,901	0,923 (+0,2%)
F1-Wert	0,894	0,934	0,889	0,906 (+1,7%)

Aus Tabelle 15 wird deutlich, dass sich im Vergleich zu Tabelle 14 der Durchschnitt aller Werte durch 50 weitere Trainingsepochen verbessert hat. Hierbei macht sich vor Allem der Anstieg des durchschnittlichen Recall-Wertes von 2,3% bemerkbar. Aus diesem Verhalten kann geschlussfolgert werden, dass das Erhöhen der Trainingsepochen die Anzahl der FN vermindert und dadurch den Recall anhebt.

Für eine weitere Steigerung der Werte wurde im Anschluss des Trainings eine Variation des Confidence-Grenzwertes (siehe Glossar) in Betracht gezogen. Durch dessen Erhöhung fallen Erkennungen mit niedrigem Confidence-Wert weg, wodurch die Anzahl der FP vermindert und die Precision erhöht werden könnte.

Da bisher der Confidence-Grenzwert bei der Validierung des Testsatzes nicht beachtet wurde, muss zunächst herausgefunden werden bei welchem Confidence-Grenzwert die besten Resultate erzielt werden. Für diesen Zweck wird die folgende F1-Kurve zur Hilfe genommen (Abbildung 19).

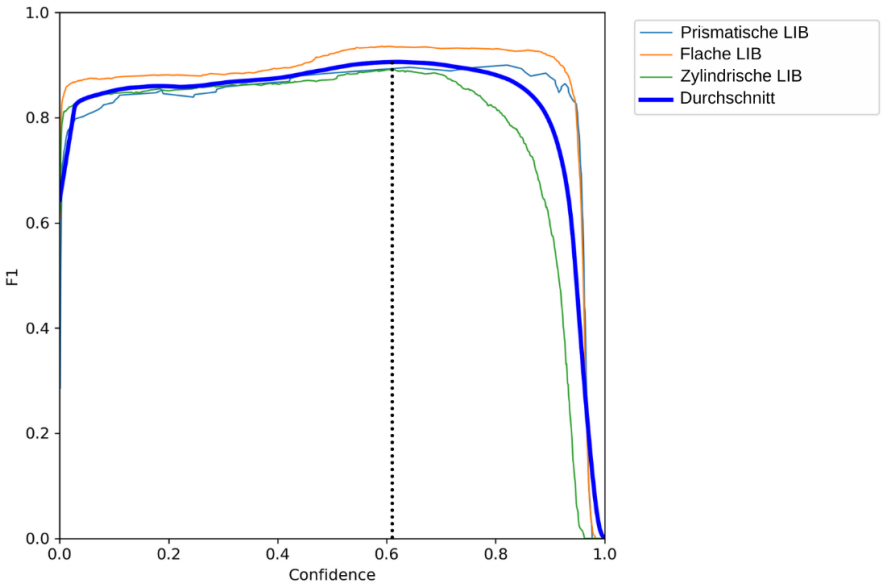

Abbildung 19: F1-Kurve aus dem sechsten Versuch

Quelle: Eigene Darstellung

Die F1-Kurve beschreibt den F1-Wert der jeweiligen Klasse, sowie den Durchschnitt in Abhängigkeit vom Confidence-Grenzwert. In Abbildung 19 steht hellblau für die prismatische, orange für die flache und grün für die zylindrische LIB. Der etwas dickere, dunkelblaue Graph beschreibt den Durchschnitt der drei LIB-Klassen.

Der höchste durchschnittliche F1-Wert ist mit der vertikalen gepunkteten Linie gekennzeichnet und liegt bei einem Confidence-Grenzwert von ca. 0,6. Aus diesem Grund wurde eine erneute Inferenz mit denselben Gewichten, aber einem Confidence-Grenzwert von 0,6 unter Verwendung des Testsatzes gestartet. In Tabelle 16 sind die daraus resultierenden Werte zusammengefasst:

Tabelle 16: Ergebnisse sechster Versuch mit Confidence-Grenzwert 0,6

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,924	0,961	0,921	0,935
Recall	0,948	0,929	0,892	0,923
MAP	0,966	0,958	0,903	0,942
F1-Wert	0,936	0,945	0,906	0,929

Die Ergebnisse aus Tabelle 16 zeigen, dass durch die Erhöhung des Confidence-Grenzwertes Precision, Recall, MAP und F1-Wert für die prismatische und zylindrische LIB deutlich angestiegen sind. Daraus kann geschlussfolgert werden, dass einige Er-

kennungen mit niedrigem Confidence-Wert bei der prismatischen und zylindrischen LIB als FP identifiziert wurden.

Um zu überprüfen, ob dabei Anzahl der Epochen und Augmentation eine Rolle spielen, werden zum Vergleich die Ergebnisse aus 4.3.2 mit einem höheren Confidence-Wert nochmal neu berechnet (Tabelle 17). Der Wert beträgt dabei wieder 0,6, da sich analog zu Abbildung 19 dort der höchste F1-Wert befindet.

Tabelle 17: Ergebnisse zweiter Versuch mit Confidence-Grenzwert 0,6

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,986	0,97	0,925	0,96
Recall	0,958	0,914	0,801	0,891
MAP	0,978	0,951	0,88	0,936
F1-Wert	0,972	0,941	0,859	0,924

In Tabelle 17 ist zu entnehmen, dass die Ergebnisse auch bei Versuch zwei durch eine Erhöhung des Confidence-Grenzwertes verbessert werden. Jedoch ist auch abzulesen, dass die Durchschnitte von Recall, MAP und F1-Wert niedriger sind und die Precision höher ist als in Tabelle 16. Es kann also behauptet werden, dass sich der Recall durch Augmentation und Erhöhung der Epochen auf 200 verbessert.

4.3.7 Siebter Versuch – Ohne Transfer Learning

Im letzten Versuch sollte dann bewiesen werden, dass die Nutzung von Transfer Learning generell zu besseren Ergebnissen beiträgt. Dafür wurde dieselbe Prozedur wie im sechsten Versuch durchgeführt, nur dass dabei nicht die aus 4.2 resultierenden Gewichte als Startgewichte verwendet werden, sondern die bereits in YOLOv5m enthaltenen Standardgewichte. Die Ergebnisse für den Testsatz nach dem Training über 200 Epochen mit dem standardmäßigen Confidence-Grenzwert von 0,001 sind in der folgenden Tabelle aufgeführt:

Tabelle 18: Ergebnisse Erkennung von LIB-Klassen – Ohne Transfer Learning

	Prismatische LIB	Flache LIB	Zylindrische LIB	Durchschnitt
Precision	0,818	0,962	0,899	0,893
Recall	0,812	0,901	0,772	0,828
MAP	0,878	0,957	0,832	0,889
F1-Wert	0,815	0,931	0,831	0,859

Aus Tabelle 18 wird deutlich, dass die Durchschnitte aller Werte deutlich niedriger sind als in Tabelle 15. Der Grund dafür ist, dass das Training ohne Transfer Learning einen niedrigeren Ausgangs-F1-Wert aufweist. In der folgenden Abbildung ist daher der Verlauf des F1-Wertes über die 200 Epochen bei einem Confidence-Grenzwert von 0,001 mit und ohne Transfer Learning dargestellt:

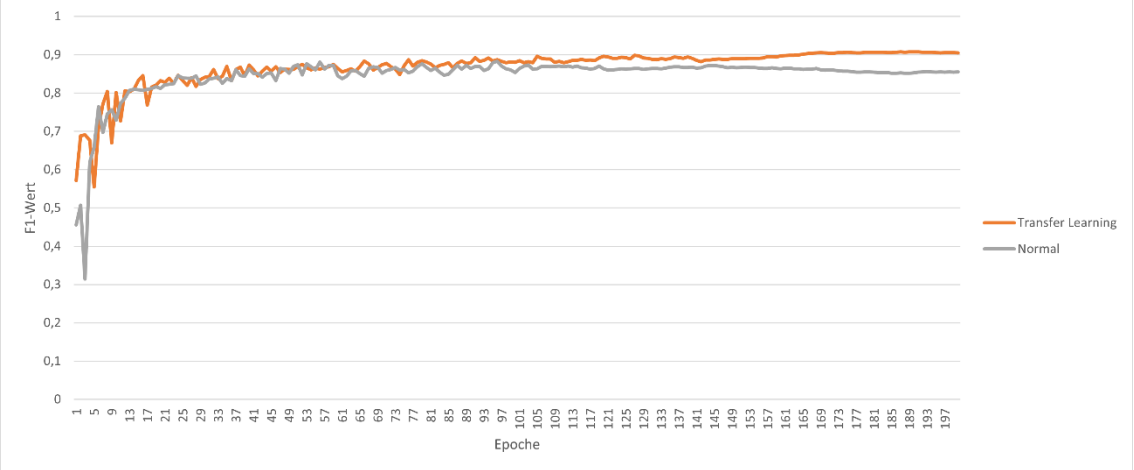

Abbildung 20: F1-Wert über 200 Epochen mit und ohne Transfer Learning

Quelle: Eigene Darstellung

Der orangene Graph aus Abbildung 20 spiegelt dabei den Trainingsverlauf mit Gewichtstransfer, der graue den Trainingsverlauf ohne Gewichtstransfer wider. Betrachtet man den Beginn der beiden Graphen, so wird deutlich, dass durch Transfer Learning bereits ein höherer Ausgangs-F1-Wert vorherrscht. Im Bereich der Epochen 1 bis 30 sind dabei in beiden Fällen deutliche Schwankungen zu erkennen, welche sich bis Epoche 120 einpendeln. Ab Epoche 100 liegt der orangene Graph stets über dem grauen und stagniert in Richtung 200. Im Bereich der Epochen 157 bis 200 ist dann eine deutliche Differenz der beiden F1-Werte zu erkennen, da mit Transfer Learning-Graph weiterhin steigt, während der normale Graph marginal sinkt. Dieses Verhalten deutet auf ein Overfitting beim Training ohne Transfer Learning hin.

Abschließend lässt sich sagen, dass der Transfer der Gewichte zu einem höheren Ausgangs-F1-Wert führt, wodurch sich über die 200 Epochen durchschnittlich bessere Ergebnisse entwickeln.

4.3.8 Zusammenfassung der sieben Versuche

Um nun nochmal alle Ergebnisse für die sieben Versuche zusammenzufassen und zu vergleichen, wird ein einheitlicher Vergleichswert geschaffen, der sich über den Mittelwert des durchschnittlichen MAP- und F1-Wertes wie folgt berechnet:

$$Vergleichswert = \frac{\emptyset MAP + \emptyset F1}{2}$$

In der folgenden Tabelle sind die Ablationen mit jeweiliger Performance, Precision, Recall und Trainingszeit aus jedem Versuch zusammengefasst:

Tabelle 19: Ablationstabelle Erkennung von LIB mit Performance-Werten

Versuch	1	2 a)	2 b)	3	4	5	6 a)	6 b)	7
EMG- und LIB-Klassen	✓								
Nur LIB-Klassen		✓	✓	✓	✓	✓	✓	✓	✓
Einfrieren der letzten Ebene				✓					
Augmentation des Datensatzes					✓	✓	✓	✓	✓
40 Epochen	✓	✓	✓						
50 Epochen				✓	✓				
150 Epochen						✓			
200 Epochen							✓	✓	✓
Confidence-Grenzwert 0,6			✓					✓	
Ohne Transfer Learning									✓
Vergleichswert	**0,906**	**0,907**	**0,93**	**0,903**	**0,901**	**0,906**	**0,936**	**0,915**	**0,874**
Ø Precision	0,943	0,952	0,96	0,927	0,903	0,923	0,934	0,935	0,893
Ø Recall	0,853	0,841	0,891	0,858	0,851	0,859	0,879	0,923	0,828
Trainingsdauer (Std.)	3,75	1,88	-	1,5	3,7	10,8	14,8	-	14,8

Es konnte also durch Anwendung von Transfer Learning, Augmentation des Datensatzes, einer Epochenanzahl von 200 und einem Confidence-Grenzwert von 0,6 die besten Resultate für die Erkennung von LIB auf X-Ray-Aufnahmen mit einem Vergleichswert von 0,936 erzielt werden. Darüber hinaus macht sich bemerkbar, dass in allen versuchen die durchschnittliche Präzision über dem durchschnittlichen Recall liegt. Das Defizit des Recall-Wertes könnte auf das erhöhte Vorkommen an sich überlappenden oder durch andere Objekte verdeckte EMG zurückzuführen sein. Die Trainingsdauer wird durch das Entfernen der EMG-Klassen verringert und durch das Erhöhen der Epochen vergrößert.

Die Zeit der Inferenz für ein einzelnes neues Bild liegt bei 0,022 Sekunden und ist bei allen Versuchen gleich, da die Architektur des Modells nicht verändert wurde.

4.3.9 Vergleich zum Stand der Technik

Um die Erkennung und Klassifikation von LIB mit dem Stand der Technik zu vergleichen, werden die Ergebnisse von Sterkens et al. (2021) aus Abbildung 5 den in 4.3.6 erzielten Werten aus Tabelle 16 gegenübergestellt (Abbildung 21).

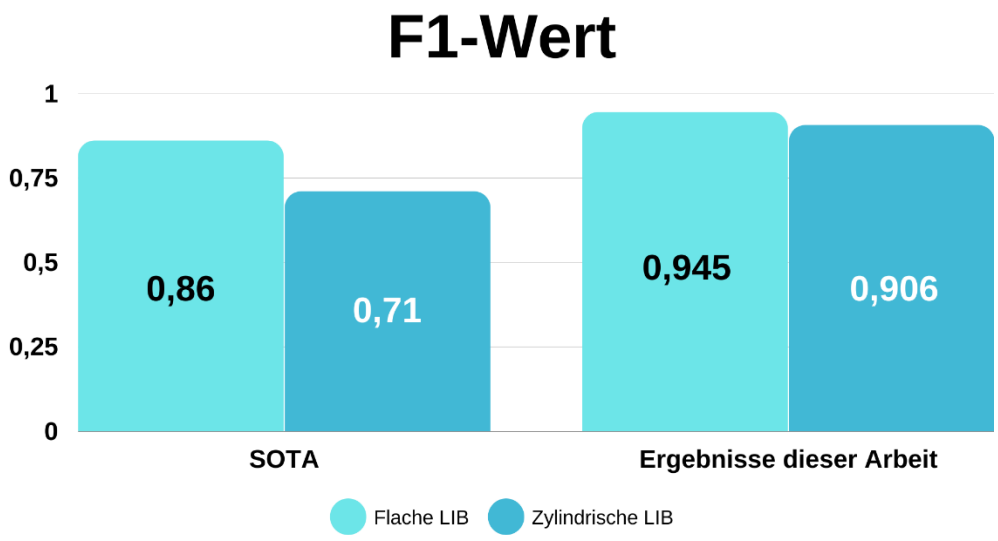

Abbildung 21: Vergleich F1-Werte mit Sterkens et al. (2021)

Quelle: Eigene Darstellung

Da sich die LIB-Klassen in der Arbeit von Sterkens et al. (2021) nur in zwei Klassen mit dieser Arbeit überschneiden und dort keine Angabe zur MAP gemacht wurde, ist in Abbildung 21 der F1-Wert für die flache und zylindrische LIB dargestellt. Der in dieser Arbeit erzielte F1-Wert übertrifft den des SOTA für die flache LIB mit 0,085 und für die zylindrische LIB mit 0,196. Dabei ist jedoch anzumerken, dass in Sterkens et al. (2021) jeweils einzelne elektronische Geräte unter Verwendung eines anderen X-Ray-

Scanners aufgenommen wurden und der darin verwendete Datensatz eine deutlich niedrigere Anzahl an Samples aufwies.

4.4 Klassifikation: Präsenz von EMG

Die Klassifikation, ob eine X-Ray-Aufnahme ein EMG enthält oder nicht wurde über zwei verschiedene Methoden durchgeführt, bei denen unterschiedliche Ergebnisse zu identifizieren sind.

4.4.1 Klassifikation durch Abzählen

In der ersten Methode wurden die besten in der vorherigen Versuchsreihe erzielten Gewichte für die Zählung der einzelnen LIB verwendet. Diese sind laut Tabelle 19 unter der Voraussetzung, dass der Confidence-Grenzwert bei 0,6 liegt, in Versuch 6 erzielt worden. Nach der in 3.3.3.1 beschriebenen Vorgehensweise wurden die Resultate in der folgenden Konfusionsmatrix festgehalten:

Tabelle 20: Konfusionsmatrix - Klassifikation durch Abzählen

Konfusionsmatrix der Testergebnisse durch Abzählen		Ermittelte Klasse	
		EMG vorhanden	Kein EMG vorhanden
Tatsächliche Klasse	EMG vorhanden	379 (TP)	4 (FN)
	Kein EMG vorhanden	21 (FP)	396 (TN)

Die Konfusionsmatrix beschreibt, wie viele der durch das Modell ermittelten Klassen korrekt oder inkorrekt klassifiziert wurden. Die grün hervorgehobenen Felder sind jeweils richtige Klassifizierungen, die roten falsche. Aus den vier Werten lassen sich dann Accuracy, Precision, Recall und F1-Wert berechnen.

Die Accuracy liegt bei ca. 0,97, woraus sich eine Fehlerquote von 3% erschließen lässt. Durch die höhere Anzahl an FP als FN liegt die Precision mit ca. 0,95 unter dem Recall, der einen Wert von 0,99 aufweist. Es wurden also auf mehr Bildern fälschlicherweise EMG als fälschlicherweise keine EMG festgestellt und je nach Anwendungskontext ist dieses Verhalten abzuwägen.

Der sich daraus ergebende F1-Wert liegt bei ca. 0,97.

4.4.2 Klassifikation durch YOLOv5m-Klassifikator

Bei der zweiten Methode wurde der YOLOv5m-Klassifikator über 20 Epochen ohne Verwendung der zuvor erhaltenen Gewichte trainiert. Dadurch fällt der Bezug zu den einzelnen LIB- und EMG-Klassen, sowie der Transfer der Gewichte weg. Die Initialgewichte des Klassifikators sind allerdings, wie bei der Standardversion von YOLOv5m, vortrainiert.

Nach dem Training wurde der Klassifikator dann für die Klassifikation desselben Testsatzes wie in 4.4.1 ausgeführt und die Ergebnisse wieder in einer Konfusionsmatrix festgehalten:

Tabelle 21: Konfusionsmatrix – Klassifikation durch YOLOv5m-Klassifikator

Konfusionsmatrix der Testergebnisse durch YOLOv5m-Klassifikator		Ermittelte Klasse	
		EMG vorhanden	Kein EMG vorhanden
Tatsächliche Klasse	EMG vorhanden	372 (TP)	28 (FN)
	Kein EMG vorhanden	12 (FP)	388 (TN)

Die daraus errechnete Accuracy liegt bei 0,95, woraus eine Fehlerquote von 5% abzuleiten ist. Die Klassifikation über den YOLOv5m-Klassifikator weist nach 20 Trainingsepochen also eine höhere Fehlerquote auf. Außerdem liegt diesmal der Wert der Precision mit ca. 0,97 über dem Recall von 0,93. Im Gegensatz zur ersten Methode werden vom Modell also häufiger EMG fälschlicherweise nicht erkannt als fälschlicherweise erkannt.

Aus den Werten lässt sich darüber hinaus ein F1-Wert von ca. 0,95 errechnen, welcher somit niedriger ist als bei der Klassifikation über das Abzählen der LIB.

4.4.3 Vergleich zum Stand der Technik

In der Forschung von Sterkens et al. (2021) wurde die Klassifikation der einzelnen Bilder analog zur ersten Methode dieser Arbeit über das Abzählen der erkannten LIB durchgeführt.

Da über diese Methode im Vergleich zur Nutzung des YOLOv5m-Klassifikators bessere Ergebnisse erzielt werden konnten, werden diese für den Vergleich zum SOTA die genutzt. In der folgenden Abbildung werden Precision, Recall und F1-Wert aus dieser Arbeit denen aus Sterkens et al. (2021) gegenübergestellt:

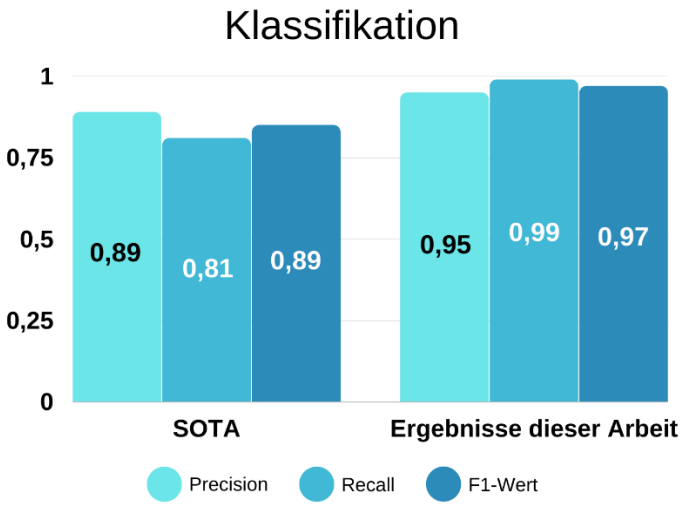

Abbildung 22: Vergleich Precision, Recall und F1-Wert mit Sterkens et al. (2021)

Quelle: Eigene Darstellung

Auch hierbei ist zu beachten, dass die in Sterkens et al. (2021) verwendeten X-Ray-Aufnahmen lediglich einzelne elektronische Geräte zeigen und die Anzahl der Samples deutlich niedriger ist.

Dennoch konnte mittels YOLOv5m die Precision um 0,06, Recall um 0,18 und F1-Wert um 0,08 angehoben und somit der bisherige Stand der Technik übertroffen werden.

4.5 Einfluss von Transfer Learning

Im Laufe des Experiments wurden an zwei Stellen trainierte Gewichte auf einen neuen Kontext transferiert (Abbildung 23).

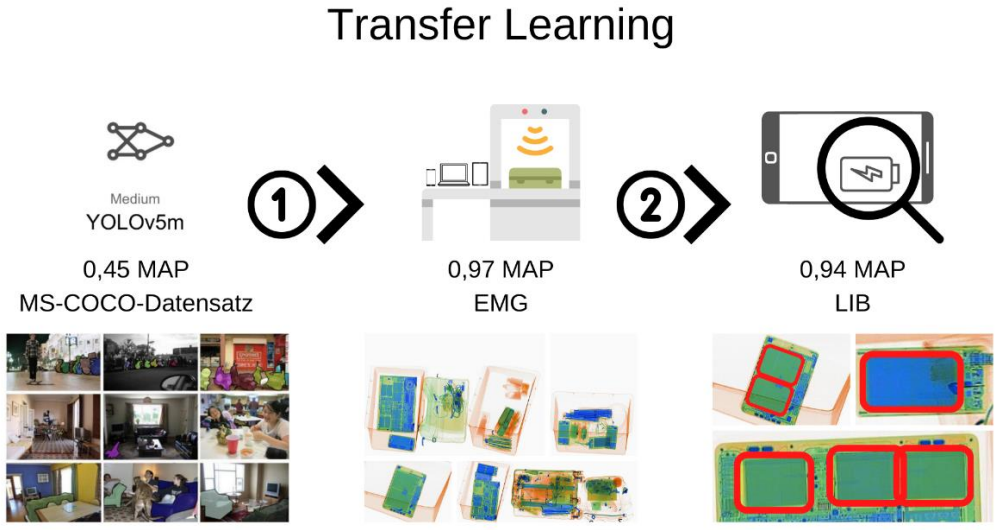

Abbildung 23: Transfer Learning Zusammenfassung

Quelle: Eigene Darstellung

Der erste Transfer wurde mit den bereits durch Ultralytics (2022) vortrainierten Gewichten für das Modell YOLOv5m durchgeführt. Diese dienten als Initialgewichte für das Training der fünf EMG-Klassen, wobei nach 20 Epochen eine durchschnittliche MAP von ca. 0,97 erreicht werden konnte.

Die daraus entstandenen Gewichte wurden dann wiederum als Initialgewichte für das Training der drei LIB-Klassen verwendet. In Tabelle 19 wird deutlich, dass bereits im ersten Versuch nach 40 Trainingsepochen ein Performance-Wert von 0,906 erreicht wurde. Im direkten Vergleich zu Versuch 7, bei dem die Nutzung der bereits trainierten Initialgewichte unterlassen und nach 200 Epochen eine Performance von 0,874 erzielt wurde, schneidet das Modell also mittels Transfer Learning trotz einem Fünftel der Trainingsepochen um ca. 3,7% besser ab.

In der obigen Versuchsreihe wurde für die Verbesserung der Ergebnisse die Anzahl der Trainingsepochen bei der Erkennung der drei LIB-Klassen angehoben, die Initialgewichte wurden allerdings in allen Versuchen beibehalten, welche durch das Training der fünf EMG-Klassen entstanden sind. Trotz der Verbesserung der Resultate durch das Anheben der Epochenanzahl, sollte in zukünftigen Versuchen zunächst die Veränderung der Parameter des ursprünglichen Trainings durchgeführt werden. Denn im Sinne des Wissenstransfers stellt die allgemeinere Aufgabe, in diesem Fall die Erkennung der fünf EMG-Klassen, die Basis für das spezifischere Training zur Erkennung der drei LIB-Klassen dar.

Dennoch ist durch den direkten Vergleich der Nutzung von Transfer Learning mit demselben Training ohne Gewichtstransfer bewiesen, dass im Kontext der Erkennung und Klassifizierung von LIB auf X-Ray-Aufnahmen nicht nur Zeit gespart, sondern auch bessere Resultate erreicht werden können. Es lässt sich außerdem behaupten, dass der Gewichtstransfer eine bessere Grundvoraussetzung für das spezialisierte Training zur Erkennung von LIB schafft, weil allgemeine Features der X-Ray-Aufnahmen nicht neu erlernt werden müssen. Dadurch wird bereits in ersten Epochen ein höherer F1-Wert erzielt als beim Training ohne Transfer Learning.

5 Zusammenfassung und Ausblick

5.1 Erkenntnisse und Bezug auf Problemstellung

In der folgenden Abbildung wird die gesamte Methodik unter Angabe der Resultate nochmal zusammengefasst:

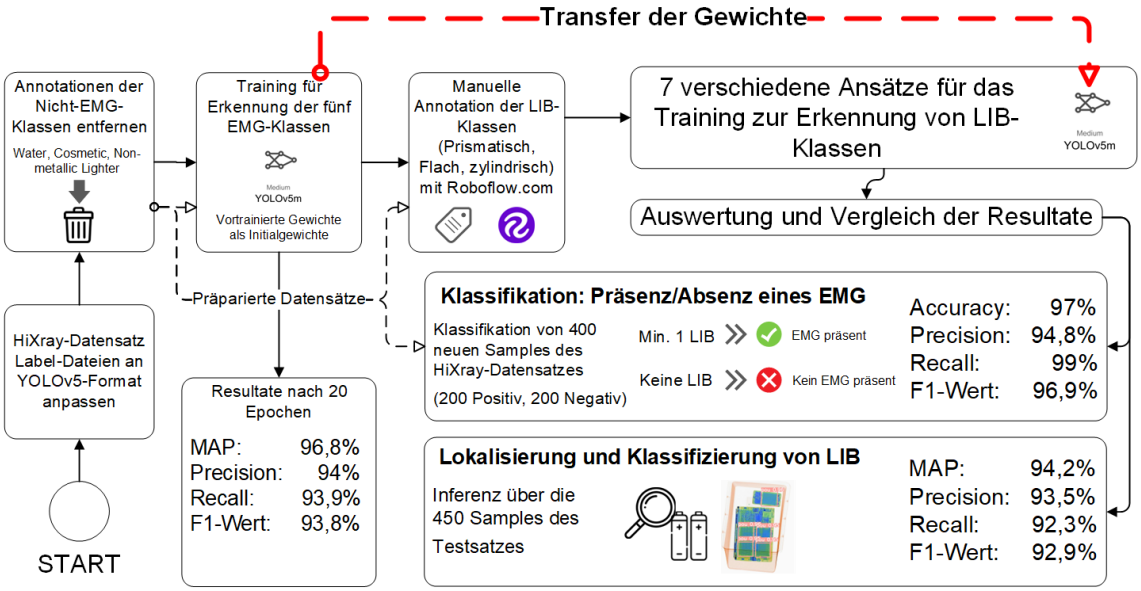

Abbildung 24: Grafische Zusammenfassung der Methodik

Quelle: Eigene Darstellung

Mittels der gewonnenen Erkenntnisse konnten die folgenden drei Fragestellungen unter Nutzung von YOLOv5m und dem Transfer der Gewichte wie folgt beantwortet werden:

5.1.1 Ist auf dem Bild ein EMG zu sehen?

In 4.4.1 wurden über die Nutzung der in 4.3.6 trainierten Gewichte die auf den X-Ray-Aufnahmen vorkommenden LIB gezählt und dadurch klassifiziert, ob mindestens ein EMG darauf präsent ist oder nicht. Mit dieser Methode konnte für den Testsatz, bestehend aus 200 Bildern mit EMG-Präsenz und 200 Bildern mit EMG-Absenz, eine Accuracy von 97% erzielt werden.

Überträgt man diese Resultate auf eine autonome Anwendung im Recycling oder der Sicherheitskontrolle am Flughafen, so ist dabei vor Allem auf die Anzahl der FN zu achten. Der Grund dafür ist, dass es, wenn bspw. in einer Sicherheitskontrolle fälschlicherweise die Absenz von EMG festgestellt wird, zu einem erhöhten Sicherheitsrisiko für die Passagiere kommen kann. Wird auf der anderen Seite fälschlicherweise die

Präsenz von EMG festgestellt, so kommt es lediglich zu einem erhöhten Zeitaufwand durch die separate Kontrolle des Gepäckstücks.

Da es um die Sicherheit von Menschen geht, kann das System nicht vollständig autonom eingesetzt und muss daher immer unter der Aufsicht des Sicherheitspersonals ausgeführt werden, solange die Fehlerquote nicht bei 0% liegt.

5.1.2 Wenn ja, wo befindet sich die LIB und zu welcher der drei Klassen gehört sie? (prismatische, flache oder zylindrische LIB)

Diese zwei Fragestellungen spielen eine Größere Rolle beim Automatisieren der Recycling-Prozesse von EMG und konnten durch die Versuchsreihe in 4.3 zusammen beantwortet werden. Die besten Ergebnisse für die Erkennung von LIB auf X-Ray-Aufnahmen konnten mittels des SOTA-Modells YOLOv5m, Transfer Learning, einem augmentierten Trainingssatz mit ca. 5700 verschiedenen Samples, einer Epochenanzahl von 200 und einem Confidence-Grenzwert von 0,6 erzielt werden. Die dabei erreichten durchschnittlichen Werte für Precision, Recall und MAP liegen bei 93,5%, 92,3% und 94,2%.

Die benötigte Zeit der Inferenz für ein neues Bild liegt bei 0,022 Sekunden, umgerechnet könnten also ca. 45 Bilder pro Sekunde verarbeitet werden. Somit ist die Übertragung auf ein Echtzeit-Objekterkennungssystem möglich.

Wie bereits zu Beginn dieser Arbeit erläutert, werden in sämtlichen Recyclinghöfen die Batterien mechanisch oder manuell aus den EMG extrahiert und sortiert. Würde man in den entsorgten EMG zunächst durch einen X-Ray-Sensor die darin befindlichen LIB lokalisieren und auch klassifizieren, so könnte daraufhin die maschinelle Separierung und Zuordnung für die spezifischen Recyclingprozesse erfolgen. Funktioniert diese Methodik vollständig autonom, so könnten erstens die Arbeitskräfte vom Sicherheitsrisiko während des gesamten Recyclingprozesses entlastet und zweitens sowohl Zeit als auch Kosten für das manuelle Sortieren gespart werden.

Auch hierbei gilt allerdings, dass für ein vollständig autonomes System eine Fehlerquote von mehr als 0% ausgeschlossen werden muss. Da auch in dieser Arbeit für Precision und Recall nicht 100% erreicht werden konnte, müssten für die Nutzung weiterhin zur Kontrolle Arbeiter in den Recyclingmanufakturen eingesetzt werden.

Nichtsdestotrotz konnte mit einem SOTA-Datensatz und -Machine-Learning-Modell der Stand der Technik verbessert werden und durch mögliche Verbesserungsstrategien der Methodik könnten in Zukunft die Recyclingprozesse von LIB, sowie die Erkennung von EMG an Sicherheitskontrollen vollständig automatisiert werden.

5.2 Mögliche Verbesserungsstrategien

5.2.1 Anpassung der Parameter und des Datensatzes

Bei der weiteren Untersuchung dieser Thematik könnten durch erneute Variation der Parameter beim Training oder Testen bessere Ergebnisse erzielt werden. Damit sind einerseits die Hyperparameter von YOLOv5 gemeint, welche in dieser Arbeit nicht verändert wurden und somit den von Ultralytics (2022) festgelegten Standards entsprechen. Andererseits könnten die Ergebnisse unter weiterer Erhöhung der Trainingsepochen oder Größe der Eingabebilder profitieren, denn die o.g. LIB-Klassen machen einen relativ kleinen Teil eines Gesamtbildes aus und könnten durch eine höhere Auflösung besser vom Modell erkannt werden.

Zudem weist der verwendete HiXray-Datensatz, wie in Abbildung 11 visualisiert wurde, Inkonsistenzen der Klassenannotationen, sowie Bildfehler in den Aufnahmen auf. Dem könnte bei weiteren Untersuchungen entgegengewirkt werden, indem der gesamte Datensatz manuell überprüft wird, wobei falsche Annotationen behoben, fehlende Annotationen hinzugefügt und Samples mit Bildfehlern entfernt werden.

Der Datensatz enthält ca. 45.000 Samples, wovon in dem hier durchgeführten Experiment aufgrund von mangelndem Arbeitsspeicher und der beanspruchten Zeit für die manuelle Annotation der LIB-Klassen lediglich weniger als die Hälfte benutzt wurde.

Führt man die Annotation für den restlichen Samples fort, oder werden in Zukunft neue thematisch passende Datensätze mit ähnlichen Annotationen veröffentlicht, so könnte die bisherige Datengrundlage erweitert werden und dadurch zu besseren Ergebnissen führen.

Außerdem ist zu beachten, dass die verwendeten Datensatzfragmente eine Imbalance in der Klassenverteilung aufweisen. Diese trug dazu bei, dass der Fokus des Trainings nicht gleichmäßig auf die verschiedenen Klassen verteilt wurde. Durch eine ausführlichere Strukturierung in der Vorbearbeitung des Datensatzes könnte diese Imbalance vermindert werden, was ggf. zu besseren durchschnittlichen Resultaten führt.

5.2.2 Verbesserung der Strategie bezüglich Transfer Learning

Um den Transfer der Gewichte zu optimieren, könnte das erste Training für die Erkennung der fünf EMG-Klassen mit einer wesentlich größeren Datenmenge, sowie Epochenanzahl durchgeführt werden.

Auf diese Weise ist es möglich, dass die fünf EMG-Klassen konsistenter und präziser erkannt werden und eine deutlich niedrigere Anzahl an Trainingsepochen für ähnliche Resultate bei der Erkennung der drei LIB-Klassen notwendig ist.

Auch die von Ultralytics (2022) bereitgestellten, vortrainierten Initialgewichte für YOLOv5m könnten unter Nutzung anderer öffentlicher Datensätze als MS-COCO (Lin et al., 2014), wie bspw. dem PASCAL-VOC- (Everingham et al., 2009), dem ImageNet-

Datensatz (Deng et al., 2009), oder der Kombination dieser, über eine höhere Anzahl von Epochen vortrainiert werden.

Danach könnten die Experimente Stück für Stück spezifischer werden, indem darauf bspw. ein Training unter Verwendung des in Tabelle 2 vorgestellten SIXray-Datensatzes mit mehr als einer Million X-Ray-Gepäckaufnahmen (Miao et al., 2019) folgt. Im Anschluss daran könnte wie oben beschrieben mit dem HiXray-Datensatz (Tao et al., 2021) mit ausschließlich den EMG-Klassen fortgefahren und zuletzt das Training für die Erkennung der drei LIB-Klassen durchgeführt werden. Im Laufe dieser Vorgehensweise würde also das Wissen des letzten Trainings auf einen immer spezielleren Kontext angewandt werden.

Wenn in den Recycling-Manufakturen auch andere Batterien als LIB mittels X-Ray-Aufnahmen automatisch verarbeitet werden sollen und zum Beispiel von diesen nur ein geringer Datensatz existiert, so könnten die aus dieser Arbeit ermittelten Gewichte auch auf den Kontext weiterer Batterietypen in X-Ray-Aufnahmen transferiert werden.

Allgemeiner betrachtet könnten die erhaltenen Gewichte auf die Erkennung jeglicher, relativ kleiner Objekte auf X-Ray-Aufnahmen transferiert werden. Ein Anwendungsbeispiel wäre die automatische Erkennung kleiner, spitzer Gegenstände bei Sicherheitskontrollen an Flughäfen.

5.2.3 Kombination verschiedener Sensoren

Darüber hinaus könnte man das trainierte Modell in einer Pipeline mit anderen Sensoren verbinden. Diese Sensoren könnten beispielsweise elektromagnetische Resonanzen der EMG und LIB messen und damit die Präsenz oder Klassifikation bestätigen.

Ein anderes Beispiel wäre im Recyclingprozess die äußerliche Untersuchung der bereits extrahierten LIB auf Komponentenbeschreibungen oder Recyclinghinweisen in Form von Texten oder Grafiken. Außerdem könnte anhand von eindeutigen äußerlichen Hinweisen auf die Marke und das jeweilige Modell der EMG eine Voraussage bezüglich Position und Art der LIB über ein weiteres NN getroffen werden. Dadurch könnten dann die Erkennungen der X-Ray-Analyse bestätigt und die Präzision ggf. erhöht werden.

5.2.4 Nutzung anderer SOTA-Architekturen oder -Datensätze

YOLOv5 ist ein öffentliches GitHub-Repository, an dem laufend Verbesserungen vorgenommen werden. Der hier vorgestellte Stand der Technik ist daher nur auf den Moment der Nutzung des Modells beschränkt.

Die zur Zeit der Versuchsdurchführung aktuelle Version wurde bereits in der Zwischenzeit von Ultralytics (2022) überarbeitet und hat sich im Vergleich zu den Resultaten aus Abbildung 3 wie folgt verbessert:

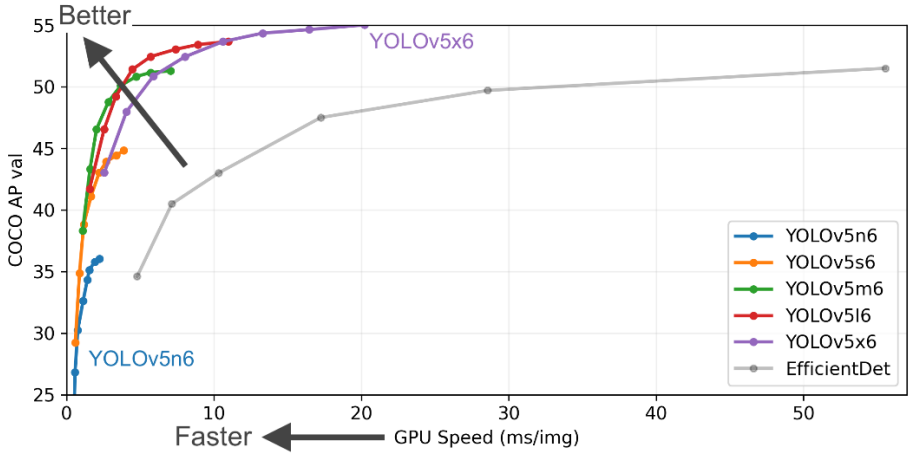

Abbildung 25: Vergleich der fünf YOLOv5.6.1-Modelle und EfficientDet

Quelle: Ultralytics (2022)

Verwendet man also für dieselbe Versuchsreihe aktuellere Versionen von YOLOv5, so sind durch Aktualisierungen der Modellarchitektur und den neuen Standards der Hyperparameter positive Auswirkungen auf Präzision und Performance zu erwarten. Da Ultralytics (2022) fünf verschiedene Varianten des Modells anbietet und in der obigen Versuchsreihe stets die mittlere, also YOLOv5m, gewählt wurde, könnte auch der Gebrauch der größeren Varianten YOLOv5l und YOLOv5x zu besseren Resultaten führen. Jedoch ist dabei jeweils die Zeit einer Inferenz zu beachten, welche mit zunehmender Modellgröße steigt.

Außerdem besteht die Möglichkeit, die Versuchsreihe auch mit anderen SOTA-Modellen durchzuführen, oder bestimmte Komponenten in der Architektur von YOLOv5 zu verändern oder zu ersetzen. Ein Beispiel ist das in 2.3 erwähnte Lateral-Inhibititon-Modul aus der Abhandlung von Tao et al. (2021).

Da aktuell im Bereich der Objekterkennung auf Bildern mittels Deep-Learning-Architekturen ständig neue Erkenntnisse gemacht und die Modelle immer präziser werden, könnte innerhalb der nächsten Jahre die Erkennung von EMG und der darin befindlichen LIB vollständig automatisiert werden.

6 Anhänge

Anhang 1: Datensatz

Anhang 1.1: Beispiel eines Samples

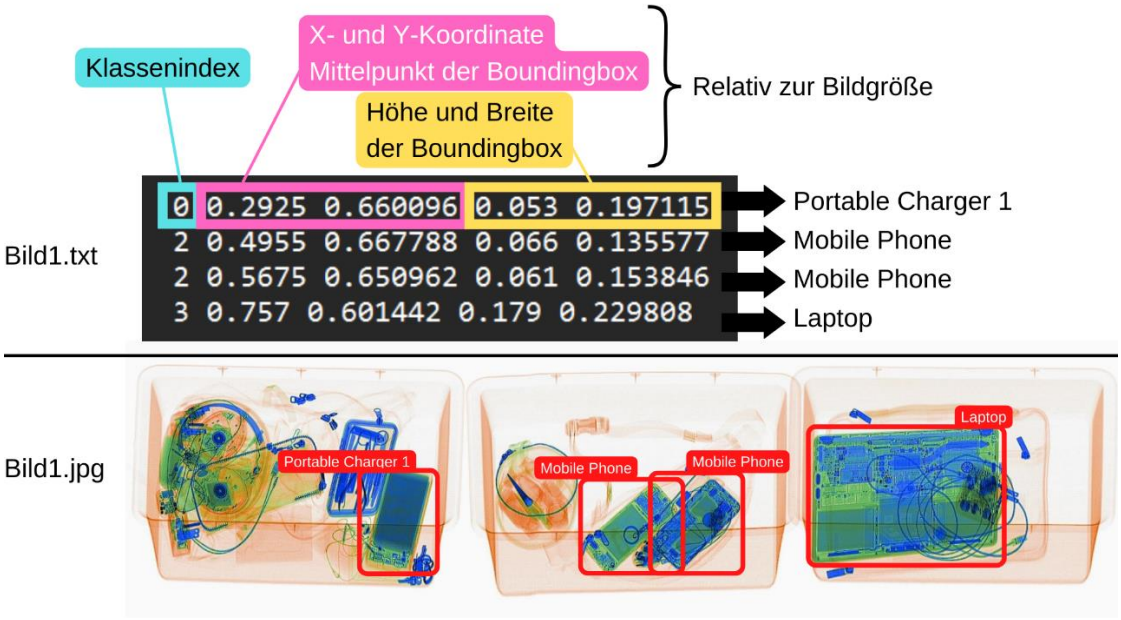

Abbildung 26: Aufbau eines Samples

Quelle: Eigene Darstellung

Anhang 1.2: Beispielbilder mit den 5 EMG-Klassen

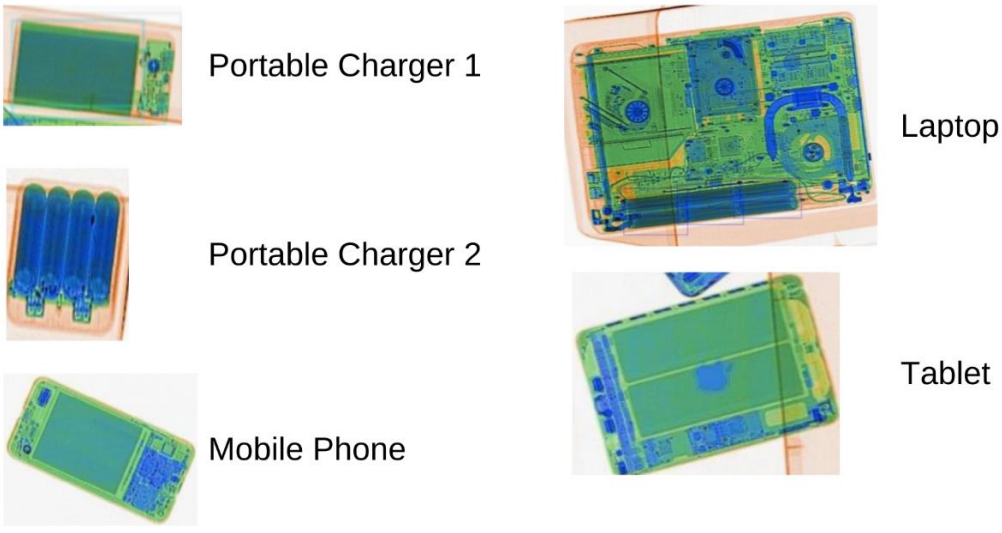

Abbildung 27: Beispiele der fünf EMG-Klassen im Datensatz

Quelle: Eigene Darstellung

Anhang 1.3: Beispielbilder mit den 3 LIB-Klassen

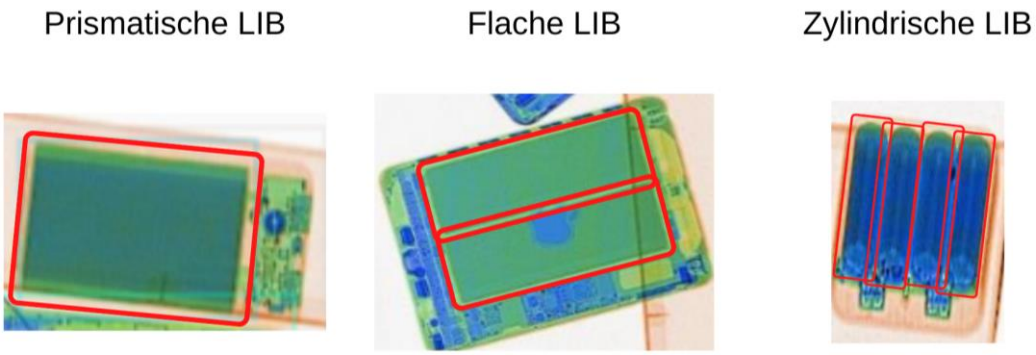

Abbildung 28: Beispiele der drei LIB-Klassen

Quelle: Eigene Darstellung

Anhang 1.4: Beispielbilder ohne jegliche Klassen

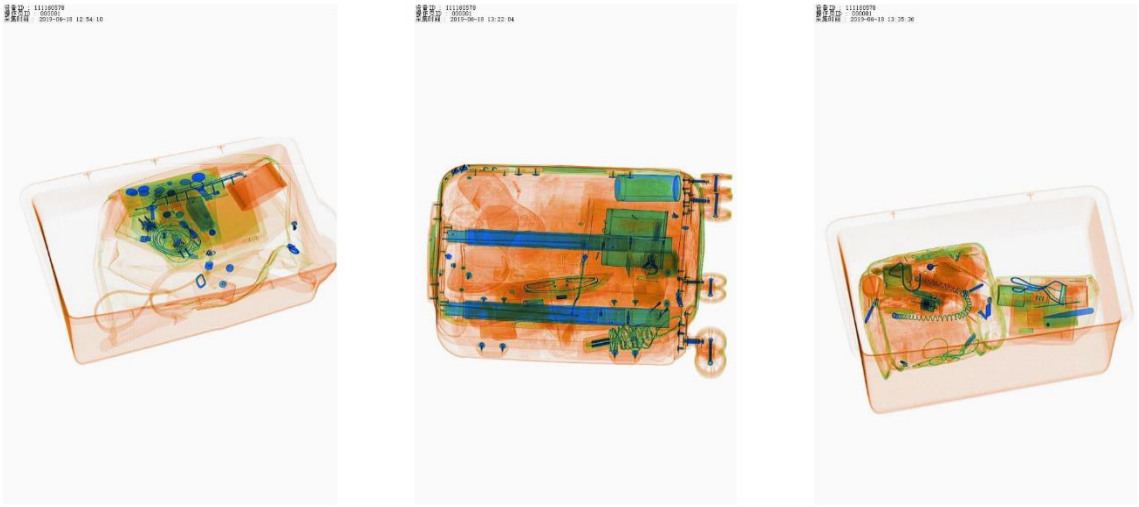

Abbildung 29: Drei Beispiele von Samples ohne jegliche Klassen

Quelle: Eigene Darstellung

Anhang 2: Ergebnisse der Probetrainings mit YOLOv5s

Tabelle 22: Ergebnisse Probetrainings mit YOLOv5s

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7
Anzahl Samples Training	100	1000	5000	10000	**15000**	20000	25000
Anzahl Samples Test	25	250	1250	2500	**3750**	5000	6250
Batch Size	10	10	50	50	50	50	50
Epochen	50	50	30	15	15	10	10
Precision	0,628	0,868	0,902	0,951	0,95	0,949	Abgebr.
Recall	0,844	0,834	0,878	0,936	0,961	0,949	Abgebr.
MAP	0,765	0,855	0,896	0,958	0,972	0,974	Abgebr.
F1-Wert	0,720	0,850	0,89	0,943	0,955	0,949	Abgebr.

Anhang 3: Programmcode

Anhang 3.1: Einrichtung von YOLOv5 in GCP

```
!git clone https://github.com/ultralytics/yolov5.git

%cd /content/yolov5

!pip install -r requirements.txt

!/content/yolov5/data/scripts/download_weights.sh
```

Anhang 3.2: Konvertierung der Label-Dateien in Python

Anhang 3.2.1: Konvertierung einer Zeile einer Label-Datei in das YOLOv5-Format

```
from PIL import Image

classes = ['PortableCharger1', 'PortableCharger2', 'MobilePhone',
'Laptop', 'Tablet', 'Cosmetic', 'Water', 'NonmetallicLighter']

def convert_line(line, img_path):
    line_array = str.split(line)
    image = Image.open(img_path + '/' + line_array[0])

    img_width = image.width
    img_height = image.height
    device_index = classes.index(line_array[1])

    width = int(line_array[4]) - int(line_array[2])
    normalized_width = width / img_width

    height = int(line_array[5]) - int(line_array[3])
    normalized_height = height / img_height

    x_center = int(line_array[2]) + (width / 2)
    normalized_x_center = x_center / img_width

    y_center = int(line_array[3]) + (height / 2)
    normalized_y_center = y_center / img_height

    new_line = str(device_index) + " " +
str(round(normalized_x_center, 6)) + " " + str(
        round(normalized_y_center, 6)) + " " +
str(round(normalized_width, 6)) + " " + str(
        round(normalized_height, 6))

    return new_line
```

Anhang 3.2.2: Konvertierung aller Label-Dateien eines Trainings- oder Testsatzes

```
def convert_all_labels(data_path, img_path):
    path_list = Path(data_path).rglob('*.txt')
    for path in path_list:
        path_in_str = str(path)
        print(path_in_str + ':')
        converted_lines = []
        with open(path_in_str, 'r') as file:
            for line in file:
                print(line)
                converted_lines.append(convert_line(line, img_path))
        print('New Lines:')
        with open(path_in_str, 'w') as file:
            for new_line in converted_lines:
                file.write(new_line + '\n')
                print(new_line)
```

Anhang 3.3: Entfernen irrelevanter Klassen in Python

```
import os

# Klassen werden im folgenden Format angegeben: Bsp.: ['1', '2', '3']

def remove_irrelevant_classes(data_path, irrelevant_classes:
list[str]):
    data_path_list = Path(data_path).rglob('*.txt')
    for path in data_path_list:
        path_in_str = str(path)

        with open(path_in_str, 'r') as file:
            lines = file.readlines()

        with open(path_in_str, "w") as f:
            for line in lines:
                current_class = str.split(line)[0]
                if current_class not in irrelevant_classes:
                    f.write(line)

    print('Done removing unnecessary labels')
```

Anhang 3.4: Aufteilung der fünf EMG-Klassen in Python

Anhang 3.4.1: Entfernen der Bilder ohne zugehörige Label-Datei

```
import os

def remove_img_without_label(data_path, img_path):
    all_labels = []
    for root, dirs, files in os.walk(data_path):
        for filename in files:
            all_labels.append(str.replace(filename, '.txt', ''))
    for root, dirs, files in os.walk(img_path):
        for img_name in files:
            if str.replace(img_name, '.jpg', '') not in all_labels:
                print('Removing: ' + img_name)
                os.remove(img_path + '\\' + img_name)
```

Anhang 3.4.2: Aufteilung des Datensatzes nach EMG-Klassen

```
import os
import random

def split_dataset(data_path, image_path, target_sample_count):
    pouch_classes = ['2', '3', '4']  # EMG-Klassen mit flachen LIB
    # Erstellen eines randomisierten Arrays der Dateien
    all_files_array = []
    pathlist = Path(data_path).rglob('*.txt')
    for path in pathlist:
        all_files_array.append(str(path))
    random.Random(40).shuffle(all_files_array)
    pris = 0  # Zähler für EMG mit prismatischen LIB
    cyli = 0  # Zähler für EMG mit zylindrischen LIB
    pouch = 0  # Zähler für EMG mit flachen LIB
    keep_files_array = []  # Array mit den zu behaltenden Dateien
    for path_in_str in all_files_array:
        if len(keep_files_array) < target_sample_count:
            current_pris = 0
            current_cyli = 0
            current_pouch = 0
            with open(path_in_str, 'r') as file:
                lines = file.readlines()
                for line in lines:
                    current_class = str.split(line)[0]
                    if current_class == '0':
                        current_pris += 1
                    if current_class == '1':
                        current_cyli += 1
                    if current_class in pouch_classes:
                        current_pouch += 1
                # Nur behalten, wenn mindestens ein EMG mit
prismatischer
                # oder zylindrischer LIB vorkommt
                if current_pris > 0 or current_cyli > 0:
                    keep_files_array.append(path_in_str)
                    pris += current_pris
                    cyli += current_cyli
                    pouch += current_pouch
            file.close()

    # Löschen der unbenutzten Samples
    pathlist = Path(data_path).rglob('*.txt')
    for path in pathlist:
        path_str = str(path)
        if path_str not in keep_files_array:
            os.remove(path_str)
    remove_img_without_label(data_path, image_path)

    # Ausgeben der erreichten Aufteilung
    print('PO1: ' + str(pris))
    print('PO2: ' + str(cyli))
    print('Other: ' + str(pouch))
    print('Filecount: ' + str(len(keep_files_array)))
```

Anhang 3.5: Aufteilung der drei LIB-Klassen in Python

Anhang 3.5.1: Filtern aller Samples, in denen bestimmte Klassen vorkommt

```
import os

# Klassen werden im folgenden Format angegeben: Bsp.: ['1', '2', '3']

def filter_irrelevant_samples(data_path, img_path, irrelevant_classes:
list[str]):
    path_list = Path(data_path).rglob('*.txt')
    for path in path_list:
        path_in_str = str(path)
        count = 0
        with open(path_in_str, 'r') as file:
            lines = file.readlines()
            for line in lines:
                current_class = str.split(line)[0]
                if current_class in irrelevant_classes:
                    count += 1
        file.close()
        if count > 0:
            os.remove(path_in_str)
    remove_img_without_label(data_path, img_path)
```

Anhang 3.5.2: Erstellen von Sample-Pools für je eine LIB-Klasse

```
# Dateipfade der 3 identischen Datensatzkopien
data_path_1_train = "Pfad/zur/Datensatzkopie1/train/labels"
data_path_1_test = "Pfad/zur/Datensatzkopie1/test/labels"
data_path_2_train = "Pfad/zur/Datensatzkopie2/train/labels"
data_path_2_test = "Pfad/zur/Datensatzkopie2/test/labels"
data_path_3_train = "Pfad/zur/Datensatzkopie3/train/labels"
data_path_3_test = "Pfad/zur/Datensatzkopie3/test/labels"
img_path_1_train = "Pfad/zur/Datensatzkopie1/train/images"
img_path_1_test = "Pfad/zur/Datensatzkopie1/test/images"
img_path_2_train = "Pfad/zur/Datensatzkopie2/train/images"
img_path_2_test = "Pfad/zur/Datensatzkopie2/test/images"
img_path_3_train = "Pfad/zur/Datensatzkopie3/train/images"
img_path_3_test = "Pfad/zur/Datensatzkopie3/test/images"

# Datensatz-Pool für Samples mit nur EMG mit prismatischer LIB
filter_irrelevant_samples(data_path_1_train, img_path_1_train, ['2',
'3', '4', '5'])
filter_irrelevant_samples(data_path_1_test, img_path_1_test, ['2',
'3', '4', '5'])
# Datensatz-Pool für Samples mit nur EMG mit zylindrischer LIB
filter_irrelevant_samples(data_path_2_train, img_path_2_train, ['1',
'3', '4', '5'])
filter_irrelevant_samples(data_path_2_test, img_path_2_test, ['1',
'3', '4', '5'])
# Datensatz-Pool für Samples mit nur EMG mit flacher LIB
filter_irrelevant_samples(data_path_3_train, img_path_3_train, ['1',
'2'])
filter_irrelevant_samples(data_path_3_test, img_path_3_test, ['1',
'2'])
```

Anhang 3.6: Hyperparameter von YOLOv5m

```
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

Anhang 3.7: Verkleinern der Bilder des 3. Datensatzes für YOLOv5-Klassifikator in Python

```
def squeeze_images(img_path, img_size: int):
    for root, dirs, files in os.walk(img_path):
        for img_name in files:
            img = cv2.imread(img_path + '\\' + img_name, 1)

            # Bild auf die angegebene Größe verkleinern
            img_stretch = cv2.resize(img, (img_size, img_size))
            cv2.imwrite(img_path + '\\' + img_name, img_stretch)
```

Anhang 3.8: Klassifikation durch Abzählen in Python

```
from PIL import Image
import torch

# Model
model = torch.hub.load('ultralytics/yolov5', 'custom',
path='last_weights_200_epochs.pt', force_reload=True)
model.conf = 0.6

# Ordner mit Bildern der Klasse „Kein EMG vorhanden“ klassifizieren
def classify_no_electronic_devices(img_path):
    files = Path(img_path).glob('*.jpg')
    true_negative = 0
    all_file_count = 0
    for file in files:
        all_file_count += 1
        path_in_str = str(file)
        img = Image.open(path_in_str)
        # Inference
        results = model(img)
        if results.pandas().xyxy[0].empty:
            true_negative += 1
    accuracy = true_negative / all_file_count
    false_positive = all_file_count - true_negative
    print('Accuracy:')
    print(str(accuracy))
    print('True Negative:')
    print(str(true_negative))
    print('False Positive:')
    print(str(false_positive))

# Ordner mit Bildern der Klasse „EMG vorhanden“ klassifizieren
def classify_electronic_devices(img_path):
    files = Path(img_path).glob('*.jpg')
    true_positive = 0
    all_file_count = 0
    for file in files:
        all_file_count += 1
        path_in_str = str(file)
        img = Image.open(path_in_str)
        # Inference
        results = model(img)
        if not results.pandas().xyxy[0].empty:
            true_positive += 1
    accuracy = true_positive / all_file_count
    false_negative= all_file_count - true_positive
    print('Accuracy:')
    print(str(accuracy))
    print('True Positive:')
    print(str(true_positive))
    print('False Negative:')
    print(str(false_negative))
```

Anhang 3.9: Klassifikation durch YOLOv5-Klassifikator in Python

```
from yolov5.classifier import *
import torch

# model
model = torch.load('best.pt',
map_location=torch.device('cpu'))['model'].float()


# Ordner mit Bildern der Klasse „Kein EMG vorhanden“ klassifizieren
def classify_no_electronic_devices(img_path):
    files = Path(img_path).glob('*.jpg')
    true_negative = 0
    all_file_count = 0
    for file in files:
        all_file_count += 1
        path_in_str = str(file)
        prediction = classify(model, size=600, file=path_in_str)
        if prediction == '1':
            true_negative += 1
    accuracy = true_negative / all_file_count
    false_positive = all_file_count - true_negative
    print('Accuracy:')
    print(str(accuracy))
    print('True Negative:')
    print(str(true_negative))
    print('False Positive:')
    print(str(false_positive))


# Ordner mit Bildern der Klasse „EMG vorhanden“ klassifizieren
def classify_electronic_devices(img_path):
    files = Path(img_path).glob('*.jpg')
    true_positive = 0
    all_file_count = 0
    for file in files:
        all_file_count += 1
        path_in_str = str(file)
        prediction = classify(model, size=600, file=path_in_str)
        if prediction == '0':
            true_positive += 1
    accuracy = true_positive / all_file_count
    false_negative = all_file_count - true_positive
    print('Accuracy:')
    print(str(accuracy))
    print('True Positive:')
    print(str(true_positive))
    print('False Negative:')
    print(str(false_negative))
```

7 Glossar

Bounding-Box: Die Bounding-Box umrahmt ein Objekt einer spezifischen Klasse auf einem Bild. Die Koordinaten dieser werden zunächst als Annotationen in den Bildern zugehörigen Text-Dateien angegeben und stellen die korrekten Klassifikationen der Objekte für das Training des Modells dar. Bei einer Inferenz trifft das Modell dann Voraussagen über Position und Klasse des jeweiligen Objektes.

Intersection over Union (IOU): Die IOU beschreibt das Verhältnis der Überschneidung von erkannter Bounding-Box und korrekter Bounding-Box zu der Gesamtfläche der beiden Bounding-Boxen. Es wird für die Auswertung der Ergebnisse ein Schwellenwert der IOU festgelegt, ab dem eine erkannte Bounding-Box als TP angesehen wird.

Confidence-Grenzwert: Der Wert der Confidence gibt an, zu welcher Wahrscheinlichkeit eine Objektklasse bei der Inferenz auf neuen Samples erkannt wurde. Der Grenzwert dieser Metrik beschreibt dementsprechend einen Wert, ab dem die Erkennung als solche gewertet wird.

Literaturverzeichnis

Leuthner, S., Fleischhammer, M. & Döring, H. (2013). Handbuch Lithium-Ionen-Batterien. Springer Vieweg. https://doi.org/10.1007/978-3-642-30653-2

Statista. (2022, 14. März). Global smartphone sales to end users 2007–2021. Abgerufen am 5. Mai 2022, von https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/

Federal Aviation Administration. (2022, 29. April). Lithium Battery Incident Chart. Abgerufen am 5. Mai 2022, von https://www.faa.gov/hazmat/resources/lithium-battery-incident-chart

Zheng, X., Zhu, Z., Lin, X., Zhang, Y., He, Y., Cao, H. & Sun, Z. (2018). A Mini-Review on Metal Recycling from Spent Lithium Ion Batteries. Engineering, 4(3), 361–370. https://doi.org/10.1016/j.eng.2018.05.018

Statistisches Bundesamt. (2022, 11. Februar). *Pressemitteilung Nr. 058 vom 11. Februar 2022* [Pressemeldung]. https://www.destatis.de/DE/Presse/Pressemitteilungen/2022/02/PD22_058_321.html

Bundespolizei. (o. D.). *Bundespolizei - Die Sicherheitskontrolle: So kommen Sie gut an Bord*. Abgerufen am 9. Mai 2022, von https://www.bundespolizei.de/Web/DE/01Sicher-auf-Reisen/01Mit-dem-Flugzeug/01Sicherheitskontrolle/sicherheitskontrolle_node.html

European Union Aviation Safety Agency. (o. D.). *Dangerous Goods*. EASA. Abgerufen am 9. Mai 2022, von https://www.easa.europa.eu/domains/passengers/dangerous-goods

Europäische Kommission. (2020, 11. März). Änderung unserer Produktions- und Verbrauchsmuster: neuer Aktionsplan für Kreislaufwirtschaft ebnet Weg zu klimaneutraler und wettbewerbsfähiger Wirtschaft mit mündigen Verbrauchern [Pressemeldung]. https://ec.europa.eu/commission/presscorner/detail/de/ip_20_420

Europäisches Parlament. (2022, 20. April). *Elektro- und Elektronikschrott in der EU: Zahlen und Fakten*. Abgerufen am 16. Mai 2022, von https://www.europarl.europa.eu/news/de/headlines/priorities/kreislaufwirtschaft/20201208STO93325/elektroschrott-in-der-eu-zahlen-und-fakten-infografik

Sortbat. (o. D.). *Sorting batteries*. Abgerufen am 16. Mai 2022, von https://www.sortbat.be/expertise/sorting-batteries

Sterkens, W., Diaz-Romero, D., Goedemé, T., Dewulf, W. & Peeters, J. R. (2021). Detection and recognition of batteries on X-Ray images of waste electrical and electronic equipment using deep learning. *Resources, Conservation and Recycling*, *168*, 105246. https://doi.org/10.1016/j.resconrec.2020.105246

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X. & Pietikäinen, M. (2019). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, *128*(2), 261–318. https://doi.org/10.1007/s11263-019-01247-4

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014*, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

Lu, X., Li, Q., Li, B. & Yan, J. (2020). MimicDet: Bridging the Gap Between One-Stage and Two-Stage Object Detection. *Computer Vision – ECCV 2020*, 541–557. https://doi.org/10.1007/978-3-030-58568-6_32

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y. & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

He, K., Gkioxari, G., Dollar, P. & Girshick, R. (2017). Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv.2017.322

Ren, S., He, K., Girshick, R. & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031

Tan, M., Pang, R. & Le, Q. V. (2020). EfficientDet: Scalable and Efficient Object Detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr42600.2020.01079

Ultralytics. (o. D.). YOLOv5 and Vision AI - Ultralytics. Abgerufen am 25. Mai 2022, von https://ultralytics.com

Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W. & Yeh, I. H. (2020). CSPNet: A New Backbone that can Enhance Learning Capability of CNN. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/cvprw50498.2020.00203

Liu, S., Qi, L., Qin, H., Shi, J. & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8759–8768. https://doi.org/10.1109/cvpr.2018.00913

He, K., Zhang, X., Ren, S. & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. Computer Vision – ECCV 2014, 346–361. https://doi.org/10.1007/978-3-319-10578-9_23

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. https://doi.org/10.48550/arXiv.1804.02767

Ultralytics. (2022, 22. Februar). Releases · ultralytics/yolov5. GitHub. Abgerufen am 27. Mai 2022, von https://github.com/ultralytics/yolov5/releases

Miao, C., Xie, L., Wan, F., Su, C., Liu, H., Jiao, J. & Ye, Q. (2019). SIXray: A Large-Scale Security Inspection X-Ray Benchmark for Prohibited Item Discovery in Overlapping Images. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2114–2123. https://doi.org/10.1109/cvpr.2019.00222

Wei, Y., Tao, R., Wu, Z., Ma, Y., Zhang, L. & Liu, X. (2020). Occluded Prohibited Items Detection: An X-ray Security Inspection Benchmark and De-occlusion Attention Module. Proceedings of the 28th ACM International Conference on Multimedia, 138–146. https://doi.org/10.1145/3394171.3413828

Wang, B., Zhang, L., Wen, L., Liu, X. & Wu, Y. (2021). Towards Real-World Prohibited Item Detection: A Large-Scale X-ray Benchmark. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 5392–5401. https://doi.org/10.1109/iccv48922.2021.00536

Tao, R., Wei, Y., Jiang, X., Li, H., Qin, H., Wang, J., Ma, Y., Zhang, L. & Liu, X. (2021). Towards Real-world X-ray Security Inspection: A High-Quality Benchmark And Lateral Inhibition Module For Prohibited Items Detection. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 10903–10912. https://doi.org/10.1109/iccv48922.2021.01074

Mery, D., Riffo, V., Zscherpel, U., Mondragón, G., Lillo, I., Zuccar, I., Lobel, H. & Carrasco, M. (2015). GDXray: The Database of X-ray Images for Nondestructive Testing. Journal of Nondestructive Evaluation, 34(4). https://doi.org/10.1007/s10921-015-0315-7

Everingham, M., van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. (2009). The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, 88(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Roboflow, Inc. (o. D.). *Roboflow: Give your software the power to see objects in images and video*. Roboflow. Abgerufen am 9. Juni 2022, von https://roboflow.com/

Deng, J., Dong, W., Socher, R., Li, L. J., Kai Li & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2009.5206848

d-side conferences. (o. D.). *Home | ESANN 2022*. ESANN. Abgerufen am 27. Juni 2022, von https://www.esann.org/

Baker, N. A., Rohrschneider, D. & Handmann, U. (2022, 10. Mai). Battery detection of XRay images using transfer learning. *30th European Symposium on Artificial Neural Networks (ESANN 2022)*.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht. Die vorgelegte Arbeit hat weder in der gegenwärtigen noch in einer anderen Fassung schon einem anderen Fachbereich der Hochschule Ruhr West oder einer anderen wissenschaftlichen Hochschule vorgelegen.

Mülheim, 03.07.2022 J. Rohrschneider

Ort, Datum Unterschrift