INSTITUT FÜR INFORMATIK
Fachbereich 12 – Informatik und Mathematik

Masterthesis

Deep Learning zur Objekterkennung und Größenapproximation antiker Keramikgefäße auf historischen Fotografien

Alicia Wirth

Abgabe am 22. November 2022

eingereicht bei
Dr. Karsten Tolle
Texttechnologie

Eidesstattliche Erklärung

gemäß der Ordnung des Masterstudiengangs Informatik 2019 §34 Abs. 16

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe.

Ebenso bestätige ich, dass diese Arbeit nicht, auch nicht auszugsweise, für eine andere Prüfung oder Studienleistung verwendet wurde.

Zudem versichere ich, dass die von mir eingereichten schriftlichen gebundenen Versionen meiner Masterarbeit mit der eingereichten elektronischen Version meiner Masterarbeit übereinstimmen.

Frankfurt am Main, den 22. November 2022 Alicia Wirth

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1
Einleitung

"History is written by the victors" ist ein berühmtes Idiom unbekannter Herkunft[1], das die Glaubwürdigkeit von Aufzeichnungen früherer Zeitalter in Frage stellt. Die frühe menschliche Geschichte ist nicht lückenlos. Da lediglich Personen höheren Standes dem Schreiben mächtig waren, liegen zudem nur spärliche Aufzeichnungen vor, deren Legitimität und Unbefangenheit angezweifelt werden kann [Met04]. Eine Wissenschaft zur Aufschlüsslung entsprechender geschichtlicher Lücken und spärlicher Aufzeichnungen ist die Archäologie [KTM+18]. Anhand materieller Überbleibsel in Form von Artefakten frühzeitlicher Gesellschaften wie Keramikgefäßen, kann die Archäologie nicht nur das Alter, die Form, die Funktion oder die sozio-ökonomische Bedeutung dieser Überbleibsel ermitteln [RYNG21], sondern auch Rückschlüsse auf die Dynamiken ihrer Produktion, Handelsflüsse oder soziale Interaktionen ziehen [ABG+20].

Insbesondere die Insel Zypern ist aufgrund ihrer durch Fremdherrschaft geprägten Geschichte und damit einhergehender kultureller Vielfalt interessant für die Archäologie [Fis17]. So war Zypern unter anderem Teil des Römischen, Osmanischen sowie Britischen Reichs und wurde zeitweise auch von Mächten wie Venedig beherrscht. Mittlerweile ist die Insel in die Republik Zypern im Süden und die Türkische Republik Nordzypern gespalten [Fis17, MS21].

Die archäologische Ausbeutung Zyperns begann bereits in der Mitte des 19. Jh. und unterlag primär dem Zweck, verwertbare Antiken zu finden, die sich sowohl in etlichen Museen als auch in Privatsammlungen wieder finden lassen [TB01]. Gerade jene verwerteten Antiken stellen eine besondere Herausforderung in der wissenschaftlich-archäologischen Bewertung und Klassifikation dar [RT21]. Häufig ist nicht einmal ihre Provenienz bekannt, wodurch ihre insbesondere für die Wissenschaft relevante Zuordnung stark erschwert wird. Mit Hilfe von Techniken der Informatik wird versucht, diesen Prozess zu automatisieren und damit zu unterstützen sowie zu beschleunigen. An eben dieser Stelle knüpft die vorliegende Masterthesis an.

[1] Häufig wird das Idiom Winston Churchill zugeschrieben, wofür jedoch kein tatsächlicher Beleg gefunden werden kann.

1.1 Aufgabenstellung

Mit der Masterthesis soll der Grundstein der LOEWE-Exploration *Künstliche Intelligenz zur Erschließung kolonialer Verwertungspraktiken archäologischer Objektsammlungen* im Rahmen des Big Data Labs der Goethe-Universität Frankfurt am Main gelegt werden. Das Ziel des Projekts ist die automatisierte Identifikation und Analyse von circa 5 000 zyprischen Antiken, die in Sammlungen auf bis zu 100 historischen Fotografien abgebildet werden. Damit soll ihre wissenschaftlich-archäologische Zuordnung unterstützt und die Automatisierung, wenn möglich, auf vergleichbare Archivbestände angewandt werden können.

Ein Teil davon soll in der Masterthesis mittels Objekterkennung unter Verwendung von Deep Learning-Techniken realisiert werden. Der Fokus wird dabei allein auf Fotografien von Keramikgefäßsammlungen gelegt. Neben der Identifikation der groben Oberkategorien, wie zum Beispiel *Amphoriskos*, und Lokalisierung der abgebildeten Keramikgefäße ist auch ihre zweidimensionale Höhe und Breite zu approximieren. Dazu kann ein auf rund der Hälfte der Fotografien abgebildeter Maßstab in Form eines Lineals herangezogen werden. Die Aufgabe der Thesis lässt sich damit in drei Teilaufgaben zerlegen:

1. Die Lokalisierung des Lineals

2. Die Erkennung und Lokalisierung der einzelnen Keramikgefäße

3. Die Approximation der Größenmaße der lokalisierten Keramikgefäße

Da aufgrund des spärlichen Datensatzes dem Training Neuronaler Netze nur mit wenig Erfolg entgegengeblickt wird, sollen verschiedene Objekterkennungsdetektoren untersucht und angewandt werden. Ihre für den Anwendungsfall erzielten Ergebnisse sollen anschließend verglichen und evaluiert werden. Daraus ergibt sich nicht nur die Notwendigkeit umfassender Recherchen, sondern auch das Herausfiltern vielversprechender Detektoren, eine Vorverarbeitung und Annotation der Daten sowie das Training und die Evaluation der Detektoren unter Verwendung adäquater Techniken. Insbesondere im Falle der Annotation der Keramikgefäße ist die interdisziplinäre Zusammenarbeit und Kommunikation mit dem Institut für Archäologische Wissenschaften der Goethe-Universität unabdingbar.

Es ist außerdem zu beachten, dass zunächst die Findung einer spezifischen, adäquaten Lösung des Anwendungsfalls zum Ziel steht und keine allgemeine Verbesserung der Genauigkeit im Hinblick auf andere Archivbestände.

1.2 Struktur der Thesis

Zur Lösung der Aufgabenstellung wird die Masterthesis schließlich folgenderweise strukturiert:

Das auf die Einleitung folgende Kapitel führt zunächst den aktuellen Forschungsgegenstand der Detektion und Größenapproximation von Objekten auf Bildern mit besonderem Fokus auf die Archäologie auf.

Im dritten Kapitel werden dann die benötigten theoretischen Grundlagen der Thesis vermittelt. Sie umfassen sowohl die vorliegende Datenbasis als auch Grundlagen, Methoden, Detektoren und Evaluationsmetriken der Objekterkennung sowie die theoretische Größenapproximation auf Bildern.

Mit diesem Hintergrund wird zunächst im vierten Kapitel ein Konzept zur praktischen Lösung der Aufgabenstellung erarbeitet. Dazu werden als erstes grobe Handlungsansätze diskutiert und anschließend mögliche Herausforderungen identifiziert. Die Handlungsansätze können schließlich zu Lösungsstrategien präzisiert werden, die dann unter der Wahl von zu verwendenden Objekterkennungsdetektoren zu einem Implementierungskonzept zusammengefasst werden.

Jenes Konzept wird schließlich im fünften Kapitel implementiert und die Ergebnisse evaluiert. Die Schritte der Implementation und Evaluation sind dabei unter einem Kapitel zusammengefasst, da die identifizierten Teilaufgaben aufeinander aufbauen und demnach zwischenzeitlich einer spezifischen Evaluation bedürfen. Zunächst wird jedoch die Implementation der benötigten Techniken sowie eine Beschreibung der Anwendung der Objekterkennungsdetektoren aufgeführt. Erst danach werden die Teilaufgaben beginnend mit der Lineal- und Keramikgefäßdetektion behandelt. Sie umfassen die Annotation und Distribution der Daten sowie das Training und eine Deskription sowie Evaluation der erzielten Ergebnisse. Dabei ist die Keramikgefäßerkennung und -lokalisierung durch zusätzliche zum Einsatz kommende Techniken und Gesichtspunkte deutlich umfangreicher. Schließlich werden die Objekterkennungsdetektoren gegenübergestellt.

Hingegen umfasst das Unterkapitel der dritten Teilaufgabe den vollständigen Weg bis hin zur Größenapproximation der Keramikgefäße, ehe die Werte anhand tatsächlicher Größenmaße evaluiert werden. Letztlich folgt eine gesamte, abschließende Evaluation anhand der in Kapitel 4 identifizierten Herausforderungen.

Die Masterthesis wird schließlich anhand einer Zusammenfassung und eines Fazits sowie vorgeschlagenen zukünftigen Verbesserungsmöglichkeiten in Kapitel 6 beendet.

2
Stand der Forschung

Das folgende Kapitel umfasst die der Bearbeitung der Masterthesis vorgegangene Recherche bezüglich der Erkennung und Größenapproximation von Objekten auf Bildern. Ein besonderer Fokus liegt dabei auf ihrem Einsatz im Feld der Archäologie. Damit bildet das Kapitel den aktuellen Forschungsstand ab und dient den kommenden Kapiteln als Grundlage.

2.1 Objekterkennung mittels Deep Learning

Aufgrund bahnbrechender Forschungsergebnisse und damit verbundenen unterschiedlichen Anwendungen, hat die maschinelle Objekterkennung auf Bild- oder Videomaterial in den vergangenen Jahren immer mehr Aufmerksamkeit gewonnen [PPMM21a]. Die Gesichtserkennung [TYRW14], das autonome Fahren [CSKX15] und die Unterstützung im medizinischen Bereich, wie beim Erkennen von Hautkrebs [EKN+17], sind nur wenige ihrer Einsatzgebiete.

Auch der Fortschritt und die Zugänglichkeit von Bildverarbeitungstechniken und Rechenleistung hinsichtlich billigeren, kleineren und hochauflösenderen Kameras oder Grafikprozessoren (GPUs) haben ihre Entwicklung in Richtung Echtzeit-Implementierung vorangetrieben. Dabei gilt die Objekterkennung auf Basis von *Neuronalen Netzwerken* als besonders vielversprechend. Im Gegensatz zu herkömmlichen Bildverarbeitungstechniken können sie komplexe Szenarien, starke Beleuchtungsunterschiede oder kleine, durcheinander auftretende Objekte besser behandeln. [PPMM21a]

Insbesondere *Deep Convulutional Neural Networks* können nachweislich mit höherer Genauigkeit klassifizieren, als voll-verbundene Neuronale Netze [TZP17] oder jene mit weniger Schichten [SHA17]. Sie können Eigenschaften von Bildern niedriger bis hoher Ebene extrahieren sowie erlernen und sind sehr robust [XTY+20, KSB+10, OBLS14]. Kapitel 3.2 bietet die theoretische Grundlage dieser Informationen.

In der Forschung existiert eine Vielzahl von Objekterkennungsdetektoren in Form von Deep Convolutional Neural Networks variierender Architekturen. Üblicherweise werden sie anhand ihrer Stufen unterschieden. Zweistufige Detektoren benutzen in einem ersten Schritt ein *Region Proposal Network* (RPN) und führen erst dann die Klassifikation und

Bounding Box-Regression auf denen von ihm generierten Regionen durch [PPMM21b]. Bekannte zweistufige Detektoren sind R-CNN [GDDM14], Fast R-CNN [Gir15], Faster R-CNN [RHGS15], Mask R-CNN [HGDG17] und R-FCN [DLHS16]. Eine neue Entwicklung sind G-RCNNs [PPMM21b], die sonst präsente, unerwünschte Regionen durch grobe Schätzungen räumlicher und temporaler Informationen exkludieren.

Im Gegensatz zu zweistufigen Detektoren verzichten einstufige Detektoren auf den ersten Schritt des RPNs. Dadurch sind sie deutlich schneller [PPMM21a]. Bekannte einstufige Detektoren sind SSD [LAE+16], EfficientDet [TPL20], CenterNet [DBX+19] und etlichen Versionen von YOLO wie Scaled-YOLOv4 [WBL21] oder YOLOR [WYL21].

Eine ausführlichere theoretische Erklärung sowie eine größere Menge einstufiger und zweistufiger Detektoren ist in Kapitel 3.6 zu finden.

Sowohl die Genauigkeit von Detektoren der Objekterkennung als auch ihre Geschwindigkeit hat in den letzten Jahren kontinuierlich zugenommen [XTY+20]. Jeder Detektor besitzt diesbezüglich gänzlich individuelle Kosten und Nutzen, die dem spezifischen Anwendungsfall entsprechend adäquat evaluiert werden müssen. Häufig zum Einsatz kommen dafür sogenannte Benchmark-Datensätze wie MS COCO [LMB+14], PASCAL VOC [EVGW+10] oder ImageNet [DDS+09], die sich darüber hinaus auch zum Vortrainieren von Modellen zur Durchführung von *Transfer Learning* eignen.

Abseits ihrer Stufen unterscheiden sich Objekterkennungsdetektoren auch anhand weiterer architektonischer Merkmale wie zum Beispiel ihrem Backbone-Netzwerk. Variationen von ResNet [HZRS16], VGG16 [SZ14] und AlexNet [KSH17] sind oft verwendete Backbone-Netzwerke in der Forschung. Demgegenüber stehen leichtgewichtigere Netzwerke wie MobileNet [HZC+17], die sich für mobile Applikationen eignen.

Trotz positiver Forschungsergebnisse existieren Bereiche in der Objekterkennung, die weiteren Fortschritts bedürfen. So stehen viele Detektoren vor einer Herausforderung, wenn überlappende, kleine Objekte zu erkennen sind. Methoden wie jene von Drid et al. [DAK20] können dem entgegenwirken, indem die Vorteile mehrerer Detektoren in einer algorithmischen Lösung kombiniert werden, namentlich Faster R-CNN [RHGS15] und YOLOv1 [RDGF16]. Ähnliche Kombinationen sind den Autor*innen folgend auch mit anderen Detektoren zu erreichen. Ebenfalls kann durch sogenanntes *Tiling*, welches beispielsweise Ünel et al. [ÖÜÖÇ19] untersuchen und bei welchem die Eingabebilder in kleinere Teilmengen zerlegt werden, oder *Grid Search* [LL19] zur Optimierung von Hyperparametern hierauf positiven Einfluss genommen werden.

Auch die Archäologie hat sich bereits zu einem Einsatzgebiet der Objekterkennung auf Basis von Neuronalen Netzen entwickelt. So haben Itkin et al. [IWD19] zum Beispiel zwei Tools zur Identifikation von Tonscherben und damit zur Unterstützung von Archäolog*innen an einer Fundstätte oder im Büro entwickelt. Das Neuronale Netzwerk

OutlineNet auf Basis von PointNet [QSMG17] identifiziert die Scherben anhand ihres Bruchumrisses, während sich das zweite Neuronale Netzwerk basierend auf ResNet-50 [HZRS16] die Verzierung der Scherben zu Nutze macht. Aufgrund des in dem Gebiet der Archäologie herrschenden Mangels an gelabelten Daten, haben die Autor*innen außerdem eine neue Technik zur Generierung synthetisch produzierter Tonscherben entwickelt und Data Augmentation angewandt.

Die mobile Applikation ArchAIDE von Anichini et al. [ABG+20] dient ebenso der Identifikation von Tonscherben anhand ihrer Form und Verzierung. Dazu werden die Tonscherben mit der App fotografiert und mit Hilfe Neuronaler Netze auf Basis von ResNet-101 [HZRS16] Übereinstimmungen mit vergleichbaren Keramikarten vorgeschlagen. Ist die korrekte Art gefunden, werden alle relevanten Informationen mit der Tonscherbe verbunden und in einer Datenbank gespeichert, die online geteilt werden kann.

Kazimi et al. [KTM+18] hingegen automatisieren den Prozess der Analyse von *Airborne Laser Scanning* (ALS) Daten zur detaillierten Messung eines bestimmten Gebiets anhand eines hierarchischen Convolutional Neural Network-Models. Auch Resler et al. [RYNG21] nutzen das Convolutional Neural Network EfficientNetB3 [TL19], um die Fundstätte und den Zeitraum archäologischer Artefakte zu erkennen.

Darüber hinaus existieren etliche weitere Anwendungsbeispiele von Neuronalen Netzen zur Objekterkennung der Archäologie. Zum Zeitpunkt der vorliegenden Masterthesis wurde sich jedoch noch nicht mit auf historischen Fotografien abgebildeten Objektsammlungen beschäftigt, deren etliche Instanzen es mittels Neuronaler Netze zu erkennen gilt.

2.2 Größenapproximation von Objekten auf Bildern

Die Größe eines Objekts gehört zu seinen wichtigsten Eigenschaften. Bereits etliche Studien über die menschliche visuelle Wahrnehmung verweisen auf deren Relevanz bei der Identifizierung und Kategorisierung eines Objekts [KO11, LKCA16]. Ähnliches kann für Künstliche Intelligenzen behauptet werden [CMDB20]. Die Betrachtung der Größe ermöglicht tiefergehende Analysen eines jenen Objekts [SIG+18] und kann damit in maschineller Form besonders im Rahmen der Objekterkennung von Interesse sein.

Bei der maschinellen Größenerkennung werden jene Objektgrößen gemessen oder approximiert, ohne dabei in direkten Kontakt mit dem Objekt zu treten. Dies kann entweder in Echtzeit über bestimmte Geräte oder Sensoren, wie zum Beispiel Lasermessgeräte, oder aber nachdem ein Bild des entsprechenden Objekts aufgenommen wurde mittels Software geschehen [TSSE20]. Insbesondere Letzteres ist im Rahmen der vorliegenden Masterthesis zu untersuchen, weswegen an dieser Stelle der Fokus auf die nachträgliche, maschinelle Größenapproximation von Objekten auf Bildern gelegt wird.

Von Nöten ist dazu ein Referenzobjekt innerhalb des Bildes mit bekannter tatsächli-

cher Größe oder Wissen um den geometrischen Aufbau der Fotoaufnahme [KDB+17, TSSE20, SIG+18]. Beide Methoden werden genauer in dem folgenden Kapitel 3.8 theoretisch aufgeführt. Da im Hinblick auf die der Masterthesis zugrundeliegenden Fotografien der geometrische Aufbau unbekannt ist, aber auf zahlreichen Bildern ein Lineal enthalten ist, wird die Untersuchung des Forschungsstands weiter in Richtung Größenapproximation auf Bildern durch Referenzobjekte spezifiziert.

Tatsächlich umgesetzt wird die maschinelle Größenapproximation in der Forschung auf verschiedene Weisen. Häufig zum Einsatz kommen Methoden, die von der Fourier-Transformation Gebrauch machen; zum Beispiel indem die Skala eines Lineals anhand des regelmäßigen Musters der Amplitude im Frequenzbereich erkannt wird. Anhand der gefundenen Skala kann vereinfacht gesprochen schließlich die Größe des Objekts beispielsweise durch seine Pixel ermittelt werden. So bei der Größenapproximation antiker Münzen auf Bildern [HZK07] oder ausgefeilter in der Forensik [BR14] und in Aquakulturen zur automatischen Größenmessung von Fischen [KDB+17].

Auch die Hough-Transformation zum Erkennen von Geraden, Kreisen und anderen geometrischen Figuren kann zur Größenapproximation, beziehungsweise einer ersten Erkennung eines Lineals, utilisiert werden, wie zum Beispiel bei der Erkennung und Größenmessung von Hämangiomen bei Kindern [OCS21]. Typischerweise sind diese Methoden eher schlicht und funktionieren primär nur unter bestimmten Bedingungen wie einem einfarbigen Hintergrund [BR14].

Entsprechend werden mittlerweile auch Neuronale Netze zur maschinellen Größenapproximation auf Bildern eingesetzt. Telahun et al. [TSSE20] verwenden zum Beispiel den Objekterkennungsdetektor Mask R-CNN [HGDG17], um mittels Semantischer Segmentierung abgebildete Maßstäbe zu extrahieren. Ihre Skala wird dann anhand der Raumfrequenz festgestellt und die entsprechende Größe über Pixel ermittelt. Ein ähnliches Vorgehen findet bei der automatischen Größenmessung von Fischen statt. Konovalov et al. [KDWJ18] erweitern ihr Vorgehen aus der zuvor aufgeführten Publikation [KDB+17] um ein Convolutional Neural Network, welches die Fotografien in einem ersten Schritt in Maßstab, Fisch und Hintergrund segmentiert, bevor die Skala des Maßstabs anhand Digitaler Fourier-Transformation ermittelt und die Größe des Fischs bestimmt werden kann. Hingegen verwenden Monkman et al. [MHKV19] R-CNNs und Passermarken anstelle eines Maßstabs, um die Größe von lokalisierten Fischen zu bestimmen.

Die der Masterthesis zugrunde liegende Datenbasis historischer Sammelaufnahmen erschwert die Verwendung von Methoden wie der Fourier-Transformation oder Neuronalen Netzen zur Erkennung eines Maßstabs aufgrund von geringen Objektgrößen und unruhigen Hintergründen. Zum aktuellen Zeitpunkt können jedoch keine wissenschaftlichen Publikationen mit vergleichbaren Datenbasen ausfindig gemacht werden.

3
Theoretische Grundlagen

Die für die Masterthesis relevanten theoretischen Grundlagen werden in diesem Kapitel aufgeführt und vermittelt. Diese umfassen zunächst die vorliegende Datenbasis, allerdings auch grundlegende Methoden der Künstlichen Intelligenz in Form von Machine Learning, Deep Learning und Computer Vision sowie ihre spezifischen Techniken der Bildannotation, Data Augmentation und Cross Validation. Ferner werden einstufige und zweistufige Objekterkennungsdetektoren mit besonderem Fokus auf YOLOR, EfficientDet und Mask R-CNN aufgeführt. Beendet wird das Kapitel mit einer genaueren Auseinandersetzung mit möglichen Evaluationsmetriken der Objekterkennung und einer rein theoretischen Beschreibung der Größenapproximation auf Bildern.

3.1 Datenbasis

Schon zur Mitte des 19. Jh. wurde nachweislich mit der archäologischen Ausbeutung der Insel Zypern in Form von Grabungen zur Findung von verwertbaren Antiken begonnen, die ihren Platz in großen Museen und Privatsammlungen finden sollten [TB01]. Auch der Archäologe Max Ohnefalsch-Richter war für Privatpersonen und Institutionen, wie die Berliner Museen oder das British Museum, auf Zypern aktiv, führte aber auch von 1880 bis 1895 auf eigene Kosten Grabungen durch. Viele seiner Funde wurden auf der Berliner Gewerbeausstellung von 1896 vorgestellt und in Form eines "Verkaufskatalogs" abgelichtet. Einige großformatige Fotografien der damals angebotenen Waren sind noch bis heute erhalten und befinden sich im Archiv des Berliner Museums für Vor- und Frühgeschichte. [RT21]

Dabei handelt es sich um rund 100 historische Fotografien mit jeweils bis zu 100 Objekten bronzezeitlicher, hellenistisch-römischer und graeco-phönikischer Antiken wie Statuen, Schmuck und Gefäßen aus Keramik. Sie sind aufgrund ihres hohen Alters in Sepia-Tönen aufgenommen und teilweise leicht verzogen sowie von niedrigerer Bildqualität.

Womöglich zur Realisierung finanzieller Einsparungen[1] sind die meisten der insgesamt bis zu 5 000 abgebildeten Antiken in Form von großen Sammlungen organisiert, wofür sie möglichst gut sichtbar in Regalen angeordnet und frontal pro Fotografie mit ungefähr gleichbleibendem Abstand zur Kamera fotografiert worden sind. Damit sind die abgebildeten Objekte sehr klein, unterschiedlich ausgerichtet und verdecken sich aufgrund ihrer schieren Menge zwangsläufig zu kleinen oder größeren Teilen gegenseitig. Für das menschliche Augen bleiben sie jedoch gut erkennbar. Nur einige Antiken sind im Detail einzeln oder in kleineren Gruppen, teilweise sogar aus verschiedenen Perspektiven, aufgenommen worden. Diese Detailaufnahmen sind jedoch nicht Gegenstand der Masterthesis.

Überdies enthalten viele, aber nicht alle Fotografien einen Maßstab in Form eines exakt 50 cm langen Lineals, das ungefähr dieselbe Entfernung zur Kamera wie die Antiken vorweist. Auch dieses ist innerhalb der großen Sammelaufnahmen klein, teilweise verdeckt oder nicht gänzlich abgebildet. Häufig ist seine Skala aufgrund seiner geringen Größe und der minderen Qualität der Fotografien nicht mehr zu entziffern.

Abbildung 3.1: Zwei vorliegende Scans der Verkaufsbilder `TCD-1` und `B-15`[2].

Im Rahmen des LOEWE-Projekts *Künstliche Intelligenz zur Erschließung kolonialer Verwertungspraktiken archäologischer Objektsammlungen* des Big Data Labs und dem Institut für Archäologische Wissenschaften der Goethe-Universität Frankfurt am Main konnten jene Verkaufsbilder aus dem Archiv des Berliner Museums für Vor- und Frühgeschichte eingescannt werden. Diese hochauflösenden Scans im JPG-Format liegen nun der Masterthesis zugrunde und sind beispielhaft in Abbildung 3.1 dargestellt. Die be-

[1] Zum Beispiel kostete eine Portraitaufnahme samt sechs Abzügen im Format 6,5 × 10,5 cm im Jahre 1870 im Atelier der Gebrüder Martin in Augsburg 1 Gulden 45 Kreuzer. Im Vergleich kosteten 500 g Schweinefleisch oder Hirsch lediglich 17 Kreuzer [Hä04]. Trotz dank der Entwicklung der Fotografie sinkender Kosten eines einzelnen Fotos erscheint die großformatige Ablichtung von bis zu 5 000 Antiken im Detail oder in kleineren Gruppen in den 1890er Jahren kaum tragbar.

[2] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 9061 (`TCD-1`) und 8353 (`B-15`).

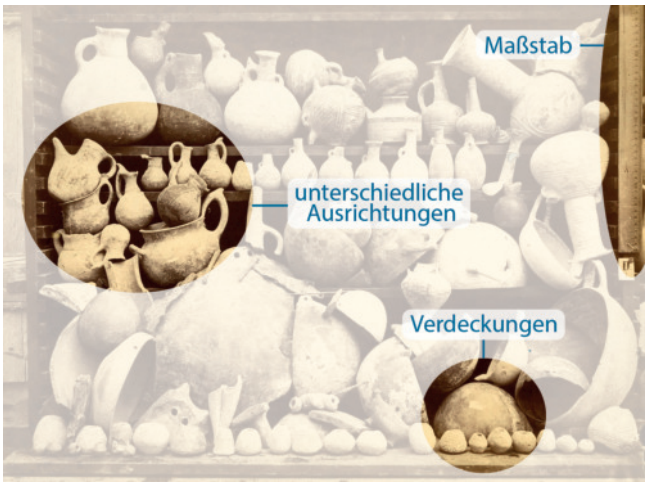

Abbildung 3.2: Nähere Beleuchtung des Scans des Verkaufsbildes `TCD-1`[2].

reits erwähnten Eigenschaften der Sammelaufnahmen werden hingegen anhand der Abbildung 3.2 näher beleuchtet. Neben den Scans werden auch weniger gut auflösende Kameraaufnahmen der Verkaufsbilder im JPG-Format zur Verfügung gestellt.

Ein grober Überblick der vorliegenden Daten wird schließlich anhand der Strahlendiagramme der Abbildung 3.3 geboten. Die Scans und Kameraaufnahmen werden dazu zunächst drei Kategorien zugeordnet: Keramikgefäße, Statuen und Schmuck sowie Waffen. Zuzüglich werden diese Kategorien noch binär anhand der Präsenz eines Lineals kategorisiert.

Relevant für die vorliegende Masterthesis sind, wie in Kapitel 1.1 definiert, jene Verkaufsbilder, welche Keramikgefäßsammlungen bergen. Zu einem späteren Zeitpunkt können jedoch auch andere Fotografien mit Lineal nützlich werden.

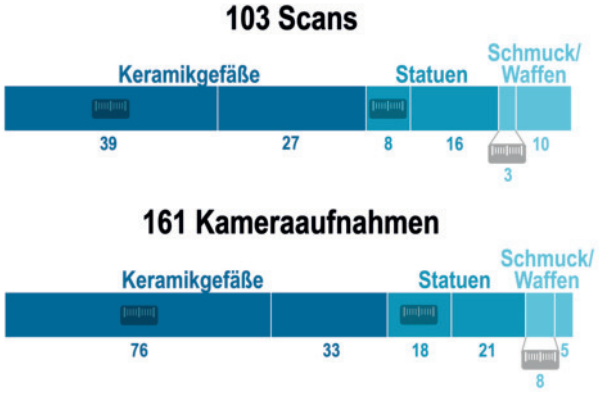

Abbildung 3.3: Die Datenbasis kategorisiert in Keramikgefäße, Statuen und Waffen/Schmuck mit und ohne Lineal.

3.2 Methoden der Künstlichen Intelligenz

Die Künstliche Intelligenz ist ein Teilgebiet der Informatik mit vielen praktischen Applikationen und aktiven Forschungsgegenständen [GBC16]. Das Finden einer einzigen, eindeutigen Definition gestaltet sich aufgrund der hohen Anzahl an möglichen Definitionen schwierig. Im Großen und Ganzen kann jedoch gesagt werden, dass das Feld der Künstlichen Intelligenz zum Ziel hat, intelligente Entitäten zu konstruieren und damit menschliches Handeln und Denken nachzubilden. Jene sollen in der Lage sein, die bestmögliche Maßnahme in einer Situation zu ergreifen [RN10]. So können durch intelligente Software beispielsweise Routineaufgaben automatisiert, Sprache und Bilder verstanden, medizinische Diagnosen gestellt oder wissenschaftliche Forschung unterstützt werden [GBC16]. Wie gut dies bewältigt werden kann, hängt stark von der Umgebung und dem zu lösenden Problem ab. Probleme, denen formale, mathematische Regeln zu Grunde liegen, mögen für Menschen schwer, jedoch für Künstliche Intelligenzen leicht zu lösen sein. Im Gegensatz dazu sind Probleme oder Aufgaben, die für Menschen leicht und gar automatisiert zu lösen sind, wie zum Beispiel das Erkennen von Gesichtern, häufig schwer für Künstliche Intelligenzen [GBC16, RN10].

Das Teilgebiet der Künstlichen Intelligenz lässt sich in viele weitere Teilgebiete gliedern. Die für die Masterthesis wichtigsten Teilgebiete sind Machine Learning, Neuronale Netze, Deep Learning und Computer Vision. Im Folgenden werden diese knapp aufgeführt.

Machine Learning

Anstelle manueller Definitionen von Regeln und Mustern kann Machine Learning als eine Reihe von Methoden verstanden werden, welche automatisch Muster in Daten erkennen. Diese können anschließend zur Vorhersage zukünftiger Daten oder zur Entscheidungsfindung verwendet werden [Mur12, SJZ21]. Ein Machine Learning-Algorithmus ist demnach in der Lage von Daten zu lernen [GBC16]. Dazu sind sowohl Trainingsdaten zum Trainieren als auch Validierungsdaten zur Evaluierung der Leistung des Algorithmus relevant. Üblicherweise wird für die verfügbaren Daten ein Verhältnis von 80:20 zur Distribution in Trainings- und Validierungsdaten verwendet [Mur12].

Grundsätzlich wird zwischen *Supervised Learning* und *Unsupervised Learning* unterschieden. Beim Supervised Learning ist das Ziel die Abbildung eines Inputs x auf einen Output y durch eine Menge an Labeln mit Input-Output-Paaren [Mur12, SJZ21]. Dies kann entweder mittels *Klassifikation* oder *Regression* erfolgen; abhängig davon, ob y kategorisch oder reellwertig ist [Mur12]. Demgegenüber steht das Unsupervised Learning. Hierbei gibt es keine klare Vorstellung des Outputs. Die Ausgangsdaten sind

demnach anders als beim Supervised Learning ungelabelt. Stattdessen wird versucht, interessante Strukturen in Daten zu identifizieren. Dabei findet in der Regel *Clustering*, *Assoziation* oder *Dimensionsreduktion* Anwendung [Mur12]. Die durchzuführende Objekterkennung und -lokalisierung der vorliegenden Thesis bewegt sich im Bereich des Supervised Learnings und setzt damit den weiteren Fokus.

Das Ziel des Machine Learnings ist stets die *Generalisierung*. Modelle sollen nicht nur gut auf ihnen bekannten Trainingsdaten arbeiten können, sondern auch auf unbekannten Daten. Eine Überprüfung erfolgt mittels einer Menge aus nicht zum Training verwendeten Testdaten, die gleich den Validierungsdaten sein können. Dadurch ergeben sich zwei Herausforderungen. Zum einen kann beim *Overfitting* die Differenz zwischen Trainings- und Test-Error zu groß werden, also genau eine oben genannte Generalisierung nicht gegeben sein. Dies nennt man auch den Generalisierungs-Error. Zum anderen kann es aber auch zu *Underfitting* kommen, wobei bereits der Trainings-Error keinen ausreichend niedrigen Wert annimmt. Beidem entgegen wirken in vielen Fällen bereits größere Datenmengen. Häufig zum Einsatz kommen außerdem Methoden wie das Transfer Learning, bei dem Gelerntes von einem Anwendungsfall für einen anderen relevant sein und anhand sogenannter *vortrainierter Weights* genutzt werden kann. So teilen sich insbesondere viele visuelle Inputkategorien Konzepte niedriger Ebenen, wie Kanten und Formen oder Änderungen der Belichtung. Spracherkennungssoftwares sollen hingegen stets valide Sätze generieren können. Es können also sowohl die Eingabe- als auch die Ausgabeschicht eines Modells von Transfer Learning profitieren. [GBC16]

Tiefergehende Einblicke in die Thematik des Machine Learnings können insbesondere im Buch *Machine Learning: A Probabilistic Perspective* [Mur12] von Murphy geboten werden.

Neuronale Netze

Inspiriert durch die Biologie des Gehirns bestehen künstliche Neuronale Netze aus mehreren gerichteten und gewichteten verbundenen Recheneinheiten, den *Neuronen* [GBC16, Sch15, NDB+19, BJS+20]. Abhängig von ihrem Anwendungsfall können jene Verbindungen äußerst individuell ausfallen und sich beispielsweise anhand ihrer Gewichte unterscheiden. Jedes Neuron kann ähnlich seinem biologischen Gegenstück aktiviert werden, wenn seine Inputsignale einen gewissen Schwellenwert erreichen [BJS+20, Sch15]. Der Einfluss eines jeden Inputsignals ist dabei durch Kantengewichte vorbestimmt [BJS+20]. Im technischen Kontext werden Inputneuronen durch umgebungswahrnehmende Sensoren aktiviert und können wiederrum folgende Neuronen aktivieren, indem durch eine Aktivierungsfunktion, wie zum Beispiel eine Sigmoidfunktion, ein Output berechnet wird [BJS+20, Sch15]. Um Lernen zu ermöglichen, müssen Kan-

tengewichte gefunden werden, die das künstliche Neuronale Netzwerk das gewünschte Verhalten zeigen lassen [Sch15].

Eine spezielle Form der Neuronalen Netze sind die *Convolutional Neural Networks* (CNNs). Im Gegensatz zu herkömmlichen Neuronalen Netzen, in denen jedes Neuron jeder Ebene mit jedem Neuron der folgenden Ebene verbunden ist, ist die Architektur von CNNs nicht vollverbunden [BJS+20]. Ein typisches CNN besteht aus drei grundlegenden, sich wiederholenden Schichten. Die *Convolution Layers* wenden Filter auf die Inputdaten oder Feature Maps an, um relevante semantische Eigenschaften oder auch *Features* zu extrahieren. Durch die *Pooling Layers* werden die Features und damit die Dimensionalität des Netzwerks reduziert. Schließlich aggregieren die *Fully-connected Layers* am Ende des Netzwerks Informationen der Feature Maps und generieren damit die finale Klassifikation [Sme19].

Prinzipiell verarbeiten CNNs Daten mit bekannter gitterähnlicher Topologie und eignen sich insbesondere für Bilddaten [GBC16]; damit sind sie auch für die vorliegende Thesis von Interesse.

Deep Learning

Ein bekanntes Teilgebiet des Machine Learnings ist das Deep Learning. Zum Ziel steht die Lösung der bereits im Überkapitel erwähnten intuitiven Probleme, wie beispielsweise die Identifizierung von Gesichtern auf Bildern, durch das Utilisieren etlicher Zwischenschichten inmitten der Eingabe- und Ausgabeschicht. Dazu wird die reale Welt von Rechnern hierarchisch konzeptualisiert: Jedes Konzept wird anhand seiner Relation zu simpleren Konzepten definiert und damit gelernt [GBC16].

Hierfür kommen die zuvor beschriebenen künstlichen Neuronale Netze zum Einsatz. Dabei unterscheiden sich Deep Learning-Algorithmen beziehungsweise *Deep Convolutional Neural Networks* von typischen Neuronalen Netzen bezüglich der Tiefe ihrer Schichten [GBC16, Sch15, NDB+19]. Sie bestehen zunächst aus einer sichtbaren Schicht oder auch Eingabeschicht, die sensorische Rohdaten entgegennimmt, wie beispielsweise die Repräsentation eines Bildes in Pixeln. Anschließend werden mehrere abstrakte Eigenschaften, wie beispielsweise Ecken und Konturen, von einer Reihe von sogenannten verborgenen Schichten extrahiert. Diese können ihre Anzahl betreffend stark variieren und geben dem Deep Learning seinen Namen. Das Modell muss dabei selbst bestimmen, welche Eigenschaften oder Konzepte relevant für die Erklärung der Beziehungen in den Rohdaten sind. Letztlich werden die Ergebnisse über die Ausgabeschicht präsentiert [GBC16, NDB+19].

Im Buch *Deep Learning* [GBC16] von Goodfellow et al. kann die Thematik mit tieferem Fokus nachgeschlagen werden.

Computer Vision

Auch die Computer Vision ist ein Teilgebiet der Künstlichen Intelligenz, enthält allerdings auch Ansätze des Machine Learnings und Deep Learnings [HNZ21]. Sie beschäftigt sich mit der automatischen Verarbeitung und Extraktion von Informationen aus visuellen Daten. Jene Informationen werden in einen nächsten Schritt zur Entscheidungsbildung verwendet - ob von einem Computer Vision-System selbst oder beispielsweise von einem Pattern Recognition-System [SQ17].

Grundsätzlich umfasst die Computer Vision eine Vielzahl von verschiedenen Bildverarbeitungsarten: Die Imitation von menschlichen Sehfähigkeiten, wie der Erkennung von Gesichtern, bis hin zur Entwicklung gänzlich neuer Sehfähigkeitskategorien, wie zum Beispiel der Erkennung von Schallwellen anhand ihrer Vibrationen in Objekten. Häufig hat sie jedoch eine einfache Objekterkennung zum Ziel, was von der Erkennung der Präsenz eines Objektes in einem Bild, über dessen genaue Position, bis hin zur Markierung eines jeden Pixels reichen kann. Näheres ist in dem Unterkapitel *Bildannotationsarten* in Kapitel 3.3 nachzulesen.

Die meisten dieser zu behandelten Aufgaben können mühelos von Menschen und den meisten Tieren durchgeführt werden, sind aber herausfordernd für Rechner. Das macht die Computer Vision zu einem der aktivsten Forschungsgebiete des Deep Learnings [GBC16].

Das Buch *Fundamentals of Computer Vision* [SQ17] von Snyder und Qi beschäftigt sich genauer mit der vorliegenden Thematik und kann als Referenzwerk dienen.

Einordnung der Thesis

Schließlich können die vorgestellten Methoden der Künstlichen Intelligenz, das Machine Learning, das Deep Learning und die Computer Vision, in Abbildung 3.4 als Mengendiagramm dargestellt und ihre Beziehungen zueinander klar gemacht werden. Da es sich bei Neuronalen Netzen mehr um ein Werkzeug als um einen konzeptionellen Bereich handelt, wird in der Abbildung auf sie verzichtet[3].

Des Weiteren kann die Masterthesis sinngemäß in der Abbildung eingeordnet werden. Die Erkennung von Keramikgefäßen und Linealen auf Bildern allein sind visuelle Aufgaben der Computer Vision, die jedoch aufgrund ihres komplexen Anwendungsfalles Deep Learning bedürfen. Demnach ist die Thesis in ihrer Schnittmenge anzuordnen.

[3]Korrekterweise sind Neuronale Netze als echte Teilmenge von Machine Learning zu verstehen, die wiederum Deep Learning als echte Teilmenge enthalten und Computer Vision schneiden [HNZ21].

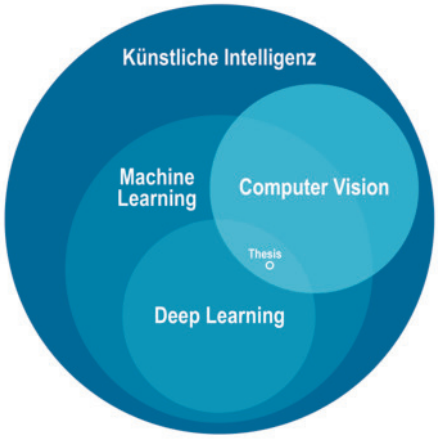

Abbildung 3.4: Wichtige Teilgebiete der Künstlichen Intelligenz als Mengendiagramm anhand [HNZ21] und die Einordnung der Thesis.

3.3 Prinzipien der Bildannotation

Durch das der Masterthesis vorgegangene Forschungsprojekt [Wir21] über den Einfluss von Bildannotationstechniken auf die Genauigkeit von Machine Learning-Modellen wurde sich bereits ausführlich mit der Theorie der Bildannotation auseinandergesetzt. Das folgende Kapitel fasst diese lediglich erneut kurz zusammen. Interessenten steht jedoch natürlich die Möglichkeit offen, auch den Bericht des Forschungsprojekts zu lesen.

Bei der Bildannotation werden detaillierte Beschreibungen oder Schlüsselworte mit Bildern assoziiert. Dies dient nicht nur der Zugriffsvereinfachung oder einer effektiveren Suche [Han08], sondern auch dem unbedingt nötigen Versehen von Datensätzen mit qualitativen Ground Truth Labeln bei überwachten Machine Learning-Methoden [SJZ21]. Dabei wird zwischen verschiedenen zu annotierenden Meta-Informationen unterschieden. Für die Thesis von Relevanz sind jedoch einzig die *inhaltsbeschreibenden Metadaten*, die die Semantik des Bildinhalts in Form der Beziehung zwischen Bildentität und realer Entität, zeitlichen Ereignissen oder tieferen Ereignissen beschreiben [Han08]. Festgehalten werden können die inhaltsbeschreibenden Metadaten unter anderem durch zuvor erwähnte Schlüsselworte, also anhand eines einzelnen Begriffs. Häufig sind diese auf ein vordefiniertes Vokabular beschränkt [SJZ21, Han08]. Typischerweise spricht man in diesem Kontext auch von dem *Label* eines Objekts.

Arten der Bildannotation

Abhängig von dem Anwendungsfall und dem Ziel, welches es mit den annotierten Bildern zu erreichen gilt, ist eine adäquate Bildannotationsart zu wählen. Die prominentesten

sind anhand beispielhafter Ausgaben dem einfacheren und schnelleren Verständnis wegen der im Forschungsprojekt erarbeiteten Tabelle 3.1 zu entnehmen.

Bildannotationsart	**Beispielhafte Ausgabe**
Bildklassifikation	„X befindet sich im Bild.“
Objekterkennung	„X befindet sich an diesen zwei Positionen im Bild.“
Semantische Segmentierung	„All diese Pixel gehören zu Gruppe X, die anderen Pixel gehören zu Gruppe Y.“
Instanzsegmentierung	„Diese Pixel gehören zu X_1, diese Pixel gehören zu X_2, diese Pixel gehören zu X_3,...“
Panoptische Segmentierung	„Diese Pixel gehören zu X_1, diese Pixel gehören zu X_2, die anderen Pixel gehören zu Gruppe Y.“

Tabelle 3.1: Beispielhafte Ausgaben der prominentesten Bildannotationsarten aus dem vorgegangenen Forschungsprojekt [Wir21].

Hervorzuheben ist dabei insbesondere die für die Masterthesis relevante *Objekterkennung*, durch die versucht wird, Entitäten in Bildern zu erkennen und im Gegensatz zur *Bildklassifikation* auch zu lokalisieren. Dabei können in einem einzelnen Bild mehrere gleiche oder ungleiche Entitäten gelabelt werden. Erst bei der *Segmentierung* wird auch der Kontext relevant, in dem sich die Entitäten im Bild befinden. Hierbei wird jedem Pixel eines Bildes oder eines ausgewählten Bildabschnitts ein Label zugeordnet. Die Segmentierung kann außerdem in weitere Unterkategorien zerlegt werden. So werden beispielsweise bei der *Semantischen Segmentierung* Entitäten desselben Typus gruppiert, wohingegen bei der *Instanzsegmentierung* jede Entität individuell gelabelt wird. Beide Segmentierungsarten lassen sich zur *Panoptischen Segmentierung* kombinieren. [CLO20, HB20]

Zusätzlich zu Tabelle 3.1 zeigt Abbildung 3.5 die ersten vier erwähnten Annotationsarten auch an einem Beispielbild.

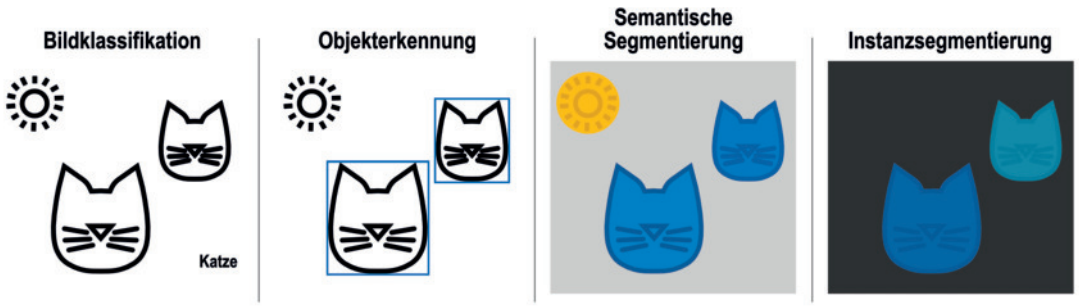

Abbildung 3.5: Die prominentesten Bildannotationsarten.

Techniken der Bildannotation

Im Kontext der durch Aufgabenstellung 1.1 vorliegenden Objekterkennung ist auch die Wahl der Bildannotationstechnik relevant. Diese richtet sich stets nach dem vorliegenden Anwendungsfall, wobei sowohl die Kosten als auch der Nutzen der Annotation stark von ihr beeinflusst werden können [MJTS19]. Die gängigsten Annotationstechniken werden im Folgenden aufgeführt.

Die in der Literatur vermutlich am häufigsten verwendete Annotationstechnik ist die *Bounding Box*. Dabei wird eine in der Regel rechteckige Box mittels zwei Ankerpunkten um die zu annotierende Entität gezogen. Jedoch existieren auch beispielsweise runde, elliptische oder dreidimensionale Ausführungen (sogenannte *3D Cuboids*) der Bounding Box [SJZ21, CLO20]. Die besten Ergebnisse erzielt sie bei symmetrischen Entitäten oder bei Entitäten, deren Form von geringem Interesse ist [CLO20].

Bei der *Key-Point-* oder auch *Landmark*-Annotation werden einzelne Punkte auf relevante Merkmale gesetzt. So können beispielsweise Charakteristika wie Augen oder die Position und Ausrichtung von Händen getrackt werden [SJZ21, CLO20]. Im Gegensatz zu der zuvor erwähnten Bounding Box ist bei dieser Annotationstechnik nur ein einzelner Ankerpunkt zu setzen.

Demgegenüber steht die *Polygon*-Annotation. Indem die höchsten Punkte der zu annotierenden Entität markiert und ihre Kanten beschriftet werden, wird sie genau nachgefahren. In der Regel handelt es sich dabei um Entitäten mit unregelmäßiger Form [SJZ21, CLO20]. Wie viele Ankerpunkte dabei nötig sind, hängt von der Entität selbst und der Ausführlichkeit ab, mit der man die Annotation vernehmen möchte. Entsprechend kann diese Annotationstechnik weitaus mehr Zeit in Anspruch nehmen als vorherige [MJTS19], dafür aber auch unter gewissen Umständen die Genauigkeit von Machine Learning-Modellen steigern [Wir21].

Während mit der Polygon-Annotation geschlossene Formen erzeugt werden, bleiben die kontinuierlichen Linien der *Polyline*-Annotation offen. Entsprechend findet sie bei Entitäten mit offener Form Anwendung [SJZ21, CLO20].

Eine bildliche Zusammenfassung der Annotationstechniken ist in Abbildung 3.6 aufzufinden. Darüber hinaus gibt es eine Vielzahl weiterer Annotationstechniken, von denen bestimmte Anwendungsfälle profitieren können.

Durchführung der Bildannotation

Zur praktischen Annotation von Bildern existiert eine Vielzahl kostenloser und kostenpflichtiger Annotationstools, die unterschiedliche Arten und Techniken auf verschiedene Weisen implementieren. Bereits in dem der Masterthesis vorgegangenen Forschungsprojekt [Wir21] wurden etwaige Annotationstools genauer beleuchtet, aber auch Publika-

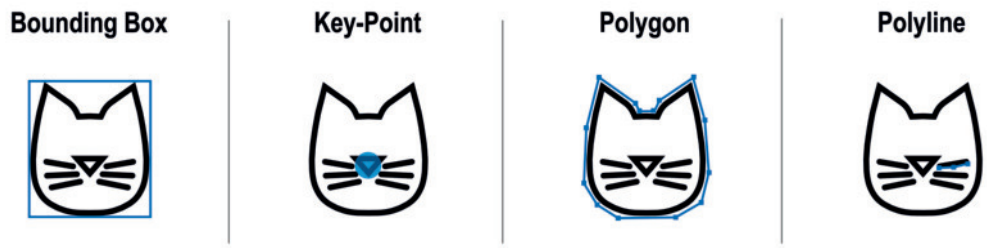

Abbildung 3.6: Die prominentesten Bildannotationstechniken.

tionen wie jene von Sager et al. [SJZ21] untersuchen sowohl die Differenzen als auch die Gemeinsamkeiten einiger ausgewählter Tools wie CVAT[4], VoTT[5] und LabelMe[6].

Demnach können sich Annotationstools neben der Bildannotationsart und -technik auch anhand ihres Automatisierungsgrades unterscheiden: *Manuell*, durch unterstützende Vorschläge *semi-automatisiert* oder *vollautomatisiert*. Eine manuelle Annotation gilt dabei zwar als wenig fehleranfällig, ist aber besonders kost- und zeitspielig [SJZ21]. Insbesondere beim kollaborativen Arbeiten [SJZ21] kann ihre inhärente Subjektivität jedoch auch zu Mehrdeutigkeit von Bildinhalten und damit abweichenden Annotationen führen [CZF+18].

Schließlich werden die Daten einer abgeschlossenen Annotation in sogenannten Exportformaten in der Regel als Pixelkoordinaten gesichert. Die prominentesten sind das COCO-Format als JSON-Datei, das Pascal VOC-Format als XML-Datei [SJZ21] und das YOLO-Format als Textdatei [Pok20].

3.4 Größere Datengrundlage durch Data Augmentation

Machine Learning-Modelle generalisieren besser, wenn sie mit größeren Datenmengen trainiert werden [GBC16]. Insbesondere die Neuronalen Netze des Deep Learnings sind auf diese angewiesen, um Overfitting zu vermeiden [SK19]. Jedoch ist die Anzahl an Daten in der Praxis begrenzt und von ihrem Anwendungsfall abhängig [GBC16].

Dem wirkt die Data Augmentation entgegen, indem die Größe und Qualität der Trainingsdaten mit scheinbar neuen, künstlichen Daten erweitert werden [GBC16, SK19]. Tatsächlich werden diese Daten durch verschiedene Augmentationstechniken aus den bereits existierenden Daten gewonnen und ermöglichen Modellen damit, umfangreichere Informationen zu extrahieren [SK19]. Dieses Vorgehen ist nicht für jede Art von Machine Learning eine gute Wahl, eignet sich aber laut Goodfellow et al. [GBC16] insbesondere

[4]CVAT, `https://github.com/openvinotoolkit/cvat`, besucht am 20.06.2022
[5]VoTT, `https://github.com/microsoft/VoTT`, besucht am 20.06.2022
[6]LabelMe, `https://github.com/CSAILVision/LabelMeAnnotationTool`, besucht am 20.06.2022

für die Klassifikation, beziehungsweise für die Objekterkennung. Da Classifier komplexe, hochdimensionale Daten entgegennehmen, um sie anhand einer einzelnen Kategorie zusammenzufassen und Bilder eine Vielzahl variierender Faktoren besitzen, können originale einfach in neue, künstliche Datenelemente transformiert werden. [GBC16].

Folgend wird der Fokus aufgrund des begrenzten Anwendungsfalls der Thesis nach Kapitel 3.1 auf die Augmentation von Bilddaten gelegt. Dabei wird zwischen zwei Arten der Augmentation unterschieden: Dem *Data Warping* und dem *Oversampling* [SK19].

Bilder werden bei dem Data Warping so transformiert, dass ihre Klassenlabel möglichst erhalten bleiben. Darunter fallen beispielsweise geometrische Transformationen, wie das vertikale Spiegeln, aber auch das Löschen zufälliger Bildausschnitte oder die Transformation des Farbraums des Bildes. Hingegen werden beim Oversampling neue, synthetische Instanzen kreiert. Dies kann beispielsweise durch das Zusammenfügen mehrerer Bilder geschehen [SK19]. Eine beispielhafte Augmentation ist in Abbildung 5.7 zu finden.

Darüber hinaus existieren jedoch noch weitere Augmentationstechniken, die nicht im Vorfeld auf Inputbildern angewandt werden. Stattdessen basieren sie auf Deep Learning. Darunter fallen Techniken wie die *Feature Space Augmentation* oder der *Neural Style Transfer* [SK19]. Eine genauere Ausführung jener Techniken würde an dieser Stelle den Rahmen der Masterthesis sprengen, kann jedoch in der Übersicht von Shorten und Khoshgoftaar [SK19] vorgefunden werden.

Durch eine Kombination der erwähnten Augmentationstechniken können riesige Datensätze generiert werden, was jedoch nicht immer von Vorteil ist. Gerade in Bereichen mit einer sehr beschränkten Datenmenge kann dies ebenfalls zu Overfitting führen. Der prinzipiell auftretende Bias von Machine Learning-Modellen kann somit zwar reduziert werden, doch die große Varianz sorgt für eine geringe Genauigkeit bezüglich der eigentlich relevanten, originalen Daten [SK19]. Auch sollte beachtet werden, dass die durchgeführten Augmentationen nicht die tatsächlichen Klassen verändern dürfen. Ein prominentes Beispiel ist die optische Erkennung der Zahlen 6 und 9, die durch Spiegeln zur jeweils anderen Zahl werden können [GBC16]. Entsprechend ist stets der vorliegende Anwendungsfall zu berücksichtigen.

In der Praxis existieren verschiedene Tools, die nicht nur eine Augmentation der vorliegenden Daten, sondern auch deren Annotationen vornehmen können. Die originalen Daten müssen damit nur ein einzelnes Mal annotiert werden, ehe sie augmentiert werden können. Beispielhafte Tools sind die Python-Libraries imgaug[7] und albumentations[8].

[7] imgaug, `https://github.com/aleju/imgaug`, besucht am 09.08.2022

[8] albumentations, `https://github.com/albumentations-team/albumentations`, besucht am 09.08.2022

3.5 Effektive Datennutzung durch Cross Validation

Wie bereits in den vorgegangenen Kapiteln etabliert, hängt die Fähigkeit eines Machine Learning-Modells stark von der Menge der ihm zum Training zur Verfügung stehenden Daten ab. Liegt nun eine eher kleine Datenmenge vor, so kann bereits ihre typische Teilung in Trainings- und Validierungsdaten zum Verhältnis von 80:20 zur Herausforderung werden [GBC16, Mur12]. Dabei geht eine geringe Menge an Validierungsdaten mit statistischer Unsicherheit des errechneten Werts der Evaluationsmetrik einher. So kann ein Algorithmus A zwar besser als ein Algorithmus B auf den entsprechenden fixen Validierungsdaten funktionieren, Algorithmus B jedoch bei einer größeren oder leicht anders gewählten Validierungsmenge überlegen sein. Dadurch wird die Bewertung beider Algorithmen deutlich erschwert [GBC16]. Gleichermaßen kann es bei einer zu kleinen Trainingsmenge zu Underfitting kommen [Mur12].

Jener Problematik soll die Cross Validation entgegenwirken. Dabei wird die Datenmenge der Größe N in K sich nicht überlappende, möglichst gleich große Teilmengen oder auch *Folds* gesplittet. Anschließend wird für jedes $k \in \{1, ..., K\}$ mit jeder Fold bis auf Fold k trainiert und mit Fold k getestet. Es ergeben sich K Iterationen und damit K Modelle. In Abbildung 3.7 wird das Verfahren der Cross Validation visualisiert. Die Evaluation eines Algorithmus wird schließlich vollzogen, indem der Durchschnitt der Evaluationsmetrik einer jeden Iteration berechnet wird. [Mur12]

Genauer wird dieses Verfahren auch als *K-Fold Cross Validation* bezeichnet. Gilt stattdessen $K = N$ so handelt es sich um die *Leave-One-Out Cross Validation*, da genau ein Datenelement zum Testen verwendet wird [Mur12]. Als *Stratified Cross Validation* wird hingegen eine Cross Validation bezeichnet, bei der die entsprechenden K Folds mit möglichst gleicher Aufteilung der zu vorhersagenden Klassen innerhalb der Trainings- und Testmengen gewählt werden [Mur12, RTL09]. Befindet sich eine Klasse bei einer Iteration beispielsweise nur in Fold k, die zum Testen verwendet wird, so kann sie nicht vorhergesagt werden und beeinflusst das Ergebnis der Evaluationsmetrik dieser Iteration negativ.

Aufgrund der zahlreichen zu generierenden Modelle gilt die Cross Validation außerdem als langsam [Mur12], was neben ihren etlichen Vorteilen einen eindeutigen Nachteil darstellen kann, der vor Anwendung Beachtung finden sollte.

3.6 Detektoren zur Objekterkennung

Die maschinelle Objekterkennung auf Basis von Deep Learning beschreibt das Ermitteln der Präsenz von Objekten vordefinierter Kategorien und, falls vorhanden, die Rückgabe ihrer räumlichen Positionen in Bild- oder Videomaterial mit Hilfe von achsenorientier-

Abbildung 3.7: Schematische Darstellung der K-Fold Cross Validation.

ten Rechtecken, den Bounding Boxen [LOW+20]. Bereits in Kapitel 3.3 sind jene Informationen in Form von Bildannotationsarten und -techniken knapp aufgeführt worden. Nachstehend folgt eine genauere Ausführung mit besonderem Fokus auf Objekterkennungsdetektoren, die in der Praxis und späteren Implementierung des Kapitels 5 zum Einsatz kommen.

Grundsätzlich werden zwei Arten der Erkennung von Objekten auf Bildern unterschieden [ZYH+13]: Die Erkennung von spezifischen Instanzen, wie beispielsweise dem Sänger Jungkook, und die Erkennung von Kategorien, wie beispielsweise Menschen. Im Kontext der Masterthesis ist Letzteres von Relevanz[9].

Das grundsätzliche Ziel der Objekterkennung ist die Erreichung einer hohen Genauigkeit sowie hohen Effizienz. Herausforderungen ergeben sich durch eine hohe Anzahl an zu erkennenden Objektkategorien, wenige Inter-Klassen-Variationen oder viele Intra-Klassen-Variationen. Beispiele sind variierende intrinsische Faktoren wie Farben oder Größen, aber auch bildabhängige Bedingungen wie wechselnde Belichtung. Demgegenüber können die Architekturen der Objekterkennungsdetektoren und die zugrundeliegende Hardware zu Einbußen im Bereich der Effizienz führen. [LOW+20]

Detektoren zur Objekterkennung bestehen aus drei Grundelementen. Das *Backbone*-Netzwerk ist das wichtigste Element der Architektur, da es nachweislich den größten Einfluss auf die Genauigkeit eines Detektors nimmt [RCCFR21]. Seine Aufgabe ist die Extraktion von semantischen Eigenschaften oder auch Features von Bildern, die der Detektor als Input erhalten hat [RCCFR21] und damit die Generierung einer Feature Map eines jeden Bildes [PPMM21a].

[9] Aufgabenstellung 1.1 folgend soll nicht eine bestimmte, antike Schüssel mit Kratzer am rechten Rand, sondern allgemein antike Schüsseln erkannt werden.

Name	Paper	Jahr	#Stufen	GitHub-Link	Update	Aktiv	Framework	Exportformat	Backbone	Zusätzliche Informationen
Cascade R-CNN	[CV18]	2018	2	/zhaoweicai/cascade-rcnn	10.09.2019	Ja	Caffe, MATLAB	COCO, Pascal VOC	vers. AlexNet, ResNet, VGG-Net	Auch mit Detectron nutzbar
CenterNet	[DBX+19]	2019	1	/Duankaiwen/CenterNet	05.06.2020	Ja	PyTorch	COCO	Hourglass-52, Hourglass-104	Auch mit TensorFlow Object Detection API nutzbar
EfficientDet	[TPL20]	2020	1	/google/automl/tree/master/efficientdet	14.04.2022	Ja	TensorFlow	Pascal VOC, COCO	EfficientNets	Auch mit TensorFlow Object Detection API nutzbar
Mask R-CNN	[HGDG17]	2017	2	/facebookresearch/detectron2	01.04.2019	Ja	PyTorch	JSON-Format	FPN, ResNet-101 & andere	Polygon-Input, ermöglicht Instanzsegmentierung, **orig. GitHub**: /facebookresearch/Detectron, **pop. GitHub**: /matterport/Mask_RCNN, auch mit TensorFlow Object Detection API nutzbar
RefineDet	[ZWB+18]	2018	1	/sfzhang15/RefineDet	18.03.2019	Ja	Caffe	Pascal VOC, COCO	VGG-16	
RetinaNet	[LGG+17]	2017	1	/facebookresearch/detectron2	11.04.2022	Ja	PyTorch	JSON-Format	ResNet-101, FPN, ResNet-50	Auch mit Detectron nutzbar
R-FCN	[DLHS16]	2016	2	/msracver/Deformable-ConvNets	15.04.2019	Ja	Caffe, MATLAB	Pascal VOC, COCO	ResNet-101	**Orig. GitHub**: /daijifeng001/r-fcn, auch mit TensorFlow Object Detection API nutzbar
Scaled-YOLOv4	[WBL21]	2021	1	/AlexeyAB/darknet	06.03.2022	Ja	darknet	YOLO	CSPDarknet53	
SSD	[LAE+16]	2016	1	/weiliu89/caffe/tree/ssd	25.10.2021	Ja	Caffe, MATLAB	Pascal VOC	VGG-16	Auch mit TensorFlow Object Detection API nutzbar
YOLOR	[WYL21]	2021	1	/WongKinYiu/yolor	05.03.2022	Ja	PyTorch	YOLO	CSPDarknet53	
YOLOv4	[BWL20]	2020	1	/AlexeyAB/darknet	06.03.2022	Ja	darknet	YOLO	CSPDarknet53	
YOLOv5	×	2020	1	/ultralytics/yolov5	12.04.2022	Ja	PyTorch	YOLO	CSPDarknet	
Fast R-CNN	[Gir15]	2015	2	/rbgirshick/fast-rcnn	23.01.2018	Nein	Caffe, MATLAB			Auch mit Detectron nutzbar
Faster R-CNN	[RHGS15]	2015	2	/ShaoqingRen/faster_rcnn	26.07.2018	Nein	Caffe, MATLAB			Auch mit Detectron & TensorFlow Object Detection API nutzbar
R-CNN	[GDDM14]	2014	2	/rbgirshick/rcnn	03.04.2017	Nein	Caffe, MATLAB			
SPP-Net	[HZRS15]	2015	2	/ShaoqingRen/SPP_net	11.07.2016	Nein	Caffe, MATLAB			
YOLOv1	[RDGF16]	2016	1	/pjreddie/darknet	14.09.2018	Nein	darknet			
YOLOv2	[RF17]	2017	1	/pjreddie/darknet	14.09.2018	Nein	darknet			
YOLOv3	[RF18]	2018	1	/pjreddie/darknet	14.09.2018	Nein	darknet			
G-CNN	[PBBC20]	2020	2	×						
G-RCNN	[PPMM21b]	2021	2	×						

Tabelle 3.2: Liste berühmter einstufiger und zweistufiger Objekterkennungsdetektoren, die nach 2014 entwickelt worden sind, ihre zugehörigen Publikationen und technischen Details. Lediglich Detektoren, deren GitHub-Repositories nach 2018 aktualisiert worden sind, werden als aktiv und damit relevant angesehen. Die letzte Aktualisierung der Tabelle ist am 01.06.2022 erfolgt.

Der *Neck* gilt als Zwischenelement und verfeinert die vom Backbone extrahierten Features, unter anderem durch ihre Fusion über mehreren Ebenen. Somit ist er besonders relevant zur Lokalisierung von Objekten. Schließlich ist der *Head* für die Vorhersage der Klasse und Regression der Bounding Box zuständig. [RCCFR21]

In Tabelle 3.2 wird eine Liste berühmter Objekterkennungsdetektoren präsentiert, die nach 2014 entwickelt worden sind. Sowohl ihre zugehörigen Publikationen als auch einige ihrer technischen Details sind in der Tabelle festgehalten. Da eine Aufführung eines jeden Detektors den Rahmen der Thesis sprengen würde, wurde sich dabei nur auf eine kleine Menge beschränkt.

Prinzipiell werden einstufige und zwei- oder mehrstufige Detektoren zur Objekterkennung anhand ihrer Architektur unterschieden. Diese werden im Folgenden in Kombination mit entsprechenden Detektoren aus der Praxis näher betrachtet.

3.6.1 Zweistufige Detektoren am Beispiel von Mask R-CNN

Wie aus ihrem Namen hervorgeht, besitzen zweistufige Objekterkennungsdetektoren zwei Aufgaben. Zunächst wird das kategorieunabhängige *Object Region Proposal* anhand konventioneller Methoden oder einem *Deep Region Proposal Network* (RPN) umgesetzt. Dadurch ergibt sich eine Menge an Regionen eines Bildes, die potenziell ein zu erkennendes Objekt enthalten können. Eigenschaften dieser Regionen werden extrahiert und im zweiten Schritt von einem kategoriespezifischen *Classifier* verwendet, um die Klassen jener Regionen zu bestimmen. [LOW+20, PPMM21a]

Damit erzielen zweistufige Detektoren zwar in der Regel eine höhere Genauigkeit und sind aufgrund ihrer flexibleren Struktur geeigneter für regionsbezogene Klassifikationen [LOW+20], generieren insbesondere bei kleinen oder überlappenden Objekten aber vermehrt unpräzise Bounding Boxen [DAK20]. Überdies benötigen sie höhere Rechen- und Speicherkapazitäten, wodurch sie typischerweise unter anderem langsamer als einstufige Detektoren sind [LOW+20]. Primär ergibt sich diese Einschränkung durch die korrelierten Stufen von zweistufigen Detektoren, aufgrund derer sie zwischen RPN und Detektionsnetzwerk geteilte Parameter benötigen. Bei Echtzeitanwendungen kann dies zu Engpässen führen [DAK20]. Mit der Einführung ähnlicher Techniken wie derer einstufiger Detektoren können jedoch auch zweistufige Detektoren in Echtzeit ausgeführt werden [LOW+20].

Folgend wird der zweistufige Detektor der Objekterkennung Mask R-CNN genauer aufgeführt.

Mask R-CNN

Der zweistufige Detektor *Mask Region-based Convolutional Neural Network*, kurz Mask R-CNN [HGDG17], mit dem Ziel der Objekt-Instanzsegmentierung aus dem Jahre 2017 erweitert den vorgegangenen Detektor Faster R-CNN [RHGS15]. Nun sollen nicht mehr Objekte lediglich identifiziert und lokalisiert werden, sondern gleichzeitig eine binäre Segmentierungsmaske für jedes Objekt generiert werden. Dies addiert nur einen kleinen Overhead zu dem Faster R-CNN Vorgänger. Ähnlich diesem wendet Mask R-CNN ein zweistufiges Verfahren an. [HGDG17]

Die erste Stufe ist dabei dieselbe wie bei Faster R-CNN. Nachdem das für Mask R-CNN festgelegte ResNet-FPN-Backbone-Netzwerk [HZRS16, LDG$^+$17] eine Feature Map für ein entsprechendes Inputbild generiert hat, kommt ein RPN zum Einsatz [HGDG17]. Dabei handelt es sich um ein Fully Convolutional Network, das eine Menge rechteckiger Objekt-Vorschläge, beziehungsweise *Proposals* und deren *Objectness-Score*[10] ausgibt und so *Regions of Interest* (RoIs) generiert. Um dies umzusetzen, wird ein kleines $n \times n$ großes Netzwerk in Form eines *Sliding Windows* über die Feature Maps geschoben. Für jeden Standort des Sliding Windows werden k Proposals generiert und jedes Sliding Window selbst wird auf einer niederdimensionalen Feature Map abgebildet. Diese wird in die vollverbundenen Schichten *box regression layer (reg)* und *box classification layer (cls)* eingespeist. Die reg-Schicht enthält $4k$ Ausgaben, die die Koordinaten der k Bounding Boxen kodieren, wohingegen die cls-Schicht $2k$ Ausgaben enthält, durch die die Wahrscheinlichkeit der Zugehörigkeit zu den Objektklassen im Gegensatz zum Hintergrund estimiert wird. [RHGS15]

Zunächst werden die k Proposals jedoch zu k Bounding Boxen parametrisiert, welche von Ren et al. [RHGS15] *Anker* genannt werden. Anker sind auf dem betreffenden Sliding Window zentriert und besitzen typischerweise eine Skalierung und ein Seitenverhältnis, wobei sie immer in einem von beiden variieren. Standardgemäß gibt es drei Skalierungen (128, 256 und 512) und drei Seitenverhältnisse (1:1, 1:2 und 2:1), wodurch es $k = 9$ Anker pro Sliding Window Standort gibt. Für eine Convolutional Feature Map der Größe $W \times H$ ergeben sich dadurch $W \times H \times k$ Anker. Schließlich wird jedem Anker im Kontext des Trainings ein positives oder negatives Label zugewiesen. Positive Label ergeben sich für Anker, die entweder den größten *Intersection over Union* (IoU)[11] oder einen IoU größer als $0,7$ mit einer Ground Truth Bounding Box besitzen. Sollte der IoU mit allen Ground Truth Bounding Boxen geringer sein als $0,3$, wird dem Anker ein negatives Label zugewiesen. [RHGS15]

[10] Der Objectness-Score misst die Zugehörigkeit zu einer Menge von Objektklassen entgegen dem Hintergrund [RHGS15].

[11] Vgl. Kapitel 3.7.

In der zweiten Stufe von Mask R-CNN gilt es, eine binäre Maske zusammen mit einer Klassenvorhersage und Bounding Box-Regression für jede durch das RPN generierte RoI auszugeben. Dazu wird jede RoI für die folgende Schicht auf einer kleineren Feature Map mit fester Größe abgebildet. Soll jedoch eine 16×16-RoI auf 7×7 abgebildet werden, kommt es zu Problemen, denn $16/7 = 2,29$, welches nicht korrekt auf der gitterartigen Struktur abgebildet werden kann. Beim noch in Faster R-CNN genutzten *RoIPooling* wird an dieser Stelle gerundet, sodass eine fehlausgerichtete 14×14-RoI auf 7×7 abgebildet wird [HGDG17, RHGS15]. Dieser Verlust von Daten hat zwar keinen Einfluss auf die Klassifikation, die gegenüber kleinen Verschiebungen robust ist, aber eine negative Auswirkung auf pixelgenaue Masken. Aus diesem Grund wendet Mask R-CNN *RoIAlign* an. Anstelle von mathematischem Runden kommt dabei die bilineare Interpolation zum Einsatz. Dabei werden die genauen Werte der Inputeigenschaften an vier regelmäßig abgetasteten Stellen jedes Blocks innerhalb einer RoI berechnet. Die erzielten Ergebnisse werden schließlich unter Verwendung von Maximal- oder Durchschnittswerten aggregiert [HGDG17]. Mit RoIAlign kann He et al. [HGDG17] folgend die Genauigkeit der Maske bereits um 10 bis 50 % verbessert werden.

Schließlich werden die Ergebnisse des RoIAlign an eine Fully Convolutional-Schicht weitergegeben und die Klassen gemeinsam mit den Bounding Boxen unabhängig von den Masken vorhergesagt [HGDG17]. Ersteres geschieht nach dem Vorbild von Faster R-CNN und damit Fast R-CNN mittels Softmax-Funktion für die Klassifikation und Regressions-Modell für die Bounding Boxen [Gir15, RHGS15]. Hingegen erfolgt die Vorhersage der Masken anhand eines weiteren Fully Convolutional Networks, welches für jede RoI eine $m \times m$ große Maske bestimmt. Damit kann das räumliche $m \times m$ Layout beibehalten werden, anstatt wie bei früheren Methoden in Vektordarstellung ohne räumliche Dimensionen übertragen werden zu müssen. [HGDG17]

Die grundlegende Netzwerkarchitektur von Mask R-CNN kann durch Abbildung 3.8 visuell nachempfunden werden.

3.6.2 Einstufige Detektoren am Beispiel von YOLOR und EfficientDet

Im Gegensatz zu zweistufigen Detektoren modellieren einstufige Detektoren[12] die Objekterkennung als simples Regressionsproblem [RCCFR21]. In einem einzigen Schritt wird sowohl die Vorhersage von potenziell interessanten Bereichen in Form von Bounding Boxen als auch die Vorhersage von Klassenlabeln anhand eines einzelnen Feed-Forward-CNNs vorgenommen [RCCFR21, LOW$^+$20]. Dabei kommt es weder zur Generierung eines Region Proposals, noch zu einer Post-Klassifizierung [LOW$^+$20].

Mit dem Wegfall des Region Proposal-Schritts sowie der Verwendung leichtgewichti-

[12] Auch *unified* Objekterkennungsdetektoren genannt.

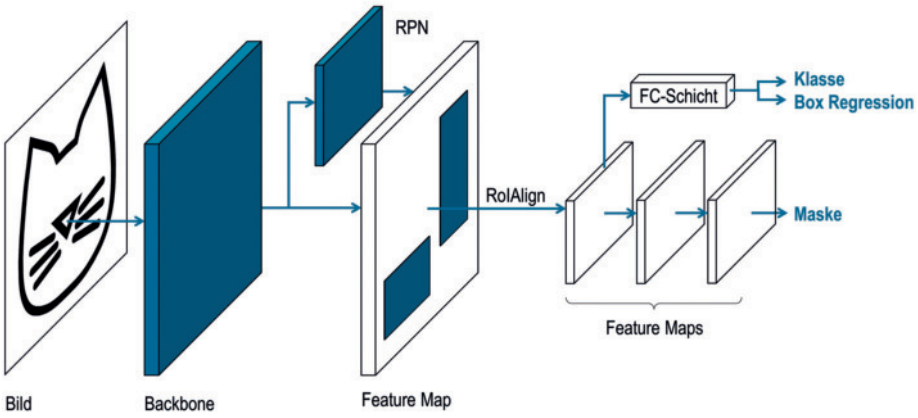

Abbildung 3.8: Die grundlegende Architektur von Mask R-CNN nach [HGDG17] und [JLK19].

gerer Backbone-Netzwerke, sind einstufige Detektoren typischerweise deutlich schneller als zweistufige. Im Vergleich erzielen ihre Detektionen jedoch in der Regel eine geringere Genauigkeit, insbesondere bei der Erkennung von kleinen Objekten [LOW+20]. Zwar sind ihre Bounding Boxen üblicherweise präziser als jene zweistufiger Detektoren, jedoch erkennen sie insgesamt weniger Objekte [DAK20].

Als nächstes werden die Objekterkennungsdetektoren YOLOR und EfficientDet genauer aufgeführt.

YOLOR

You Only Learn One Representation oder YOLOR [WYL21] aus dem Jahre 2021 ist eine neue Version des bekannten YOLO-Detektors und damit ein einstufiger Detektor zur reinen Objekterkennung. Im Gegensatz zu anderen Objekterkennungsdetektoren soll YOLOR in der Lage sein, verschiedene Aufgaben zu bewältigen und effektiver zu gestalten, indem das Lernen des menschlichen Gehirns imitiert wird.

Dazu integriert YOLOR sowohl implizites als auch explizites Wissen mit geringen Kosten. Implizites Wissen wird unterbewusst anhand von Erfahrungen erlangt und unterstützt das menschliche Gehirn beim Bewältigen mehrerer Aufgaben. So ist es Menschen möglich, ein einziges Datenelement aus verschiedenen Perspektiven zu betrachten und damit verschiedenste Informationen zu erlangen. Eine Definition, wie genau implizites Wissen funktioniert und erhalten werden kann, existiert nicht. Demgegenüber steht explizites Wissen, welches einer gemachten Beobachtung direkt entspricht. Es basiert im Sinne von YOLOR also auf den Inputdaten und den Features, die aus ihnen extrahiert werden können. [WYL21]

Angewendet werden kann implizites Wissen zum Beispiel bei dem *Kernel Space Alignment*. So ist in Multi-Task- und Multi-Head-Netzwerken das Kernel Space Misalignment ein häufig auftretendes Problem. Anhand von Addition und Multiplikation der Output-Features und impliziten Repräsentationen kann ein Kernel translatiert, rotiert und skaliert werden, sodass jedes Output-Kernel eines Netzwerks ausgerichtet ist. Dies ist insbesondere relevant bei der Feature-Ausrichtung von großen und kleinen Objekten in FPNs [LDG+17], die insbesondere die Erkennung von Objekten unterschiedlicher Skalierung unterstützen. [WYL21]

Um implizites Wissen tatsächlich in ein Netzwerk zu integrieren, müssen konventionelle Netzwerke erweitert werden. Ein konventionelles Netzwerk kann als folgende Formel dargestellt werden:

$$y = \min_{\epsilon} f_\theta(x) + \epsilon$$

Dabei stellt y das Ziel einer bestimmten Aufgabe, x eine Beobachtung, θ die Menge an Parametern des Netzwerks, f_θ die Operation des Netzwerks und ϵ den Error dar. Nun muss der Error ϵ minimiert werden, um das Ziel y zu erreichen. Dazu muss ϵ modelliert werden. Um dies zu tun, werden implizites und explizites Wissen zum Training eines einstufigen Netzwerks verwendet. Es ergibt sich folgende Gleichung:

$$y = \min_{\epsilon + g_\phi(\epsilon_{ex}(x), \epsilon_{im}(z))} f_\theta(x) + \epsilon + g_\phi(\epsilon_{ex}(x), \epsilon_{im}(z))$$

ϵ_{ex} und ϵ_{im} sind dabei Operationen zur Modellierung des expliziten und impliziten Errors einer Beobachtung x und dem latenten Code z, wohingegen g_ϕ eine aufgabenspezifische Operation zur Kombination oder Selektion von Informationen von explizitem und implizitem Wissen darstellt. Da einige Methoden zur Integration von explizitem Wissen in f_θ existieren, kann die Gleichung folgendermaßen umgeschrieben werden:

$$y = f_\theta(x) \star g_\phi(z)$$

Durch $\star$ werden dabei mögliche Operationen zur Kombination von f_θ und g_ϕ dargestellt. Im Falle der Publikation sind dies Addition, Multiplikation und Konkatenation.

Dabei kann das implizite Wissen selbst abhängig von seiner Komplexität als Vektor, Neuronales Netzwerk oder Matrix repräsentiert werden. Der Vektor gilt dabei als direkte, implizite Repräsentation. Dem fügt das Neuronale Netzwerk Gewichte hinzu und führt eine lineare Kombination oder Nicht-Linearisierung durch. Hingegen werden zur Matrixfaktorisierung mehrere Vektoren mit vorheriger Wissensbasis und einem Koeffizienten genommen. [WYL21]

Schließlich lassen sich die vorgestellten Methoden des impliziten Wissens in die Netzwerkarchitektur von YOLOR einspeisen. Als Basismodell wird dabei das YOLOv4-CSP-Modell der Scaled-YOLOv4-Reihe [WBL21] verwendet [WYL21], welches sich aus der CSPisierung von YOLOv4 [BWL20] ergibt. CSPisierung beschreibt dabei die Anwendung der Konzepte von Wang et al. [WLW+20]. Im Kontext von YOLOv4-CSP wird die Hälfte der Output-Features über den direkten Hauptpfad des Netzwerks geleitet, wodurch eine größere Menge an semantische Informationen generiert werden, während die andere Hälfte der Features über einen Umgehungspfad geleitet werden, der eine größere Menge an räumliche Informationen erhält [Boc20].

Für das Backbone-Netzwerk von YOLOv4-CSP kommt CSPDarknet53 zum Einsatz, dem ein CSPisiertes PANet [LQQ+18] als Erweiterung von FPN [LDG+17] sowie ein integriertes SPP-Modul als Neck folgen [WBL21]. Angelehnt an YOLOv4 wird YOLOv3 [RF18] als Head des Netzwerks verwendet. Durch YOLOR wird nun implizites Wissen in Form von Kernel Space Alignments (auch Feature Alignments), Prediction Refinement des Modells [WYL21] durch eine Korrektur seitens der Loss-Funktion [Tew22] und Multi-Task Learning an bestimmten Stellen des YOLOv4-CSP-Modells angewandt. Aufgaben, die das Multi-Task Learning inkludieren sind die Objekterkennung, die Multi-Label-Bildklassifikation und die Feature-Einbettung [WYL21].

Die entsprechenden Stellen des impliziten Wissens können der Netzwerkarchitektur von YOLOR in Abbildung 3.9 entnommen werden.

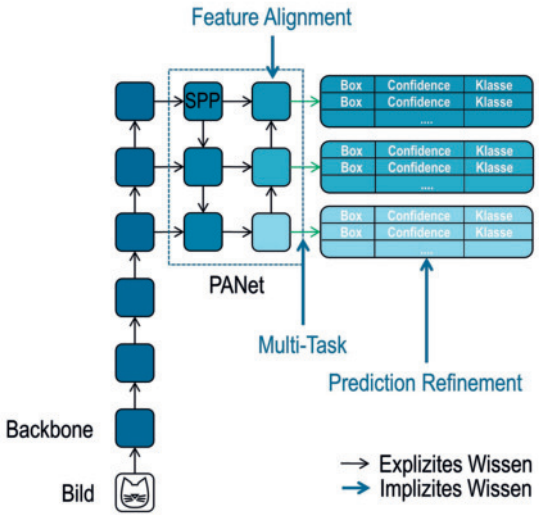

Abbildung 3.9: Die grundlegende Architektur von YOLOR nach [WYL21] und [WBL21].

EfficientDet

EfficientDet [TPL20] ist ein einstufiger Detektor zur Objekterkennung aus dem Jahre 2020, dessen Fokus insbesondere bei der Skalierung von Modellen liegt. Typischerweise werden unterliegende Basisdetektoren zur Steigerung der Genauigkeit skaliert, indem größere Backbone-Netzwerke eingesetzt werden. Diese Methoden sind jedoch in ihren Skalierungsdimensionen eingeschränkt. Auch die Optimierung der Repräsentation und Verarbeitung von Multi-Scale-Features wird mit EfficientDet thematisiert [TPL20].

EfficientNets [TL19] bilden das Backbone-Netzwerk von EfficientDet. Dabei wird das Basisnetzwerk EfficientNet-B0 bis zu EfficientNet-B7 in den Dimensionen Tiefe, Breite und Auflösung anhand eines *Compound-Koeffizienten* skaliert und an der Feature-Extraktion eines Bildes beteiligt. Die extrahierten Features der Level 3 bis 7, namentlich der Netzwerke EfficientNet-B3 bis EfficientNet-B7, werden schließlich dem entwickelten Feature-Netzwerk *Bi-Directional Feature Pyramic Network* oder auch BiFPN [TL19] übergeben.

Mit BiFPN soll die *Multi-Scale Feature Fusion* zur Aggregation von Features unterschiedlicher Auflösungen effektiv umgesetzt werden. Mathematisch ausgedrückt existiert eine Liste von Multi-Scale-Features $\vec{P}^{in} = (P_{l_1}^{in}, P_{l_2}^{in}, ...)$, wobei $P_{l_i}^{in}$ das Feature des Levels l_i repräsentiert. Nun soll eine Transformation f gefunden werden, die verschiedene Features aggregiert und eine Liste neuer Features ausgibt, also $\vec{P}^{out} = f(\vec{P}^{in})$. Für das konventionelle FPN [LDG+17] und ein Bild der Auflösung 640×640 bedeutet das, dass P_3^{in} das Feature Level 3 mit der Auflösung 80×80 (da Auflösung$/2^{l_i} = 80$) repräsentiert. [TPL20]

BiFPN soll nun jedoch am Beispiel von PANet [LQQ+18] und NAS-FPN [GLL19] skalenübergreifende Verbindungen optimieren. Dazu enthält es nur noch Knoten mit mehr als einer eingehenden Kante, da diese dem Feature-Netzwerk intuitiv einen größeren Beitrag leisten. Um mehr Features ohne größere Kosten zu fusionieren, besitzt jeder Inputknoten eine Kante zu dem Outputknoten seines Levels. Zusätzlich wird jeder bidirektionale Pfad als eine Schicht des Feature-Netzwerks verstanden und jede Schicht mehrfach wiederholt, sodass die Feature Fusion auch auf höheren Ebenen ermöglicht wird. [TPL20]

Die Feature Fusion selbst bewertet alle Features nun nicht mehr gleich, sondern ist gewichtet. Dies liegt dem Gedanken zugrunde, dass typischerweise Features in unterschiedlicher Auflösung auch unterschiedlich zu dem Output beitragen. Demnach werden vom Netzwerk erlernbare Gewichte für jeden Input hinzugefügt. Die tatsächliche gewichtete Fusion erfolgt über die Methode *Fast Normalized Fusion* mit $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$, wobei $w_i \geq 0$ und $\epsilon = 0,0001$ gilt. Schließlich werden die fusionierten Features in ein

Klassen- und Box-Netzwerk eingespeist, welches die Objektklasse und die Bounding Box vorhersagt [TPL20].

Zur Entwicklung einer Menge an Modellen, die mit optimierter Genauigkeit und Effizienz ein weites Spektrum an verschiedenen Bedingungen von Ressourcen bedienen können, wird das EfficientDet Basismodell anhand eines Compound-Koeffizienten ϕ skaliert. Die Autor*innen sprechen dabei von ihrer *Compound Scaling* Methode.

Sowohl das Backbone-Netzwerk als auch das BiFPN und das Box-/Klassen-Vorhersage-Netzwerk sowie die Input Bildauflösung wird jeweils skaliert. Es entstehen acht EfficientDet-Modelle, namentlich EfficientDet-D0 ($\phi = 0$) bis EfficientDet-D7 ($\phi = 7$), wobei EfficientDet-D7 das größte Netzwerk bildet, das gleichermaßen auch die größte Genauigkeit erreicht [TPL20].

Abbildung 3.10 zeigt die grundlegende Netzwerkarchitektur von EfficientDet.

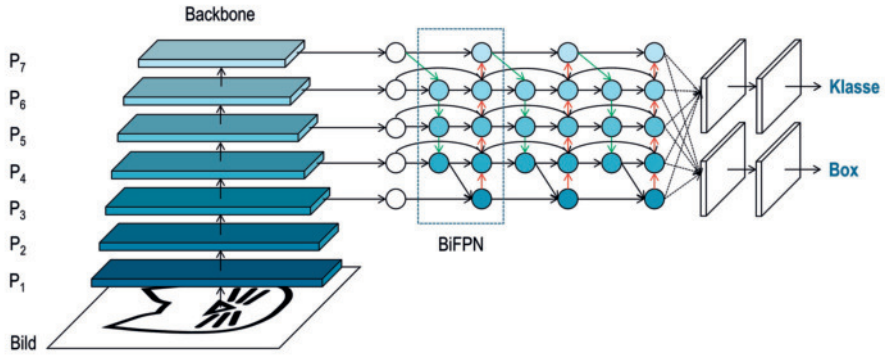

Abbildung 3.10: Die grundlegende Architektur von EfficientDet nach [TPL20].

3.7 Evaluationsmetriken der Objekterkennung

Um die Genauigkeit eines trainierten Modells eines Objekterkennungsdetektors zu evaluieren, werden entsprechende Metriken benötigt. Die Genauigkeit beschreibt dabei im Allgemeinen den Anteil an Datenelementen, für die der Detektor korrekte Ausgaben liefert [GBC16]. Dazu kommt im Kontext der Objekterkennung der isolierte Test- oder Validierungsdatensatz zum Einsatz, dessen Ground Truth-Annotationen mit den von dem Detektor vorhergesagten Bounding Boxen verglichen werden. Die bekannteste Metrik hierfür ist die *Average Precision* (AP) [PNDS20]. Im Folgenden werden diese sowie grundlegende Konzepte aufgeführt, die mit ihr verbunden sind.

Zunächst sind die grundlegenden Konzepte zur Evaluation von Machine Learning-Modellen im Hinblick auf die Objekterkennung in Tabelle 3.3 festgehalten. Im Kontext

dieser spielen TNs in Wirklichkeit keine Rolle, da eine unendliche Menge an Bounding Boxen pro Bild existiert. Aus diesem Grund werden in der Objekterkennung keinerlei Metriken verwendet, die TN enthalten [PNDS20].

True Positive (TP)	Eine korrekte Vorhersage einer Ground-Truth Bounding Box.
True Negative (TN)	Ein korrektes Ausbleiben von Vorhersagen falscher Ground-Truth Bounding Boxen.
False Positive (FP)	Eine fehlerhafte Vorhersage eines nicht existenten oder an anderer Stelle platzierten Objekts.
False Negative (FN)	Ein fehlerhaftes Ausbleiben der Vorhersage einer Ground-Truth Bounding Box.

Tabelle 3.3: Die grundlegenden Evaluationskonzepte TP, TN, FP und FN des Machine Learnings im Hinblick auf die Objekterkennung nach [PNDS20].

Um zu etablieren, was als tatsächlich korrekte und fehlerhafte Detektion gilt, wird häufig die Metrik *Intersection over Union* (IoU) herangezogen. Anhand ihr wird die Similarität zweier Mengen evaluiert, indem ihre überlappenden Werte, beziehungsweise in der Objekterkennung ihre Flächen verglichen werden. Genauer gesagt wird berechnet, zu welchem Grad die Fläche der von dem Detektor vorhergesagte Bounding Box B_p mit der Ground-Truth Bounding Box B_{gt} übereinstimmt. Es ergibt sich folgende Formel:

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

Wird der berechnete IoU nun mit einem vordefinierten Schwellenwert t verglichen, kann eine Detektion als korrekt, wenn $IoU \geq t$ gilt, oder fehlerhaft, wenn $IoU < t$ gilt, klassifiziert werden. Häufig werden Werte wie $t = 50\%$ verwendet. [PNDS20]

In der Praxis wird t mit der *Confidence* verglichen, mit der ein Detektor eine Vorhersage getroffen hat [PPD+21].

Überdies basieren die meisten Evaluationsmetriken auf *Precision* P und *Recall* R:

$$P = \frac{TP}{TP + FP}$$
$$R = \frac{TP}{TP + FN}$$

P gibt dabei den Prozentsatz korrekter positiver Vorhersagen und damit die mögliche Fähigkeit des Detektors an, lediglich relevante Objekte zu identifizieren. Hingegen wird durch R der Prozentsatz korrekter positiver Vorhersagen über allen Ground Truth-Werten beschrieben. In Zuge dessen ergibt sich durch R die mögliche Fähigkeit des

Detektors, alle relevanten Ground Truth-Bounding Boxen vorherzusagen.

Beide Werte können anhand einer $P \times R$-Kurve dargestellt werden, die ihren Trade-Off in Form der Confidence eines Detektors darstellt. Wird FP durch einen Detektor beispielsweise niedrig eingeschätzt, ist P hoch, wodurch jedoch viele TP übersehen werden können. Dadurch ergibt sich ein hoher FN und damit ein niedriger R. Gleichermaßen kann R durch viele TPs steigen, wodurch gleichermaßen die FPs steigen können, was wiederum P sinken lässt. Ein guter Objektdetektor sollte alle Ground Truth-Objekte finden und gleichermaßen nur relevante Objekte identifizieren, also $FN = 0$ und $FP = 0$ anpeilen. Eine große Fläche unterhalb der Kurve gibt dabei einen hohen P- sowie R-Wert an. Um einen genauen Wert zu messen, kommt die N-Punkt- oder Alle-Punkte-Interpolation ins Spiel. [PNDS20, PPD+21]

Die N-Punkt-Interpolation wird beispielsweise zur Evaluation von Detektionen von COCO verwendet, wobei $N = 101$[13] gilt. Sie findet damit auch Anwendung bei YOLOR[14] und der TensorFlow Object Detection API, wenn entsprechende COCO-Metriken zur Evaluation ausgewählt werden. Es ergibt sich folgende Formel des APs nach [PNDS20, PPD+21]:

$$AP_N = \frac{1}{N} \sum_{n=1}^{N} P_{interp}(R)$$

Dabei gilt $P_{interp}(R) = \max_{\widetilde{R}:\ \widetilde{R} \geq R} P(\widetilde{R})$. Wird nun durch alle Punkte interpoliert, ergibt sich hingegen

$$AP_{all} = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1})$$

mit $P_{interp}(R_{n+1}) = \max_{\widetilde{R}:\ \widetilde{R} \geq R_{n+1}} P(\widetilde{R})$.

Anhand jener AP-Metriken jeder Klasse lässt sich nun vielmehr der *Mean Average Precision* (mAP) über allen Klassen wie folgt berechnen [PNDS20]:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

Auch er stellt ein bekanntes Maß zur Evaluation von Objekterkennungsdetektoren dar und soll in der vorliegenden Masterthesis zum Einsatz kommen. Dabei ist insbesondere die Definition des IoU von Relevanz. So existiert der mAP@0.5 unter einem IoU von

[13] Vgl. `recThrs` der Evaluationsmetriken des COCO-Exportformats nach dem COCO Consortium [Con Jb].

[14] Vgl. Zeile 134 des Programmcodes `https://github.com/WongKinYiu/yolor/blob/main/utils/metrics.py` der YOLOR-Implementierung nach Wang et al. [WYL21], besucht am 30.06.2022

50 %, wohingegen der mAP@[.5:.95] den durchschnittlichen Wert unter einem IoU von 50 bis 95 % mit einer Schrittweite von 5 % angibt [RHGS15].

3.8 Größenapproximation auf Bildern

Bei der maschinellen Größenapproximation werden die Längenmaße eines Objekts in Echtzeit oder nachträglich ohne direkten Kontakt mit dem Objekt approximiert [TSSE20]. Dadurch kann das Objekt tiefergreifend analysiert werden [SIG+18]. Jener Umstand sowie der aktuelle Forschungsstand einer nachträglichen maschinellen Größenapproximation auf Bildern sind bereits in Kapitel 2.2 dargelegt worden. In diesem Kapitel wird demgegenüber vermehrt auf den theoretischen Rahmen der Größenapproximation auf Bildern, etwaiger Herausforderungen sowie Methodiken eingegangen.

Bei dem Fotografieren eines Objekts wird seine Dreidimensionalität in der realen Welt in den zweidimensionalen Raum eines Fotos übertragen. Dabei gehen relevante Informationen wie die Objekttiefe [SIG+18] verloren oder werden verfälscht. So hat die Perspektive, in der ein Objekt abgelichtet wurde, einen Einfluss auf die schlussendliche Abbildung des Objekts in der Fotografie und damit auf die nachträgliche Größenapproximation. Fotografiert man beispielsweise eine Hausfassade, so scheinen die Fenster, die näher an der Kamera waren im Bild größer als jene, die weiter weg lagen, obwohl in der realen Welt alle Fenster dieselbe Größe besitzen [Cri02]. Jene Problematiken können behandelt werden, in dem unter anderem mehrere Fotografien aus verschiedenen Perspektiven des Zielobjekts aufgenommen werden, um unter anderem die Dreidimensionalität des Objekts zu rekonstruieren. Dies ist jedoch nicht immer möglich [Cri02]. Liegt beispielsweise eine große zeitliche oder räumliche Distanz zwischen der Fotografie des Objekts und dem aktuellen Wunsch seine Größe zu approximieren, gestalten sich neue Aufnahmen aus anderen Perspektiven als schwierig. Dasselbe gilt natürlich, wenn das Objekt gar nicht mehr oder nicht mehr auf dieselbe Weise existiert.

Typischerweise kommen in diesen Situationen Referenzobjekte zum Einsatz, die sich im Bild befinden und deren tatsächliche Größe bekannt ist, beziehungsweise bekannt sein muss [SIG+18, KDB+17, Ros16]. Dabei wird sich dem Pixel pro Metrik-Verhältnis des Referenzobjekts aus Bildgröße und tatsächlicher Größe bedient, um dann anhand der Pixel des zu berechnenden Objekts auch dessen tatsächliche Größe zu ermitteln [SIG+18]. Um den zuvor beschriebenen Herausforderungen nicht zu unterliegen, ist es wichtig, dass das Referenzobjekt und das zu berechnende Objekt komplanar sind und damit ungefähr dieselbe Distanz zur Kamera vorweisen können. Darüber hinaus sollte das Objektiv der Kamera möglichst auf Höhe der Objekte orthogonal zur Ebene stehen, sodass mögliche Verzerrungen weitestgehend vermieden werden können [Ros16].

Eine andere Möglichkeit der Größenapproximation von Objekten auf Bildern ergibt

sich, wenn der geometrische Aufbau des Bildes bekannt ist [KDB+17, Bon14]. Mittels einfacher Trigonometrie kann so die Größe eines Objekts anhand des Abstands zur Kamera d, der Brennweite der Kamera b und der Größe des Objekts auf dem Sensor der Kamera h_{sensor} berechnet werden. Damit ergibt sich die Formel $h_{real} = (h_{sensor}/b) \cdot d$ [Bon14].

4
Konzept

Erste Konzepte zur Behandlung der Aufgabenstellung 1.1 werden in diesem Kapitel entworfen. Sie dienen dem besseren Verständnis der möglichen Lösungsstrategien und als Grundlage der folgenden Implementation. In einem ersten Schritt werden demnach mögliche Handlungsansätze zur Lösung der zuvor definierten Teilaufgaben erarbeitet: Der Lineallokalisierung, der Keramikgefäßerkennung und -lokalisierung sowie der Größenapproximation der Keramikgefäße. Anschließend werden Herausforderungen identifiziert, die konzeptionell behandelt werden und dann in das folgende Implementierungskonzept einfließen.

4.1 Handlungsansätze zur Lösung der Aufgabenstellung

In Kapitel 1.1 ist die in der Thesis zu bearbeitende Aufgabenstellung in drei Teilaufgaben zerlegt worden. Folglich werden mögliche erste Handlungsstrategien für jene Teilaufgaben in diesem Kapitel erarbeitet und grob evaluiert.

4.1.1 Lokalisierung des Lineals

Die Lokalisierung des Lineals in den Fotografien kann prinzipiell über drei verschiedene Methoden funktionieren. Zuerst besteht die Möglichkeit, sich vielen Publikationen aus Kapitel 2.2 folgend konventionelle Techniken der Bildverarbeitung zu Nutze zu machen, wie beispielsweise der Digitalen Fourier Transformation. Durch die Fourier Transformation kann die Skala eines Lineals anhand des regelmäßigen Musters ihrer Amplitude im Frequenzbereich des Bildes ermittelt und damit nicht nur für die Größenapproximation [HZK07], sondern auch zur Lokalisierung verwendet werden. In den vorherigen Kapiteln 2 und 3.2 ist bereits angedeutet worden, dass dieser Ansatz sich für die vorliegende Datenbasis aufgrund ihrer in Kapitel 3.1 beschriebenen vielen, kleinen Objekte und verminderter Bildqualität nicht eignet und damit für das weitere Vorgehen ausgeschlossen werden kann.

Eine andere Möglichkeit kann das Deep Learning und damit die maschinelle Erkennung mittels Deep Convolutional Neural Networks bieten. Jedoch wird auch jener

aufgrund der geringen, aber komplexen Datenbasis nur mit geringen Erfolgserwartungen entgegengeblickt. Die präzise Lokalisierung des Lineals ist für eine sinnvolle Größenapproximation unter Verwendung eben jenes Lineals unabdingbar. Nichtsdestotrotz wird ein Ausschluss dieser Methode mit Absprache mit dem Big Data Labs an aktueller Stelle nicht vorgenommen, da jegliche Versuche in jenem Bereich dem LOEWE-Projekt zugutekommt.

Schließlich kann die Lokalisierung des Lineals auch manuell anhand der sowieso für die Deep Learning-Herangehensweise benötigte Annotation der Lineale erfolgen. Aufgrund der geringen Datenmenge und der rechteckigen Form des Lineals, ist die Annotation überdies einfach und schnell von Menschenhand durchführbar. Damit würde sie sich auch für den Anwendungsfall lohnen, wenn sie nicht sowieso durch die Deep Learning-Herangehensweise gegeben wäre. So kann sie jedoch im Zweifelsfall ohne Mehraufwand zur Lokalisierung des Lineals verwendet werden.

4.1.2 Erkennung und Lokalisierung der Keramikgefäße

Im Gegensatz zu den Handlungsansätzen der Lokalisierung des Lineals aus Kapitel 4.1.1 können bei der Erkennung und Lokalisierung der einzelnen Keramikgefäße lediglich Methoden des Deep Learnings zum Einsatz kommen.

Ein manuelles Vorgehen ergibt aufgrund der Komplexität der Formen und der hohen Anzahl der Keramikgefäße pro Fotografie technisch keinerlei Sinn und würde für die aktuelle, digitale Lage der Archäologie keinen Fortschritt bedeuten. Auch für konventionelle Techniken der Bildverarbeitung ist die Datenbasis in diesem Fall zu komplex.

4.1.3 Größenapproximation der Keramikgefäße

Durch das in den Fotografien vorhandene Lineal existiert ein bekanntes und vor allem verlässliches Größenmaß, welches wie in Aufgabenstellung 1.1 definiert zur Größenapproximation der zweidimensionalen Gesamthöhe und -breite der Keramikgefäße verwendet werden kann. Als Voraussetzung gelten die den beiden vorherigen Kapiteln 4.1.1 und 4.1.2 folgenden lokalisierten Lineale und Keramikgefäße. Die Grundidee bildet die Berechnung der Pixel der generierten Bounding Boxen eines in den Fotografien abgebildeten und in der realen Welt gültigen metrischen Maßes (hier durch das Lineal gegeben), durch welches die zweidimensionale Gesamthöhe und -breite der lokalisierten Gefäße anhand ihrer Pixel und einfacher Mathematik approximiert werden kann. Dennoch existieren verschiedene Möglichkeiten zur genauen Durchführung der Approximation, welche im Folgenden ausgeführt werden.

Zunächst kann die Skala des abgebildeten Lineals in menschlicher Manier herangezogen werden. Über konventionelle Techniken der Bildverarbeitung können, wie bereits in

Kapitel 4.1.1 erwähnt, die einzelnen Striche der Skala identifiziert werden und als Maß gelten. Bereits zwei direkt nebeneinander liegende Striche der Skala genügen, um die Anzahl an Pixeln für 1 cm festzustellen und damit die Größe der Gefäße zu approximieren. Gleichermaßen können auch die Zahlen der Skala mittels Techniken der Computer Vision erkannt und ihr Abstand in Pixeln ähnlich verwertet werden. Vorausgesetzt ist dabei, dass die Skala des Lineals leserlich und damit von guter Qualität ist.

Stattdessen kann auch die Gesamtlänge des sowieso lokalisierten Lineals verwendet werden. Da es gilt, den spezifischen Anwendungsfall der Aufgabenstellung 1.1 zu lösen und demnach immer dasselbe, exakt 50 cm lange Lineal in den Fotografien abgebildet ist, gestaltet sich dieses Vorgehen als äußerst effizient und einfach. Dennoch können Probleme auftreten, wenn das Lineal nicht in seiner ganzen Länge oder schief abgebildet wird. Diese werden in den unmittelbar folgenden Kapiteln genauer behandelt.

Auch andere Referenzobjekte können als Maß der realen Welt für die Größenapproximation herhalten, wie beispielsweise die standardisierten Regale, in denen die Keramikgefäße auf den Fotografien einsortiert sind. Die Ermittlung ihrer Größe beruht jedoch auf Schätzungen oder bestenfalls der manuellen Verwendung des Lineals. Entsprechend eignen sie sich nicht für die grundlegende Größenapproximation, können aber bei Spezialfällen zum Einsatz kommen; zum Beispiel, wenn kein Lineal abgebildet ist.

4.2 Herausforderungsanalyse

Bevor ein konzeptionelles Vorgehen zur Lösung der Aufgabenstellung 1.1 erarbeitet werden kann, müssen zunächst mögliche Herausforderungen erkannt werden, die mit der komplexen Datenbasis 3.1 einhergehen. Erst dann können sie adäquat im weiteren Verlauf behandelt und bestmöglich aufgelöst werden. Der Fokus liegt dabei wie in Kapitel 1.1 definiert auf Fotografien von Keramiksammlungen. Andere Objektkategorien oder Fotografien, die sehr wenige Objekte enthalten, werden nicht beachtet. Zudem betreffen Herausforderungen bezüglich der Lokalisierung von Objekten im Umkehrschluss auch die Größenapproximation – abhängig davon, wie diese im Endeffekt durchgeführt wird. Schließlich werden die möglichen Herausforderungen kategorisiert und folgend aufgeführt.

Grundlegende Herausforderungen

Betreffen Herausforderungen den kompletten Datensatz an Sammelaufnahmen und nehmen Einfluss auf folgende Kategorien, handelt es sich um *grundlegende Herausforderungen*, die anschließend erläutert werden.

HG1) Kleiner Datensatz. Der vorliegende Datensatz besteht aus lediglich 103

Scans und 161 Kameraaufnahmen. Ist nun der Fokus Aufgabenstellung 1.1 folgend gesetzt und werden überdies unpassende Fotografien aussortiert, bleiben 43 verwendbare Scans und 76 verwendbare Kameraaufnahmen mit und ohne Maßstab. Die erfolgreiche Anwendung von Techniken des Deep Learnings wird damit erschwert.

HG2) **Fehlende Farben.** Aufgrund ihres hohen Alters liegen die Fotografien nur in dem Farbton Sepia vor. Damit können die abgebildeten Objekte nicht mehr direkt anhand ihrer tatsächlichen Farbe erkannt und von anderen Objekten unterschieden werden. Es ergeben sich weniger Inter-Klassen-Variationen der Objekte.

HG3) **Mangelnde Bildqualität.** Das Alter der Fotografien bringt außerdem qualitative Einbuße mit sich. Zu Teilen sind die Fotografien unscharf, zu hoch oder niedrig belichtet sowie zu kontrastreich oder -arm. Insbesondere die Kameraaufnahmen der Fotografien sind davon betroffen und zudem verbogen. Weniger Inter-Klassen-Variationen der Objekte können unter anderem eine Folge sein.

HG4) **Viele kleine Objekte.** Da es sich bei den Fotografien um Sammelaufnahmen handelt, beinhalten sie viele und dadurch klein erscheinende Objekte, wodurch deren Auflösung sowie maschinelle Erkennung leidet.

HG5) **Unterschiedliche Ausrichtungen der Objekte.** Die einzelnen Objekte sind mit verschiedener Häufigkeit unterschiedlich in den Fotografien ausgerichtet und positioniert. Ihre Lokalisierung gestaltet sich damit durch viele Intra-Klassen-Variationen schwieriger.

HG6) **Verdeckung der Objekte.** Sowohl die Antiken als auch das Lineal können von anderen Gegenständen zu kleinen oder großen Teilen verdeckt werden. Hierdurch wird ihre maschinelle Erkennung aufgrund möglicher vieler Intra-Klassen- und weniger Inter-Klassen-Variationen erschwert.

Linealspezifische Herausforderungen

In dieser Kategorie werden linealspezifische Herausforderungen zugeordnet. Die zuvor definierten Herausforderungen **HG1** bis **HG6** betreffen darüber hinaus ebenso das Lineal in Form von beispielsweise schlechter Qualität und damit unleserlicher Skala oder starker Biegung des Lineals in den Kameraaufnahmen. Ein praktischer Überblick einiger dieser Herausforderungen für die Lineallokalisierung ist durch das Baumdiagramm der Scans der Keramikgefäßsammlungen in Abbildung 4.1 zu gewinnen.

HL1) Fehlendes Lineal. Nicht auf jeder Fotografie ist ein Lineal abgebildet, wodurch ein anderes Maß zur Größenapproximation benötigt wird.

HL2) Nicht vollständig abgebildete Lineallänge. Viele Fotografien bilden nur Teile des Lineals ab. Dadurch ergibt sich nicht nur die erwähnte Herausforderung **HG6**, sondern auch spezifische Herausforderungen, die die Größenapproximation anhand der Lineallänge betreffen. So ist jene problemlos durchzuführen, wenn die Länge des Lineals weiterhin vollständig abgebildet ist. Ist dem nicht so, ergeben sich zwei erschwerende Teilherausforderungen:

HL2.1) Einzig die Lineallänge ist nicht vollständig abgebildet.

HL2.2) Die Lineallänge und -höhe sind nicht vollständig abgebildet.

HL3) Schiefe Ausrichtung des Lineals. Ist das Lineal nicht möglichst gerade ausgerichtet und werden Bounding Boxen zur Ermittlung seiner Größe in Pixeln verwendet, die im Gegensatz zu dem Lineal immer parallel zu den Bildrändern sind, kann die Größenapproximation der Keramikgefäße stark verfälscht werden. Auch **HG5** kommt erneut zum Tragen.

HL4) Verwechslung von dem Lineal mit anderen Objekten. Automatisierten Methoden kann die Unterscheidung von dem Lineal und Regalbrettern, länglichen Keramikgefäßen und anderen Objekten aufgrund ihrer ähnlichen Erscheinung schwerfallen.

Keramikgefäßspezifische Herausforderungen

Schließlich werden Herausforderungen, die neben **HG1** bis **HG6** die Keramikgefäße betreffen, dieser Kategorie zugeordnet. Gewisse Teile der Herausforderungen sind ebenso auf die anderen Objektkategorien der Statuen sowie Waffen/Schmuck zu übertragen. Für die vorliegende Masterthesis sind diese allerdings nicht von Relevanz.

HK1) Eingeschränkte Größenapproximation. Jegliche Größenapproximation, die über das Approximieren der gesamten, zweidimensionalen Breite und Höhe eines Objekts hinaus geht, gestaltet sich schwierig. So ist beispielsweise die Größe eines frontal ausgerichteten Tellers problemlos zu approximieren. Ist der Teller allerdings in einem 45 Grad Winkel zur Kamera ausgerichtet, ist seine tatsächliche Breite nicht zu bestimmen. Dies lehnt sich außerdem an **HG5** an. Gleichermaßen sind spezifische Maße, wie der Innenradius eines Krughalses, schwierig zu approximieren und zu Teilen ausrichtungsabhängig.

HK2) Gemeinsame Darstellung von Objekten unterschiedlicher Kategorien. Insbesondere im Kontext der Masterarbeit, bei der nur Fotografien der Keramiksammlungen bearbeitet werden sollen, ist das gelegentliche Auftauchen von Nicht-Keramikgefäßen in jenen Fotografien negativ und kann zu vermehrt falschen Detektionen führen.

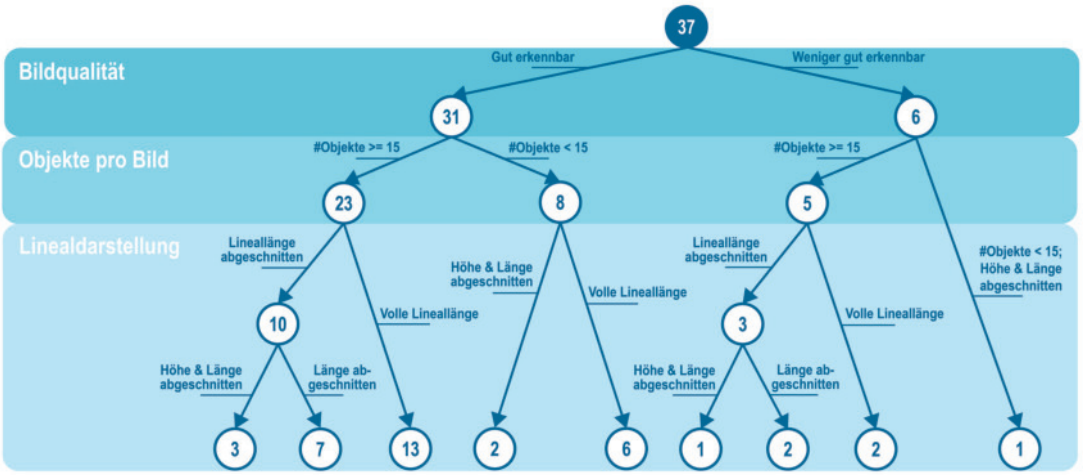

Abbildung 4.1: Gliederung der Scans von Keramikgefäßsammlungen mit Lineal in Bildqualität, Anzahl der Objekte pro Bild und Linealdarstellung[1].

4.3 Mögliche Lösungsstrategien der Herausforderungen

Aufgrund ihres Umfangs ist das Ziel der Masterthesis nicht die Lösung jeder identifizierten Herausforderung. Dennoch werden im folgenden Kapitel etwaige Strategien zur Auflösung der Herausforderungen festgestellt und aufgeführt. Damit werden insbesondere die ersten Handlungsansätze aus Kapitel 4.1 weiter präzisiert.

Vergrößerung des Datensatzes

Wie bereits in Kapitel 3.2 beschrieben, kann eine größere Datenmenge den typischen Herausforderungen des Machine Learnings entgegenwirken. Da der vorliegende Datensatz in **HG1** und Kapitel 3.1 bereits als sehr klein identifiziert worden ist, stellt seine Aufblähung einen ersten möglichen Schritt zur Lösung einiger Herausforderungen dar. Den Herausforderungen aller **HG** sowie **HL4** und **HK2** würde bereits die verbesserte

[1] Eine Gliederung aller Scans und Kameraaufnahmen von Objektsammlungen mit Lineal ist in Abbildung A.1 im Appendix dargestellt.

Erkennung von Objekten entgegenkommen, die mit einem größeren Datensatz einhergehen würde. Leider gestaltet sich eine derartige Vergrößerung eines Datensatzes in den meisten Fällen jedoch weniger einfach.

So ist bezüglich des spezifischen Anwendungsfalls der Masterthesis eine Hinzunahme von Bildern aus dem Internet oder durch andere Museen auszuschließen. Durch eine derartige Aufblähung relevanter Daten würde zwar zu einer Generalisierung hingearbeitet werden, aber nicht unbedingt zu der Lösung des Anwendungsfalls. Ein positiver Einfluss wäre nur durch der vorliegenden Datenbasis äußerst ähnlichen Bildern gegeben, welche der Thesis nicht vorliegen.

Ein Datensatz kann jedoch auch künstlich aufgebläht werden, indem Data Augmentation nach Kapitel 3.4 zum Einsatz kommt. Durch jene können künstliche Datenelemente aus den originalen generiert werden, die beim Training benutzt werden können. Auch diese unterliegen jedoch der Bedingung, den originalen Daten noch möglichst ähnlich sehen zu müssen.

Effektive Datennutzung

Im Hinblick auf Herausforderung **HG1** bietet sich außerdem die in Kapitel 3.5 beschriebene Cross Validation zur effektiveren, tatsächlichen Nutzung aller Datenelemente sowie der damit einhergehenden besseren Bewertung der Modelle der Objekterkennungsdetektoren an. Aufgrund der vielen, ungleich häufig auftretenden Klassen erscheint zunächst die erwähnte Stratified Cross Validation als beste Option, um einer ungleichen Verteilung der Klassen innerhalb der Trainings- und Validierungsmenge entgegenzuwirken. Da jedoch aufgrund des Fokus auf Fotografien der Keramikgefäßsammlungen pro Fotografie mehrere Objekte verschiedener Klassen abgebildet werden, gestaltet sich dies schwierig[2] und sprengt damit den bereits breit gefassten Rahmen der Masterthesis.

Eine weniger optimale, aber einfache Herangehensweise zur Sicherung, dass jede Klasse zumindest einmal in der Trainings- und Validierungsmenge auftritt, ist die Leave-One-Out Cross Validation. Zwar ist jene Form der Cross Validation leicht zu implementieren, jedoch sprengt auch sie den Rahmen der Masterthesis – allerdings auf Grund der nötigen Trainingszeit. Allein im Fall der Scans der Keramikgefäßsammlungen würden 43 Modelle trainiert werden müssen, deren Epochen oder Schritte überdies zur Realisierung von Zeiteinsparungen nicht groß reduziert werden könnten, um immer noch aussagekräftige Modelle zu generieren.

Stattdessen bleibt die K-Fold Cross Validation mit einem geringeren K und einer

[2] Beispielsweise ist das Modul `StratifiedKFold` von sklearn wie viele andere Implementationen nicht auf die Multi-Label-Klassifikationen ausgelegt. Vgl. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html`, besucht am 16.11.2022 und den Stack Overflow-Beitrag von Tsatsoulis [Tsa18].

manuell nötigen Überprüfung und Anpassung der Trainings- und Validierungsmenge, sodass Klassen mit Sicherheit in beiden Mengen vorkommen.

Identifikation geeigneter Objekterkennungsdetektoren

Es existiert eine Reihe an verschiedenen Detektoren zur Objekterkennung mit unterschiedlichen Eigenschaften. Entsprechend eignen sich auch verschiedene Detektoren besser zur Lösung von bestimmten Anwendungsfällen. So gibt es Detektoren oder Kombinationen von Detektoren, wie durch Drid et al. [DAK20] gezeigt, die sich beispielsweise besser für die Behandlung von vielen, kleinen Objekten, wie in Herausforderung **HG4** erwähnt, verwenden lassen. Aus jenem Grund werden in der vorliegenden Thesis der Aufgabenstellung 1.1 folgend mehrere Detektoren zur Objekterkennung getestet und evaluiert. Erst im folgenden Kapitel 4.4 werden entsprechend drei Detektoren zum weiteren Vorgehen ausgewählt, doch eine Liste mehrere Detektoren, ihre grundlegenden Eigenschaften und Vorgehensweisen sind bereits in den Kapiteln 3.6 und 2.1 theoretisch beschrieben worden.

Anpassung der Bilder

Insbesondere im Hinblick auf **HG2** und **HG3** können Scans und Kameraaufnahmen manuell und ganz individuell über ein Bildbearbeitungsprogramm angepasst werden. So kann zum Beispiel der Papprand jeder Fotografie entfernt oder ihr Kontrast und ihre Belichtung verändert werden. Theoretisch ist es sogar möglich, die Biegung der Kameraaufnahmen oder Objekte anderer Kategorien, wie in **HK2** erläutert, zu entfernen. Auch wenn die vorliegende Datenbasis klein ist, ist eine für jeden Scan oder Kameraaufnahme individuelle manuelle Anpassung mit viel Arbeit verbunden. Gleiches gilt für die Implementation einer maschinellen Lösung. Die Herausforderungen **HG1**, **HG4** bis **HG6** sowie alle **HL** und **HK1** würden dennoch bestehen bleiben. Entsprechend würden die hohen Kosten einer perfekten Anpassung zur Lösung von **HG2** und **HG3** möglicherweise nur einen geringen Nutzen erzielen.

Bildannotation in Form von Polygonen

Im vorgegangenen Forschungsprojekt [Wir21] wurde bereits deutlich, dass die zeitaufwendige Polygon-Bildannotation der Bounding Box-Bildannotation bezüglich der erzielten Genauigkeit von Machine Learning-Modellen überlegen ist. Es ist naheliegend, dass sie auch dem komplexen Anwendungsfall der Masterthesis zugutekommen kann, indem die Erkennung der Objekte verbessert wird. Näher beschrieben werden die verschiedenen Techniken der Bildannotation in Kapitel 3.3.

Mögliche Herausforderungen, die einen Vorteil aus der Polygon-Annotation ziehen können sind **HG2** bis **HG6** sowie **HK2**, da durch sie die tatsächliche Form der zu erkennenden Objekte stärker zum Tragen kommen könnte. Überdies könnten schiefe Ausrichtungen des Lineals, wie in Herausforderung **HL3** beschrieben, korrekt bemessen werden. Auch bezüglich **HK1** könnte eine spätere Größenapproximation spezifischer Faktoren der Keramikgefäße erleichtert werden.

Nutzung verschiedener Referenzobjekte

Nach Herausforderung **HL1** ist nicht auf jeder Fotografie ein Lineal abgebildet, das zur Approximation der Größe von den Keramikgefäßen verwendet werden kann. Entsprechend müssen andere dargestellte Objekte als Größenreferenz herhalten, wie zum Beispiel die auf jeder relevanten Fotografie enthaltenen, gleichen Regalbretter. Ihre tatsächliche Größe kann beispielsweise über jene Fotografien, die ein Lineal enthalten, manuell approximiert und anschließend für Fotografien ohne Lineal verwendet werden.

Spezieller Umgang mit nicht ganz abgebildeter Lineallänge

Herausforderung **HL2** verdeutlicht, dass nicht auf jeder Fotografie die Lineallänge ganz abgebildet wird. Dadurch wird eine in Kapitel 4.1.3 beschriebene, eigentlich einfache Größenapproximation anhand der Lineallänge erschwert. Eine Entfernung jener Fotografien aus dem sowieso schon spärlichen Datensatz ist überdies keine Option. Motiviert wird diese Aussage durch den Überblick der Lineale innerhalb der Keramikgefäßsammlungen in Abbildung 4.1. Demnach wären ganze 16 von 37 Fotografien zu entfernen, was gewiss nicht tragbar ist.

Maschinell könnten jene in ihrer Länge nicht ganz abgebildeten Lineale über die Koordinaten ihrer Vorhersagen oder Annotationen erkannt werden. Befinden sich diese nah am Bildrand, was über die Größe der Bilder in Pixeln errechnet werden kann, ist das entsprechende Bild näher zu betrachten. Ob nun die Lineallänge oder -breite aus dem Bild herausragt, kann anhand der Abstände zwischen den Koordinaten ermittelt werden. Da die Lineallänge typischerweise in der Datenbasis nicht derart weit aus dem Bild ragt, stellt der größere Abstand die Lineallänge dar. Ragt dieser aus dem Bild, ergibt sich Herausforderung **HL2.1**. Ragt hingegen nur die Linealbreite aus dem Bild, kann die Größenapproximation problemlos durchgeführt werden.

Schließlich kann im Fall von Herausforderung **HL2.1.** das tatsächliche Verhältnis der Seiten des gänzlich abgebildeten Lineals verwendet werden, um die volle Lineallänge zu rekonstruieren. Nachfolgend könnte sie zur Größenapproximation der Keramikgefäße verwendet werden.

Wird nun sowohl die Lineallänge als auch -breite nicht abgebildet, ist das Seitenverhältnis nicht mehr aussagekräftig. Dadurch ergibt sich Herausforderung **HL2.2**, deren Lösung schwieriger ist. So zum Beispiel, wenn die Lineallänge und -breite zu gleich vielen Prozenten über den Bildrand ragen und die tatsächlichen Seitenverhältnisse fälschlicherweise mimen.

Optional können für diesen Fall bestimmte, bereits in Kapitel 4.1.3 erwähnte, Herangehensweisen anhand der Linealskala zum Tragen kommen, die jedoch eine hohe Bildqualität und mehr Aufwand voraussetzen. Eine andere Möglichkeit ist das Heranziehen des tatsächlichen Verhältnisses von Skala-Strichen zur Linealbreite, von welchem weiter auf die Verhältnisse der Lineallänge und -breite geschlossen werden kann. Doch auch für diesen Fall würde zunächst die Länge der Skala-Striche erkannt werden müssen, die darüber hinaus ebenfalls nicht gänzlich abgebildet sein könnten.

Alternativ kann die Rekonstruktion der Lineallänge auch manuell anhand der zuvor erwähnten Ansatzpunkte über die durchgeführte Annotation erfolgen. Dabei wird die Annotation entsprechend so angepasst, dass sie die komplette Lineallänge umschließt, auch wenn diese über den Bildrand hinaus geht. Aufgrund der eingeschränkten Größe des Datensatzes, bildet dieses Vorgehen eine adäquate Lösungsstrategie der vorliegenden Grundherausforderung **HL2**.

4.4 Wahl der Objekterkennungsdetektoren

Aufgabenstellung 1.1 und Kapitel 4.3 folgend gilt es, verschiedene Objekterkennungsdetektoren zu testen, um im Endeffekt den für den Anwendungsfall geeignetsten zu identifizieren. In Tabelle 3.2 sind bereits einige berühmte Detektoren recherchiert und aufgeführt worden, von denen jedoch nur ein Bruchteil im Rahmen der Thesis getestet werden kann. Durch das vorliegende Kapitel soll die Wahl dreier Objekterkennungsdetektoren zum Lösen der Aufgabenstellung 1.1 getroffen und motiviert werden.

Ein wichtiger Faktor ist die Aktualität der Objekterkennungsdetektoren. Bereits in Tabelle 3.2 wurde sich nur auf Detektoren fokussiert, die nach 2014 entwickelt geworden sind und deren GitHub-Repository nach 2018 weiterhin aktiv geblieben ist. Somit kann die Menge an interessanten Detektoren verkleinert werden. Ebenso ist es bei der Entscheidung von Bedeutung, dass zum Zeitpunkt der Masterthesis eine wissenschaftliche Publikation und zu ihr gehöriger Code vorliegt. Aus diesem Grund können auch die Detektoren YOLOv5, G-CNN [PBBC20] und G-RCNN [PPMM21b] ausgeschlossen werden. Es bleibt eine Menge von zehn Detektoren.

Schließlich ist es von Interesse, sowohl ein- als auch zweistufige Detektoren aufgrund

ihrer in Kapitel 3.6 festgehaltenen, groben Unterschiede im Hinblick auf die Aufgabenstellung 1.1 zu testen und ihre Ergebnisse zu evaluieren. Die Wahl sollte also mindestens auf einen ein- und einen zweistufigen Detektor fallen.

Ein sehr berühmter einstufiger Detektor, der stets gute Werte in Genauigkeit und Effizienz erzielt, ist YOLO. Von YOLO gibt es etliche Versionen, einige der neusten sind YOLOv4 [BWL20], sein überlegener Nachfolger Scaled-YOLOv4 [WBL21] und YOLOR [WYL21]. Aufgrund der generellen Popularität des YOLO-Detektors, seiner extremen Aktualität, seiner guten Ergebnisse und nachweislichen Überlegenheit gegenüber Scaled-YOLOv4[3], wird YOLOR von Wang et al. [WYL21] zur Lösung der Aufgabenstellung 1.1 herangezogen.

Außerdem fällt die Wahl auch auf den einstufigen Detektor EfficientDet des Google Brain Teams [TPL20]. Der Grund dafür ist neben seiner guten Ergebnisse der Genauigkeit die Tatsache, dass er hinsichtlich Tabelle 3.2 der aktuellste einstufige Detektor ist, der nicht YOLO-basiert ist. Zwar ist EfficientDet nachweislich weniger effizient als einige der neueren YOLO-Versionen[2], im Fall der Masterthesis spielt Effizienz im Vergleich zur Genauigkeit jedoch nur eine geringfügige Rolle.

Eine genauere Beschreibung der Architektur beider Detektoren ist in Kapitel 3.6.2 zu finden.

Zuletzt wird der sehr bekannte Detektor Mask R-CNN von He et al. [HGDG17] ausgewählt, um zweistufige Detektoren zu repräsentieren. Auch er ist bereits in Kapitel 3.6.1 näher beschrieben worden.

Im Gegensatz zu einstufigen Detektoren lag der Fokus der letzten Jahre nicht auf zweistufigen Detektoren, weswegen es an dieser Stelle weniger Auswahl gibt. Dies mag der Tatsache zu Schulden sein, dass zweistufige Detektoren typischerweise langsamer sind und der aktuelle Trend sich eher in Richtung Echtzeit-Detektion bewegt, wie bereits in den Kapiteln 2.1 und 3.6 angeführt. Dennoch kann Mask R-CNN trotz seines Alters gute Ergebnisse bezüglich der Genauigkeit erzielen und ist deshalb auch in dem der Masterthesis vorgegangenen Forschungsprojekt [Wir21] zum Einsatz gekommen.

Insbesondere durch die Möglichkeit, Polygon-Annotationen als Input zu verwenden, die sich im Forschungsprojekt den einfachen Bounding Boxen gegenüber als überlegen herausgestellt haben, bildet Mask R-CNN einen interessanten Detektor zum Lösen der Aufgabenstellung 1.1. Dies ist auch der Grund, weshalb sich nicht für den eigentlich jüngeren Detektor Cascade R-CNN [CV18] entschieden wird, der durch sein dennoch hohes Alter und R-CNN Basis auch nicht für einen möglichen vierten Detektor von Interesse ist. Stattdessen wäre der äußerst aktuelle Detektor G-RCNN [PPMM21b] attraktiv,

[3] Vgl. die von Bochkovskiy durchgeführte *Waymo Self-Driving Challenge* anhand seines GitHub-Beitrags [Boc21].

hätte er nicht aufgrund von fehlendem Code in einem vorherigen Schritt ausgeschlossen werden müssen.

4.5 Konzept der Implementierung

Anhand der vorgegangenen Kapitel kann schließlich ein Konzept für die folgende Implementierung sowie Evaluation der in Aufgabenstellung 1.1 identifizierten Teilaufgaben erarbeitet werden. Wie bereits definiert, bleibt der Fokus dabei auf Fotografien von Keramikgefäßsammlungen, deren Papprand jeweils manuell entfernt wird.

Zunächst soll für die erste Teilaufgabe der Lokalisierung des Lineals Deep Learning zum Einsatz kommen. Entsprechend gilt es, innerhalb Fotografien, die ein Lineal abbilden, jenes Lineal mittels des bereits durch das vorgegangene Forschungsprojekt [Wir21] bekannte Annotationstool CVAT zu annotieren. CVAT bietet die Möglichkeit, verschiedene Annotationstechniken zu verwenden und die getätigten Annotationen kostenfrei als verschiedene Exportformate herunterzuladen. Des Weiteren kann es auch über eine Weboberfläche verwendet werden. Damit können die Lineale aufgrund ihrer rechteckigen, symmetrischen Form und aufgrund erheblicher Zeiteinsparungen mit Bounding Boxen annotiert werden.

Im Hinblick auf eine möglichst hohe Genauigkeit werden dabei sowohl Scans als auch Kameraaufnahmen der Fotografien mit Lineal verwendet. Neben Fotografien der Keramiksammlungen werden ihnen ähnliche Sammlungen von Statuen mit Lineal ebenfalls hinzugenommen und annotiert. Durch die spätere Einschränkung auf Scans für die folgende zweite Teilaufgabe und die Ermöglichung eines besseren Vergleichs der Detektoren werden an dieser Stelle jedoch ebenfalls lediglich Scans von Keramikgefäßsammlungen zur Lineallokalisierung verwendet.

Dazu werden schließlich die in Kapitel 4.4 ausgewählten Detektoren der Objekterkennung, YOLOR, EfficientDet und Mask R-CNN utilisiert. Für YOLOR kommt die praktische Implementierung[4] von Wang et al. [WYL21] des in Kapitel 3.6.2 beschriebenen Publikation zum Einsatz. Hingegen wird für EfficientDet [TPL20] des Kapitels 3.6.2 und Mask R-CNN [HGDG17] des Kapitels 3.6.1 die TensorFlow Object Detection API[5] verwendet. Diese Entscheidung wird motiviert durch ihre einfache Verwendung, große Beliebtheit und damit einer großen Anzahl an Ressourcen, sowie der Möglichkeit, sowohl Mask R-CNN als auch EfficientDet zu trainieren. Für alle Detektoren müssen die annotierten Scans der Keramikgefäßsammlungen für die Lineallokalisierung zunächst in

[4] YOLOR-Implementation, `https://github.com/WongKinYiu/yolor`, besucht am 14.07.2022

[5] TensorFlow 2 Object Detection API, `https://github.com/tensorflow/models/tree/master/research/object_detection`, besucht am 03.06.2022

ihr entsprechendes Exportformat gebracht werden. Die Inputbildgrößen der Detektoren werden dabei so groß und ähnlich wie ressourcen- und objektdetektorenmäßig möglich gewählt. Etwaige Verzerrungen werden durch Auffüllen mit schwarzen Balken zu vermieden.

Erst dann können die Detektoren unter Verwendung der in den Kapiteln 3.5 sowie 4.3 beschriebenen K-Fold Cross Validation trainiert und validiert werden. Dazu wird $K = 5$ für ein ungefähres Verhältnis von 80:20 für Trainings- und Validierungsdaten nach 3.2 gewählt. Anschließend werden die durchschnittlich erreichten mAP@[.5:.95]-Werte und Trainingsdauern der Modelle jedes Detektors pro Fold sowie ihre Vorhersagen auf den Validierungsdaten betrachtet. Im Rahmen der Thesis wird folglich mAP als Synonym für mAP@[.5:.95] verwendet. Auf eine Evaluation anhand anderen Metriken wird aufgrund des Umfangs der Masterthesis verzichtet.

Auch für die zweite Teilaufgabe der Erkennung und Lokalisierung der Keramikgefäße soll Deep Learning verwendet werden. Deswegen werden die Gefäße innerhalb Fotografien von Keramiksammlungen als erstes mit CVAT anhand einer der Thesis bereitgestellten Liste der abgebildeten Gefäßformen[6] annotiert. Dazu werden dieses Mal jedoch Polygon-Annotationen verwendet. Aufgrund fehlender Expertise bezüglich der korrekten Bezeichnung der Keramikgefäße, wird die Thesis durch das Institut für Archäologische Wissenschaften der Goethe-Universität unterstützt.

Durch die hohe Anzahl an Keramikgefäßen pro Sammelaufnahme und der dennoch vorhandenen Schwierigkeit des Findens ihrer jeweiligen korrekten Bezeichnung ist mit einem sehr hohen Zeitaufwand zu rechnen. Rechnet man mit einer durchschnittlichen Anzahl von 50 zu annotierenden Keramikgefäßen pro Bild, einer positiv eingeschätzten Identifikationsdauer von 5 Minuten pro Objekt und einer Annotationsdauer von 1 Minute pro Objekt, so ist bereits bei 43 zu verwendenden Scans der Keramikgefäßsammlungen mit einer insgesamten Dauer von 12 900 Minuten zu rechnen. Mit fast 100 zusätzlichen Kameraaufnahmen der Keramikgefäßsammlungen wäre die Annotation in jeder Form im Rahmen der Masterarbeit nicht tragbar, auch wenn die Motive doppelt vorkommen und die Identifikationsdauer pro Objekt damit reduziert werden könnte. Entsprechend werden an dieser Stelle nur noch Scans der Keramikgefäßsammlungen verwendet, wodurch wie aufgeführt auch die Limitierung der Dateien bezüglich der Lineallokalisierung der ersten Teilaufgabe motiviert wird.

Dem Gedanken folgend, dass ein Detektor der Objekterkennung, der im Fall der ersten Teilaufgabe nicht überzeugen konnte, mit hoher Wahrscheinlichkeit auch im Fall der komplexeren zweiten Teilaufgabe nicht überzeugen kann, werden an dieser Stel-

[6] Die Liste der abgebildeten Gefäßformen ist ebenfalls über den Zugang zur praktischen Implementierung innerhalb des Kapitels A.1 zu betrachten.

le die beiden zuvor als überlegen identifizierten Detektoren verwendet. Dabei werden sie analog anhand der K-Fold Cross Validation mit $K = 5$ mit ihren entsprechenden Exportformaten trainiert und validiert.

An dieser Stelle wird aufgrund der hohen Komplexität jedoch nicht nur mit der tatsächlichen, kleinen Datenbasis trainiert, sondern ebenfalls Aufblähungen mittels Data Augmentation vorgenommen. Diese ist bereits in den Kapiteln 3.4 und 4.3 beschrieben worden. Um dabei nicht nur einen Schluss über die allgemeine Nützlichkeit der Verwendung der Data Augmentation bezogen auf den vorliegenden Anwendungsfall und die eingesetzten Objekterkennungsdetektoren zu treffen, wird die Datenbasis in einem dritten Schritt noch mehr aufgebläht. Damit soll auch grob messbar werden, ob die größere Aufblähung einen positiven oder negativen Effekt auf die Modelle der Detektoren im Vergleich zur geringeren Aufblähung hat. Dies kann insbesondere die Einschätzung ihrer Verwendung im späteren Verlauf des LOEWE-Projekts vereinfachen.

Die durch die Cross Validation und die Data Augmentation generierten 15 Modelle eines jeden Detektors werden schließlich anhand ihrer mAP-Werte, Trainingsdauern und Vorhersagen auf den Validierungsdaten evaluiert. Da sich eine manuelle und genaue Evaluation der Vorhersagen aufgrund der hohen Anzahl an Objekten als schwierig gestaltet, wird sich dabei auf ein einziges, für Laien gut erkennbares Keramikgefäß fokussiert.

Zuletzt wird die dritte Teilaufgabe der Größenapproximation der Keramikgefäße vorgenommen. Dazu kommt die Gesamtlänge des Lineals nach Kapitel 4.3 zum Einsatz, welches für eine sinnige Größenapproximation äußerst gut lokalisiert sein muss. Dies gestaltet sich insbesondere im Hinblick auf nicht vollständig abgebildete Lineale der Herausforderung **H2** schwierig. Aufgrund des kaum tragbaren Mehraufwands anderer maschineller Herangehensweisen für nur insgesamt 16 Bilder, wird die Lokalisierung des Lineals an dieser Stelle durch die in der ersten Teilaufgabe sowieso umgesetzte Annotation vorgenommen. Die Annotationen der nicht vollständig abgebildeten Lineallängen der entsprechenden 16 Bilder werden dazu manuell erweitert. Schließlich kann die entsprechende Lineallänge in Pixeln pro Bild aus der Annotationsdatei gezogen werden. Anhand ihr kann ermittelt werden, wie viele Pixel den tatsächlichen 50 cm des Lineals entsprechen.

Ist nun jedoch gemäß Herausforderung **HL1** kein Lineal abgebildet, so wird die tatsächliche Dicke der auf jeder Fotografie vorhandenen Regalbretter Kapitel 4.3 folgend herangezogen. Sie kann anhand der anderen Fotografien über das abgebildete Lineal bestimmt werden. Ihre ebenfalls benötigte Dicke in Pixeln lässt sich mit manuell gesetzten Punkten ermitteln. Diese Punkte können während der Größenapproximation aktiv auf dem als Fenster aufpoppenden Bild gesetzt und verrechnet werden.

Um auch die Koordinaten der Keramikgefäße zu erhalten, tätigt der beste Objekterkennungsdetektor unter seiner besten Augmentationsstufe in Teilaufgabe 2 Bounding Box-Vorhersagen der Gefäße. Jene Koordinaten können letztlich mit denen des innerhalb der Fotografie ebenfalls abgebildeten Lineals oder Regalbretts verrechnet werden, um auch die tatsächliche Gesamtbreite und -höhe der entsprechenden Keramikgefäße zu ermitteln. Die erlangten Ergebnisse werden auf den entsprechenden Bildern sowie als CSV-Datei ausgegeben. In einem nächsten Schritt können sie anhand der durch das Berliner Museum für Vor- und Frühgeschichte bereitgestellten tatsächlichen Größen einiger Objekte evaluiert werden.

Schließlich wird das gesamte Konzept der Implementierung schematisch in Abbildung 4.2 dargestellt.

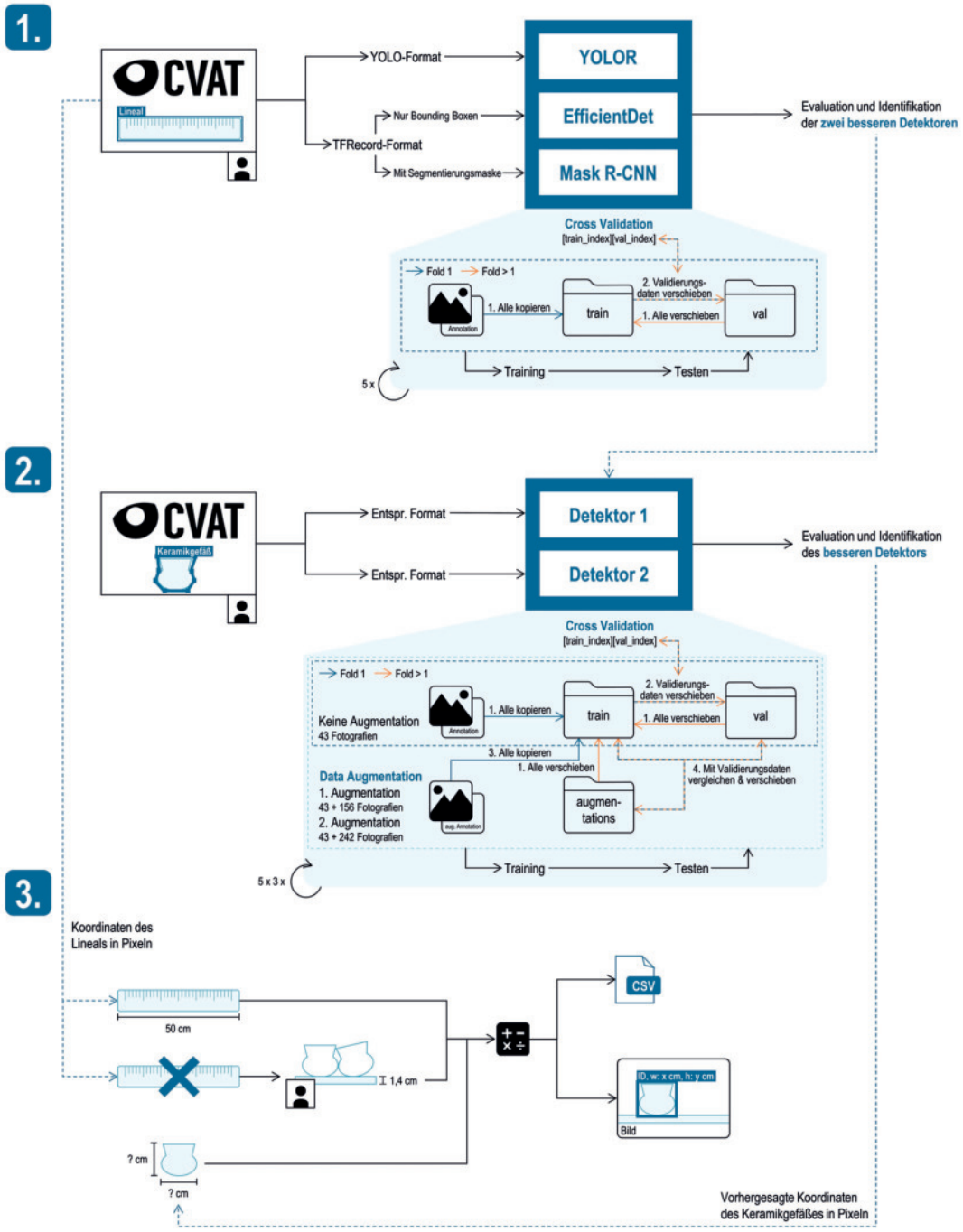

Abbildung 4.2: Schematische Darstellung des Implementierungskonzepts.

5
Implementierung und Evaluation

Im folgenden Kapitel wird die praktische Implementation zur Lösung der Aufgabenstellung 1.1 sowie ihre zum Fortschritt benötigte spezifische Evaluation anhand des ermittelten Konzepts 4.5 durchgeführt und erläutert. Dazu wird zunächst die zugrunde liegende Hardware beschrieben, welche die Implementierung bündelt. Anschließend werden sowohl die manuelle Annotation der als relevant erachteten Daten als auch die Cross Validation der gewählten Objekterkennungsdetektoren und ihre Ergebnisse im Hinblick auf die Teilaufgaben der Lokalisierung des Lineals aufgeführt. Ihren Ergebnissen folgend wird dasselbe mit den zwei besseren Detektoren auch für die Erkennung und Lokalisierung der Keramikgefäße unter Hinzunahme von Data Augmentation umgesetzt. Schließlich wird die Größenapproximation der Keramikgefäße anhand zuvor ermittelter Erkenntnisse vorgenommen und abgebildet. Zuletzt werden die in Kapitel 4.2 identifizierten Herausforderungen abschließend geprüft.

Die durchgeführte Implementierung kann anhand der URL innerhalb des Kapitels A.1 heruntergeladen werden. Etwaige erwähnte, hauptsächlich selbstgeschriebene Python-Skripte zur Unterstützung der Implementierung sind zudem in Tabelle A.3 zusammengefasst.

5.1 Zugrunde liegende Hardware

Zunächst wird in diesem Kapitel kurz auf die der Implementation zugrunde liegende Hardware eingegangen. Diese kann insbesondere bei der Einordnung oder Einschätzung ressourcenlastiger Tätigkeiten, wie der Trainingsdauer, auf der aktuellen oder auch abweichenden Hardware unterstützen.

Ein von dem Big Data Lab der Goethe-Universität bereitgestellter Rechner mit Betriebssystem Ubuntu 22.04 soll dabei die Implementation bündeln und eine Evaluation unter möglichst gleichen Bedingungen ermöglichen. Der Rechner verfügt über 32 GB RAM und eine NVIDIA GeForce RTX 3090 GPU über 24 GB. Um jene GPU zu utilisieren ist außerdem CUDA 11.2 und cuDNN 8.2.1 installiert. Die tatsächliche Implementation wird schließlich mittels Anaconda 4.12.0 umgesetzt, um etwaige Inkompatibilitäten der zu verwendenden Libraries leichter behandeln zu können.

In den entsprechenden Anaconda-Environments installierte Packages oder Libraries können als `yaml`-Datei der bereitgestellten Implementierung über die URL innerhalb des Kapitels A.1 nachempfunden werden. Spezifische vorgenommene Änderungen werden in den folgenden Kapiteln behandelt.

5.2 Implementierung der Cross Validation

Zur Ermöglichung eines besseren Vergleichs der Objekterkennungsdetektoren wird zunächst für alle Trainingsphasen von Machine Learning-Modellen die Cross Validation implementiert. Zum Einsatz kommt dazu primär das Modul `KFold` von *sklearn*[1] und eine automatisch generierte Liste der Dateinamen der entsprechenden Bilder sowie Annotationen innerhalb des Ordners `all_images`.

Zunächst wird eine `KFold`-Instanz mit den Parametern `n_splits = 5` für die Anzahl der Folds und damit einem ungefähren Verhältnis der Trainings- und Validierungsmenge von 80:20 sowie `Shuffle = True` zur tatsächlichen Randomisierung der Datenelemente generiert. Diese Anzahl an Folds ist sowohl für die Lokalisierung des Lineals als auch der Erkennung und Lokalisierung der Keramikgefäße gültig. Mit der Methode `split`, die die erwähnte Liste der Bildnamen als Parameter entgegennimmt, und einer `for`-Schleife können nun die entsprechenden partitionierten Indizes für die Trainings- und Validierungsdaten in Form von Listen zum Einsatz kommen. Ist eine neue K-Fold-Partition nicht nötig, zum Beispiel, weil eine bereits bekannte verwendet werden soll oder aber weil manuelle Änderungen vorgenommen werden müssen, so können entsprechende Indizes für die Trainings- und Validierungsmenge pro Fold in die Liste `split_into` eingetragen werden. Diese können anschließend über eine einfache `for`-Schleife über die Folds abgerufen werden.

Mit den Indizes der Trainings- und Validierungsdaten einer Fold können schließlich entsprechende Bilder und Annotationen automatisiert in `train`- und `val`-Ordner gepackt werden.

Handelt es sich um die erste Fold, werden die Ausgangsordner innerhalb des Ordners `all_images` aller Bild- und Annotationsdaten zunächst mittels `os.popen` und dem Linux-Befehl `cp -r` in den `train`-Ordner kopiert. Anschließend wird eine `for`-Schleife über jeden Validierungsindex des Folds mit den entsprechenden Listen der Bild- sowie Annotationsdateinamen eine jede Datei ausfindig machen und sie von `train` nach `val` verschieben.

Für alle anderen Folds > 1 werden schließlich alle Bild- und Annotationsdateien von `val` in `train` verschoben, dann wieder über den Validierungsindex des Folds entspre-

[1] `sklearn.model_selection.KFold`, `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html`, besucht am 22.08.2022

chender Dateien ausfindig gemacht und von `train` nach `val` geschoben. Dieses Vorgehen kann der schematischen Darstellung des Implementierungskonzept der Abbildung 4.2 nachempfunden werden. Für das Verschieben der Validierungsbilder beider Fälle kann dazu auch der folgende Programmcode dienen.

```
for j in range(0, len(test_index)):
    # Get image and label file name out of list of all file names
    img = img_files[test_index[j]]
    label = label_files[test_index[j]]
    os.rename(TRAIN_DIR + '/images/' + img, VAL_DIR + '/images/' + img)
    os.rename(TRAIN_DIR + '/labels/' + label, VAL_DIR + '/labels/' + label)
```

Befinden sich damit alle Bilder und Annotationen in den entsprechenden Ordnern, so können der Trainings- als auch der Testbefehl des zu verwendenden Detektors durchgeführt und ein Zähler der Folds um eins erhöht werden. Außerdem wird durch den Testbefehl eine Textdatei erstellt, die den mAP eines jeden Modells einer Fold speichert. Wie genau dies für jeden Detektor umgesetzt wird, wird in den nächsten Kapiteln erläutert.

Schließlich kann anhand einer jenen Textdatei nach dem Training der Durchschnitt über alle eingetragenen mAP der Folds berechnet und ausgegeben werden.

5.3 Implementierung der Data Augmentation

Zur Erweiterung der Datenmenge und damit der möglichen Verbesserung der Objekterkennungsdetektoren im Bezug auf die Erkennung und Lokalisierung der Keramikgefäße, kommt die zuvor erwähnte Python-Library imgaug zum Einsatz. Diese ermöglicht neben der Augmentation der Bounding Box-Annotation insbesondere auch jene der Polygone. Mit dem selbstgeschriebenen Skript `augmentation_poly.py` wird sie implementiert.

Zunächst wird dazu die COCO-Annotationsdatei[2] geladen und kopiert. Ein beispielhafter Ausschnitt einer COCO-Annotationsdatei ist innerhalb des Kapitels A.3 im Appendix zu finden. Im nächsten Schritt gilt es, die Bild-ID, den Dateinamen und die vorliegenden Annotationen zu erhalten. Dies ist über zwei `for`-Schleifen über alle Einträge des Keys `images` und `annotations` der Annotationsdatei möglich. Mit der Bedingung, dass sowohl die Bild-ID innerhalb des `images`-Keys als auch des `annotations`-Keys übereinstimmen, können die entsprechenden Werte korrekt zugeordnet und in ein Dictionary pro Bild eingefügt werden. Da ein Bild mehrere annotierte Objekte enthalten kann, werden insbesondere die Annotationen in eine Liste aus mehreren Dictionaries gespeichert.

[2] Vgl. das Format des COCO-Exportformats [Con Ja].

Pro Annotation bestehen diese aus ihrer Klasse und ihrer Polygon-Annotation.

Dabei ist die Polygon-Annotation, die in der COCO-Annotationsdatei unter dem `segmentation`-Key innerhalb `annotations` zu finden ist, für spätere Zwecke weiter anzupassen. Anstatt einer Liste bestehend aus einzelnen, abwechselnden `x`- und `y`-Punkten, werden die Punkte zu Listen aus `x`- und `y`-Koordinaten gepaart.

Die bildspezifischen Dictionaries werden schließlich in eine Liste gespeichert, die daraufhin mittels `for`-Schleife durchgearbeitet wird. Jedes Bild wird anhand des eingefügten Dateinamens geladen und für jede zugehörige Annotation und Klasse wird eine `Polygon`-Instanz via imgaug erstellt. Alle `Polygon`-Instanzen werden in eine Liste gespeichert, welche mit den Größenmaßen des aktuellen Bildes anschließend eine `PolygonsOnImage`-Instanz generiert. Auf jene Instanz und das aktuelle Bild können dann die von imgaug selbstständig gestalt- und kombinierbaren Augmentoren [Jun20] angewandt werden. Welche im Rahmen der Thesis zum Einsatz kommen, ist in Kapitel 5.7.2 festgehalten. Dadurch ergibt sich sowohl das augmentierte Bild als auch alle zugehörigen augmentierten Polygon-Annotationen.

Mittels der Funktion `remove_out_of_image` werden schließlich jene augmentierten Annotationen entfernt, die komplett außerhalb eines Bildes liegen. Bleiben danach noch Annotationen übrig, so wird zunächst das augmentierte Bild mit `cv2.imwrite` unter neuem Dateinamen gespeichert. Für die augmentierten Annotationen ist etwas mehr Aufwand nötig:

Zunächst wird ein neuer Eintrag für den Key `images` der COCO-Annotationsdatei nach ihrem Muster vorbereitet und befüllt. Das generierte Dictionary wird dann in die zu Beginn kopierte COCO-Annotationsdatei geschrieben. Um auch die augmentierten Annotationen zu schreiben, muss jede Annotation einzeln betrachtet werden. Insbesondere müssen die Polygon-Annotationen zurück in ihre ursprüngliche Form von getrennten `x`- und `y`-Punkten übersetzt werden. Erst dann kann auch der Eintrag des Keys `annotations` als Dictionary nach dem originalen Muster der COCO-Annotationsdatei vorbereitet werden. Insbesondere die Bild-ID, wie auch die Annotations-ID wird dabei zunächst mit der `max`-Funktion über die entsprechenden Keys der COCO-Annotationsdatei ermittelt und anschließend pro Element um 1 inkrementiert.

Schließlich werden alle neuen Werte über `json.dump` in die zuvor kopierte COCO-Annotationsdatei eingefügt.

Mit dem selbstgeschriebenen Skript `seg_to_bbox.py` werden schließlich aus dem augmentierten `segmentation`-Key der neuen Annotationsdatei die Bounding Box-Werte für den `bbox`-Key einer jeden Annotation gezogen. Dazu werden pro Annotation die maximalen und minimalen x- und y-Punkte des `segmentation`-Keys ermittelt und anschließend mittels `min(x), min(y), max(x) - min(x)` und `max(y) - min(y)` in das

benötigte $[x, y, w, h]$-COCO-Format des `bbox`-Keys übersetzt.

Um die Augmentationen nun in einem zusätzlichen Schritt manuell zu überprüfen, wird die Funktion `cv2.polylines` zu dem Codeabschnitt hinzugefügt, bei welchem jede Annotation eines Bildes einzeln behandelt wird. Durch `cv2.polylines` werden die Annotationen auf das entsprechende Bild gezeichnet und anschließend mittels `cv2.imwrite` in einem neuen Ordner `check` gespeichert. Dies dient hauptsächlich zur Identifikation von Objekten, die nur noch zu einem kleinen Teil auf dem augmentierten Bild abgebildet und nicht mehr zu identifizieren sind. Ihre Annotation ragt dabei über den Bildrand hinaus. Jene Fälle werden manuell aus der neuen Annotationsdatei entfernt.

Zwar könnte die zuvor erwähnte Funktion `remove_out_of_image`, mit dem Parameter `partly` auf `True` gesetzt, dies ebenso automatisiert übernehmen, jedoch würden dann auch Objekte und ihre augmentierten Annotationen entfernt werden, welche nur zu einem sehr geringen Teil über den Bildrand hinaus gehen. Da diese Objekte teilweise jedoch noch gut zu erkennen sind und damit behalten werden können, wird auf die automatisierte Umsetzung verzichtet.

Stattdessen sind ihre Annotationen dem Bildrand entsprechend anzupassen, um als valide zu gelten. Dazu kommt das neue Skript `adapt_seg_bbox_values.py` zum Einsatz. Dieses passt die Werte der `segmentation`-Keys an.

Für die Punkte in dem `segmentation`-Key gilt dabei folgendes: Sind die Werte negativ, werden sie auf 0.0 gesetzt; sind `x`-Punkte, die über ihren geraden Index zu identifizieren sind, größer als die Bildbreite, werden sie auf die Bildbreite gesetzt; sind die `y`-Werte, die über ihren ungeraden Index zu identifizieren sind, größer als die Bildhöhe, werden sie auf die Bildhöhe gesetzt.

Des Weiteren werden Bilder, die zu stark augmentiert und damit unpassend geworden sind, manuell gelöscht. Mit dem bereits erwähnten Skript `adapt_annotation_folder.py` lassen sich die entsprechenden Einträge der neuen Annotationsdatei leicht auf sich tatsächlich im Ordner befindende Bilder reduzieren.

Einbindung in die Cross Validation

Schließlich sind auch die Daten der Augmentation in die Implementierung der Cross Validation aus Kapitel 5.2 einzubinden. Dies funktioniert größtenteils analog zu dem zuvor erwähnten Kapitel.

Zunächst wird der Ordner `augmentations`, der die Ordner `images`, `labels` und `backup` birgt, auf derselben Stufe wie die `train`- und `val`-Ordner generiert. Wie der in Kapitel 5.2 erstellte Ordner `all_images` enthält der Ordner `backup` alle neuen aug-

mentierten Bilder und Annotationen. Für Fold 1 werden dann analog mittels einer Liste aller augmentierten Dateinamen alle augmentierten Bilder und Annotationen von dem `backup`- in den `train`-Ordner kopiert.

Jedoch sollen augmentierte Bilder und Annotationen, deren originale Bilder und Annotationen in dem vorliegenden Fall zur Validierung verwendet werden, weder beim Training, noch bei der Validierung zum Einsatz kommen. Insbesondere bei Verwendung in der Trainingsphase können jene Daten bereits Informationen über die eigentlich noch ungesehenen Validierungsbilder übermitteln und damit die Genauigkeit des Detektors fälschlicherweise in die Höhe treiben. Jedoch sollten auch bei der Validierung lediglich originale Daten zum Einsatz kommen, um ein für den Anwendungsfall sinnvolles Ergebnis ermitteln zu können. Entsprechend gehören jene augmentierten Bilder und Annotationen wieder aussortiert.

Prinzipiell kann dies ähnlich zu Kapitel 5.2 über eine `for`-Schleife über die Liste der für die Fold identifizierten Indizes der originalen Testdaten geschehen. Allerdings besitzen die augmentierten Daten abgeänderte Dateinamen und können nicht so einfach verschoben werden. Die Dateinamen folgen der Regel, dass dem originalen Dateinamen ein Unterstrich und eine Identifikation der verwendeten Augmentoren-Kombination angehängt wird. So ist beispielsweise `TCD-3_flip` das horizontal gespiegelte Bild des originalen Bildes `TCD-3`. Anhand einer `if`-Abfrage, die alle Einträge der Liste der augmentierten Dateinamen mit den zur Validierung nötigen Einträge der Liste der tatsächlichen Dateinamen gegenprüft, können die passenden augmentierten Bilder und Annotationen schließlich von `train` nach `augmentations` verschoben werden. Relevant dabei ist insbesondere der Unterstrich, der den originalen Dateien im Rahmen der Überprüfung angehängt werden sollte, um Fehler zu vermeiden. Da einige Dateinamen echte Teilmengen anderer Dateinamen sind, wie `ECA-1` und `ECA-10`, werden fehlerhafte Verschiebungen von augmentierten Daten, deren originale Daten sich nicht in der Validierungsmenge befinden, aus dem `train`-Ordner heraus vermieden. Die Identifizierung und Verschiebung der Validierungsbilder dieses Falles wird mit dem folgenden Programmcode umgesetzt:

```
for aug_img, aug_label in zip(aug_img_files, aug_label_files):
    ...
    for j in range(0, len(test_index)):
        if os.path.splitext(img_files[test_index[j]])[0] + '_' in aug_img:
            os.rename(TRAIN_DIR + '/images/' + aug_img, AUG_DIR + '/images/'
                + aug_img)
            os.rename(TRAIN_DIR + '/labels/' + aug_label, AUG_DIR + '/labels/
                ' + aug_label)
```

Für Folds > 1 werden erneut analog zu Kapitel 5.2 alle augmentierten Dateien von dem

Ordner `augmentations` in den Ordner `train` verschoben. Ebenso werden jene augmentierten Daten, deren originale Daten für die aktuelle Fold zur Validierung verwendet werden, analog zum Vorgehen für Fold 1 identifiziert und in `augmentations` geschoben. Eine Darstellung des gesamten Vorgehens kann ebenfalls dem schematischen Vorgehen des Implementierungskonzepts der Abbildung 4.2 entnommen werden.

Schließlich befinden sich alle Daten in ihren korrekten Ordnern und das Training, sowie die Validierung und das Testen jeder Fold kann erfolgen.

5.4 Anwendung des YOLOR-Detektors

Zunächst werden in diesem Kapitel die für die Thesis wichtigsten Aspekte der Anwendung des einstufigen Detektors YOLOR aufgeführt. Dazu wird die praktische Implementierung der in Kapitel 3.6.2 beschriebenen Publikation von Wang et al. [WYL21] verwendet. Es wird sich dabei außerdem an dem Artikel *How to Train YOLOR on a Custom Dataset* [SS21] orientiert.

Installation

Die Grundlage des Detektors bildet Python 3.7.13, welches in der folgenden Installation Inkompatibilitäten mit den zu verwendenden Versionen von `Cython` sowie `pycocotools` vermeidet. Damit kann das entsprechende GitHub-Repository geklont und installiert werden. Zur korrekten Installation auf dem vorgegebenen Rechner sind `Torch` und `Torchvision` jedoch entgegen der mitgelieferten Datei `requirements.txt` mit dem folgenden Kommando für eine fehlerfreie Verwendung zu installieren:

```
pip install torch==1.7.1+cu110 torchvision==0.8.2+cu110 -f https://download.
    pytorch.org/whl/torch_stable.html
```

Um auch unmittelbar vor dem Trainingsende keinen Error zu enthalten, muss innerhalb der geklonten Datei `/yolor/utils/plots.py` nach Zeile 97 `pred = pred.cpu()` in die Schleife eingefügt werden [Wan21a].

Wang et al. [WYL21] stellen überdies die vortrainierten Weights YOLOR-P6[3] und YO-LOR-W6[4] zur Verfügung, von denen das zu trainierende Modell profitieren kann. Dabei wird sich dem Artikel [SS21] folgend und aufgrund des durch die Autor*innen übermittelten Unterschieds von weniger als 1 % ihres APs für YOLOR-P6 entschieden.

[3] YOLOR-P6 Weights, `https://drive.google.com/uc?export=download&id=1Tdn3yqpZ79X7R1Ql0zNlNScB1Dv9Fp76`, besucht am 14.06.2022

[4] YOLOR-W6 Weights, `https://drive.google.com/uc?export=download&id=1UflcHlN5ERPdhahMivQYCbWWw7d2wY7U`, besucht am 14.06.2022

Entsprechend diesem dient nun die geklonte Konfigurationsdatei `/yolor_p6.cfg` der tatsächlichen Konfigurationsdatei des Anwendungsfalles der Thesis als Vorlage.

Training

Um ein YOLOR-Modell zu trainieren, müssen zunächst entsprechende Bilder sowie ihre Annotationen im YOLO-Format[5] als Textdateien zum Training und zur Validierung zur Verfügung gestellt werden. Ein beispielhafter Auszug einer Annotation des YOLO-Exportformats kann dem Kapitel A.3 im Appendix nachempfunden werden. Der Pfad des Ordners, in dem die Dateien gespeichert werden, wird dann in der Datei `data.yaml` angepasst. Ebenso müssen die Anzahl der Klassen innerhalb `nc` und die Klassenlabel als Liste innerhalb `names` in der Datei angegeben werden. In der Datei `data.names` gilt es zusätzlich, die Namen der Klassen erneut mit einer Zeile pro Klasse aufzuführen.

Des Weiteren muss die Konfigurationsdatei `/yolor_p6.cfg` dem YOLOR vorgegangenen Detektor YOLOv4 folgend[6] angepasst werden. Zunächst ist an vier Stellen die Variable `classes` zu verändern, deren Wert aus der zuvor generierten Datei `data.yaml` erhalten werden kann. Auch die Variable `filters` unmittelbar über `classes` im Bereich `convolutional` ist anhand der Formel $\texttt{filters} = (\texttt{classes} + 5) \cdot 3$ an vier Stellen anzupassen. Andere `filters`-Variablen müssen dabei unberührt bleiben. `max_batches` ergibt sich ebenfalls anhand der Klassen durch die Formel $\texttt{max\_batches} = \texttt{classes} \cdot 2\,000$, wobei jedoch $\texttt{max\_batches} \geq 6\,000$ gilt. Schließlich ergibt sich `steps` aus $\texttt{steps} = \texttt{max\_batches} \cdot 0,8, \texttt{max\_batches} \cdot 0,9$.

Anhand des in Artikel [SS21] aufgeführten Jupyter Notebooks werden insbesondere `classes` und `filters` automatisiert über der Funktion `writetemplate(line, cell)` mittels *cell magics* von IPython nach Schiele [Sch21] umgesetzt. Die oben erwähnten Formeln für `max_batches` sowie `steps` werden der Vollständigkeit halber an den entsprechenden Stellen ergänzt. Damit kann die Konfigurationsdatei abhängig von den vorliegenden Daten automatisch und ohne typische Fehlerquellen, wie zum Beispiel Tippfehler, generiert werden.

Erst dann kann das Training mittels der Datei `/yolor/train.py` und übergebenen Parametern wie folgt begonnen werden. Im Falle der Cross Validation wird insbesondere der Name des Modells gleich der aktuellen Fold gesetzt.

```
python /yolor/train.py --batch-size {BATCHGRÖßE} --epochs {EPOCHENZAHL} --
    data {PFAD ZU DATA.YAML} --cfg {PFAD ZU YOLOR_P6.CFG} --weights {PFAD ZU
     YOLOR_P6.PT} --project {PFAD ZUM SPEICHERORT DES MODELLS} --name {NAME
    DES MODELLS} --hyp ./yolor/data/hyp.scratch.1280.yaml --device 0
```

[5] Vgl. das Format des YOLO-Exportformats anhand Pokhrel [Pok20].
[6] Vgl. die Einstellungen der YOLOv4-Konfigurationsdatei nach Bochkovskiy [Boc22].

Dabei behält YOLOR dem Autor Wang [WYL21] folgend[7] die Seitenverhältnisse eines jeden Bildes bei. Damit muss kein Bild für das Training verkleinert oder quadratisch zugeschnitten werden und es werden keine negativen Einflüsse durch nicht genau auf den Detektor angepasste Inputbilder generiert.

Testen

Zum Testen eines YOLOR-Modells werden etwaige mAP-Werte über die Validierungsbilder mittels `/yolor/test.py` wie folgt ermittelt. Dazu sollten nun anstelle der vortrainierten Weights die `best_overall.pt`-Weights[8] innerhalb des `weights`-Ordners des entsprechenden neu trainierten YOLOR-Modells gewählt werden.

```
python /yolor/test.py --data {PFAD ZU DATA.YAML} --cfg {PFAD ZU YOLOR_P6.CFG
    } --weights {PFAD ZU BEST_OVERALL.PT DES TRAINIERTEN MODELLS} --names {
    PFAD ZU DATA.NAMES} --project {PFAD ZUM SPEICHERORT DES MODELLS} --name
    {NAME DES MODELLS} --device 0 --save-map
```

Optional könnte auch unter anderem die Flag `conf` gesetzt werden, die angibt, wie sicher der Detektor bezüglich einer Vorhersage sein soll, um sie zu zählen. Per Default ist jener Wert auf 0,001 gesetzt, was an dieser Stelle beibehalten wird, um ein vollumfängliches Bild der getesteten Detektionen zu erlangen.

Zusätzlich wird die Flag `save-map` neu entwickelt, durch welche der mAP in einer Textdatei gespeichert wird. Mit ihr kann pro Fold der Cross Validation die Textdatei mit ihrem mAP aktualisiert werden. Dazu ist lediglich notwendig, an allen Positionen, an denen die anderen Flags innerhalb `/yolor/test.py` gesetzt werden, die neue `save-map`-Flag einzufügen sowie, sollte sie `True` sein, den aktuellen mAP innerhalb der Variable `map` in die Textdatei `map.txt` einzufügen. Schließlich können nach dem Training einer jeden Fold jene in der Textdatei gespeicherten mAP-Werte zum Berechnen des Durchschnitt-mAP der gesamten Cross Validation verwendet werden.

Vorhersage der Bounding Boxen

Um nun auch aktiv Ergebnisse begutachten zu können, können Objekte auf Bildern lokalisiert und klassifiziert werden, in dem für das Objekt eine Bounding Box gezogen und ein Klassenlabel hinzugefügt wird. Dafür wird `/yolor/detect.py` wie folgt verwendet. An dieser Stelle sollte insbesondere die Confidence des Modells für Vorhersagen

[7] Vgl. einen GitHub-Beitrag von Wang [Wan21b], einem der Schöpfer*innen von YOLOR.

[8] Für YOLOR existieren auch andere Weights, wie zum Beispiel `best_ap50.pt`. Für den Zweck der Thesis wird sich jedoch für `best_overall.pt` entschieden, um die tatsächlich allumfassend beste Performanz erhalten zu können.

über die `conf`-Flag auf einen höheren Wert gesetzt werden, da ansonsten zu viele Bounding Boxen generiert werden, was ihre korrekte Einschätzung für das menschliche Auge erschwert.

```
python /yolor/detect.py --source {PFAD ZU DEN VALIDIERUNGSBILDERN} --cfg {
    PFAD ZU CONFIG.CFG} --weights {PFAD ZU BEST_OVERALL.PT DES TRAINIERTEN
    MODELLS} --output {PFAD ZUM SPEICHERORT DER GELABELTEN BILDER} --names {
    PFAD ZU DATA.NAMES} --conf {CONFIDENCE} --device 0
```

Zusätzlich können unter anderem Flags wie `save-txt` hinzugefügt werden, wodurch neben den durch das Modell gelabelten Bildern auch die vorhergesagten Bounding Box-Koordinaten im YOLO-Format als Textdatei gespeichert werden.

5.5 Anwendung des EfficientDet- und Mask R-CNN-Detektors

Zur praktischen Einsetzung des einstufigen Detektors EfficientDet [TPL20] aus Kapitel 3.6.2 und des zweistufigen Detektors Mask R-CNN [HGDG17] aus Kapitel 3.6.1 wird die TensorFlow 2 Object Detection API verwendet. Diese Entscheidung ist bereits durch das Implementierungskonzept 4.5 motiviert worden. Im folgenden Kapitel werden die wichtigsten Aspekte ihrer Verwendung aufgeführt. Orientiert wird sich dabei insbesondere an dem TensorFlow 2 Object Detection Tutorial [Vla20] und dem Artikel *How to Train a TensorFlow 2 Object Detection Model* [Sol20].

Installation

Zunächst werden für die TensorFlow 2 Object Detection API Python 3.8.13 sowie TensorFlow 2.8.0 installiert. Letzteres kann dabei auch mit `pip` installiert werden. Anschließend wird dem Tutorial [Vla20] folgend das entsprechende TensorFlow *Models* GitHub-Repository[9] geklont, Protobuf-Libraries heruntergeladen und kompiliert sowie die Object Detection API installiert.

Vortrainierte Weights sind im TensorFlow 2 Detection Model Zoo[10] aufgeführt und können hier oder über Code mitsamt ihrer Konfigurationsdatei heruntergeladen werden. Für Mask-RCNN existiert zum Zeitpunkt der Thesis lediglich das Modell `Mask R-CNN Inception ResNet V2 1024x1024`, wohingegen verschiedene Versionen von D0 bis D7 von EfficientDet mit unterschiedlichen Bildgrößen existieren. Hierbei wird sich

[9] TensorFlow *Models*, `https://github.com/tensorflow/models`, besucht am 29.05.2022

[10] TensorFlow 2 Detection Model Zoo, `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md`, besucht am 29.05.2022

für `EfficientDet D4 1024x1024` entschieden, da höhere Versionen aufgrund der hohen benötigten Ressourcen zu Problemen bei der Ausführung geführt haben und damit die gleiche Größe der Inputbilder wie von Mask R-CNN gewählt werden kann.

Training

Zum Trainieren von EfficientDet- oder Mask-RCNN-Modellen mittels der TensorFlow 2 Object Detection API müssen entsprechende Bilder und ihre Annotationen als `tfrecord`-Dateien zum Training und zur Validierung zur Verfügung gestellt werden. Zusätzlich muss eine Datei `label_map.pbtxt` erstellt werden, die für jede vorherzusagende Klasse deren ID und Name im folgenden Format enthalten soll.

```
item {
    id: {ID}
    name: {KLASSENNAME}
}
```

Schließlich muss auch die Konfigurationsdatei erstellt werden. Dazu kann die `pipeline.config`-Datei des heruntergeladenen vortrainierten Modells kopiert und an entsprechenden Stellen angepasst werden. Zwar unterscheidet sich die Konfigurationsdatei beider Modelle, die vorzunehmenden Änderungen jedoch kaum.

Zunächst ist `num_classes` an die Anzahl an Klassen anzupassen. Diese können aus der Datei `label_map.pbtxt` gezogen werden. Sowohl EfficientDet als auch Mask R-CNN sollten den `image_resizer` `keep_aspect_ratio_resizer` mit entsprechenden `min_dimension: 1024`, `max_dimension: 1024` und `pad_to_max_dimension: true` vorweisen. Damit wird das Seitenverhältnis eines jeden Inputbildes beibehalten und keine Verzerrungen vorgenommen, die die Ergebnisse des Modells negativ beeinflussen können.

`batch_size` innerhalb `train_config` muss auf einen für den Rechner tragbaren Wert angepasst werden, was im Falle der Thesis 1 ist. Zur Vermeidung von `NaN`-Werten während des Trainings sollte die `learning_rate_base` auf einen Wert um die 0,0012 gesetzt werden. Überdies sind die Werte `num_steps` an die Wünsche der Anzahl an Trainingsschritten des Modells anzupassen. `fine_tune_checkpoint_type` ist auf `‘‘detection’’`, `fine_tune_checkpoint_version` auf `V2` und, um nicht mit TPU zu trainieren, `use_bfloat16` auf `false` zu setzen.

Zuletzt sind Pfade zu bestimmten Dateien zu vermerken. `fine_tune_checkpoint` sollte auf den Pfad `/checkpoint/ckpt-0` des entsprechenden vortrainierten Modells verweisen. Sowohl für `train_input_reader` als auch für `eval_input_reader` ist der Pfad zur `label_map.pbtxt` für den Eintrag `label_map_path` zu setzen. Zusätzlich ist der Pfad zu den entsprechenden Inputbildern und Annotationen als `tfrecord`s zum Trainieren

oder Validieren für den Eintrag `input_path` zu setzen. Für `input_path` sollte dabei lediglich der Ordner, in dem die `tfrecord`-Dateien liegen angegeben werden und alle in ihm liegenden Dateien mittels {`PFAD ZU TFRECORDS`}`/*` inkludiert werden.

Insbesondere im Falle von Mask R-CNN sollte außerdem der Eintrag `num_parallel_batches: 1` ergänzt werden, um vermehrte Abstürze beim Training zu vermeiden.

Jedwede Änderungen lassen sich schließlich analog zu Kapitel 5.4 mit der Funktion `writetemplate(line, cell)` und *cell magics* von IPython automatisiert umsetzen. Schließlich kann das Training mit dem folgenden Befehl begonnen werden:

```
python /models/research/object_detection/model_main_tf2.py --pipeline_config
     _path={PFAD ZU PIPELINE.CONFIG} --model_dir={PFAD ZUM SPEICHERORT DES
    MODELLS} --num_eval_steps={ANZAHL SCHRITTE BIS EVALUATION}
```

Testen

Um die Genauigkeit der Modelle zu evaluieren, können unterschiedliche mAP- sowie AP- und AR-Werte über die Validierungsbilder mit dem folgenden Befehl ermittelt werden. Dabei müssen insbesondere die Flags `model_dir` und `checkpoint_dir` identisch sein.

```
python /models/research/object_detection/model_main_tf2.py --pipeline_config
     _path={PFAD ZU PIPELINE.CONFIG} --model_dir={PFAD ZUM MODELL} --check
    point_dir={PFAD ZUM MODELL}
```

Wird jener Befehl während der Trainingsdauer ausgeführt, so werden für jeden Evaluationsschritt neben den üblichen `loss`-Metriken auch die anderen Evaluationsmetriken, wie die mAP-Werte gespeichert, welche später mit TensorBoard visualisiert werden können. Bei Modellen, die viele Ressourcen benötigen, wie zum Beispiel Mask R-CNN, bietet sich dies jedoch nicht an, da es damit vermehrt zu Abstürzen während des Trainings kommen kann.

Darüber hinaus müssen für das Testen von Mask R-CNN, insbesondere während des Trainings, einige Änderungen an Dateien vorgenommen werden, um den sonst auftretenden Fehler `ValueError: Category stats do not exist` zu vermeiden.

Zunächst ist Zeile 240 bis 244 der von GitHub-Benutzer*in *pialin* angepassten Datei `coco_tools.py` nach Zeile 287 in der gleichnamigen Datei des Packages `object_detection` im generierten Anaconda-Environment einzufügen. Ebenso ist Zeile 499 bis 648 der Datei `cocoeval.py` derselben Benutzer*in nach Zeile 496 in der gleichnamigen Datei des Packages `pycocotools` im generierten Anaconda-Environment hinzuzufügen. [Pia18]

Um auch im Falle von EfficientDet und Mask R-CNN die entsprechenden mAP-Werte jeder Fold der Cross Validation in einer Textdatei zu speichern, wird außerdem die Datei `/models/research/object_detection/model_lib_v2.py` angepasst. Dazu wird bei jedem Aufrufen von `model_lib_v2.py` der Wert innerhalb `eval_metrics['Detection Boxes_Precision/mAP']` in die Textdatei `map.txt` geschrieben. Jene gespeicherten Werte können schließlich nach dem Training zur Berechnung des durchschnittlichen mAP der Cross Validation verwendet werden.

Vorhersage der Bounding Boxen

Zur Betrachtung tatsächlicher Vorhersagen von Objekten auf Bildern muss zunächst das entsprechende Modell über den folgenden Befehl als Inference Graph exportiert werden.

```
python /models/research/object_detection/exporter_main_v2.py --pipeline_con
    fig_path={PFAD ZU PIPELINE.CONFIG} --trained_checkpoint_dir={PFAD ZUM
    MODELL} --output_directory={PFAD ZUM SPEICHERORT DES GRAPHEN} --input_
    type image_tensor
```

Dann kommt die Datei `inference_object_detection.ipynb` auf Grundlage des offiziellen Object Detection Tutorials von TensorFlow [GRH+21] unter Verwendung der `pycocotools`-Library zum Einsatz. Dabei wird der gespeicherte Inference Graph des Modells mittels der Funktion `load_model` geladen.

Weiterhin werden auch die Pfade zu `label_map.pbtxt` zur Übersetzung der vorhergesagten IDs in Klassennamen, und zu dem Ordner mit Validierungsbildern definiert, sodass schließlich der Pfad eines jeden Bildes verwendbar wird. Mittels einer Schleife über jedem Bild wird die Funktion `show_inference` aufgerufen, welche wiederum die Funktion `run_inference_for_single_image` aufruft. Im Falle von Mask R-CNN muss die Funktion `run_mrcnn_inference_for_single_image` ergänzt und statt `run_inference_for_single_image` verwendet werden, um einen `Index out of range`-Error zu vermeiden [Wei20].

Beide zuletzt erwähnten Funktionen generieren gleichermaßen die Vorhersagen der Bounding Boxen des geladenen Modells, wohingegen `show_inference` ihre Visualisierung umsetzt und die entsprechenden Bilder direkt im Notebook anzeigt. Innerhalb der zuletzt erwähnten Funktion kann außerdem der Parameter `min_score_threshold` für die Funktion `visualize_boxes_and_labels_on_image_array` festgelegt werden, um nur Vorhersagen von Bounding Boxen mit einer bestimmten Confidence-Höhe anzuzeigen. Zusätzlich kann auch der Parameter `instance_masks=None` ergänzt werden, um im Falle von Mask R-CNN keine Masken anzuzeigen. Da sowohl YOLOR als auch EfficientDet

keine Masken generieren, können sie auch für eine gleichwertige Gegenüberstellung mit Mask R-CNN ignoriert werden.

Im Code ergänzt wird außerdem, dass `show_inference` die Bilder mit vorhergesagten Bounding Boxen über die Methoden `fromarray` und `save` des `PIL`-Moduls `Image` in einem Ordner speichert.

5.6 Lokalisierung des Lineals

Die Implementierung der Lokalisierung des Lineals über Objekterkennungsdetektoren wird in folgendem Kapitel behandelt. Zunächst werden dazu die relevanten Daten sowie deren Annotation und Distribution anhand der Cross Validation beschrieben. Danach folgen das Training und die Ergebnisse sowie die spezifische Evaluation der Lokalisierung mit den Detektoren YOLOR, EfficientDet und Mask R-CNN. Schließlich werden die Detektoren einander gegenübergestellt, um die beiden besseren Detektoren für die nächste Teilaufgabe zu identifizieren.

5.6.1 Annotation und Distribution der Daten

Um die Lokalisierung des Lineals dem Kapitel 4.5 folgend mit möglichst hoher Genauigkeit zu vollziehen, werden zunächst Scans und Kameraaufnahmen, die Lineale abbilden, annotiert. Dabei wird sich überdies nicht nur auf Keramikgefäßsammlungen mit Lineal fokussiert, sondern auch auf ähnliche Sammlungen von Statuen mit Lineal. Dateien mit äußerst schlecht zu erkennendem Lineal werden dabei aussortiert.

Durch die Unterstützung einer studentischen Hilfskraft der Informatik werden schließlich 134 Scans und Kameraaufnahmen über die Online-Oberfläche des Annotationstools CVAT mit Bounding Boxen annotiert. Dabei werden die Bounding Boxen möglichst eng um die Lineale gezogen und überlappende Objekte ignoriert, um pro Bild nur eine einzelne Box zu ziehen. Wie erwartet ist der gesamte Annotationsprozess innerhalb weniger Stunden durchzuführen.

Tatsächlich zum Einsatz für die Objekterkennungsdetektoren kommen jedoch im Endeffekt nur die Scans der Keramiksammlungen mit Lineal. Jene Limitation ist angelehnt an die Dateien der Identifizierung und Lokalisierung der Keramikgefäße. Dieser Umstand ist bereits in Kapitel 4.5 erläutert worden. Damit bleiben 37 annotierte Scans. Von diesen werden jeweils ca. 80 % (29 oder 30 Scans) zum Trainieren und 20 % (7 oder 8 Scans) zum Validieren der Modelle der einzelnen Cross Validation Folds verwendet.

Abbildung 5.1 zeigt die Ausrichtung der Lineale innerhalb der Fotografien der Validierungsmenge pro Fold und ermöglicht damit einen besseren Überblick über die tatsächlichen Datenelemente der Folds und deren spätere Ergebnisse, aber auch der verwendeten

Trainingsdaten. Diese sind innerhalb der Abbildung durch alle Folds außer der aktuell betrachteten dargestellt.

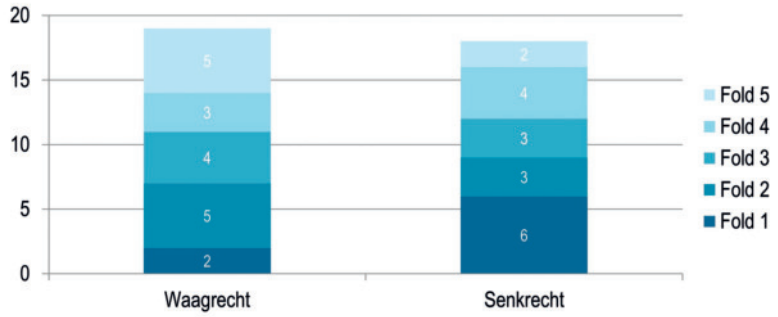

Abbildung 5.1: Gliederung der Lineale der Validierungsbilder in waagrechte und senkrechte Ausrichtungen pro Cross Validation Fold.

5.6.2 Lokalisierung mit YOLOR

In diesem Kapitel wird die Lineallokalisierung mittels des in Kapitel 3.6.2 beschriebenen, einstufigen Objekterkennungsdetektors YOLOR [WYL21] realisiert. Dazu werden die in dem Kapitel 5.4 herausgearbeiteten Aspekte verwendet.

Generierung der detektorspezifischen Inputdaten

Wie vorgegangene Detektoren der YOLO-Familie benötigt YOLOR die Annotationsdateien im YOLO-Exportformat. Dabei wird für jedes Bild eine Textdatei generiert, in der sich in jeder Zeile die entsprechende Annotation in Form von der ID der Klasse und den Bounding Box-Koordinaten im YOLO-Format befindet. Das Annotationstool CVAT ermöglicht den Export der manuell bewältigten Annotation im YOLO-Exportformat über wenige Klicks. Die Textdateien und Bilder werden schließlich in den Ordner `all_images` eingefügt, von wo aus sie nach Kapitel 5.2 automatisiert durch die Cross Validation in die zur Fold passende `train`- und `val`-Ordner eingefügt werden.

Anstelle der von CVAT generierten Datei `obj.data`, die Informationen und Pfade der Daten enthält, wird von YOLOR jedoch die `yaml`-Datei `data.yaml` benötigt, die bereits in Kapitel 5.4 erwähnt worden ist. Die von CVAT generierte Datei `train.txt`, die die Pfade eines jeden Bildes enthält, kann damit gelöscht werden. Überdies wird die Datei `obj.names`, die die einzelnen Klassennamen pro Spalte enthält (im vorliegenden Fall lediglich *ruler*) zum besseren Verständnis in `data.names` umbenannt.

Deskription und Evaluation der Ergebnisse

Schließlich werden fünf YOLOR-Modelle unter Verwendung der fünf Folds der Cross Validation über je 1 500 Epochen trainiert. Die Ergebnisse in Form der mAP-Werte und der Trainingsdauer der einzelnen Modelle sowie deren Durchschnittswert, werden in Tabelle 5.1 aufgeführt. Der durchschnittliche mAP beträgt 0,304 bei einer durchschnittlichen Trainingsdauer von rund 7 h. Insbesondere die erste Fold erreicht dabei den besten mAP von 0,442 und die fünfte Fold den schlechtesten mAP von 0,111. Ein möglicher Erklärungsversuch diesbezüglich ergibt sich durch das Balkendiagramm der Abbildung 5.1, welches die Ausrichtung der Lineale innerhalb der Validierungsmenge darstellt. So sind fünf der sieben Lineale der Validierungsbilder der fünften Fold waagrecht angeordnet, jedoch nur zwei der acht Lineale der Validierungsbilder der ersten Fold. Damit liegt der Schluss nahe, dass waagrecht angeordnete Lineale von YOLOR schlechter erkannt werden. Dies weist auf Herausforderung **HG5** hin. Grund dafür liefert bereits die Anordnung der Objekte in Regalen innerhalb der Fotografien. Die waagrecht angeordneten Regalbretter ähneln **HL4** sowie teilweise **HG2** folgend insbesondere waagrecht angeordneten Linealen aufgrund ihrer Dicke sowie Farbe stark.

YOLOR		
Fold	**mAP**	**Trainingsdauer**
1	0,442	7 h 9 min
2	0,356	6 h 53 min
3	0,253	8 h 18 min
4	0,359	7 h 9 min
5	0,111	6 h 2 min
ø	**0,304**	**7 h 6 min**

Tabelle 5.1: Ergebnisse der Lineallokalisierung der YOLOR-Modelle auf Basis der Validierungsdaten pro Cross Validation Fold.

Werden nun auch mögliche Lineale der Validierungsmenge von den Modellen vorhergesagt und damit entsprechende Bounding Boxen auf den Bildern gezogen, bestätigt sich die Annahme bei einer betrachteten Confidence von 3 %. Alle Modelle sind nicht gut darin, waagrecht angeordnete Lineale zu erkennen und verwechseln sie häufig mit Regalbrettern. Dies ist jedoch nicht die einzige Problematik. Häufig werden auch kleine Ausschnitte der Wand hinter oder des Bodens unter den Regalen am Bildrand fälschlicherweise als Lineal identifiziert. Durch ihre Farbe und Form, die durch den Bild- und Regalrand rechteckig erscheint, liegt auch hier eine hohe Ähnlichkeit zu waagrechten und senkrechten Linealen vor, wodurch **HG2** angenommen werden kann. Dies tritt bei allen Folds bis auf der vierten auf, innerhalb welcher weniger derartige Motive vorliegen. Zusätzlich kann auch eine zu hohe oder zu niedrige Belichtung der Bilder nach Heraus-

forderung **HG3** dazu führen, dass sich das Lineal nicht mehr gut vom Rest des Bildes abhebt und damit schwerer zu erkennen ist. Dies führt auch zu vermehrten falschen Lineallokalisierungen. Gleichermaßen können auch weiterhin, **HL4** entsprechend, andere Objekte aufgrund ihrer Farbe und länglichen Form eine Ähnlichkeit zu Linealen aufweisen und verwechselt werden, beispielsweise innerhalb Fold 3 und Fold 5.

Erhöht man nun den Schwellwert der Confidence des Modells bezüglich seiner zu behaltenden Vorhersagen, fallen viele der falschen Vorhersagen aufgrund ihrer geringen Confidence zwar weg, gleichermaßen aber auch viele der korrekten Vorhersagen. Diesen schreiben die Modelle nämlich typischerweise ebenfalls häufig keine hohe Confidence zu.

Eine weitere Schwierigkeit wird geschaffen, wenn große Teile eines Lineals vom Bildrand abgeschnitten und damit schwerer zu lokalisieren sind. Dies ist von Herausforderung **HL2** festgehalten und insbesondere bei der fünften Fold der Fall. Damit wird die fünfte Fold außerdem von der zweiten Fold abgehoben, die zwar auch ganze fünf von acht Validierungsbilder mit waagrechten Linealen enthält, dafür allerdings weniger Bilder mit zu hoher oder niedriger Belichtung und zu stark abgeschnittenen Linealen.

Abbildung 5.2 zeigt zwei der Fotografien mit den Vorhersagen zweier YOLOR-Modelle der entsprechenden Folds. Sie bildet die zuvor erwähnten Problematiken der Fotografien gut ab.

Abbildung 5.2: Vorhersagen der YOLOR-Modelle zur Lineallokalisierung auf den Validierungsbildern `TCD-1` und `PG-5`[11].

5.6.3 Lokalisierung mit EfficientDet

Auch mit dem einstufigen Detektor EfficientDet [TPL20] aus Kapitel 3.6.2 wird die Lineallokalisierung im Folgenden praktisch auf Basis von der TensorFlow 2 Object Detec-

[11] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 9061 (`TCD-1`) und 8364 (`PG-5`).

tion API realisiert. Dazu werden die in Kapitel 5.5 herausgearbeiteten Vorgehensweisen angewandt.

Generierung der detektorspezifischen Inputdaten

Zur Benutzung der TensorFlow 2 Object Detection API müssen die einzuspeisenden Daten als `tfrecord`s vorliegen. Auch wenn dies ein mögliches Exportformat von CVAT darstellt, so ermöglichen jene exportierten `tfrecord`s keine einfache Unterteilung mehr in Trainings- und Validierungsmenge sowie die Exklusion der zusätzlich annotierten zu entfernenden Kameraaufnahmen. Die CVAT-Weboberfläche bietet überdies keine Möglichkeit, die Bilder und Annotationen im Nachhinein noch in entsprechende Mengen anzuordnen – dies ist nur im Vorfeld über verschiedene Tasks möglich. Dementsprechend werden die getätigten Annotationen der Lineale im COCO-Exportformat heruntergeladen.

Anhand des geklonten Skripts `/models/research/object_detection/dataset_tools/create_coco_tf_record.py`[12] der TensorFlow 2 Object Detection API lassen sich aus den COCO-Annotationen `tfrecord`-Bruchstücke generieren. Der Einfachheit halber werden der `create_coco_tf_record.py`-Datei dazu drei Flags hinzugefügt, die die Anzahl der Bilder pro Menge angeben und später die Anzahl der generierten Bruchstücke der Funktion `_create_tf_record_from_coco_annotation` bestimmen. Per Default sind jene auf 100 für die Trainingsdaten und 50 für Validierungs- sowie Testdaten festgelegt worden, was im Falle der Masterthesis zu Fehlern geführt hat.

Nachdem die `tfrecord`-Bruchstücke aus der COCO-Annotationsdatei generiert worden sind, werden mit Hilfe des selbstgeschriebenen Skripts `change_tf_record_name.py` ihre Dateinamen in die der entsprechenden Bilder umbenannt. Die in Kapitel 5.2 implementierte Cross Validation fügt diese dann automatisiert der aktuellen Fold entsprechend in den korrekten `train`- oder `val`-Ordner ein.

Im Gegensatz dazu kann jedoch die für die Daten von der TensorFlow 2 Object Detection API benötigte `label_map.pbtxt` über das Exportieren der annotierten Daten als `tfrecord` in CVAT erhalten werden – oder aber manuell erstellt werden, was insbesondere im Falle von wenigen Klassen tragbar ist.

Deskription und Evaluation der Ergebnisse

Es werden fünf EfficientDet-Modelle nach den Verteilungen der fünf Folds der Cross Validation mit 12 500 Schritten trainiert und die mAP-Werte sowie Trainingsdauer in der Tabelle 5.2 festgehalten. Der über die fünf Folds durchschnittliche mAP beläuft sich

[12] `create_coco_tf_record.py`, `https://github.com/tensorflow/models/blob/master/research/object_detection/dataset_tools/create_coco_tf_record.py`, besucht am 19.09.2022

für EfficientDet auf 0,314 und die durchschnittliche Trainingsdauer auf rund 1 h und 20 min. Dabei erreicht die dritte Fold den besten mAP von 0,443 und die fünfte Fold den schlechtesten mAP von 0,21. Fold 4 ist nur 2,3 % besser als Fold 5. Ähnlich zu der Lineallokalisierung mit YOLOR des vorgegangenen Kapitels 5.6.2 und der Aufschlüsselung der Ausrichtung des Lineals anhand des Balkendiagramms in Abbildung 5.1 mag dies an den schwieriger zu lokalisierenden waagrecht ausgerichteten Linealen nach **HG5** liegen.

EfficientDet		
Fold	**mAP**	**Dauer**
1	0,312	1 h 19 min
2	0,372	1 h 19 min
3	0,443	1 h 18 min
4	0,233	1 h 19 min
5	0,210	1 h 18 min
ø	**0,314**	**1 h 19 min**

Tabelle 5.2: Ergebnisse der Lineallokalisierung der EfficientDet-Modelle auf Basis der Validierungsdaten pro Cross Validation Fold.

Mit Blick auf die Ergebnisse der anderen Folds und auf die Vorhersagen der Bounding Boxen der Modelle mit einem Confidence-Schwellenwert von 1 % scheint der Einfluss der Linealausrichtungen an dieser Stelle jedoch nicht derart groß zu sein: Im Gegensatz zu den YOLOR-Modellen werden mehr waagrechte Lineale grob lokalisiert. Die Confidence, mit der EfficientDet Lineale vorhersagt, liegt bei ungefähr 15 bis 80 %. Viele der vorhergesagten Bounding Boxen befinden sich an den richtigen Stellen, umschließen das Lineal meist jedoch nicht voll oder teilen sich dasselbe Lineal mit anderen Bounding Boxen.

Bereits aufgeführte Problematiken, wie das fehlerhafte Identifizieren von Regalbrettern oder Objektteilen als Lineal aufgrund ihrer Form und Farbe nach den Herausforderungen **HL4** und **HG2**, treten auch bei EfficientDet innerhalb der ersten, zweiten und vierten Fold auf – allerdings deutlich seltener als bei YOLOR. Auch eine zu hohe oder niedrige Belichtung sowie größere abgeschnittene Teile des Lineals nach **HG3** und **HL2** scheinen weniger Einfluss auf die Erkennung zu nehmen.

Die Vorhersagen zweier EfficientDet-Modelle derselben Fotografien des vorherigen Kapitels 5.6.2 werden in Abbildung 5.3 dargestellt.

5.6.4 Lokalisierung mit Mask R-CNN

Zuletzt wird auch der zweistufige Detektor Mask R-CNN [HGDG17], beschrieben in Kapitel 3.6.1, im Kontext der Lineallokalisierung praktisch unter Verwendung von der

Abbildung 5.3: Vorhersagen der EfficientDet-Modelle zur Lineallokalisierung auf den Validierungsbildern `TCD-1` und `PG-5`[11].

TensorFlow 2 Object Detection API nach Kapitel 5.5 angewandt.

Generierung der detektorspezifischen Inputdaten

Analog zur Vorbereitung der Eingabedaten für EfficientDet der TensorFlow 2 Object Detection API in Kapitel 5.6.3 müssen auch an dieser Stelle die Daten als `tfrecord`s vorliegen. Da diese aus den Annotationen über CVAT nicht sinngemäß generierbar und in Trainings- und Validierungsmenge aufzusplitten sind, werden die Annotationen gleichermaßen im COCO-Exportformat heruntergeladen. Ein beispielhafter Auszug einer Annotation jenes Formats ist in Kapitel A.3 im Appendix einzusehen.

Mask R-CNN der TensorFlow 2 Object Detection API benötigt nun jedoch Segmentierungsmasken, um funktionieren zu können. Im Falle der Lineallokalisierung ist jedoch wie in Kapitel 5.6.1 beschrieben lediglich mit Bounding Boxen annotiert worden. Über das selbstgeschriebene Skript `bbox_to_seg.py` können jene im COCO-Annotationsformat vorhandenen Koordinaten der Bounding Boxen der Form $[x, y, w, h]$ nun in die benötigten Segmentierungskoordinaten $[x_1, y_1, x_2, y_2, \ldots x_n, y_n]$ umgewandelt werden. Dazu müssen im Kontext der Lineallokalisierung lediglich die vier Eckpunkte der Bounding Box anhand (x, y), $(x + w, y)$, $(x + w, y + h)$ und $(x, y + h)$ berechnet und in die leere Liste des entsprechenden `segmentation`-Keys innerhalb des Keys `annotations` eingefügt werden.

Erst dann kann das von der TensorFlow 2 Object Detection API gebotene Skript `/models/research/object_detection/dataset_tools/create_tf_record.py` verwendet werden, um `tfrecord`s analog zu EfficientDet zu generieren. Der einzige Unterschied besteht darin, dass nun zur Inklusion der Segmentierungsmasken die Flag `include_masks` explizit auf `True` gesetzt werden muss.

Anhand des selbstgeschriebenen Skripts `change_tf_record_name.py` werden schließlich die `tfrecord`s in jenen Scan umbenannt, den und dessen Annotationen sie speichern. Anschließend können sie in den Ordner `all_images` eingefügt und dann automatisiert von der Cross Validation in den für den aktuellen Fold korrekten `train`- oder `val`-Ordner geschoben werden.

Schließlich lässt sich die benötigte `label_map.pbtxt` zur Zuordnung der IDs und Klassen gleich EfficientDet in Kapitel 5.6.3 herunterladen oder generieren.

Deskription und Evaluation der Ergebnisse

Ebenso werden fünf Mask R-CNN-Modelle den fünf Folds der Cross Validation entsprechend mit je 12 500 Schritten trainiert. Ihre erreichten mAP-Werte sowie Trainingsdauern werden in Tabelle 5.3 festgehalten. Die Durchschnittswerte belaufen sich für den mAP auf 0,632 und für die Trainingsdauer auf 53 min. Dabei ist Fold 3 mit einem mAP von 0,821 das beste und Fold 5 mit einem mAP von 0,457 das schlechteste Modell.

Mask R-CNN		
Fold	**mAP**	**Dauer**
1	0,549	54 min
2	0,738	54 min
3	0,821	51 min
4	0,595	52 min
5	0,457	54 min
ø	**0,632**	**53 min**

Tabelle 5.3: Ergebnisse der Lineallokalisierung der Mask R-CNN-Modelle auf Basis der Validierungsdaten pro Cross Validation Fold.

Betrachtet man die Vorhersagen der einzelnen Modelle mit einem Confidence-Schwellwert von 1 %, so wird deutlich, dass waagrecht angeordnete Lineale im Fall von Mask R-CNN fast genauso gut wie senkrecht angeordnete erkannt werden. Insbesondere die Confidence der Bounding Boxen, die tatsächlich Lineale identifiziert haben, sind typischerweise bei über 95 %, auch wenn das Lineal nicht vollstens lokalisiert werden konnte.

Die anderen für YOLOR in Kapitel 5.6.2 und EfficientDet in Kapitel 5.6.3 aufgeführten Problematiken können jedoch auch für Mask R-CNN identifiziert werden. So werden teilweise Regalbretter insbesondere der Folds 1, 2 und 3 aufgrund ihrer Farbe und Form Herausforderungen **HG2** und **HL4** folgend fälschlicherweise als Lineal erkannt. Innerhalb Fold 2 sowie Fold 4 wird außerdem auch der Teil der Wand zwischen dem Regal und dem Bildrand wieder fälschlicherweise als Lineal betitelt. Beides kommt jedoch nur selten und mit sehr geringer Confidence im Vergleich zu den möglichst korrekt lokalisierten Linealen vor. Dasselbe gilt für die wenigen Objekte, die insbesondere innerhalb

der zweiten und fünften Fold weiterhin nach **HL4** fälschlicherweise als Lineal erkannt werden. Dabei werden jedoch nicht nur Teile der Objekte, die mit ihrer länglichen Form nah an die Form eines Lineals kommen, als solches identifiziert, wie es bei YOLOR und EfficientDet der Fall war. Stattdessen werden komplette Objekte wie *Bowls*, deren Bounding Boxen sogar beinahe quadratisch und in keinem Fall länglich sind, fälschlicherweise mit einer Confidence von meist unter 10 % als Lineal identifiziert. Da die Confidence der Lineale im Vergleich zu den falschen Bounding Box-Vorhersagen derart hoch ist, würde es sich anbieten, den Confidence-Schwellwert zu erhöhen, um falsche Vorhersagen auszusortieren, korrekte Vorhersagen aber genauso gut beibehalten zu können. Bei den Modellen der beiden vorherigen Detektoren würden zwangsläufig vermehrt korrekte Vorhersagen verloren gehen, weshalb sich dieses Vorgehen bei ihnen nicht anbietet.

Schließlich zeigt Abbildung 5.4 beispielhafte Vorhersagen zweier Mask R-CNN-Modelle für dieselben Fotografien der vorhergegangenen Objekterkennungsdetektoren in den Kapiteln 5.6.2 und 5.6.3.

Abbildung 5.4: Vorhersagen der Mask R-CNN-Modelle zur Lineallokalisierung auf den Validierungsbildern `TCD-1` und `PG-5`[11].

5.6.5 Gegenüberstellung der Detektoren

Nun können die zuvor erhaltenen und spezifisch evaluierten Ergebnisse der Lineallokalisierung mittels YOLOR in Kapitel 5.6.2, EfficientDet in Kapitel 5.6.3 und Mask R-CNN in Kapitel 5.6.4 ins Verhältnis zueinander gesetzt werden. Insbesondere der Detektor Mask R-CNN kann mit dem höchsten durchschnittlichen mAP-Wert von 0,632 glänzen, der in nicht einmal 1 h Training erreicht werden konnte. Auch die betrachteten Vorhersagen der Bounding Boxen sind unter Rücksichtnahme der in Kapitel 5.6.4 erwähnten negativen Aspekte, die prinzipiell jeder Detektor besitzt, gut. Des Weiteren ist zu erwähnen, dass es sich bei Mask R-CNN um einen zweistufigen Detektor handelt,

der im Vergleich zu den anderen beiden Detektoren einen größeren Vergleichswert bietet. Dies liegt insbesondere auch an der Möglichkeit, Polygon-Annotationen, wie eigentlich für die Keramikgefäßerkennung gefordert, utilisieren zu können. Entsprechend steht außer Frage, dass Mask R-CNN auch für die Keramikgefäßerkennung und -lokalisierung zum Einsatz kommen wird.

Schließlich muss nur noch eine Entscheidung zwischen EfficientDet und YOLOR getroffen werden. Werden an dieser Stelle lediglich die durchschnittlichen mAP-Werte und Trainingsdauern betrachtet, würde die Wahl ganz klar auf EfficientDet fallen. Dessen Modelle erreichen durchschnittliche Werte von 0,314 innerhalb 1 h 20 min. Demgegenüber erreichen die YOLOR-Modelle einen fast gleichen durchschnittlichen Wert von 0,304 in 7 h. Allein aus Effizienzgründen ist EfficientDet im getesteten Fall YOLOR klar überlegen. Auch durch die Betrachtung der vorhergesagten Bounding Boxen beider Modelle ist eine eher geringe, aber doch klare Überlegenheit von EfficientDet zu erkennen, da YOLOR-Modelle im direkten Vergleich häufiger waagrechte Lineale nicht lokalisieren können. Allerdings ist gerade diese hohe durchschnittliche Trainingsdauer nicht typisch für YOLOR, doch auch durch einen erneuten Trainingsversuch unter gleichen Voraussetzungen nach Kapitel 5.1 nicht zu verschmälern.

Ferner sind jedoch auch die Ergebnisse der zuvor ohne Cross Validation trainierten Modelle zur Lineallokalisierung sowie zur Keramikgefäßerkennung und -lokalisierung zu berücksichtigen. Sie sind in Tabelle A.2 und A.1 im Appendix festgehalten. In beiden Fällen ist das YOLOR-Modell im Hinblick auf den mAP den anderen Detektoren überlegen und weist keine deutlich längere Trainingsdauer auf. Im Fall der Erkennung und Lokalisierung der Keramikgefäße erreicht es sogar eine deutlich kürzere Trainingsdauer als Mask R-CNN. Aufgrund dieser zusätzlichen Informationen und dem Wissen um die klare Überlegenheit im Bezug auf die relevanteste Teilaufgabe der Keramikgefäßerkennung und -lokalisierung, wird sich an dieser Stelle doch noch für YOLOR entschieden.

5.7 Erkennung und Lokalisierung der Keramikgefäße

Die praktische Erkennung und Lokalisierung der einzelnen Keramikgefäße wird schließlich in diesem Kapitel behandelt. Ähnlich dem vorgegangenen Kapitel 5.6 folgt die Beschreibung relevanter Daten und deren Annotation sowie Distribution anhand der Cross Validation. Anschließend werden sowohl das Training als auch die Ergebnisse und spezifische Evaluation der Lokalisierung mit den Detektoren YOLOR und Mask R-CNN unter Verwendung verschiedener Stufen der Data Augmentation aufgeführt. Dabei wird insbesondere auch der Umgang der Modelle mit unannotierten Objekten näher betrachtet. Schließlich werden die Detektoren gegenübergestellt und das beste Modell des besseren Detektors für die darauffolgende Größenapproximation verwendet.

5.7.1 Annotation und Distribution der Daten

Für die in diesem Kapitel als Ziel gesetzte Erkennung und Lokalisierung der einzelnen Keramikgefäße pro Fotografie von Keramikgefäßsammlungen gilt es, jene Keramikgefäße zunächst mit CVAT zu annotieren. Ob die entsprechenden Fotografien dabei ein Lineal abbilden oder nicht, ist zunächst nicht von Relevanz. Dem Implementierungskonzept des Kapitels 4.5 folgend soll die Annotation mit Polygonen durchgeführt werden, um eine im Forschungsprojekt [Wir21] erkannte höhere Genauigkeit im Vergleich zu Bounding Boxen zu erzielen. In diesem Fall ist jedoch auch insbesondere die korrekte Identifikation der Keramikgefäße relevant, zum Beispiel als *Amphoriskos*, welche jedoch seitens der Informatik aufgrund mangelnder Expertise nicht zu bewältigen ist. In diesem Zuge stellt das Institut für Archäologische Wissenschaften der Goethe-Universität drei studentische Hilfskräfte zur Unterstützung zur Verfügung.

Dennoch nimmt, wie in Kapitel 4.5 approximiert, bereits die Annotation der Scans der Fotografien trotz Expertise äußerst viel Zeit in Anspruch. Aufgrund dessen wird auf die Hinzunahme und Annotation der über 100 weiteren Kameraaufnahmen im Rahmen der Masterthesis verzichtet. Dies hat zu Folge, dass sich auch im Fall der Lineallokalisierung nur auf Scans mit Keramikgefäßsammlungen fokussiert wird.

Schließlich werden die Hilfskräfte auf dieser Basis anhand einer Videokonferenz samt erstelltem PDF als Tutorial in die Bildannotation und das Annotationstools CVAT eingeführt. Hierbei stellt die Web-Oberfläche CVATs einen besonderen Vorteil dar, da durch sie eine Installation über die Kommandozeile umgangen werden kann.

Unter anderem werden die Hilfskräfte diesbezüglich dazu angehalten, die Keramikgefäße guter Annotationsmannier folgend möglichst eng mit Polygonen zu annotieren. Um Löcher in oder an Gefäßen soll dabei herum annotiert werden, beispielsweise um Henkel von Gefäßen. Wird ein Objekt hingegen nicht vollständig abgebildet, weil es von einem anderen Objekt verdeckt wird, so soll grob so annotiert werden, als sei das Objekt vollständig abgebildet. Für jedes dargestellte Objekt existiert damit nur eine einzige Annotation.

Sind die Hilfskräfte überdies nicht in der Lage ein Objekt zu identifizieren, ist es dennoch mit dem Klassenlabel *unknown* zu annotieren. Gleichermaßen sind Objekte, deren Klasse nicht innerhalb CVAT vertreten ist, da sie nicht auf der bereitgestellten Liste von Gefäßformen standen, mit *missing_label* zu annotieren. Beide Klassenlabel können zu einem späteren Zeitpunkt durch erneute Kommunikation mit dem Institut für Archäologische Wissenschaften korrigiert werden. Im Kontext der Masterthesis werden die Klassenlabel aufgrund begrenzter Zeit jedoch vorerst hingenommen.

Des Weiteren existieren nicht wenige Objekte, die laut den Hilfskräften keine Keramikgefäße sind und deshalb unannotiert bleiben.

Die studentischen Hilfskräfte bewältigen die zeitspielige Annotation der Scans in ca. anderthalb Monaten. Von den entsprechenden annotierten Scans werden schließlich ca. 80 % (34 oder 35 Scans) zum Trainieren und 20 % (8 oder 9 Scans) zum Validieren der YOLOR- und Mask R-CNN-Modelle der Cross Validation Folds verwendet. Die tatsächliche Anzahl der Gefäße sowie die Anzahl an Bildern pro annotierter Klasse werden in Tabelle 5.4 dargestellt.

Klasse	**#Gefäße**	**#Bilder**
Alabastron	1	0
Amphora	41	18
Amphoriskos	20	17
Animal-shaped Vase	9	6
Askos	21	12
Basket	0	0
Bottle	47	17
Bowl	261	30
Cooking Pot	26	12
Cup	14	7
Dish	57	19
Flask	26	7
Goblet	5	4
...		
Horn-shaped Vase	1	0
Hydria	1	0
Jar	14	11
Jug	404	37
Juglet	189	31
Lid	4	2
missing_label	15	12
Oil Lamp	135	18
Rhyton	0	0
Ring-Vessel	3	3
Strainer	1	0
Support	0	0
unknown	0	0

Tabelle 5.4: Die Anzahl der Gefäße und Bilder pro Keramikgefäßklasse mit ■ = später entfernte Klassen.

Vor dem Training der Objekterkennungsdetektoren sind jene Klassen noch einmal genauer zu evaluieren. Klassen, die auf weniger als zwei Fotografien auftreten, können entweder nicht zum Training oder nicht zur Validierung verwendet werden. Dadurch können sich Ergebnisse verfälschen oder ausbleiben, weswegen sie samt ihrer Annotationen manuell entfernt werden. Gleichermaßen werden die Klassen *unknown* und *missing_label* entfernt, da sie in der Praxis keinen Mehrwert bieten und lediglich der vereinfachten Korrektur der Annotation zu einem späteren Zeitraum über der Masterthesis hinaus dienen. Damit bleiben 17 Klassen, deren Anzahl innerhalb der Validierungsmenge pro Fold in Abbildung 5.5 dargestellt werden.

Da jene Klassen aufgrund ihrer hohen Anzahl sowie der hohen Anzahl an Objekten selbst schwer manuell zu evaluieren sind, wird sich im Folgenden nach Kapitel 4.5 auf eine einzige Keramikgefäßklasse fokussiert. Die Entscheidung fällt dabei schließlich

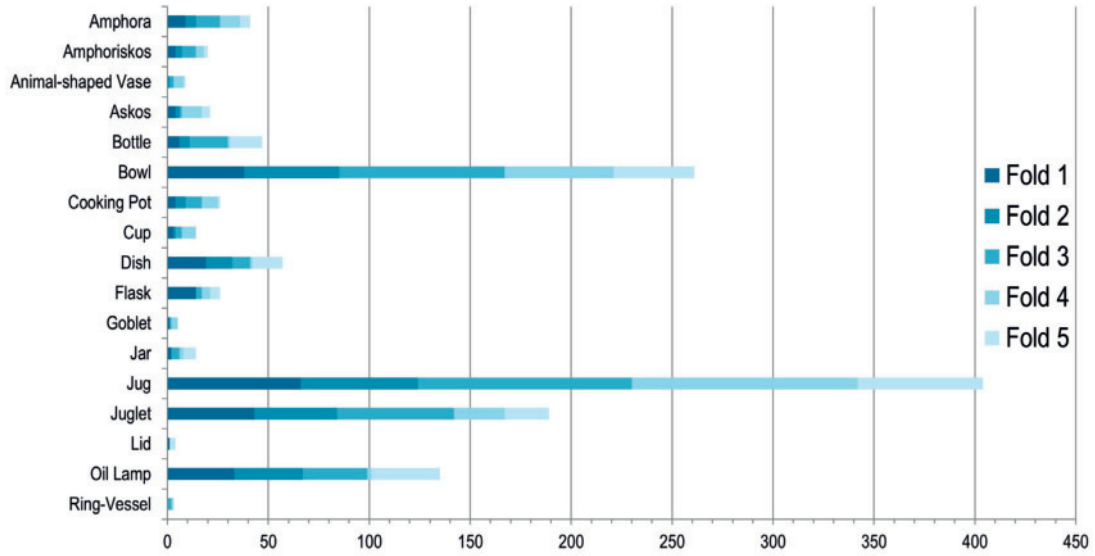

Abbildung 5.5: Gliederung der Validierungsbilder in ihre Klassen pro Cross Validation Fold.

auf Gefäße der Klasse *Amphoriskos*. Erste Versuche[13] konnten bereits zeigen, dass jene Gefäße selten, aber nicht nie von Modellen identifiziert werden können. Überdies beläuft sich ihre Anzahl in den originalen Daten nur auf insgesamt 20 Stück und sie sind für Laien mehr oder weniger gut erkennbar, weswegen mit ihnen manuell im Kontext einer Evaluation gut umzugehen ist.

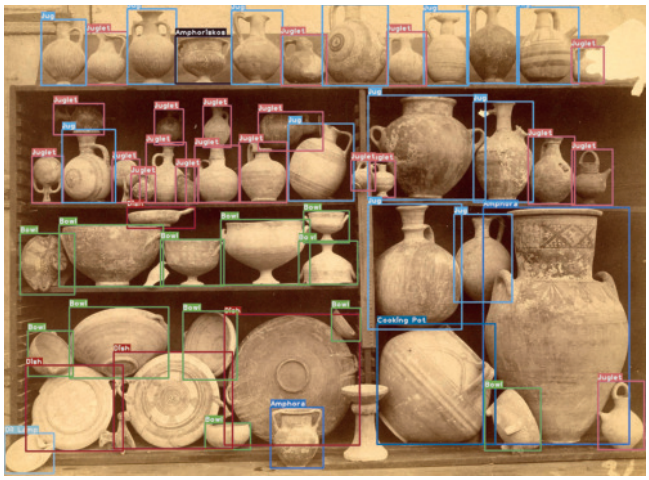

Abbildung 5.6: Die Bounding Box-Annotation des Verkaufsbildes `GCA-2`[14].

[13] Die Ergebnisse jener Versuche sind in Tabelle A.1 im Appendix festgehalten.

[14] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 9051 (`GCA-2`).

Abbildung 5.6 zeigt eine beispielhafte in Bounding Boxen übersetzte Ground Truth-Annotation eines Verkaufsbildes.

5.7.2 Anwendung der Data Augmentation

Aufgrund der geringen Datenbasis und dem größeren Fokus auf die Erkennung und Lokalisierung der Keramikgefäße, wird hierbei die Data Augmentation, implementiert in Kapitel 5.3, angewendet und ihr Einfluss auf die Genauigkeit der Objekterkennungsdetektoren in den nächsten Unterkapiteln genauer untersucht. Dazu werden zwei unterschiedlich große Mengen augmentierter Daten generiert.

Für die erste Menge kommen die in Tabelle 5.5 grob deskriptiv aufgeführten Augmentationsfunktionen zum Einsatz. Die Spalte # stellt dabei die Anzahl an Augmentoren dar, die zufällig aus der angegebenen Menge innerhalb der Spalte *Augmentor* pro Funktion ausgewählt werden. Da insbesondere darauf geachtet wurde, augmentierte Daten zu generieren, die der tatsächlich Datenbasis noch bis zu einem gewissen Teil ähneln, bleibt die Auswahl an Augmentoren limitiert und repetitiv in verschiedenen Funktionen.

Nr.	#	Augmentor	Deskriptive Parameter
1	1	Vertikale Spieglung	100 % der Bilder
2	2	Vertikale Spieglung	100 % der Bilder
		Entfernung von Pixelreihen am Bildrand	Zwischen 1 % und 25 %
		Rotation	Zwischen -5° und 5°
3	1	Multiplizieren jedes Pixels	Faktor zwischen 0,5 und 1,5
		Verwischung mit Gaußschem Kernel	σ zwischen 1 und 2
		Kontrastreduktion	Geringste Stärke von 1
4	2	Entfernung von Pixelreihen am Bildrand	Zwischen 1 % und 25 %
		Rotation	Zwischen -20° und 20°
		Translation in x- und y-Richtung	Je zwischen -20 % und 20 %
		Verwischung mit Gaußschem Kernel	σ zwischen 1 und 2
		Kontrastreduktion	Geringste Stärke von 1
5	2	Perspektivische Transformation	Skala zwischen 1 % und 15 %
		Verwischung mit Gaußschem Kernel	σ zwischen 1 und 2
		Kontrastreduktion	Geringste Stärke von 1

Tabelle 5.5: Deskriptiver Überblick der fünf für die Augmentation 1 verwendeten Augmentationsfunktionen.

Zunächst werden alle fünf Augmentationsfunktionen verwendet, jedoch werden anschließend unpassende Bilder manuell aussortiert und die Annotationsdatei mittels des Skripts `adapt_annotation_folder.py` aktualisiert. Schlussendlich bleiben für jede Fotografie drei augmentierte Versionen übrig, für jene mit der Klasse *Amphoriskos* sogar falls möglich fünf, ansonsten vier. Durch die größere Anzahl an augmentierten Bildern mit

dem Objekt *Amphoriskos* soll besonders die Evaluation seiner Erkennung größere Gewichtung bekommen.

Schließlich stehen 156 augmentierte Bilder und ihre Annotationen mit Abstrich der augmentierten Versionen der Validierungsbilder zum Training zur Verfügung. Folglich wird dabei von Augmentation 1 gesprochen.

Die zweite Menge an augmentierten Daten erweitert die erste Menge um Versionen mit den in Tabelle 5.6 dargestellten Augmentationsfunktionen. In diesen Fall werden keine der neuen Datenelemente mehr aussortiert, sodass am Ende 242 augmentierte Bilder mit Abstrich der augmentierten Versionen der Validierungsbilder zum Training zur Verfügung stehen. Folglich wird dabei von der Augmentation 2 gesprochen.

Nr.	**#**	**Augmentor**	**Deskriptive Parameter**
6	1	Einfügen grauer Rechtecke	Zwischen 1 bis 3 Rechtecke der Größe 0,2
7	2	Vertikale Spieglung	100 % der Bilder
		Multiplizieren jedes Pixels	Faktor zwischen 0,5 und 1,2
		Rotation	Zwischen -10° und 0°

Tabelle 5.6: Deskriptiver Überblick der zwei zusätzlichen für die Augmentation 2 verwendeten Augmentationsfunktionen.

Dem Kapitel 5.3 folgend müssen beide Mengen nach ihrer Generierung über das Skript `augmentation_poly.py` wie beschrieben teilweise händisch und mit anderen Skripten weiterbearbeitet werden. Eine beispielhafte augmentierte Version der zuvor aufgeführten Fotografie `GCA-2` in Abbildung 5.6 ist durch Abbildung 5.7 dargestellt.

Abbildung 5.7: Augmentierte Version des Verkaufsbildes `GCA-2`[14] in Abbildung 5.6 anhand der Augmentatsfunktion 5 nach Tabelle 5.5.

5.7.3 Erkennung und Lokalisierung mit YOLOR

Analog zu Kapitel 5.6.2 wird in diesem Kapitel die Erkennung und Lokalisierung der Keramikgefäße mit dem Objekterkennungsdetektor YOLOR [WYL21] auf Basis von Kapitel 5.4 implementiert.

Generierung der detektorspezifischen Inputdaten

Wie im Kapitel der Lineallokalisierung mittels YOLOR erwähnt, benötigt YOLOR die Dateien der Annotation im YOLO-Exportformat. Nun liegen Kapitel 5.7.1 folgend jedoch keine benötigten Bounding Box-, sondern Polygon-Annotationen vor. Diese kann das Annotationstool CVAT nicht eigenständig in die Bounding Boxen des YOLO-Formats übersetzen – ein Export führt zu leeren Annotationsdateien. Stattdessen wird die Annotation der Keramikgefäße im COCO-Exportformat heruntergeladen und weiterbearbeitet. Zur Verdeutlichung liegen beispielhafte Auszüge von einer Annotation beider Exportformate innerhalb Kapitel A.3 im Appendix vor.

Es kommt das Skript `bbox_coco_to_yolo.py` zum Einsatz, um die Werte des `bbox`-Keys der Form $[x_c, y_c, w_c, h_c]$ innerhalb des `annotations`-Keys in YOLO-Koordinaten umzuwandeln. Dazu werden insbesondere die Werte $d_w = 1/Bildbreite$ und $d_h = 1/Bildhöhe$ genutzt. Die tatsächlichen Werte x_y, y_y, w_y und h_y des YOLO-Formats ergeben sich schließlich folgendermaßen.

$$
\begin{aligned}
x_y &= ((2 \cdot x_c + w_c)/2) \cdot d_w \\
y_y &= ((2 \cdot y_c + h_c)/2) \cdot d_h \\
w_y &= w_c \cdot d_w \\
h_y &= h_c \cdot d_h
\end{aligned}
$$

Mittels einer Schleife über alle Annotationen können sie mit der Klassen ID, die sich über den Key `category_id` innerhalb `annotations` ergibt, in entsprechende Textdateien pro Bild geschrieben werden.

Anschließend werden die originalen Bilder und Textdateien in den Ordner `all_images` und die augmentierten Bilder und Textdateien in den Ordner `augmentations` eingefügt. Anschließend können die entsprechenden Daten durch die Cross Validation, die in Kapitel 5.2 implementiert und in Kapitel 5.3 angepasst wurde, automatisiert in die passenden `train`- und `val`-Ordner verschoben werden.

Auch die Datei `data.yaml` mit den entsprechenden Pfaden zu den Ordnern, der Anzahl der Klassen und ihren Namen sowie die Datei `data.names` mit einer Liste der Klassennamen ist manuell zu generieren.

Deskription und Evaluation der Ergebnisse

Schließlich können der Cross Validation entsprechend jeweils fünf YOLOR-Modelle mit je 1 000 Epochen ohne augmentierte Bilder, mit zusätzlichen 156 augmentierten Bildern sowie mit zusätzlichen 242 augmentierten Bildern trainiert werden. Alle mAP-Ergebnisse und die Trainingsdauern der einzelnen Modelle werden in Tabelle 5.7 aufgeführt. Mit YOLOR können dabei durchschnittliche, mit den augmentierten Bildern steigende mAP-Werte von 0,488 ohne Augmentation, 0,517 für die Augmentation 1 und 0,551 für die Augmentation 2 erreicht werden. Auch die durchschnittliche Trainingsdauer steigt mit der erhöhten Anzahl an Trainingsbildern von 3 h 41 min ohne Augmentation auf bis zu 7 h bei der Augmentation 2.

Für alle drei Augmentationsstufen gilt dabei die vierte Fold mit mAP-Werten von 0,265 bis 0,352 als die schlechteste und die dritte Fold mit mAP-Werten von 0,426 bis 0,475 als die zweitschlechteste. Jene Werte steigen ebenfalls mit Hinzunahme von augmentierten Bildern mit der Ausnahme von der dritten Fold der Augmentation 1 auf die Augmentation 2. Diese sinkt um 0,9 %.

Die Folds mit den besten mAP-Werten sind die übrigen Fold 1, Fold 2 und Fold 5. Ihre Werte belaufen sich auf 0,579 bis 0,594 bei keiner Augmentation, auf 0,535 bis 0,636 bei der Augmentation 1 sowie auf 0,636 bis 0,646 bei der Augmentation 2. Die Differenzen jener Werte sind typischerweise sehr gering, mit der Ausnahme der fünften Fold im Fall der Augmentation 1, die im Vergleich zur ersten Fold mit 0,634 und der zweiten Fold mit 0,636 lediglich bei 0,535 liegt. Welche der drei Folds schließlich den höchsten mAP-Wert erreichen, ist pro Augmentationsstufe anders.

	YOLOR					
	Keine Augmentation		**Augmentation 1**		**Augmentation 2**	
Fold	**mAP**	**Dauer**	**mAP**	**Dauer**	**mAP**	**Dauer**
1	0,58	3 h 58 min	0,634	5 h 51 min	0,646	6 h 40 min
2	0,579	3 h 33 min	0,636	6 h 13 min	0,644	6 h 52 min
3	0,426	3 h 21 min	0,484	6 h 16 min	0,475	6 h 47 min
4	0,265	3 h 34 min	0,3	5 h 41 min	0,352	7 h 9 min
5	0,594	3 h 58 min	0,535	5 h 54 min	0,636	7 h 17 min
ø	**0,488**	**3 h 41 min**	**0,517**	**5 h 59 min**	**0,551**	**6 h 57 min**

Tabelle 5.7: Ergebnisse der Keramikgefäßerkennung und -lokalisierung der YOLOR-Modelle auf Basis der Validierungsdaten pro Cross Validation Fold und Augmentationsstufe.

Der Grund für die schlechte Performanz der dritten und vierten Fold ergibt sich durch die Anzahl an Objekten pro Bild pro Fold. Die entsprechenden Werte sind in Tabelle 5.8 festgehalten. Während die erste, zweite und fünfte Fold sich auf durchschnittlich um

die 30 Objekte pro Bild belaufen, weisen die dritte Fold durchschnittlich 50,4 Objekte pro Bild und die fünfte Fold durchschnittlich 51,63 Objekte pro Bild auf. Anhand dieser Information lässt sich darauf schließen, dass Fotografien, die eine größere Anzahl an Objekten abbilden, schlechtere mAP-Werte erzielen und damit von YOLOR schlechter zu identifizieren sind. Dies ist bereits in Herausforderung **HG4** festgehalten worden. Zusätzlich mag es der Tatsache geschuldet sein, dass die Anordnung der Objekte in den Regalen innerhalb der Fotografien mit hoher Anzahl an Objekten stets ungezügelter wird: Gerade diese Fotografien weisen viele Verdeckungen und eine große Varianz in der Ausrichtung der Objekte auf, vermutlich um alle im Regal unterbringen zu können.

Fold	#Objekte	#Objekte/Bild
1	288	32
2	266	29,6
3	454	50,4
4	413	51,63
5	265	33,13

Tabelle 5.8: Die Gesamtzahl der Objekte und die durchschnittliche Anzahl der Objekte pro Validierungsbild jeder Cross Validation Fold.

Abbildung 5.8 zeigt die Bounding Box-Vorhersagen der Fotografie `GCA-2` keiner Augmentation, der Augmentation 1 und der Augmentation 2 in dieser Reihenfolge unter dem Confidence-Schwellwert von 40 %. Um die getätigten Vorhersagen besser bewerten zu können, kann Abbildung 5.6 herangezogen werden, die die Ground Truth Bounding Box-Annotationen enthält.

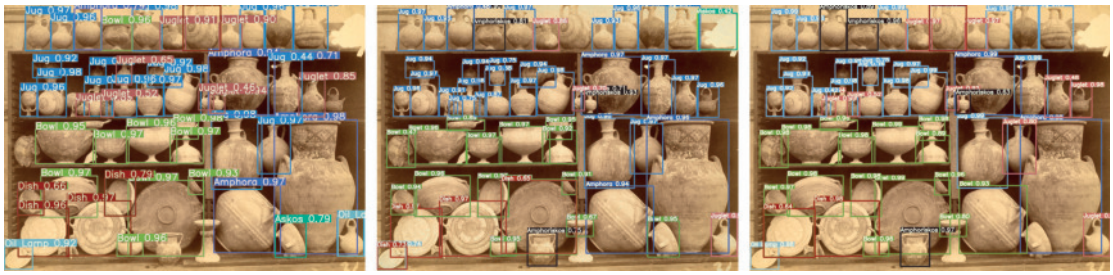

Abbildung 5.8: Vorhersagen der YOLOR-Modelle zur Keramikgefäßerkennung und -lokalisierung (von links) ohne Augmentation, mit der Augmentation 1 und der Augmentation 2 auf dem Validierungsbild `GCA-2`[14].

Da sich eine manuelle Evaluation aufgrund der vielen Objekte und Klassen, wie in Kapitel 5.7.1 erwähnt, schwierig gestaltet, wird sich auf die Klasse *Amphoriskos* fokussiert. Ob und wie die 20 innerhalb der originalen Fotografien dargestellten *Amphoriskoi* erkannt und lokalisiert werden können, lässt sich aus Tabelle 5.10 lesen. Prinzipiell ist eine Verbesserung der Vorhersagen durch die augmentierten Daten und damit Herausforde-

rung **HG1** zu beobachten. Ganze zwei *Amphoriskoi* mehr können beim Vergleich des Falles ohne Augmentation und der Augmentation 1 korrekt lokalisiert und klassifiziert werden. Für die Augmentation 2 kann noch ein weiterer *Amphoriskos* korrekt lokalisiert, aber nicht korrekt klassifiziert werden. Da es sich dabei aber nur um einen von insgesamt zwanzig *Amphoriskoi* handelt, ist die Aussagekraft dieser faktischen Verbesserung anzuzweifeln. Ähnliches könnte über den Vergleich von fehlender Augmentation und den anderen beiden Augmentationsstufen zum Tragen kommen. An dieser Stelle lohnt es sich jedoch, einen genaueren Blick auf die korrekt lokalisierten und klassifizierten *Amphoriskoi* zu werfen. Die falsch lokalisierten und falsch klassifizierten bleiben hierbei außen vor, da diesen aufgrund der fehlenden Grunderkennung keine größere Relevanz zugrunde liegt.

Für die drei korrekt lokalisierten und klassifizierten *Amphoriskoi* der Modelle ohne augmentierte Bilder liegen für jeden *Amphoriskos* trotz der Confidence-Schwelle von 40 % weitere Bounding Boxen für dasselbe Objekt vor. Zwei der drei *Amphoriskoi* werden zusätzlich als *Amphora* oder *Jar* klassifiziert, wobei die korrekte Klassifizierung jedoch die höchste Confidence besitzt. Hingegen wird der andere *Amphoriskos* zusätzlich sowohl als *Cooking Pot* als auch als *Jar* klassifiziert, wobei *Jar* die höchste und *Amphoriskos* die zweithöchste Confidence vorweisen kann. Überdies werden in zwei der drei Fälle auch ein anderes Objekt irrtümlich einzig als *Amphoriskos* klassifiziert.

Für die Augmentation 1 treten keine doppelten Klassifizierungen unter dem Confidence-Schwellwert von 40 % mehr auf, jedoch wird bei drei der fünf Fotografien mit den korrekt vorhergesagten *Amphoriskoi* ein weiteres Objekt fälschlicherweise als *Amphoriskos* klassifiziert. Im Fall von der Fotografie `GCA-2` handelt es sich dabei sogar um vier weitere Objekte. Schließlich tritt dieses Problem bei der Augmentation 2 nur noch ein einziges Mal ebenfalls bei `GCA-2` auf, wobei sich die falschen weiteren *Amphoriskoi* jedoch auf drei beschränken. Mehrere falsche Klassifizierungen der korrekt erkannten *Amphoriskoi* gibt es nicht mehr.

Damit kann mit Fokus auf die Identifizierung und Lokalisierung der Objekte der Klasse *Amphoriskos* gleichermaßen von einer Verbesserung durch Data Augmentation gesprochen werden.

5.7.4 Erkennung und Lokalisierung mit Mask R-CNN

Nun gilt es die Erkennung und Lokalisierung der Keramikgefäße auch mit Mask R-CNN [HGDG17] durchzuführen. Ähnlich wie im unmittelbar vorgegangenen Kapitel 5.7.3 für YOLOR, orientiert sich auch dieses Kapitel an der Lineallokalisierung desselben Detektors mit der TensorFlow 2 Object Detection API beschrieben in Kapitel 5.6.4.

Generierung der detektorspezifischen Inputdaten

Die Inputdaten für Mask R-CNN der TensorFlow Object Detection API müssen, wie bereits mehrfach beschrieben, als `tfrecord`s vorliegen. Jedoch liegen dieses Mal bereits die von Mask R-CNN benötigten Segmentierungsmasken der Keramikgefäße vor. Die Erlangung der passenden Inputdaten ist damit fast analog zu jener der Lineallokalisierung mit EfficientDet aus Kapitel 5.6.3: Die Annotation wird als COCO-Format heruntergeladen und mit `/models/research/object_detection/dataset_tools/create_tf_record.py` in `tfrecord`-Bruchstücke umgewandelt, deren Dateinamen mit `change_tf_record_name.py` geändert werden. Der einzige Unterschied zu Kapitel 5.6.3 ist, dass die Flag `include_masks` bei der Generierung der `tfrecord`-Bruchstücke auf `True` gesetzt wird. Des Weiteren kann die benötigte `label_map.pbtxt` manuell generiert oder aufgrund der hohen Anzahl an Klassen einfacher über das sonst nicht benutzbare `tfrecord`-Exportformat von CVAT heruntergeladen werden.

Deskription und Evaluation der Ergebnisse

Zuletzt werden der Cross Validation entsprechend je fünf Mask R-CNN Modelle mit 15 000 Schritten für die drei Augmentationsstufen, keine Augmentation, die Augmentation 1 mit 156 zusätzlichen augmentierten Bildern und die Augmentation 2 mit 242 zusätzlichen augmentierten Bildern trainiert. Eine Zusammenfassung der mAP-Ergebnisse und Trainingsdauern ist in Tabelle 5.9 zu entnehmen. Insbesondere die durchschnittlichen mAP-Werte von 0,369 ohne augmentierte Bilder, 0,37 für die Augmentation 1 und 0,359 für die Augmentation 2 sind dabei zu betrachten. Im Gegensatz zu der Verbesserung um mindestens 2,9 % und höchstens 3,4 % der YOLOR-Modelle ist im Fall von Mask R-CNN lediglich eine Verbesserung um 0,1 % durch Hinzunahme der 156 augmentierten Bilder und eine Verschlechterung um 1,1 % durch weitere Hinzunahme von 86 augmentierten Bildern zu beobachten. Diese Differenzen zwischen den Augmentationsstufen sind derart gering und offensichtlich schwankend, sodass hierbei von keiner Verbesserung durch Data Augmentation gesprochen werden kann. Ebenso pendelt sich die durchschnittliche Trainingsdauer aller Augmentationsstufen auf ca. 6 h ein.

Analog zu YOLOR gelten Fold 1, Fold 2 und Fold 5 als jene mit den besten nachgewiesenen mAP-Ergebnissen und konkurrieren um den besten Platz. Auch ihre Differenzen innerhalb der Augmentationsstufen sind gering, jedoch nicht so gering wie bei YOLOR. So belaufen sich die Werte der drei Folds ohne augmentierte Daten auf ein Intervall von 0,423 bis 0,493, bei der Augmentation 1 auf ein Intervall von 0,425 bis 0,455 und bei der Augmentation 2 auf ein Intervall von 0,425 bis 0,433.

Ebenfalls stellen sich Fold 3 und Fold 4 als die Folds mit den schlechtesten mAP-Werten heraus. Fold 4 mit mAP-Werten zwischen 0,213 bis 0,282 über alle Augmenta-

tionsstufen wird nur im Fall der Augmentation 1 von Fold 3 mit mAP-Werten zwischen 0,251 bis 0,298 über alle Augmentationsstufen als Fold mit dem schlechtesten mAP-Ergebnis abgelöst. Der Grund für ihre schlechte Performanz lässt sich erneut durch die bereits für die Evaluation von den YOLOR-Modellen herausgefilterte durchschnittliche Anzahl an Objekten pro Bild nach Herausforderung **HG4** erklären. Diese ist in Tabelle 5.8 festgehalten, die verdeutlicht, dass Fold 3 und Fold 4 20 Objekte mehr als die besser abschneidenden Folds abbilden. Im Gegensatz zu YOLOR ist die Differenz der mAP-Werte zwischen Fold 3 und Fold 4 jedoch deutlich geringer.

	Mask R-CNN					
	Keine Augmentation		**Augmentation 1**		**Augmentation 2**	
Fold	**mAP**	**Dauer**	**mAP**	**Dauer**	**mAP**	**Dauer**
1	0,43	5 h 53 min	0,425	6 h 1 min	0,433	6 h
2	0,493	5 h 57 min	0,437	6 h 10 min	0,428	6 h 7 min
3	0,258	6 h 8 min	0,251	5 h 11 min	0,298	5 h 18 min
4	0,24	5 h 47 min	0,282	5 h 41 min	0,213	5 h 34 min
5	0,423	6 h 16 min	0,455	6 h 15 min	0,425	5 h 59 min
ø	**0,369**	**6 h**	**0,37**	**5 h 51 min**	**0,359**	**5 h 48 min**

Tabelle 5.9: Ergebnisse der Keramikgefäßerkennung und -lokalisierung der Mask R-CNN-Modelle auf Basis der Validierungsdaten pro Cross Validation Fold und Augmentationsstufe.

Werden nun auch die getroffenen Bounding Box-Vorhersagen der Mask R-CNN Modelle mit einem Confidence-Schwellwert von 0,1 % betrachtet, fällt zunächst auf, dass nur sehr wenige Objekte trotz des geringen Schwellwerts überhaupt erkannt werden. Dies hängt wie in Abbildung 5.9 ersichtlich nicht mit der Objektgröße innerhalb der Fotografie zusammen, sondern unter Umständen mit den anderen aufgeführten Herausforderungen – insbesondere **HG1** und **HG4**. Die Confidence der lokalisierten Objekte ist typischerweise mit 90 bis 100 % jedoch ziemlich hoch. Diese Beobachtung wird deutlicher, wenn der Fokus auf das Objekt *Amphoriskos* gerichtet wird. Inwiefern dieses korrekt vorhergesagt werden kann, lässt sich durch Tabelle 5.10 ermitteln. Ganze 17 von 20 *Amphoriskoi* konnten im Fall ohne Augmentation nicht einmal lokalisiert werden, 14 im Fall der Augmentation 1 und 16 im Fall der Augmentation 2. Nur bei den Modellen der Augmentation 1 können zwei *Amphoriskoi* korrekt lokalisiert und klassifiziert werden, bei den anderen beiden Augmentationsstufen lediglich ein einziger. Damit kann bei der Hinzunahme augmentierter Bilder von keiner Augmentation hin zur Augmentation 1 von einer kleinen Verbesserung gesprochen werden – primär, da mehr *Amphoriskoi* überhaupt lokalisiert, wenn auch nicht korrekt klassifiziert werden können. Diese geringe Steigerung scheint jedoch durch die Hinzunahme von noch mehr augmentierten Bildern für die Augmentation 2 wieder fast gänzlich zu verschwinden.

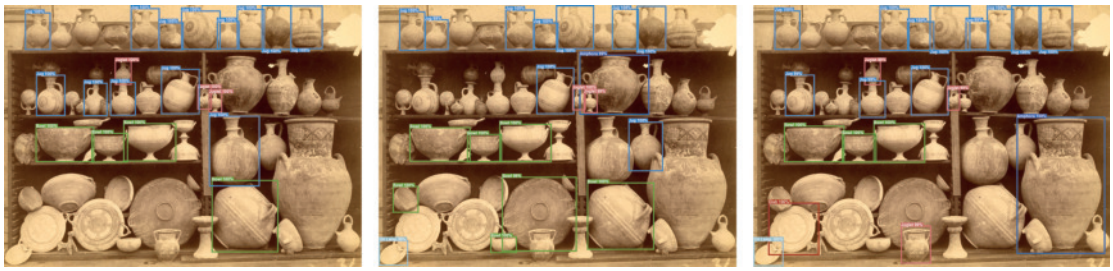

Abbildung 5.9: Vorhersagen der Mask R-CNN-Modelle zur Keramikgefäßerkennung und -lokalisierung (von links) ohne Augmentation, mit der Augmentation 1 und der Augmentation 2 auf dem Validierungsbild `GCA-2`[14].

Für alle drei Stufen gilt außerdem, dass der korrekt lokalisierte und klassifizierte *Amphoriskos* der Fotografie `ECA-5` außerdem auch als *Jar*, im Fall der Augmentation 1 sogar auch als *Amphora* klassifiziert worden ist. Für keine Augmentation sowie die Augmentation 2 besitzt die Klasse *Amphoriskos* für das korrekte Objekt jedoch die höchste Confidence – bei der Augmentation 1 jedoch die geringste. Überdies wird von allen drei Augmentationsstufen zusätzlich ein weiteres, gleiches Objekt fälschlicherweise als *Amphoriskos* klassifiziert. Tatsächlich handelt es sich bei diesem Objekt um ein *Juglet*, welches bereits ohne augmentierte Daten über 400 Mal im Datensatz auftritt und daher gut identifiziert werden können sollte. Für Mask R-CNN kann schließlich durch diese Betrachtung keine neue Information gewonnen werden. Der Schluss liegt nahe, dass das Hinzunehmen von noch mehr augmentierten Bildern an dieser Stelle eher zu einer Stagnation, möglicherweise sogar einer Verschlechterung führen würde.

5.7.5 Detektorverhalten gegenüber unannotierten Objekten

Durch die Herausforderungsanalyse in Kapitel 4.2 wurde bereits mittels Herausforderung **HK2** erwähnt, dass einige abgebildete Objekte keine Keramikgefäße sind und damit für die Keramikgefäßerkennung und -lokalisierung unannotiert bleiben müssen. Des Weiteren sind jene Annotationen von Keramikgefäßen, die zunächst mit *missing_label* klassifiziert worden sind, für das Training der Modelle der Objekterkennung mittels YOLOR und Mask R-CNN entfernt worden, wie in Kapitel 5.7.1 nachzulesen. Es bietet sich an, diese Vorfälle ebenfalls genauer zu betrachten. Dazu werden die Vorhersagen der Modelle der Detektoren mit den üblichen Confidence-Schwellenwerten, 40 % für YOLOR-Modelle und 0,1 % für Mask R-CNN-Modelle, herangezogen.

Zunächst werden die Keramikgefäße, die zuvor mit `missing_labels` annotiert worden sind, genauer betrachtet. Dabei ist es schwierig, das tatsächlich gewünschte, optimale Verhalten der Detektoren zu bestimmen. Werden die *missing_label*-Keramikgefäße nicht

	YOLOR			Mask R-CNN		
Bild	**k. A.**	**A. 1**	**A. 2**	**k. A.**	**A. 1**	**A. 2**
ECA-1	■	■	■	■	■	■
ECA-4	■	■	■	■	■	■
ECA-5	■	■	■	■	■	■
ECA-6	■	■	■	■	■	■
ECB-2	■	■	■	■	■	■
GCA-1	■	■	■	■	■	■
GCA-2	■	■	■	■	■	■
GCB-2	■	■	■	■	■	■
	■	■	■	■	■	■
	■	■	■	■	■	■
GCB-3	■	■	■	■	■	■
TCA-2	■	■	■	■	■	■
TCB-2	■	■	■	■	■	■
TCB-3	■	■	■	■	■	■
TCD-1	■	■	■	■	■	■
TCD-3	■	■	■	■	■	■
TCE-1	■	■	■	■	■	■
TCE-2	■	■	■	■	■	■
	■	■	■	■	■	■
TCE-3	■	■	■	■	■	■

■	3	5	5	1	2	1
■	12	12	13	2	4	3
■	5	3	2	17	14	16

Tabelle 5.10: Die *Amphoriskos*-Vorhersagen der YOLOR- und Mask R-CNN-Modelle aller Augmentationsstufen auf den entsprechenden Validierungsbildern unterteilt in ■ = lokalisiert und korrekt klassifiziert, ■ = lokalisiert und falsch klassifiziert, ■ = nicht lokalisiert.

lokalisiert und klassifiziert, stimmen die Vorhersagen der Detektoren mit der Ground Truth-Annotation überein. Werden sie hingegen lokalisiert und klassifiziert, so kann es auch möglich sein, dass die Detektoren dazu in der Lage sind, getätigte Fehler bei der Ground Truth-Annotation auszumerzen. So ist beispielsweise insbesondere das Keramikgefäß am rechten Rand des mittleren Regalbretts des Bildes `TCD-2` innerhalb Fold 2 von den Modellen aller Augmentationsstufen beider Detektoren mit hoher Confidence als *Amphora* klassifiziert geworden. Aus der Sicht eines Laien sieht diese vorhergesagte Klassifikation korrekt aus. Noch mehr so mit als *Amphora* klassifizierten Objekten der Fotografie `GCB-2` der getätigten Ground Truth-Annotation. Beide entsprechenden Ausschnitte der Fotografien sind in Abbildung 5.10 dargestellt. Natürlich kann diese

Annahme aufgrund fehlender Expertise im Bereich der Archäologie auch inkorrekt sein.

Abbildung 5.10: Annotiertes *missing_label*-Keramikgefäß des Bildes `TCD-2` und als *Amphora* annotierte Objekte des Bildes `GCB-2` in blau[15].

Nichtsdestotrotz sind alle Vorhersagen der *missing_label*-Keramikgefäße durch die Detektoren der drei Augmentationsstufen zur zukünftigen Behandlung in der Tabelle A.4 im Appendix aufgeführt. Dabei fällt außerdem auf, dass YOLOR deutlich mehr der *missing_label*-Keramikgefäße klassifiziert, was mit Hinzunahme von augmentierten Bildern für die 1. und Augmentation 2 um jeweils drei ungleiche Objekte steigt.

Hingegen werden von den Mask R-CNN-Modellen weitaus weniger *missing_label*-Keramikgefäße klassifiziert, was jedoch auch daran liegen kann, dass Mask R-CNN grundsätzlich weniger Objekte lokalisiert und klassifiziert. Dabei ist die Augmentationsstufen betreffend zunächst für die Augmentation 1 ein *missing_label*-Keramikgefäß weniger klassifiziert worden als im Vergleich zu keiner Augmentation. Von der Augmentation 1 zur Augmentation 2 werden hingegen wieder zwei *missing_label*-Keramikgefäße mehr klassifiziert.

Ob dieses Verhalten nun als positiv oder negativ zu beurteilen gilt, ist aus Laiensicht nicht klar aufzuschlüsseln.

Ähnliche Bewertungseinschränkungen gelten für die vielen bereits von Beginn an unannotierten Objekte auf den Bildern. Im Gegensatz zu den *missing_label*-Keramikgefäßen sind sie jedoch häufiger von den Modellen beider Detektoren tatsächlich auch nicht lokalisiert und klassifiziert geworden.

Die Position der Objekte scheint dabei nicht ausschlaggebend zu sein, wie in Abbildungen 5.8 und 5.9 im Vergleich zu dem annotierten Bild in Abbildung 5.6 teilweise ersichtlich. Auch die Größe der Objekte in den Fotografien scheint keinen Einfluss auf

[15] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 9064 (`TCD-2`) und 8375 (`GCB-2`).

Abbildung 5.11: Annotationen und Vorhersagen der YOLOR-Modelle der Augmentation 2 mit Confidence von 40 % sowie der Mask R-CNN-Modelle der Augmentation 1 mit Confidence von 0,1 % der Verkaufsbilder `GCA-1` und `TCA-1`[16] mit vielen unannotierten Objekten.

die korrekte Nicht-Klassifizierung der in der Annotation unannotierten Objekte zu nehmen. Dies wird durch das Auftreten von kleinen und großen Objekten, die durch die Vorhersagen der Detektoren korrekterweise unklassifiziert bleiben, deutlich.

[16] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 9050 (`GCA-1`) und 8299 (`TCA-1`).

Abbildung 5.11 zeigt diesen Umstand anhand der Annotationen sowie Vorhersagen der Modelle zweier Fotografien mit vielen unannotierten Objekten verschiedener Positionen und Größen auf.

Stattdessen scheint es eine andere Eigenschaft der Objekte, sehr wahrscheinlich ihre Form, zu sein, durch die die Detektoren sie unklassifiziert lassen. Dies scheint sich nach einem groben manuellen Vergleich auch nicht zwischen den verschiedenen Augmentationsstufen zu verändern. Dabei ist jedoch zu erwähnen, dass die Modelle von Mask R-CNN prinzipiell weniger Objekte erkennen und diesbezüglich natürlich auch wenige unannotierte Objekte klassifizieren.

5.7.6 Gegenüberstellung der Detektoren

Schließlich werden die erhaltenen Ergebnisse der YOLOR- und Mask R-CNN-Modelle der unterschiedlichen Augmentationsstufen für die Erkennung und Lokalisierung der Keramikgefäße der beiden Kapitel 5.7.3 und 5.7.4 gegenübergestellt. Vergleicht man zunächst den durchschnittlichen mAP-Wert der jeweils besten Augmentationsstufe, so sind die YOLOR-Modelle der Augmentation 2 mit 0,551 denen der Augmentation 1 von Mask R-CNN mit 0,37 um 18,1 % überlegen. Dabei beträgt unter den in Kapitel 5.1 festgelegten Hardwarevoraussetzungen die durchschnittliche Trainingsdauer der entsprechenden Augmentationsstufen der YOLOR-Modelle ca. 7 h und die er Mask R-CNN-Modelle ca. 6 h.

Des Weiteren fällt auf, dass YOLOR für deutlich mehr Keramikgefäße Bounding Boxen vorhersagen kann und in diesem Zuge auch mehr Keramikgefäße klassifiziert. Dies ist besonders bei Fotografien zu beobachten, die viele, kleine Keramikgefäße abbilden.

Dies deckt sich ungefähr mit den im früheren Versuch ohne Cross Validation erzielten Ergebnissen, die innerhalb Tabelle A.1 im Appendix festgehalten worden sind. Damit ergibt sich eindeutig, dass dem Implementierungskonzept des Kapitels 4.5 folgend mit den YOLOR-Modellen der Augmentation 2 weitergearbeitet und die Größenapproximation unterstützt wird.

5.8 Größenapproximation der Keramikgefäße

Zum Schluss folgt die tatsächliche Größenapproximation der einzelnen Keramikgefäße basierend auf den zuvor erzielten Ergebnissen der Kapitel 5.6 und 5.7 sowie dem Implementierungskonzept 4.5. Sowohl die Erlangung der Koordinaten der Bounding Boxen des Lineals und der Keramikgefäße als auch die Umrechnung der Pixel in Zentimeter wird dabei vorgenommen. Zusätzlich gilt es, eine Größenapproximation trotz nicht vollständig abgebildeter Lineale der Herausforderung **HL2** und gänzlich fehlender Li-

neale der Herausforderung **HL1** möglichst sinngemäß durchzuführen. Die Ergebnisse der Größenapproximation werden schließlich auf den entsprechenden Bildern dargestellt und in einer CSV-Datei gespeichert, ehe sie manuell anhand der tatsächlichen Größenmaße weniger Objekte evaluiert werden.

5.8.1 Erhalt der Koordinaten der Bounding Boxen

Zur Durchführung einer Größenapproximation sind zunächst die Koordinaten der Bounding Boxen sowohl von dem Lineal als auch von den einzelnen, erkannten Keramikgefäßen nötig. Sie werden verwendet, um die Höhe und Breite der Bounding Boxen in Pixeln zu erhalten, welche schließlich in einem nächsten Schritt in Zentimeter übersetzt werden. Der Erhalt der Koordinaten beider Typen wird im Folgenden beschrieben.

Koordinaten des Lineals

Wie in Kapitel 4 dargelegt werden keine in Kapitel 5.6 angewandten Objekterkennungsdetektoren zur automatischen Lokalisierung des Lineals und seinen Koordinaten zum Einsatz kommen, um seine Gesamtlänge zur Approximation der Kermaikgefäße zu nutzen. Zusätzlich zu den sowieso angeführten Argumenten motivieren auch die in den Kapiteln 5.6.2, 5.6.3 und 5.6.4 ermittelten Ergebnisse jene Entscheidung, die zwar besser als erwartet, aber nicht gut genug sind, um eine den realen Werten möglichst nahe Approximation durchzuführen.

Entsprechend wird die sowieso durch Kapitel 5.6.1 vorhandene Annotation des Lineals im COCO-Format herangezogen. Dazu wird die Annotation nicht einfach verwendet, sondern im Hinblick auf nicht vollständig abgebildete Lineale der Herausforderung **HL2** angepasst. Davon sind 16 der insgesamt 37 zu verwendenden Scans von Fotografien von Keramiksammlungen mit Lineal betroffen. Diese Fotografien enthalten **HL2.1** und **HL2.2** folgend Lineale, deren Länge sowie Länge und Breite über den Bildrand hinausragen, wodurch die gewollte, eigentlich einfache Nutzung der Gesamtlänge des Lineals zur Approximation erschwert wird. Aus diesem Grund wird nun die Annotation der Lineale jener Fälle manuell über den Bildrand hinaus erweitert: Es werden Bounding Boxen über den Bildrand hinausgezogen, die die Gesamtlänge des Lineals in Abhängigkeit des tatsächlich auf den Fotografien abgebildeten Teils nachempfinden.

Leider ermöglichen CVAT, aber auch andere Annotationstools wie LabelMe oder makesense.ai[17], jedoch nicht das Ziehen von Bounding Boxen über den Bildrand hinaus. Stattdessen werden die 16 Bilder manuell in die Richtungen vergrößert, beziehungsweise mit schwarzen Balken gefüllt, in die die Lineallänge bei voller Abbildung passen würde.

[17] makesense.ai, `https://www.makesense.ai/`, besucht am 28.08.2022

Dies kann kurzerhand mit einem beliebigen Bildbearbeitungsprogramm mit Augenmaß vorgenommen werden. Anschließend werden die entsprechenden erweiterten Bilder in CVAT mit neuen Bounding Boxen annotiert.

Die Annotation kann schließlich über folgendes einfaches Vorgehen umgesetzt werden: Dazu wird zunächst um einen Teil des Lineals, zum Beispiel 10 cm, eine Bounding Box gezogen. Anschließend wird diese Bounding Box in Richtung der nicht abgebildeten Seite am Lineal angelegt, zum Beispiel mit der linken Kante bei 40 cm, um nicht abgebildete 50 cm zu erreichen. Dies wird so lange wiederholt, bis das Lineal anhand seines abgebildeten Bereichs und den Bounding Boxen komplett abgebildet wird. Schließlich kann eine einzelne Bounding Box über den gesamten Bereich gezogen werden und alle anderen Hilfs-Boxen gelöscht werden.

Sind nun alle 16 Bilder annotiert, wird die neue Annotationsdatei heruntergeladen und verwendet, um entsprechende Werte in der vorherigen Annotationsdatei aus Kapitel 5.6.1 zu ersetzen. Dies betrifft nicht nur den Key `bbox` innerhalb `annotations` eines jeden Bildes, sondern auch die neuen Werte für `width` und `height` innerhalb des `images`-Keys.

Mit Hilfe der neuen Annotationsdatei wird schließlich die Länge in Pixeln eines jeden Lineals pro Bild ermittelt. Dabei kommt die Funktion `get_ruler_length` zum Einsatz, die anhand ihrer Parameter die entsprechende Datei lädt und schließlich über den entsprechenden Bildnamen sowie der damit einhergehenden `ID` in der Annotationsdatei auf die zu dem Bild gehörige Annotation, beziehungsweise Bounding Box-Koordinaten in `bbox` zugreifen kann. Die tatsächliche Länge des Lineals lässt sich dann über den größeren Wert des dritten und vierten Eintrags, namentlich `w` und `h` der Liste der Koordinaten des bereits mehrfach erwähnten COCO-Exportformats identifizieren und zurückgeben. Offensichtlich gibt der größere Wert an dieser Stelle die Lineallänge in Pixeln an.

Koordinaten der Keramikgefäße

Um nun auch die benötigten Koordinaten der einzelnen Bounding Boxen der Keramikgefäße zu erhalten, werden die Erkenntnisse aus Kapitel 5.7 verwendet. Bereits in Kapitel 4 ist aufgeführt worden, dass an dieser Stelle eine andere Methode als die Verwendung von Deep Learning-Methoden keinerlei Sinn ergibt. Entsprechend werden hier die in Kapitel 5.7.3 trainierten YOLOR-Modelle angewendet, um zunächst die Bounding Boxen der einzelnen Gefäße vorherzusagen. Es wird sich insbesondere für die Modelle der Augmentation 2 entschieden, da diese, wie in Kapitel 5.7.3 und 5.7.6 evaluiert, die besten Ergebnisse erzielen.

Zur Vorhersage der Bounding Boxen kommt der in Kapitel 5.4 vorgestellte Befehl

über das Skript `/yolor/detect.py` mit der Flag `save-txt` sowie `conf 0.4` zum Einsatz. Sowohl die Bilder als auch die Weights der YOLOR-Modelle werden dabei entsprechend durch alle fünf trainierten Folds passend zu ihren Validierungsbildern iteriert. Damit werden die Detektionen nun nicht nur auf den Bildern dargestellt, sondern ihre tatsächlichen Werte pro Bild in einer Textdatei im YOLO-Exportformat pro Bild vermerkt.

Als nächstes kann die Funktion `yolo_to_coco` verwendet werden. Diese utilisiert nun die entsprechenden Textdateien, um pro Zeile die vorhergesagte Klassen-ID, also den ersten Wert, sowie die YOLO-Koordinaten der Bounding Boxen, also die restlichen vier Werte, zu extrahieren. Über die für YOLOR stets nötige Datei `data.names` kann schließlich die Klassen-ID in ihren tatsächlichen Namen übersetzt werden.

Anschließend werden die YOLO-Koordinaten in COCO-Koordinaten umgewandelt und in eine Liste gespeichert. Die Übersetzung ergibt sich folgendermaßen [Gos21]:

$$
\begin{aligned}
w_c &= w_y \cdot w \\
h_c &= h_y \cdot h \\
x_c &= x_y \cdot w - (w_c/2) \\
y_c &= y_y \cdot h - (h_c/2)
\end{aligned}
$$

Dabei stellt θ_c die entsprechende COCO-Koordinate und θ_y die YOLO-Koordinate dar. Des Weiteren sind w und h die Breite sowie Höhe des entsprechenden Bildes. Beide Exportformate sind mit einem beispielhaften Ausschnitt innerhalb Kapitel A.3 im Appendix festgehalten, wodurch die Umwandlung verdeutlicht werden kann. Schließlich werden sowohl die neuen COCO-Koordinaten der Bounding Box als auch die entsprechen Klassennamen zurückgegeben.

5.8.2 Umgang mit fehlendem Lineal

Ist nun gemäß Herausforderung **HL1** kein Lineal abgebildet, muss ein anderes Referenzobjekt der Größenapproximation dienen. Bereits in Kapitel 4.5 ist sich diesbezüglich für die augenscheinlich auf fast jeder Fotografie abgebildeten, größtenteils gleichen Regalbretter entschieden worden. Die Dicke eines Regalbretts lässt sich über Fotografien, die ein Lineal abbilden, manuell approximieren. Folglich wird sie allgemein auf 1,4 cm geschätzt. Zunächst gilt es jedoch zu erkennen, ob überhaupt ein Lineal in einer Fotografie abgebildet wird oder nicht.

Dazu wird die in dem vorgegangenen Kapitel 5.8.1 beschriebene Funktion `get_ruler_length` erweitert. Dies basiert auf der Grundlage, dass alle Fotografien mit Lineal in der erweiterten Annotationsdatei des Lineals zu finden sind. Um dies zu überprüfen,

wird ein Zähler `amount` eingefügt, der jeden betrachteten Eintrag in `images` der erweiterten Annotationsdatei zählt. Existiert ein Eintrag mit einem Lineal, so werden, wie in Kapitel 5.8.1 beschrieben, seine Annotation in Form von Bounding Box-Koordinaten anhand der generierten Bild-ID der Annotationsdatei ermittelt, die Schleife unterbrochen und entsprechende Werte sowie 0 zurückgegeben. Erreicht `amount` nun aber die Anzahl an Einträgen innerhalb des `images`-Key und ist eben dieser letzte Eintrag nicht gleich dem aktuell zu bearbeitenden Bild, so werden 1 und die Ergebnisse der neuen Funktion `no_ruler` zurückgegeben.

`no_ruler` öffnet ein Fenster mit dem spezifische Bild ohne Lineal nun mit dem Python-Package OpenCV, beziehungsweise `cv2`. Ist das Bild nun an einer oder beiden Seiten größer als der Desktop, mit dem es betrachtet werden soll, kann es nicht ganz dargestellt werden und muss runterskaliert werden. Dies wird mittels `cv2.resize` und der Variable `resize_factor` übernommen. Letztere ergibt sich entweder durch die Breite des Desktops geteilt durch die Breite des Bildes oder analog durch die Höhe des Desktops geteilt durch die Höhe des Bildes. Dabei gilt es stets jene Bildseite als Faktor zu verwenden, die im Verhältnis größer als die entsprechende Desktopseite ist. Ansonsten würde das Bild nicht genug runterskaliert werden, sodass ein Teil eben jener im Verhältnis größeren Seite zur Desktopseite weiterhin nicht dargestellt werden kann.

Insbesondere kann an dieser Stelle nicht lediglich der größere Wert der Bildhöhe und -breite gewählt werden, da Desktops in der Regel eine größere Breite als Höhe besitzen. Entsprechend könnte ein Bild, dessen Breite minimal größer ist als seine Höhe, zwar in der Breite korrekt abgebildet werden, in seiner Höhe jedoch durch die Ränder des Desktops abgeschnitten werden.

Schließlich kann der `resize_factor` verwendet werden, um mittels Multiplikation sowohl die Bildbreite als auch die Bildhöhe passend mit `cv2.resize` zu skalieren. Es ergibt sich folgender Code, wobei `img_shape[1]` die Bildbreite und `img_shape[0]` die Bildhöhe darstellt.

```
# if the image is bigger than the screen, take action
if img_shape[1] > screen_w or img_shape[0] > screen_h:

    # look at ratio of screen size to image size
    if (screen_w/img_shape[1]) < (screen_h/img_shape[0]):
        resize_factor = screen_w/img_shape[1]

    else:
        resize_factor = screen_h/img_shape[0]
    no_ruler_img = cv2.resize(no_ruler_img, (int(img_shape[1] *
```

```
        resize_factor), int(img_shape[0] * resize_factor)))
```

Darüber hinaus wird ein `setMouseCallback` auf jenes geöffnete Fenster gesetzt, wodurch Benutzer*innen über zwei Klicks mit der rechten Maustaste erlaubt wird, zwei Punkte zu setzen, die die Regalbrettdicke markieren sollen. Dazu wird die Funktion `draw_circle` aufgerufen, die entsprechende Punkte nicht nur rot markiert, sondern auch ihre Koordinaten in eine Liste einfügt und die Klicks zählt. Die Koordinaten werden dabei wieder durch den `resize_factor` geteilt, um ihre tatsächlichen Werte ohne jegliche Skalierung zu erhalten.

Werden überdies in der Funktion `no_ruler` nun zwei Klicks vermeldet, so wird kein anderer Klick erlaubt und das Fenster geschlossen. Schließlich wird die Distanz zwischen beiden durch Benutzer*innen erhaltenen Koordinaten in Pixeln über die allseits bekannte Formel $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ berechnet und zurückgegeben.

5.8.3 Umrechnung von Pixel in Zentimeter

Somit kann die Höhe und Breite in Pixeln der einzelnen Bounding Boxen der Keramikgefäße mit denen der erhaltenen Lineallänge oder Regalbrettdicke verwendet werden. Da die tatsächliche Größe der Lineallänge und Regalbrettdicke bekannt ist, kann auch die tatsächliche Höhe und Breite der Keramikgefäße ermittelt werden.

Diesbezüglich wird über die durch die Funktion `yolo_to_coco` erhaltene Liste an Bounding Boxen im COCO-Exportformat sowie Klassen der erkannten Objekte, beschrieben in Kapitel 5.8.1, iteriert. Ferner wird der Rückgabewert der Funktion `get_ruler_length` aus Kapiteln 5.8.1 und 5.8.2 verwendet, um aufzudecken, ob das entsprechend in seinen Objektgrößen zu approximierende Bild ein Lineal enthält oder nicht. Enthält es ein Lineal, so hat der Rückgabewert der Funktion `get_ruler_length` den Wert 0 an erster Stelle, falls nicht den Wert 1. Unter diesen Voraussetzungen wird nun die Funktion `calculate_size` mit der Lineallänge oder Regalbrettdicke in Pixeln, den Bounding Boxen der Keramikgefäße und den als Referenz dienenden Zentimetern aufgerufen. Im Falle des Lineals beträgt der letzte Wert 50, während er ohne Lineal die approximierten 1,4 Zentimeter der Regalbrettdicke beträgt.

Die Funktion `calculate_size` verwendet schließlich jene übergebenen Werte, um $(\text{Referenz}_{cm}/\text{Referenz}_{Pixel}) \cdot \text{Gefäß}_{Pixel}$ für die entsprechende Höhe oder Breite der Objekte in Zentimetern zu berechnen und zurückzugeben. Dabei stellt Referenz die Lineallänge oder Regalbrettdicke dar.

5.8.4 Ausgabe der Ergebnisse

Zuletzt sind die erhaltenen Ergebnisse lediglich adäquat auszugeben. Zum einen wird dafür eine CSV-Datei mit Kopf `Image, Object ID, Class, Width, Height, BBox_x, BBox_y, Bbox_w` und `Bbox_h` generiert, die pro Zeile die entsprechenden Werte der einzelnen Keramikgefäße pro Bild enthält. Die Objekthöhe und -breite werden dabei in Zentimetern angegeben und entsprechend auf die erste Nachkommastelle gerundet. Überdies werden die einzelnen Objekte pro Bild zufällig mit einer aufsteigenden ID versehen, um die Werte schließlich auch einem bestimmten Objekt auf den Bildern zuordnen zu können. Werden jene IDs durch die Labels oder Bounding Boxen anderer Objekte überdeckt, so können die ebenfalls gespeicherten Koordinaten der Bounding Box eines Objekts verwendet werden, um es auf dem Bild zu identifizieren.

Um dies tatsächlich tun zu können, sollen die erhaltenen Werte nun auch auf den einzelnen Bildern abgebildet werden. Aus diesem Grund wird die Funktion `display_size` definiert. Diese zieht eine Bounding Box über das gerade behandelte Objekt eines Bildes, indem es die durch die Funktion `yolo_to_coco` erhaltenen Koordinaten und `cv2.rectangle` verwendet. Zusätzlich werden sowohl die zuvor erwähnte Objekt-ID als auch die Größenmaße in Zentimetern über jener Bounding Box dargestellt, indem `cv2.rectangle` und `cv2.putText` verwendet werden. Ersteres dient dabei nur der besseren Lesbarkeit als Unterlage des Textes. Schließlich werden die entsprechenden Bilder mittels `cv2.imwrite` in einem Ordner auf dem Rechner gespeichert.

Abbildung 5.12 zeigt die auf zwei Fotografien illustrierte Größenapproximation. Die sich teilweise überlagernden Label bestimmter Objekte verdeutlichen den Sinn der Objekt-IDs sowie der CSV-Datei, die alle relevanten Werte der Objekte enthält.

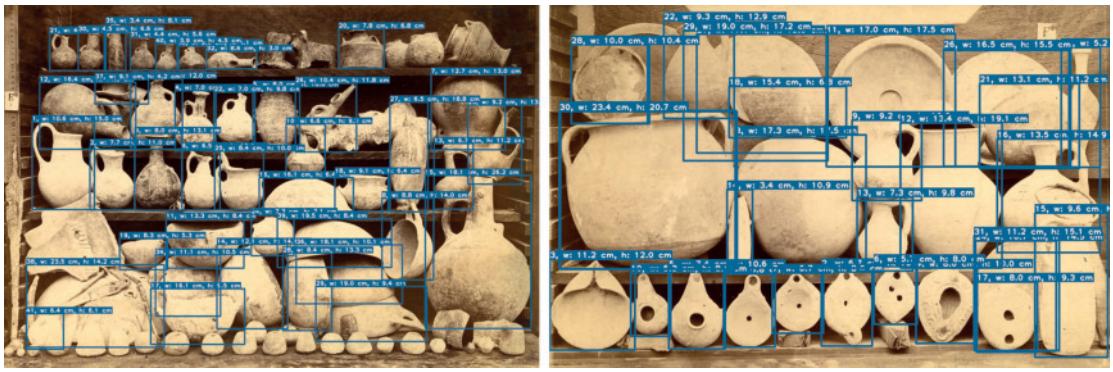

Abbildung 5.12: Ausgabe der approximierten Größenmaße über den vorhergesagten Bounding Boxen auf der Fotografie `TCE-1` mit und `TCF-3` ohne Lineal[18].

[18] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 9067 (`TCE-1`) und 9114 (`TCF-3`).

5.8.5 Evaluation anhand tatsächlicher Größenmaße

Letztlich bietet es sich an, die Größenapproximation der Keramikgefäße anhand ihrer tatsächlich durch das Berliner Museum für Vor- und Frühgeschichte gemessenen Größenmaße zu evaluieren. Da einige Keramikgefäße nicht vorliegen oder gar während des Krieges beschädigt worden sind, und viele Maße zum Zeitpunkt der Masterthesis noch nicht vollständig digitalisiert und den Bildern zugeordnet werden konnten, werden lediglich 12 Keramikgefäße von 9 Fotografien zur Evaluation verwendet. Dies stellt jedoch kein Problem dar, denn die Evaluation kann aufgrund der Schwierigkeit der automatisierten Zuordnung der Keramikgefäß-IDs der Größenapproximation zu den tatsächlichen IDs des Museums nur manuell vorgenommen werden. Dennoch sollten die folgenden Ergebnisse nicht als vollumfänglich angesehen werden. Die identifizierten Herausforderungen **HG5** sowie **HK1** des Kapitels 4.2 machen eine manuelle Herangehensweise aufgrund der verschiedenen Ausrichtung der Objekte, der Zweidimensionalität der Größenapproximation und den möglichen spezifischen Maßen unerlässlich.

Die Tabelle 5.11 fasst den Vergleich mit den tatsächlichen Daten der spezifischen Keramikgefäße zusammen. Dabei wird nicht nur der Bildname und die Fold, in der sich das entsprechende Bild zur Validierung befindet angegeben, sondern auch die tatsächliche und durch die Größenapproximation generierte ID[19] sowie die Breite und Höhe in Zentimetern der Gefäße. Insbesondere jene tatsächlichen Größenmaße, die mit einem Asterisk versehen sind, haben nicht vorgelegen und sind stattdessen über einen anderen tatsächlichen Wert sowie die entsprechenden Pixel des Bildes manuell approximiert worden. Entsprechend sind diese Werte höchstwahrscheinlich nicht völlig korrekt und mit Vorsicht zu betrachten. Überdies sind tatsächliche und approximierte Höhen- und Breitenmaße aufeinander abzustimmen, denn sie beschreiben nicht unbedingt dasselbe Maß eines Keramikgefäßes. So zeigt Abbildung 5.13 einen liegenden *Jug*, dessen Höhe auf der Fotografie tatsächlich seinen Durchmesser darstellt. Analoges gilt für die tatsächliche Höhe und die Breite auf der Fotografie.

Für die Tabelle 5.11 sind die tatsächlichen Größenmaße aus diesem Grund der Einfachheit halber den approximierten Größenmaßen zugeordnet worden. Somit sind in der Tabelle 5.11 die auf Abbildung 5.13 vermerkte Breite sowie Größe auch als genau jene Breite sowie Größe dargestellt und die tatsächlichen Maße an sie angepasst. Dies folgt viel mehr dem Grund, dass die tatsächlichen Größenmaße nicht unbedingt die Gesamtbreite oder -höhe darstellen, sondern auch andere, unterschiedliche Größenmaße, wie beispielsweise der Durchmesser im Fall von Abbildung 5.13. Eine umgekehrte Zuordnung der approximierten zu den tatsächlichen Größenmaßen kann jedoch unter anderen

[19] Nach der Nummerierung des der Thesis bereitgestellten PDFs `Sammlung_MOR_NUMMERN`, welches ebenfalls im Rahmen der praktischen Implementierung, deren Zugang sich in Kapitel A.1 befindet, hochgeladen wurde.

		ID		Klasse			Breite		Höhe		
Fold	**Bild**	**Real**	**Approx.**	**Real**	**Annot.**	**Approx.**	**Real**	**Approx.**	**Real**	**Approx.**	**ΔA**
1	ECA-5	1	12	Amphoriskos	Amphora	Bottle	9,7* cm	8,9 cm	13,3 cm	11,8 cm	18,6 %
		2	14	Mini-Jug	Juglet	Jug	5,1* cm	4,8 cm	7 cm	6,1 cm	18 %
		14	4	Juglet	Juglet	Jug	8 cm	6,9 cm	10 cm	7,9 cm	31,9 %
	TCE-4	1	5	Jug	Jug	Jug	50 cm	44,8 cm	33 cm	27,2 cm	26,1 %
	TCF-3	20	3	Oil Lamp	Oil Lamp	Oil Lamp	11,4 cm	11,2 cm	9,9 cm	12 cm	19,1 %
2	TCE-1	23	1	Juglet	Jug	Jug	8,1* cm	10,6 cm	12* cm	15 cm	63,6 %
3	L-3	1	3	Oil Lamp	Oil Lamp	Oil Lamp	6,4 cm	6,3 cm	9 cm	9 cm	1,6 %
4	ECA-1	32	2	Jug	Jug	Jug	8,5 cm	8,4 cm	16,2 cm	16,5 cm	0,6 %
	TCD-1	63	48	Bowl	Bowl	Jug	11,1 cm	12,4 cm	6,8 cm	8 cm	31,4 %
5	ECA-4	8	11	Bowl	Bowl	Bowl	11 cm	9,8 cm	12,8 cm	9,9 cm	31,1 %
		16	20	Skyphos	Bowl	Bowl	9 cm	8,8 cm	6,9 cm	7,2 cm	2 %
	GCA-3	36	17	Oil Lamp	Oil Lamp	Oil Lamp	6,2 cm	6,1 cm	7,6 cm	7,8 cm	1 %
											ø ≈ 20,4 %

Tabelle 5.11: Gegenüberstellung der tatsächlichen und approximierten Größenmaße einiger Keramikgefäße sowie die prozentuale Differenz ihres Flächeninhalts und ihre annotierte, tatsächliche und vorhergesagte Klasse mit ■ = Fotografie ohne Lineal.

Gesichtspunkten in der Praxis sinnvoller sein.

Abbildung 5.13: Ausgabe der approximierten Größenmaße des liegenden *Jugs* des Bildes `TCE-4`[19].

Zusätzlich sind auch die tatsächliche, die für die Thesis annotierte und die von YOLOR vorhergesagte Klasse des Keramikgefäßes in der Tabelle angegeben. Die tatsächlichen Klassen liegen dabei auf Deutsch vor und wurden grob übersetzt. Im Gegensatz dazu liegen die anderen Klassen auf Englisch vor. Es fällt auf, dass nicht jede tatsächliche Klasse mit der annotierten Klasse übereinstimmt und insbesondere Klassen wie *Skyphos* vorkommen, die nicht Teil der für die Masterthesis übermittelten Klassen ist.

Schließlich werden die prozentualen Differenzen der tatsächlichen und approximierten Breite und Höhe in Form des Flächeninhalts ihrer Bounding Box ausgerechnet. Es ergibt sich eine durchschnittliche Differenz von ca. 20,4 %. Dabei ist auffällig, dass die Größe von bestimmten Gefäßen, wie der *Oil Lamp* des Bildes `L-3` mit der approximierten ID 3, besonders gut, aber ebenso Gefäße, wie die *Jug* des Bildes `TCE-4` mit der approximierten ID 5, eher schlecht approximiert werden. Dabei werden beide Gefäße ziemlich gut von ihren vorhergesagten Bounding Boxen umschlossen. Auf beiden zugehörigen Bildern wird jedoch kein Lineal abgebildet, sodass die tatsächliche Länge in Pixeln durch die Regalbrettdicke, die von Benutzer*innen händisch gesetzt werden muss, approximiert worden ist. Die durchschnittliche Differenz der Flächeninhalte anderer Gefäße auf Bildern ohne abgebildetes Lineal ergibt 19,2 %. Hingegen kann von Bildern mit einem abgebildeten Lineal eine durchschnittliche Differenz der Flächeninhalte von 21,6 % erreicht werden. Dadurch lässt sich jedoch schließen, dass die Regalbrettdicke sehr wohl als Linealersatz mit den in 5.8.2 approximierten tatsächlichen Werten herangezogen werden kann, wenn kein Lineal abgebildet wird, wodurch Herausforderung **HL1** für die vorliegenden Bilder als gelöst gilt. Dabei ist jedoch zu beachten, dass darauf nur auf Grundlage der wenigen Keramikgefäße mit übermittelten tatsächlichen Größenmaße geschlossen wird und sich

[19] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 8319 (`TCE-4`).

diese Tatsache unter Evaluation weiterer Gefäße ändern kann.

Die größere Herausforderung liegt klar bei **HK1**. Durch die nicht gerade Positionierung der Gefäße zur Kameralinse, kann es zu Abweichungen zwischen den tatsächlichen und approximierten Größenmaßen von mehreren Zentimetern kommen. Im Gegensatz dazu beläuft sich die Abweichung bei möglichst gerade zur Kamera positionierten Gefäßen nur auf wenige Millimeter. Abbildung 5.14 zeigt zwei derartige Beispiele. Auch welches tatsächliche Größenmaß schließlich zu welchem approximierten Größenmaß passt, hängt völlig von der Ausrichtung der Keramikgefäße ab und bedarf daher manueller Eingriffe.

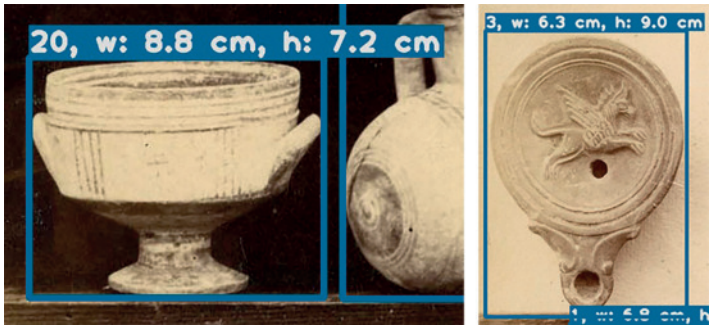

Abbildung 5.14: Ausgabe der approximierten Größenmaße zweier unterschiedlich zur Kamera ausgerichteten Objekte der Bilder `ECA-4` und `L-3`[20].

Des Weiteren können natürlich selbst im Fall möglichst korrekt positionierter Keramikgefäße nicht alle Größenmaße anhand des Bildes approximiert werden. So fällt beispielsweise die eigentliche Höhe der *Oil Lamp* mit approximierter ID 17 der Fotografie `GCA-3` weg und lediglich ihre Länge und Breite können anhand des Bildes als zweidimensionale Höhe und Breite approximiert werden.

Nichtsdestotrotz kann anhand des vorliegenden Anwendungsfalles und den zu verwendenden Ressourcen eine solide Größenapproximation der Gesamthöhe und -breite der Keramikgefäße durchgeführt werden.

5.9 Abschließende Evaluation der Herausforderungen

Schließlich werden die in Kapitel 4.2 identifizierten Herausforderungen anhand der in den vorherigen Kapiteln 5.6, 5.7 und 5.8.5 evaluierten Ergebnisse einer abschließenden Evaluation im Hinblick auf ihre Bewältigung und Folgenschwere[21] unterzogen. Dazu werden zunächst Herausforderungen aufgeführt, die mittlerweile als bewältigt oder wenig

[20] Vgl. die Scans der Verkaufsbilder von Max Ohnefalsch-Richter im Archiv des Berliner Museums für Vor- und Frühgeschichte [LOE21], Inv. Nr. 8277 (`ECA-4`) und 8356 (`L-3`).

[21] D.h. wie schwerwiegend die Herausforderungen tatsächlich für die Detektoren sind.

folgenschwer gelten. Es folgen Herausforderungen, die weder als stark, noch als gering folgenschwer gelten. Abschließend werden jene Herausforderungen aufgeführt, die als sehr folgenschwer und damit problematisch identifiziert werden können.

Sowohl Herausforderung **HL1** als auch Herausforderung **HL2** sind im Rahmen der Masterthesis bewältigt worden. Die Verwendung der Regalbrettdicke als substituiertes Referenzobjekt zur Größenapproximation im Sinne von **HL1** ist in Kapitel 5.8.5 aufgrund der geringen durchschnittlichen Differenzen der Flächeninhalte der approximierten und tatsächlichen Größe als gut erachtet worden; dies betrifft jedoch nur die 12 evaluierten Keramikgefäße. Ebenso kann die Wahl der Verwendung der Lineallänge aufgrund jener Werte positiv bewertet werden. Diesbezüglich gilt außerdem die manuelle Erweiterung der wenigen nicht vollständig abgebildeten Lineallängen nach **HL2** im Sinne einer Kosten-Nutzen-Analyse als effizient. Eine automatisierte Umsetzung wäre demgegenüber zur Erzielung derselben Lösung deutlich aufwendiger, aber sicherlich auf Basis der in Kapitel 4.3 aufgeführten grundlegenden Gedanken zu bewältigen.

Da das Lineal meist ziemlich gerade zum Bildrand angeordnet ist und damit zu den gezogenen, achsenorientierten Bounding Box-Annotationen passt, ist außerdem Herausforderung **HL3** wenig relevant.

Stattdessen ordnen sich die Herausforderungen **HG5**, **HG6** und **HK2** im Mittelfeld hinsichtlich ihrer Bewältigung und Folgenschwere an. Wie bereits in Kapitel 5.7.5 aufgegriffen, ist hier insbesondere **HK2** hervorzuheben. Soweit eine tatsächliche Bewertung als positiv oder negativ im Sinne des Kapitels überhaupt möglich ist, sind beide Ergebnisse der Detektoren YOLOR und Mask R-CNN eher als gut zu verstehen. Im Hinblick auf die restlichen beiden Herausforderungen sind die Objekterkennungsdetektoren unter bestimmten Umständen gut in ihrer Bewältigung, unter anderen Umständen jedoch weniger gut. Speziell im Falle der Keramikgefäßerkennung und -lokalisierung ist YOLOR gut darin, auch teilweise verdeckte Keramikgefäße nach **HG6** zu lokalisieren. Demgegenüber steht Mask R-CNN, welches prinzipiell weniger Objekte, diesbezüglich allerdings auch einige verdeckte, lokalisiert. Dasselbe gilt für unterschiedlich ausgerichtete Keramikgefäße nach **HG5**. Im Fall der Lineallokalisierung erscheint dies für alle drei getesteten Detektoren schwieriger, was unter anderem an der im Vergleich, beispielsweise zur Keramikgefäßklasse *Jug*, geringeren Anzahl an Linealen in der Datenbasis liegen kann.

Generell kann davon ausgegangen werden, dass alle drei Herausforderungen durch eine größere Datenbasis als deutlich weniger folgenschwer angesehen werden könnten.

Ähnliches gilt für fast alle folgenden schwerwiegenden Herausforderungen, weshalb die

kleine Datenbasis betreffende Herausforderung **HG1** hier an erster Stelle angeführt wird. Zwar hat insbesondere die Lineallokalisierung mit lediglich 37 Bildern und damit 37 Linealen eine bessere Genauigkeit als erwartet erzielt, jedoch noch keine, die auch ohne die aufgeführten Problematiken der Lösung für **HL2** in Kapitel 4.3 für eine sinnige Größenapproximation zum Einsatz kommen könnte. Durch eine größere Datenbasis ähnlicher Fotografien wären möglicherweise andere Ergebnisse zu erwarten.

Im Fall der Erkennung und Lokalisierung der Keramikgefäße ist dies durch eine künstliche Aufblähung des Datensatzes mit Data Augmentation getestet worden. Dabei kann für YOLOR eine Verbesserung des durchschnittlichen mAP-Wertes von 6,3 % durch eine Hinzunahme von 242 augmentierten Bildern beobachtet werden. Jedoch ist fraglich, inwieweit jener Wert noch zu steigern ist – insbesondere wenn nach Kapitel 4.5 und 5.7.2 möglichst sinnige augmentierte Versionen geschaffen werden sollen, die die Wahl von Augmentoren stark limitieren. Für den Detektor Mask R-CNN kann keine derartige Verbesserung nachgewiesen werden. An dieser Stelle bietet sich die Untersuchung der Hinzunahme von anderen, aber ähnlichen Fotografien an, die zum Zeitpunkt der Masterthesis jedoch nicht vorliegen.

Entsprechend ist **HG1** trotz getätigter Versuche und insbesondere aufgrund der Abhängigkeit anderer Herausforderungen zu ihr weiterhin eine schwerwiegende Herausforderung.

Davon sind auch Herausforderungen **HG2**, **HG3**, **HG4** sowie **HL4** betroffen. Im Falle fehlender Farben von **HG2** und mangelnder Bildqualität von **HG3** ist ihr tatsächlicher Einfluss zwar nur schwierig zu messen, jedoch klar aufgrund der fehlerhaften Vorhersagen von Regalbrettern, Objekten und anderem als Lineal innerhalb Kapitel 5.6 beobachtet worden. Durch diesen Umstand wird außerdem die Verwechslung anderer Objekte mit dem Lineal nach Herausforderung **HL4** unterstrichen.

Da die Erkennung und Lokalisierung der Keramikgefäße prinzipiell schwieriger manuell zu evaluieren ist, kann in diesem Fall für **HG2** und **HG3** keine spezifische Aussage getroffen werden. Jedoch ist deutlich, dass zumindest die hohe Anzahl an abgebildeten Objekten nach Herausforderung **HG4** einen schlechten Einfluss auf die Genauigkeit der Detektoren zur Keramikgefäßerkennung und -lokalisierung nimmt, wie in Kapitel 5.7 gezeigt.

Schließlich bleibt auch Herausforderung **HK1** eine schwerwiegende Herausforderung, die nicht nur eine automatisierte Evaluation im Rahmen der Masterthesis unmöglich gestaltet, sondern auch eine manuelle nach Kapitel 5.8.5 erschwert. Größenmaße, die über die in den Fotografien als Gesamtbreite und -höhe verstandenen Maße hinausgehen, sind im Nachhinein nicht mehr aus jenen Fotografien zu ermitteln. Die Hinzunahme anderer Fotografien, möglicherweise getätigt durch die Museen, die die entsprechenden Objekte beherbergen, wäre zwar möglich, aber weit vom aktuellen Anwendungsfall entfernt.

6
Resümee und Fazit

Ziel der vorliegenden Thesis war es, mittels Techniken des Deep Learnings einen Grundstein für die LOEWE-Exploration *Künstliche Intelligenz zur Erschließung kolonialer Verwertungspraktiken archäologischer Objektsammlungen* zu legen, die die automatisierte Klassifikation und Zuordnung von bis zu 5 000 zyprischen Antiken auf rund 100 Fotografien mit und ohne Lineal der 1890er Jahre anstrebt. Unter diesem Hintergrund wurde die Masterthesis mit dem Fokus auf Fotografien der Keramiksammlungen in drei Aufgabenbereiche zerlegt: Die Lokalisierung des Lineals, die Erkennung und Lokalisierung der einzelnen Keramikgefäße und die Größenapproximation der lokalisierten Keramikgefäße. Für das Deep Learning existierten diesbezüglich etliche Herausforderungen, angefangen bei der geringen Datenmenge, der hohen Anzahl an Objekten pro Fotografie und der nicht oder nicht vollständigen Abbildung des Lineals.

Im Folgenden werden die Behandlung und Ergebnisse der Aufgabenbereiche vorerst dargelegt und anschließend anhand möglicher Ansätze für weitere Optimierungen ein Fazit gebildet.

Zunächst wurde die Lokalisierung des Lineals auf 37 Scans von Keramikgefäßsammlungen mit Lineal anhand der gewählten einstufigen Objekterkennungsdetektoren YOLOR und EfficientDet sowie mit dem zweistufigen Objekterkennungsdetektor Mask R-CNN vorgenommen. Dazu wurden die Lineale als erstes mit Bounding Boxen annotiert, bevor die Detektoren der Objekterkennung unter Verwendung der Technik Cross Validation mit fünf Folds trainiert und getestet wurden.

Anhand der Tabellen 5.1 für YOLOR, 5.2 für EfficientDet und 5.3 für Mask R-CNN wurden die erreichten mAP-Werte sowie die Trainingsdauern pro Fold zusammengefasst. Die Modelle von Mask R-CNN schnitten mit der im Vergleich geringsten Trainingsdauer mit dem höchsten mAP am besten ab. Hingegen erreichten die Modelle von YOLOR und EfficientDet ähnliche, aber um die Hälfte schlechtere mAP-Werte als die Mask R-CNN-Modelle. Insbesondere die YOLOR-Modelle benötigten dazu jedoch trotz gleicher Bedingungen nach Kapitel 5.1 eine durchschnittlich siebenfach längere Trainingsdauer. Gemein hatten jedoch alle drei Detektoren, dass sie sowohl Regalbretter als auch ähnlich farbige und geformte Teile von Wänden, Böden und Objekten fälschlicherweise eben-

falls als Lineal lokalisierten. Andere Herausforderungen ergaben sich durch die schlechte Belichtung einiger Fotografien sowie die Anordnung der Lineale.

Damit werden die folgenden Erkenntnisse für die Lokalisierung des Lineals im Rahmen aller Erkenntnisse der Masterthesis festgehalten.

Top 11 Erkenntnisse der Masterthesis
Lokalisierung des Lineals

1. Mask R-CNN-Modelle schnitten am besten ab.
2. Regalbretter sowie Teile von der Wand, dem Boden und Objekten wurden oft mit Linealen verwechselt, wenn ihre Farbe oder Form sich geähnelt haben.

Die Erkennung und Lokalisierung der einzelnen Keramikgefäße auf nun viel mehr 43 Scans von Keramikgefäßsammlungen mit und ohne Lineal machte sich nun die zuvor erhaltenen Ergebnisse zu Nutze. Diesbezüglich wurden nur noch die Objekterkennungsdetektoren Mask R-CNN und YOLOR unter Verwendung der Cross Validation mit fünf Folds trainiert und getestet. Die Entscheidung für YOLOR und gegen EfficientDet wurde dabei insbesondere durch die zuvor getätigten Experimente ohne Cross Validation, die im Appendix unter den Tabellen A.1 und A.2 festgehalten wurden, motiviert.

Als erstes mussten die einzelnen Keramikgefäße jedoch durch Unterstützung des Instituts für Archäologische Wissenschaften der Goethe-Universität im Hinblick auf die Genauigkeit mit Polygonen annotiert werden. Es ergaben sich 17 zu erkennende Keramikgefäßklassen.

Überdies wurden die Detektoren der Objekterkennung noch zwei weitere Male mit Cross Validation trainiert und getestet, indem der kleine Datensatz zu zwei unterschiedlichen Graden mittels Data Augmentation künstlich, aber so sinngemäß wie möglich, aufgebläht wurde. Im ersten Fall kamen dabei 156 augmentierte Bilder hinzu, im zweiten Fall wurden diese um weitere 86 augmentierte Bilder erweitert.

Die Ergebnisse und Trainingsdauern der Modelle beider Detektoren wurden in den Tabellen 5.7 für YOLOR und 5.9 für Mask R-CNN festgehalten. Dabei war zu beobachten, dass die mAP-Werte der YOLOR-Modelle denen von Mask R-CNN bei ungefähr gleicher Trainingsdauer um 11 bis 18 % überlegen waren. Im Falle der YOLOR-Modelle konnte überdies eine sich steigende Verbesserung durch beide Augmentationsgrade erkannt werden. Bei den Modellen von Mask R-CNN lag keine Verbesserung durch Data Augmentation vor. Deutlich wurde durch die Evaluation der einzelnen Cross Validation Folds der Modelle jedoch, dass je mehr Objekte durchschnittlich pro Fotografie pro Fold abgebildet wurden, desto schlechtere mAP-Werte wurden erreicht. Nachdem auch die Vorhersagen der Modelle untersucht wurden, konnte außerdem beobachtet werden, dass

Mask R-CNN prinzipiell deutlich weniger Objekte überhaupt lokalisiert hat. Durch die Betrachtung der Keramikgefäßklasse *Amphoriskos* konnten jene Umstände anhand der Tabelle 5.10 noch unterstrichen werden.

Hervorzuheben war außerdem der Umgang der Detektoren mit unannotierten Objekten innerhalb der Fotografien. Diese waren entweder unannotiert, weil sie keine Keramikgefäße sind oder weil die entsprechende Keramikgefäßklasse zum Zeitpunkt der Annotation durch das Institut für Archäologische Wissenschaften nicht vorlag. Im ersten Fall wurden viele Objekte unabhängig ihrer Größe und Position korrekterweise nicht von den Objekterkennungsdetektoren lokalisiert und klassifiziert. Dies wurde vor allem durch Abbildung 5.11 ersichtlich. Hingegen wurden im zweiten Fall insbesondere von den Modellen von YOLOR fast alle nicht annotierten Keramikgefäße lokalisiert und versucht zu klassifizieren. Auch Mask R-CNN lokalisierte und klassifizierte einige wenige Objekte. Etwaige Klassifikationen wurden innerhalb Tabelle A.4 festgehalten. Aufgrund mangelnder Expertise im Bereich der Archäologie und möglicher Fehlerpunkte innerhalb der manuellen Annotation war dieser Umstand jedoch nur schwierig als positiv oder negativ zu bewerten.

Damit kann die Liste der Erkenntnisse der Masterthesis um die folgenden Erkenntnisse der Erkennung und Lokalisierung der Keramikgefäße erweitert werden.

Erkennung und Lokalisierung der einzelnen Keramikgefäße

3. YOLOR-Modelle mit der höchsten Augmentationsstufe schnitten am besten ab.
4. Durch Data Augmentation konnte die Genauigkeit von YOLOR-Modellen gesteigert werden. Die Genauigkeit von Mask R-CNN-Modellen blieb größtenteils unverändert.
5. Je mehr Objekte durchschnittlich auf Fotografien abgebildet wurden, desto schlechter waren die mAP-Werte der Cross Validation Folds.
6. Mask R-CNN-Modelle lokalisierten grundsätzlich weitaus weniger Keramikgefäße als YOLOR-Modelle.
7. Unannotierte *missing_label*-Keramikgefäße wurden häufig von YOLOR-Modellen und weniger von Mask R-NN-Modellen erkannt.
8. Unannotierte Objekte, die keine Keramikgefäße sind, wurden oft unabhängig von ihrer Größe oder Form korrekterweise nicht lokalisiert und klassifiziert.

Letztlich wurde auch die Approximation der Gesamtbreite und -höhe der lokalisierten Keramikgefäße umgesetzt. Dazu kam zunächst das auf den meisten Fotografien abge-

bildete Lineal zum Einsatz. Durch dieses konnte der tatsächlichen Lineallänge von 50 cm auch eine Größe in Pixeln zugewiesen werden, mit der dann die tatsächliche Größe anderer abgebildeter Objekte ermittelt werden konnte. Zur Erhaltung der Lineallänge in Pixeln kam jedoch keine der zuvor trainierten und getesteten Detektoren für die Lineallokalisierung zum Einsatz, da sie trotz allem noch zu ungenau waren und die Approximation damit zu stark verfälschen würden. Stattdessen wurde die Lineallänge in Pixeln aus den annotierten Bounding Boxen gezogen. Dazu wurde insbesondere die Annotation von in ihrer Länge nicht vollständig abgebildeten Linealen manuell erweitert. Diese Herangehensweise war deutlich tragbarer als eine automatisierte, die für gerade mal 16 Fälle noch immer etliche Herausforderungen stellen würde, beispielsweise wenn sowohl die Lineallänge als auch -breite nicht mehr vollständig abgebildet werden würden. Somit konnte auch identifiziert werden, ob sich ein Lineal in der entsprechenden Fotografie befindet. War dem nicht so, wurde die Dicke der in jeder Fotografie abgebildeten Regalbretter verwendet, deren tatsächliche Größe manuell über Fotografien mit Lineal approximiert werden konnte. Ihre Dicke in Pixeln pro Fotografie wurde ebenfalls manuell ermittelt, indem Nutzer*innen bei der Größenapproximation für jede Fotografie ohne Lineal automatisiert aufgefordert wurden, die Regalbrettdicke mit zwei Punkten in einem sich öffnenden Fenster zu markieren. Diese Umsetzung ist auch auf andere Fotografien zu übertragen und bietet sich damit mehr an als eine vollständige Annotation der Regalbretter, wie es bei den Linealen der Fall war.

Schließlich diente der in der Teilaufgabe zur Erkennung und Lokalisierung der Keramikgefäße als besser identifizierter Objekterkennungsdetektor YOLOR der besten Augmentationsstufe zur Erlangung der Größenmaße der einzelnen Keramikgefäße in Pixeln. Damit konnten die tatsächlichen Größenmaße der einzelnen Keramikgefäße errechnet werden. Diese wurden dann mit ID sowohl auf dem entsprechenden Bild als auch in einer CSV-Datei samt der Keramikgefäßklasse und den Bounding Box-Koordinaten gespeichert.

Die approximierten Größenmaße von 12 Keramikgefäßen konnten dann anhand ihrer tatsächlichen Größenmaße manuell evaluiert werden. Etwaige Ergebnisse wurden innerhalb Tabelle 5.8.5 festgehalten. Es ergab sich eine gesamte durchschnittliche Differenz der Flächeninhalte der tatsächlichen und approximierten Gesamthöhe und -breite von 20,4 %. Dabei wurden insbesondere Keramikgefäße, die möglichst gerade zur Kamera ausgerichtet sind, besser approximiert. Für Fotografien, die keine Lineale abbilden, wurde eine positive Differenz von 1,2 % zum Durchschnitt festgestellt. Dabei war jedoch zu beachten, dass sich etwaige Ergebnisse nur auf die 12 evaluierten Keramikgefäße beziehen und sie unter Evaluation von mehr Gefäßen anders ausfallen könnten. Nichtsdestotrotz konnte darauf geschlossen werden, dass die Substitution der Lineallänge mit der Regalbrettdicke zur Größenapproximation im vorliegenden Anwendungsfall eine gu-

te und einfache Lösung darstellt. Beschränkt blieb die Größenapproximation jedoch auf die auf den Bildern abgebildete Gesamtbreite und -höhe der Objekte, an die die tatsächlichen Größenmaße zur Evaluation angepasst werden musste. Aufgrund verschiedener Ausrichtungen sowie tatsächlicher Maße, die über die Objektbreite und -höhe hinausgingen, musste dies manuell erfolgen.

Mit der Hinzunahme der Erkenntnisse der Größenapproximation der Keramikgefäße vervollständigt sich die Liste der wichtigsten Erkenntnisse.

Größenapproximation der lokalisierten Keramikgefäße

9. Es treten überschaubare Differenzen zwischen den approximierten und tatsächlichen Größenmaßen auf.
10. Die Verwendung sowie manuelle Erweiterung der Annotation der Lineale zur Erlangung der Lineallänge in Pixeln ist Kosten-Nutzen-technisch effizient.
11. Die Regalbrettdicke ist ein gutes Substitut für die Lineallänge, sollte diese nicht abgebildet sein.

Schlussfolgernd konnten damit zwar einige der identifizierten Herausforderungen der Datenbasis, wie das fehlende oder abgeschnittene Lineal, aufgelöst werden, doch bleiben auch viele Herausforderungen bestehen. Dies betrifft insbesondere jegliche Art der Objekterkennung auf den Fotografien der Keramikgefäßsammlungen. Die wohl relevanteste Herausforderung, deren Lösung andere Herausforderungen begünstigen könnte, ist die geringe Datenbasis. Sie kann lediglich im Rahmen von YOLOR anhand Data Augmentation zu einem kleinen Teil behandelt werden.

Zukünftig sollte daher vor allem eine genauere Untersuchung der Data Augmentation unter verschiedenen Gesichtspunkten unternommen werden. Im Rahmen der Masterthesis wurde primär geprüft, ob ein Einfluss für die Objekterkennungsdetektoren YOLOR und Mask R-CNN besteht und ob dieser allein durch die Hinzunahme von einer größeren Anzahl augmentierter Daten noch zu verbessern ist. Darüber hinaus sollte auch genauer untersucht werden, welche Augmentoren und spezifischer, welche Parameter sich am gewinnbringendsten oder gar negativ auf den vorliegenden Anwendungsfall auswirken.

In diesem Zuge sollte gleichermaßen in Betracht gezogen werden, unabhängige Datenbasen aus dem Internet oder von anderen kooperierenden Museen hinzuzunehmen. Diese müssen der vorliegenden Datenbasis jedoch so ähnlich wie möglich sein, um überhaupt

einen positiven Einfluss auf die Genauigkeit der Objekterkennung nehmen zu können. Zumindest eine farbliche und qualitative Ähnlichkeit könnte sicherlich durch Anpassung über ein Bildbearbeitungsprogramm zustande kommen. Im Kontext der Masterthesis lagen derartige Datenbasen nicht vor und aufgrund der limitierten Zeit konnten sie auch nicht erworben werden, weswegen diese Option in Kapitel 4.3 ausgeschlossen wurde.

Gleichermaßen kann auch versucht werden, die aktuellen Fotografien über ein Bildbearbeitungsprogramm passend zu färben oder ihre Belichtung, ihren Kontrast und ihre Qualität anzupassen. Dies wurde bereits in dem vorgegangenen Kapitel 4.3 erwähnt, ist aber mit viel manuellem Aufwand verbunden, da die Anpassung für jede Fotografie individuell erfolgen müsste. Der tatsächliche Nutzen kann überdies trotz des hohen manuellen Aufwands nicht gut im Vorfeld abgeschätzt werden.

In Anbetracht der Tatsache, dass die Erkennung und Lokalisierung der Keramikgefäße definitiv mit einer höheren Anzahl an Objekten pro Fotografie und damit auch einer höheren Anzahl an kleineren Objekten leidet, bietet sich überdies eine Zerlegung der Fotografie in kleinere Teile an. Objekterkennungsdetektoren können anschließend ihre Vorhersagen auf den entsprechenden Teilen der Fotografien tätigen, welche zum Schluss wieder zusammengefügt werden. In der Literatur spricht man dabei von *Tiling*, welches beispielsweise in der Publikation von Ünel et al. [ÖÜÖÇ19] angewandt wurde.

Abseits der Eingabedaten kann des Weiteren versucht werden, für eine bessere Nutzung und Evaluation der Daten statt der K-Fold Cross Validation die Leave-One-Out Cross Validation für die Keramikgefäßerkennung und -lokalisierung zu utilisieren. Im Rahmen der Masterthesis hätte diese, wie zuvor in Kapitel 4.3 erwähnt, trotz der zugrundeliegenden Hardware 5.1 zu viel Zeit in Anspruch genommen. Generell ist sie jedoch zumindest einen Versuch wert.

Einem längeren Training des Ist-Zustands der Objekterkennungsdetektoren wird demgegenüber nicht mit viel Zuversicht entgegengeblickt, da die Detektoren bereits ziemlich ausgereizt erscheinen[1]. Stattdessen kann beispielsweise nach der Publikation von Liashchynskyi und Liashchynskyi [LL19] die Technik *Grid Search* angewendet werden, um die optimalen Hyperparameter der Objekterkennungsdetektoren zu ermitteln und ihre Genauigkeit damit zu steigern. Auch eine Kombination mehrerer Detektoren, um ihre besten Eigenschaften zu utilisieren, bietet sich nach der Publikation von Drid et al. [DAK20] an. Ebenfalls können andere Detektoren der Objekterkennung abseits YOLOR, EfficientDet und Mask R-CNN getestet und verglichen werden.

[1] Etwaige TensorBoard-Graphen, die dies bestätigen, sind innerhalb des Ordners `runs` für YOLOR oder `my_models` für EfficientDet und Mask R-CNN der innerhalb Kapitel A.1 bereitgestellten Implementierung hinterlegt.

Zuletzt ist ebenfalls zu erwähnen, dass die Annotation der Keramikgefäße erneut von Experten überarbeitet werden sollte, um insbesondere jene aktuell mit *missing_label* versehene Keramikgefäße im Nachhinein korrekt zu klassifizieren.

Auch eine automatisierte Lokalisierung des Lineals bietet sich unter der Annahme von Modellen der Objekterkennung mit höherer Genauigkeit für die Größenapproximation natürlich an. Diesbezüglich müsste ebenfalls eine automatisierte Lösung für nicht vollständig abgebildete Lineale gefunden werden. Etwaige Vorschläge über das Utilisieren der Seitenverhältnisse wurden bereits in Kapitel 4.3 aufgeführt.

Schließlich kann die Masterthesis anhand ihres großen Umfangs an ersten Experimenten zweifelsohne den wichtigen Grundstein der LOEWE-Exploration legen. Auf diesem kann nun durch weitere Arbeit in Form von teilweise erwähnten Verbesserungsmöglichkeiten und weiteren Evaluationen im Namen des Projekts aufgebaut werden. Für die Zukunft ergibt sich vor allem die Möglichkeit, die wissenschaftlich-archäologische Identifikation von Antiken oder Zuordnung zu Antiken auf Fotografien durch Techniken des Deep Learnings und Nutzung von Größenmaßen zu unterstützen und zu beschleunigen. Damit kann zunächst der Erkenntnisgewinn in der archäologischen Praxis vereinfacht werden, doch auch eine Anwendung auf gänzlich andere Archivbestände oder Anwendungsgebiete ist denkbar.

Appendix

A.1 Zugriff auf die praktische Implementierung

Die in der Masterthesis erarbeitete praktische Implementierung kann über die folgende URL heruntergeladen werden. Diese enthält sowohl die verwendeten Python-Skripte als auch etwaige Dokumente, die verwendeten Scans und Kameraaufnahmen der Verkaufsbilder sowie deren Annotationen. Überdies sind die verwendeten Detektoren und der Programmcode zum Trainieren, Testen und Vorhersagen im Rahmen der Cross Validation sowie deren Ergebnisse beigefügt. Auch die trainierten Weights der einzelnen Modelle der Detektoren liegen damit vor, sind allerdings aufgrund von limitiertem Speicherplatz lediglich auf die tatsächlich verwendeten beschränkt. Im Rahmen von YOLOR ist insbesondere auch der Programmcode für die Größenapproximation enthalten.

`https://hessenbox-a10.rz.uni-frankfurt.de/getlink/fi1Seb7GKLaWExYpGRyyU/`

A.2 Frühere Ergebnisse trainierter Modelle

Vor der Umsetzung der Lineallokalisierung in Kapitel 5.6 und Keramikgefäßerkennung und -lokalisierung in Kapitel 5.7 sind die Detektoren YOLOR, EfficientDet und Mask R-CNN zunächst ohne Cross Validation anhand eines 70/20/10-Splits der Trainings-, Validierungs- und Testdaten untersucht worden. Die erreichten Ergebnisse sind in den Tabellen A.1 und A.2 festgehalten.

Detektor	**Epochen/Schritte**	**mAP**	**Dauer**
YOLOR	3 000 Epochen	0,498	6 h 1 min
	5 720 Epochen	0,48	15 h 7 min
Mask R-CNN	75 000 Schritte	0,324	16 h 16 min
	100 000 Schritte	0,327	22 h 29 min

Tabelle A.1: Erste Ergebnisse der Keramikgefäßerkennung und -lokalisierung der YOLOR- und Mask R-CNN-Modelle ohne Cross Validation.

Detektor	Epochen/Schritte	mAP	Dauer
YOLOR	2 000 Epochen	0,275	5 h 43 min
	3 000 Epochen	0,65	5 h 33 min
EfficientDet	50 000 Schritte	0,373	3 h 39 min
	75 000 Schritte	0,223	5 h 28 min
Mask R-CNN	50 000 Schritte	0,354	3 h 26 min
	75 000 Schritte	0,354	5 h 12 min

Tabelle A.2: Erste Ergebnisse der Lineallokalisierung der YOLOR-, EfficientDet- und Mask R-CNN-Modelle ohne Cross Validation.

A.3 Beispielhafte Ausschnitte von Annotationsdateien

Im Zuge der Implementierung in Kapitel 5 kommen häufig COCO-Annotationsdateien zum Einsatz oder werden angepasst. Aufgrund dessen folgt ein beispielhafter Ausschnitt einer solchen Annotationsdatei mit ihren für die Implementierung wichtigsten Einträgen:

```
{ ...
    "categories": [ ...
        {
            "id": 12,
            "name": "Jug", ...
        }, ...
    ],
    "images": [ ...
        {
            "id": 7,
            "width": 1381,
            "height": 1673,
            "file_name": "TCB-4.jpg", ...
        }, ...
    ],
    "annotations": [ ...
        {
            "id": 177,
            "image_id": 7,
            "category_id": 12,
            # "segmentation": [[x1, y1, ... xn, yn]]
            "segmentation": [[ 162.64, 537.19, ...]], ...
            # "bbox": [x, y, w, h]
            "bbox": [62.12, 241.51, 285.58, 320.02], ...
        }, ...
```

Im Gegensatz dazu ergibt sich der einzelne Eintrag desselben Objekts anhand der Annotationsdatei `TCB-4.txt` des entsprechenden Bildes `TCB-4` im YOLO-Exportformat durch folgenden Eintrag. Dabei weicht insbesondere die ID des Objekts *Jug* von der der COCO-Annotationsdatei ab, da beim YOLO-Exportformat nicht mit 1, sondern mit 0 angefangen wird die IDs zu vergeben.

```
# ID, x, y, w, h
... 11 0.14837798696596669 0.23999999999999996 0.20679217958001445
    0.19128511655708305 ...
```

A.4 Angewandte Python-Skripte

Während der Umsetzung der Implementierung in Kapitel 5 sind viele selbstgeschriebene Python-Skripte zum Einsatz gekommen. Zur Schaffung eines besseren Überblicks werden sie und ihre groben Funktionsweisen in der anschließenden Tabelle A.3 aufgeführt.

Skript	**Beschreibung**
`adapt_annotation_folder.py`	Löscht alle Einträge von Bildern aus der COCO-Annotationsdatei, die sich nicht im angegebenen Ordner befinden.
`adapt_seg_bbox_values.py`	Passt alle Werte des `segmentation`- und `bbox`-Keys der COCO-Annotationsdatei an, sodass sie nicht mehr über den Bildrand ragen.
`augmentation_poly.py`	Augmentiert Bilder und ihre Polygon-Annotationen mit imgaug anhand definierter Augmentationsfunktionen.
`bbox_coco_to_yolo.py`	Übersetzt die Bounding Box-Annotationen von COCO in Bounding Box-Annotationen von YOLO.
`bbox_to_seg.py`	Übersetzt die Werte des `bbox`-Keys in Werte des `segmentation`-Keys der COCO-Annotationsdatei.
`change_tf_record_name.py`	Nennt die `tfrecord`-Bruchstücke in die ihnen enthaltenen Bilder um.
`check_all_annotations.py`	Zeichnet alle Bounding Box- oder Polygon-Annotationen auf ihre entsprechenden Bilder, um sie zu überprüfen.
`seg_to_bbox.py`	Übersetzt die Werte des `segmentation`-Keys in Werte des `bbox`-Keys der COCO-Annotationsdatei.

Tabelle A.3: Die hauptsächlich selbstgeschriebenen Python-Skripte zur Unterstützung der Masterthesis.

A.5 Klassifikation der *missing_label*-Keramikgefäße

Im Rahmen des Kapitels 5.7.5 sind die Vorhersagen der *missing_label*-Keramikgefäße durch die YOLOR-Modelle mit einer Confidence von 40 % und durch die Mask R-CNN-Modelle mit einer Confidence von 0,1 % untersucht worden. Eine genauere Auflistung der einzelnen vorhergesagten Klassen der Modelle unterschiedlicher Augmentationsstufen ist Tabelle A.4 zu entnehmen.

		YOLOR			**Mask R-CNN**		
Bild	**ID**	**k. A.**	**A. 1**	**A. 2**	**k. A.**	**A. 1**	**A. 2**
ECA-1	35	Jug	Jug	Jug			Jug
ECB-1	5		Bowl	Bowl			
GCA-1	76	Bowl	Bowl	Bowl			
	118	Juglet		Juglet			
GCA-2	44		Bowl	Bowl			
GCB-1	80	Bowl	Bowl, Juglet	Bowl			
GCB-2	30		Jug	Jug			
TCA-2	7	Amphora	Amphora	Amphora	Jug		
TCB-2	12	Amphora	Amphora	Amphora	Jug	Jug	Jug
	27	Bowl	Bowl	Bowl	Bowl	Bowl	Bowl
TCD-1	8		Jug				
TCD-2	19	Amphora	Amphora	Amphora, Jug	Amphora	Amphora	Amphora
TCE-1	47	Bowl	Bowl	Bowl	Bowl	Bowl	Bowl
TCF-1	49	Bowl	Bowl	Bowl			Bowl

Tabelle A.4: Klassifikationen der *missing_label*-Keramikgefäße aller Augmentationsstufen der YOLOR- und Mask R-CNN-Modelle.

A.6 Überblick von Objektsammlungen mit Lineal

Innerhalb der Abbildung 4.1 hat sich der Überblick einiger Herausforderungen der Lineallokalisierung lediglich auf Scans von Keramikgefäßsammlungen bezogen. Im Rahmen der folgenden Abbildung A.1 auf der nächsten Seite wird die Gliederung hingegen sowohl mittels Scans als auch Kameraaufnahmen von Objektsammlungen vorgenommen.

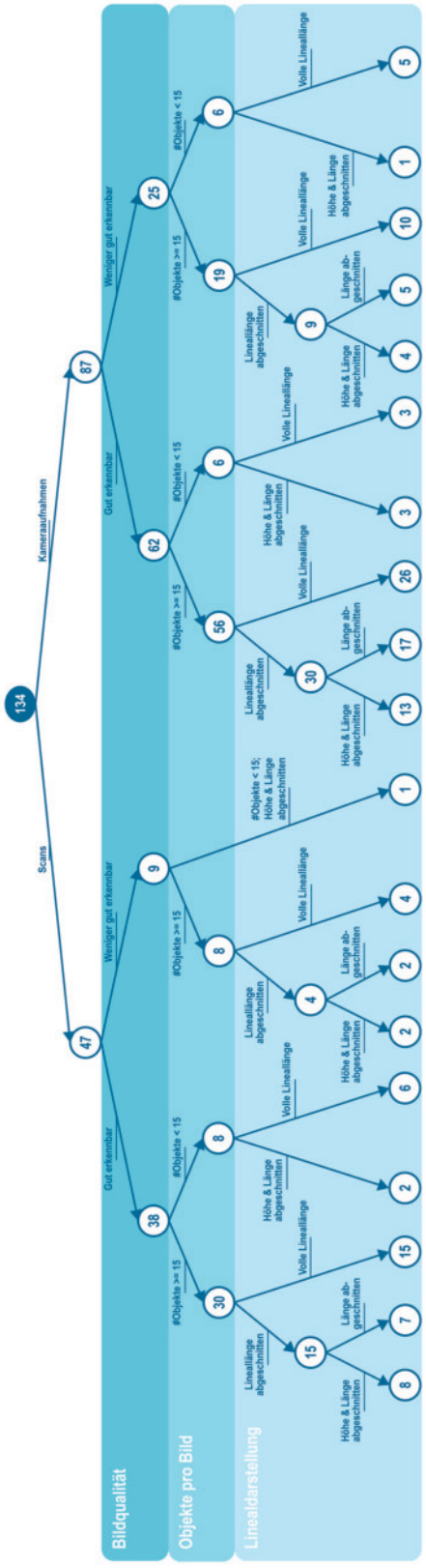

Abbildung A.1: Gliederung der Scans und Kameraaufnahmen von Objektsammlungen mit Lineal in Bildqualität, Anzahl der Objekte pro Bild und Linealdarstellung.

Literaturverzeichnis

[ABG+20] Francesca Anichini, Francesco Banterle, Jaume Garrigós, Marco Callieri, Nachum Dershowitz, Nevio Dubbini, Diego L. Diaz, Tim Evans, Gabriele Gattiglia, Katie Green et al. Developing the ArchAIDE application: A digital workflow for identifying, organising and sharing archaeological pottery using automated image recognition. *Internet Archaeology*, 52, 2020.

[BJS+20] Mirza R. Baig, Thomas V. Joseph, Nipun Sadvilkar, Mohan K. Silaparasetty und Anthony So. *The Deep Learning Workshop*. Packt Publishing, 1. Aufl., 2020.

[Boc20] Alexey Bochkovskiy. Scaled YOLO v4 is the best neural network for object detection on MS COCO dataset. `https://alexeyab84.medium.com/scaled-yolo-v4-is-the-best-neural-network-for-object-detection-on-ms-coco-dataset-39dfa22fa982`, 2020. Besucht am 16.11.2022.

[Boc21] Alexey Bochkovskiy. GitHub-Beitrag: Waymo self-driving challenge: YOLOR(Scaled-YOLOv4-based) is the best speed/accuracy, better than YOLOv5/Detr. `https://github.com/pjreddie/darknet/issues/2454`, 2021. Besucht am 16.11.2022.

[Boc22] Alexey Bochkovskiy. How to train (to detect your custom objects). `https://github.com/AlexeyAB/darknet/#how-to-train-to-detect-your-custom-objects`, 2022. Besucht am 16.11.2022.

[Bon14] Melia Bonomo. Estimating Focal Length. `https://laser.physics.sunysb.edu/_melia/Summer2014/tennis.html`, 2014. Besucht am 14.07.2022.

[BR14] Abhir Bhalerao und Gregory Reynolds. Ruler Detection for Autoscaling Forensic Images. *International Journal of Digital Crime and Forensics*, 6(1):9–27, 2014.

[BWL20] Alexey Bochkovskiy, Chien-Yao Wang und Hong-Yuan M. Liao. Yolov4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*, 2020.

[CLO20] CLOUDFACTORY. Image Annotation for Computer Vision: A Guide to Labeling Visual Data for Your Machine Learning Project. `https://www.cloudfactory.com/image-annotation-guide`, 2020. Besucht am 14.06.2022.

[CMDB20] Agnese Chiatti, Enrico Motta, Enrico Daga und Gianluca Bardaro. Fit to Measure: Reasoning About Sizes for Robust Object Recognition. *arXiv preprint arXiv:2010.14296*, 2020.

[Con Ja] COCO Consortium. Data format. `https://cocodataset.org/#format-data`, o. J. Besucht am 16.11.2022.

[Con Jb] COCO Consortium. Detection Evaluation. `https://cocodataset.org/#detection-eval`, o. J. Besucht am 16.11.2022.

[Cri02] Antonio Criminisi. Single-View Metrology: Algorithms and Applications. In *Joint Pattern Recognition Symposium*, S. 224–239. Springer, 2002.

[CSKX15] Chenyi Chen, Ari Seff, Alain Kornhauser und Jianxiong Xiao. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In *Proceedings of the IEEE International Conference on Computer Vision*, S. 2722–2730, 2015.

[CV18] Zhaowei Cai und Nuno Vasconcelos. Cascade R-CNN: Delving into High Quality Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 6154–6162, 2018.

[CZF+18] Qimin Cheng, Qian Zhang, Peng Fu, Conghuan Tu und Sen Li. A Survey and Analysis on Automatic Image Annotation. *Pattern Recognition*, 79:242–259, 2018.

[DAK20] Khaoula Drid, Mebarka Allaoui und Mohammed Lamine Kherfi. Object Detector Combination for Increasing Accuracy and Detecting More Overlapping Objects. In *International Conference on Image and Signal Processing*, S. 290–296. Springer, 2020.

[DBX+19] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang und Qi Tian. CenterNet: Keypoint Triplets for Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, S. 6569–6578, 2019.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li und Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, S. 248–255. IEEE, 2009.

[DLHS16] Jifeng Dai, Yi Li, Kaiming He und Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[EKN+17] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau und Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn und Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[Fis17] Kevin D. Fisher. *Cyprus, Archaeology of*, S. 1–21. Springer International Publishing, Cham, 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell und Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 580–587, 2014.

[Gir15] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, S. 1440–1448, 2015.

[GLL19] Golnaz Ghiasi, Tsung-Yi Lin und Quoc V. Le. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, S. 7036–7045, 2019.

[Gos21] Prabhjot Gosal. Converting bounding box annotations from YOLO to COCO format. `https://www.youtube.com/watch?v=qESxRSqsAGw`, 2021. Besucht am 16.11.2022.

[GRH+21] Tyler Ganter, Vivek Rathod, Mitchel Humpherys, Karim Nosseir et al. Object Detection API Demo. `https://github.com/tensorflow/models`

/blob/master/research/object_detection/colab_tutorials/object_detection_tutorial.ipynb, 2021. Besucht am 16.11.2022.

[Han08] Allan Hanbury. A survey of methods for image annotation. *Journal of Visual Languages & Computing*, 19(5):617–627, 2008.

[HB20] Abdul M. Hafiz und Ghulam M. Bhat. A survey on instance segmentation: state of the art. *International Journal of Multimedia Information Retrieval*, S. 171–189, 2020.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár und Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, S. 2961–2969, 2017.

[HNZ21] M. Q. Huang, J. Ninić und Q. B. Zhang. BIM, machine learning and computer vision techniques in underground construction: Current status and future perspectives. *Tunnelling and Underground Space Technology*, 108:103677, 2021.

[HZC+17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto und Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

[HZK07] Michael Herrmann, Sebastian Zambanini und Martin Kampel. Image-Based Measurement of Ancient Coins. In *Proc. of VAST*, Bd. 7, S. 55–62, 2007.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren und Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren und Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 770–778, 2016.

[Hä04] Franz Häußler. *Fotografie in Augsburg, 1839 bis 1900: Mit einem Bildteil aus den Fotoschätzen des Stadtarchivs Augsburg*, Bd. 1. Wißner-Verlag, 2004.

[IWD19] Barak Itkin, Lior Wolf und Nachum Dershowitz. Computational Ceramicology. *arXiv preprint arXiv:1911.09960*, 2019.

[JLK19] Hwejin Jung, Bilal Lodhi und Jaewoo Kang. An automatic nuclei segmentation method based on deep convolutional neural networks for histopathology images. *BMC Biomedical Engineering*, 1(1):1–12, 2019.

[Jun20] Alexander Jung. Overview of Augmenters. `https://imgaug.readthedocs.io/en/latest/source/overview_of_augmenters.html`, 2020. Besucht am 16.11.2022.

[KDB+17] Dmitry A. Konovalov, Jose A. Domingos, Casey Bajema, Ronald D. White und Dean R. Jerry. Ruler Detection for Automatic Scaling of Fish Images. In *Proceedings of the International Conference on Advances in Image Processing*, S. 90–95, 2017.

[KDWJ18] Dmitry A. Konovalov, Jose A. Domingos, Ronald D. White und Dean R. Jerry. Automatic Scaling of Fish Images. In *Proceedings of the 2nd International Conference on Advances in Image Processing*, S. 48–53, 2018.

[KO11] Talia Konkle und Aude Oliva. Canonical visual size for real-world objects. *Journal of Experimental Psychology: Human Perception and Performance*, 37(1):23–37, 2011.

[KSB+10] Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, Yann Cun et al. Learning Convolutional Feature Hierarchies for Visual Recognition. *Advances in Neural Information Processing Systems*, 23, 2010.

[KSH17] Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6):84–90, 2017.

[KTM+18] Bashir Kazimi, Frank Thiemann, Katharina Malek, Monika Sester und Kourosh Khoshelham. Deep Learning for Archaeological Object Detection in Airborne Laser Scanning Data. In *Proceedings of the 2nd Workshop On Computing Techniques For Spatio-Temporal Data in Archaeology And Cultural Heritage*, Bd. 15, 2018.

[LAE+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu und Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision*, S. 21–37. Springer, 2016.

[LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan und Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 2117–2125, 2017.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He und Piotr Dollár. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, S. 2980–2988, 2017.

[LKCA16] Bria Long, Talia Konkle, Michael A. Cohen und George A. Alvarez. Mid-Level Perceptual Features Distinguish Objects of Different Real-World Sizes. *Journal of Experimental Psychology: General*, 145(1):95–109, 2016.

[LL19] Petro Liashchynskyi und Pavlo Liashchynskyi. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. *arXiv preprint arXiv:1912.06059*, 2019.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár und C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, S. 740–755. Springer, 2014.

[LOE21] LOEWE-Exploration: Künstliche Intelligenz zur Erschließung kolonialer Verwertungspraktiken archäologischer Objektsammlungen. Scans der Verkaufsbilder zyprischer Antiken von Max Ohnefalsch-Richter. Archiv des Beliner Museums für Vor- und Frühgeschichte, 2021.

[LOW+20] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu und Matti Pietikäinen. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128(2):261–318, 2020.

[LQQ+18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi und Jiaya Jia. Path Aggregation Network for Instance Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 8759–8768, 2018.

[Met04] Gabriele Metzler. *Einführung in das Studium der Zeitgeschichte*, Bd. 2433. UTB, 2004.

[MHKV19] Graham G. Monkman, Kieran Hyder, Michel J. Kaiser und Franck P. Vidal. Using machine vision to estimate fish length from images using

regional convolutional neural networks. *Methods in Ecology and Evolution*, 10(12):2045–2056, 2019.

[MJTS19] James F. Mullen Jr., Franklin R. Tanner und Phil A. Sallee. Comparing the Effects of Annotation Type on Machine Learning Detection Performance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[MS21] Farid Mirbagheri und Emilios A. Solomou. *Historical Dictionary of Cyprus*, S. XXV–XLIV. Rowman & Littlefield, 2021.

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[NDB+19] Giang Nguyen, Stefan Dlugolinsky, Martin Bobák, Viet Tran, Álvaro López García, Ignacio Heredia, Peter Malík und Ladislav Hluchỳ. Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, 52(1):77–124, 2019.

[OBLS14] Maxime Oquab, Leon Bottou, Ivan Laptev und Josef Sivic. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 1717–1724, 2014.

[OCS21] Serban Oprisescu, Mihai Ciuc und Alina Sultana. Automatic Segmentation and Measurement of Infantile Hemangioma. *Symmetry*, 13(1):138, 2021.

[ÖÜÖÇ19] F. Özge Ünel, Burak O. Özkalayci und Cevahir Çiğla. The Power of Tiling for Small Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[PBBC20] Sankar K. Pal, Debasmita Bhoumik und Debarati B. Chakraborty. Granulated deep learning and Z-numbers in motion detection and object recognition. *Neural Computing and Applications*, 32(21):16533–16548, 2020.

[Pia18] Pialin [Alias]. GitHub-Beitrag: TF's COCO evaluation wrapper doesn't actually support include_metrics_per_category and all_metrics_per_category. `https://github.com/tensorflow/models/issues/4778/#issuecomment-430262110`, 2018. Besucht am 16.11.2022.

[PNDS20] Rafael Padilla, Sergio L. Netto und Eduardo A. B. Da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. In *2020 International Conference on Systems, Signals and Image Processing*, S. 237–242. IEEE, 2020.

[Pok20] Sabina Pokhrel. Image Data Labelling and Annotation—Everything you need to know. `https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1`, 2020. Besucht am 29.05.2022.

[PPD+21] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto und Eduardo A. B. da Silva. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10(3):279, 2021.

[PPMM21a] Sankar K. Pal, Anima Pramanik, Jhareswar Maiti und Pabitra Mitra. Deep learning in multi-object detection and tracking: state of the art. *Applied Intelligence*, 51(9):6400–6429, 2021.

[PPMM21b] Anima Pramanik, Sankar K. Pal, Jhareswar Maiti und Pabitra Mitra. Granulated RCNN and Multi-Class Deep SORT for Multi-Object Detection and Tracking. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(1):171–181, 2021.

[QSMG17] Charles R. Qi, Hao Su, Kaichun Mo und Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 652–660, 2017.

[RCCFR21] Ivan Rodriguez-Conde, Celso Campos und Florentino Fdez-Riverola. Optimized convolutional neural network architectures for efficient on-device vision-based object detection. *Neural Computing and Applications*, S. 1–33, 2021.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick und Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 779–788, 2016.

[RF17] Joseph Redmon und Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 7263–7271, 2017.

[RF18] Joseph Redmon und Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick und Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[RN10] Stuart Russell und Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 3. Aufl., 2010.

[Ros16] Adrian Rosebrock. Measuring size of objects in an image with OpenCV. `https://pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/`, 2016. Besucht am 14.07.2022.

[RT21] Matthias Recke und Karsten Tolle. Projektantrag der LOEWE-Exploration: Künstliche Intelligenz zur Erschließung kolonialer Verwertungspraktiken archäologischer Objektsammlungen. 2021.

[RTL09] Payam Refaeilzadeh, Lei Tang und Huan Liu. Cross-validation. *Encyclopedia of Database Systems*, 5:532–538, 2009.

[RYNG21] Abraham Resler, Reuven Yeshurun, Filipe Natalio und Raja Giryes. A deep-learning model for predictive archaeology and archaeological community detection. *Humanities and Social Sciences Communications*, 8(1):1–10, 2021.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[Sch21] Veit Schiele. IPython-Magie. `https://jupyter-tutorial.readthedocs.io/de/latest/workspace/ipython/magics.html`, 2021. Besucht am 16.11.2022.

[SHA17] Hasbi A. Shiddieqy, Farkhad Ihsan Hariadi und Trio Adiono. Implementation of deep-learning based image classification on single board computer. In *2017 International Symposium on Electronics and Smart Devices*, S. 133–137. IEEE, 2017.

[SIG+18] Muhammad A. Shahid, Muhammad A. Iftikhar, Zaheer A. Gondal, Saima Rathore und Muhammad Adnan. Object Size Measurement through Images: An Application to Measuring Human Foot Size. In *2018 International Conference on Frontiers of Information Technology*, S. 298–302. IEEE, 2018.

[SJZ21] Christoph Sager, Christian Janiesch und Patrick Zschech. A survey of image labelling for computer vision applications. *Journal of Business Analytics*, S. 91–110, 2021.

[SK19] Connor Shorten und Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):1–48, 2019.

[Sme19] Kousai Smeda. Understand the architecture of CNN. `https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7`, 2019. Besucht am 01.06.2022.

[Sol20] Jacob Solawetz. How to Train a TensorFlow 2 Object Detection Model. `https://blog.roboflow.com/train-a-tensorflow2-object-detection-model/`, 2020. Besucht am 30.05.2022.

[SQ17] Wesley E. Snyder und Hairong Qi. *Fundamentals of Computer Vision*. Cambridge University Press, 2017.

[SS21] Samrat Sahoo und Jacob Solawetz. How to Train YOLOR on a Custom Dataset. `https://blog.roboflow.com/train-yolor-on-a-custom-dataset/`, 2021. Besucht am 14.06.2022.

[SZ14] Karen Simonyan und Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TB01] Veronica Tatton-Brown. Cyprus in the 19th Century AD. *Fact, Fancy and Fiction*, 2001.

[Tew22] Anubhav Tewari. YOLOR model architecture. `https://iq.opengenus.org/yolor/`, 2022. Besucht am 16.11.2022.

[TL19] Mingxing Tan und Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, S. 6105–6114. PMLR, 2019.

[TPL20] Mingxing Tan, Ruoming Pang und Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, S. 10781–10790, 2020.

[Tsa18] Christos I. Tsatsoulis. Stack Overflow-Beitrag: Sklearn StratifiedKFold: ValueError: Supported target types are: ('binary', 'multiclass'). Got 'multilabel-indicator' instead. `https://stackoverflow.com/a/48512157`, 2018. Besucht am 16.11.2022.

[TSSE20] Michael Telahun, Daniel Sierra-Sossa und Adel S. Elmaghraby. Heuristic Analysis for In-Plane Non-Contact Calibration of Rulers Using Mask R-CNN. *Information*, 11(5):259, 2020.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato und Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 1701–1708, 2014.

[TZP17] Ye Tao, Ming Zhang und Mark Parsons. Deep learning in photovoltaic penetration classification. In *2017 IEEE Power & Energy Society General Meeting*, S. 1–5. IEEE, 2017.

[Vla20] Lyudmil Vladimirov. TensorFlow 2 Object Detection API tutorial. `https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/`, 2020. Besucht am 17.09.2022.

[Wan21a] Chien-Yao Wang. GitHub-Beitrag: can't convert cuda:0 device type tensor to numpy. Use Tensor.cpu(). `https://github.com/WongKinYiu/yolor/issues/113#issuecomment-937791665`, 2021. Besucht am 16.11.2022.

[Wan21b] Chien-Yao Wang. GitHub-Beitrag: Image size for training. `https://github.com/WongKinYiu/yolor/issues/114#issuecomment-937618535`, 2021. Besucht am 16.11.2022.

[WBL21] Chien-Yao Wang, Alexey Bochkovskiy und Hong-Yuan M. Liao. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, S. 13029–13038, 2021.

[Wei20] Xinjun Wei. GitHub-Beitrag: InvalidArgumentError: Index out of range using input dim 1; input has only 1 dims [Op:StridedSlice] name: strided_slice. `https://github.com/tensorflow/models/issues/9255/#issuecomment-698693650`, 2020. Besucht am 16.11.2022.

[Wir21] Alicia Wirth. Einfluss von Bildannotationstechniken auf die Genauigkeit von Machine Learning-Modellen zur Objekterkennung. Goethe-Universität Frankfurt am Main. `http://www.bigdata.uni-frankfurt.de/wp-content/uploads/2021/12/2021_FP_Alicia_online.pdf`, 2021. Besucht am 29.05.2022.

[WLW+20] Chien-Yao Wang, Hong-Yuan M. Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh und I-Hau Yeh. CSPNet: A New Backbone That Can Enhance Learning Capability of CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, S. 390–391, 2020.

[WYL21] Chien-Yao Wang, I-Hau Yeh und Hong-Yuan M. Liao. You Only Learn One Representation: Unified Network for Multiple Tasks. *arXiv preprint arXiv:2105.04206*, 2021.

[XTY+20] Youzi Xiao, Zhiqiang Tian, Jiachen Yu, Yinshu Zhang, Shuai Liu, Shaoyi Du und Xuguang Lan. A review of object detection based on deep learning. *Multimedia Tools and Applications*, 79(33):23729–23791, 2020.

[ZWB+18] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei und Stan Z. Li. Single-Shot Refinement Neural Network for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 4203–4212, 2018.

[ZYH+13] Xin Zhang, Yee-Hong Yang, Zhiguang Han, Hui Wang und Chao Gao. Object class detection: A survey. *ACM Computing Surveys*, 46(1):1–53, 2013.