Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe. Die Stellen, die anderen Werken (gilt ebenso für Werke aus elektronischen Datenbanken oder aus dem Internet) wörtlich oder sinngemäß entnommen sind, habe ich unter Angabe der Quelle und Einhaltung der Regeln wissenschaftlichen Zitierens kenntlich gemacht. Diese Versicherung umfasst auch in der Arbeit verwendete bildliche Darstellungen, Tabellen, Kartenskizzen und gelieferte Zeichnungen.

Mir ist bewusst, dass Täuschungen nach der für mich gültigen Studien- und Prüfungsordnung / nach § 6 RaPO / § 48 BayVwVfG geahndet werden.

Die Zustimmung zur elektronischen Plagiatsprüfung wird erteilt.

_____ _____

Ort, Datum Unterschrift des Verfassers / der Verfasserin

Kamera-basierte Erkennung von Flurförderzeugen inkl. Aufbereitung von Trainingsdaten

Studienarbeit von

Taim Alshara

20. März 2022

Autor:
Taim Alshara
Matrikelnummer:
22 19 804

Erstprüfer:
Prof. Dr. Doll

TECHNISCHE HOCHSCHULE ASCHAFFENBURG

FAKULTÄT INGENIEURWISSENSCHAFTEN

WÜRZBURGER STRASSE 45

D-63743 ASCHAFFENBURG

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Bounding Box	Begrenzungsrahmen
CNN	Faltungsneuronale Netze
mAP	Mittlere durchschittliche Genauigkeit
R-CNN	Regionale Faltungsneuronale Netze
Fast R-CNN	Schnelle Regionale Faltungsneuronale Netze

1 Einleitung

Das Trainieren eines Objekterkennungsmodells mit einem benutzerdefinierten Datensatz eigener Wahl von Grund auf ist ein langwieriger Prozess.
Begonnen wird mit dem Aufbau eines Modells unter Verwendung eines Feature Pyramid Network in Kombination mit einem Region Proposal Network, wenn man sich für die auf Region Proposal (dt. *Regionsvorschlägen*) basierende Methoden wie Faster R-CNN entscheidet. Als Alternative Vorgehensweise, können auch One-Shot-Detektoralgorithmen wie SSD und YOLO verwendet werden.
Mit beiden Vorgehensweisen ist es etwas kompliziert zu arbeiten, wenn die Netzwerke von Grund auf neu implementiert werden. Optimal wäre, ein Framework, in dem modernste Modelle wie Fast oder Faster R-CNNs problemlos verwendet werden können.

1.1 Motivation

Facebook Research hat vorgefertigte Detectron2-Versionen veröffentlicht, die die lokale Installation erheblich vereinfachen. (Getestet auf Linux und Windows)
Neben der Veröffentlichung von PyTorch Version 1.3 veröffentlichte Facebook auch eine grundlegende Neufassung seines Objekterkennungs-Frameworks Detectron.
Das neue Framework heißt Detectron2 und ist nun in PyTorch statt in Caffe2 implementiert.
Detectron2 ermöglicht es, Objekterkennungsmodelle einfach zu verwenden und zu erstellen sowie modernste Modelle wie Fast, Faster und Mask R-CNNs problemlos zu verwenden

1.2 Aufgabenstellung

Ziel der Arbeit ist es ein eigenes Modell mit einem benutzerdefinierten Datensatz mit Detectron2 zu trainieren. Alle im “model zoo” der Detectron2-Bibliothek vorhandenen Modelle sind auf dem COCO Dataset vortrainiert. Es muss nur der benutzerdefinierte Datensatz anhand des vortrainierten Modells optimiert werden.

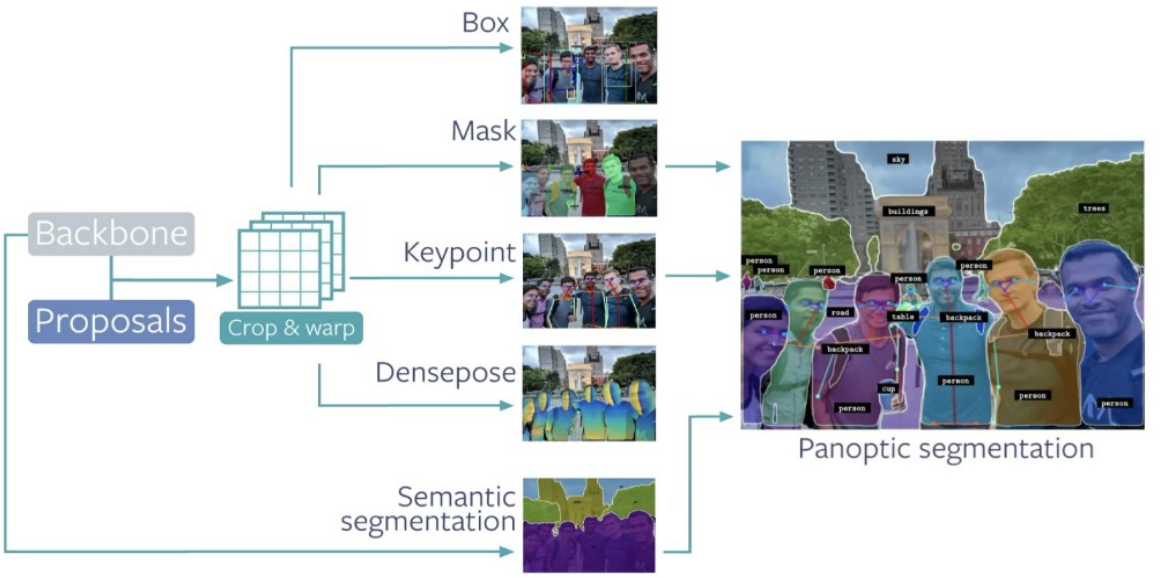

Abbildung 1: Arten von Objekterkennungsmodellen, die Detectron2 bietet. [1]

In Abbildung 1 sind die verschiedene Arten von Objekterkennungsmodellen, die Detectron2 bietet, zu sehen.
Die Instanzerkennung bezieht sich auf die Klassifizierung und Lokalisierung eines Objekts mit einem umgebenden Rahmen (engl. Bounding Box).
In dieser Arbeit werden Flurförderzeuge aus Bildern mit dem Faster RCNN-Modell aus dem “model zoo” des Detectron2 Identifiziert.
Implementiert wird ein Modell, bei dem die Ausgabe auf die Weise, wie es in Abbildung 2 zu sehen ist, erfolgen würde.

Abbildung 2: Erwartetes Resultat

1.3 Gliederung

Die Arbeit ist in vier Kapitel unterteilt. Kapitel 1 enthält die Motivation und Aufgabenstellung sowie die Gliederung. Anschließend werden verschiedene Architekturen von Netzen in Kapitel 2 vorgestellt und erklärt. Kapitel 3 zeigt die einzelnen Schritte zum Trainieren eines benutzerdefinierten Modells. Das Fazit im Kapitel 4 schließt diese Arbeit ab.

2 Netzwerkarchitekturen

In Kapitel 2.1 werden die CNNs (d.h. Faltungsneuronale Netze) vorgestellt. Anschließend wird die Architektur von einer einfachen R-CNN-Netzwerkarchitektur gezeigt. Im Abschluss erfolgt dann eine Einführung des Mask R-CNN.
Die Abbildung [3] zeigt, wie sich die CNN-Architekturen im laufe der Zeit entwickelt haben.

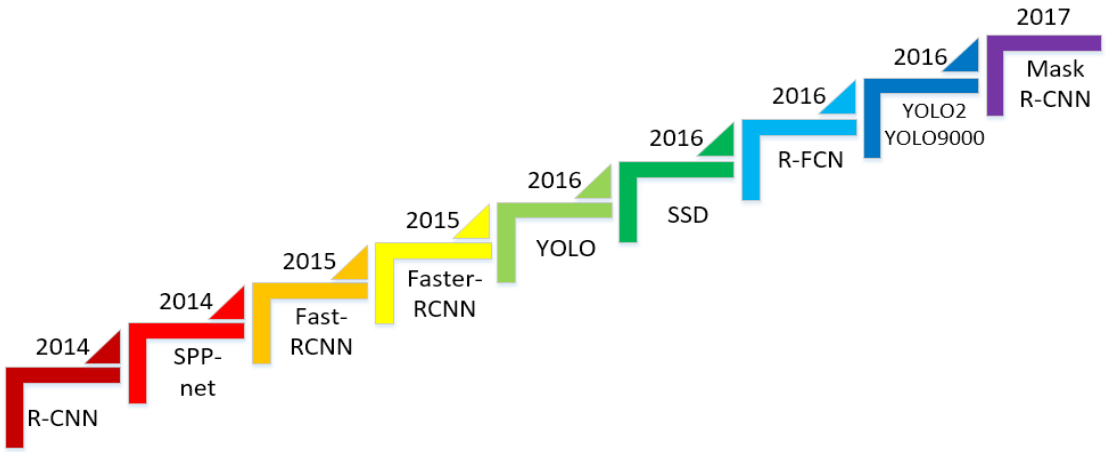

Abbildung 3: Entwicklung der CNN-Architekturen [8]

2.1 CNN (Convolutional Neural Network – Faltungsneuronale Netze)

CNN ist ein tiefes neuronales Netzwerk, das für die Bildanalyse entwickelt wurde. Es verfügt über hervorragende Fähigkeiten in der sequentiellen Datenanalyse sowie in der Verarbeitung natürlicher Sprache. Convolutional Neural Network (CNN) ist die Hauptkategorie der Bildklassifizierung, Objekterkennung, Bilderkennung usw. Es nimmt Input und klassifiziert den Output. Es nimmt das Bild von Pizza, Burger, Sandwich, Getränken als Eingabe und klassifiziert es in diese Kategorie. Im Gegensatz zu herkömmlichen mehrschichtigen Perzeptron-Architekturen verwendet das CNN zwei Prozesse namens Faltung und Bündelung, um ein Bild auf seine wesentlichen Merkmale zu reduzieren, und verwendet diese Merkmale, um das Bild zu verstehen und zu klassifizieren. Abbildung [4] zeigt ein Beispiel eines CNN-Netzwerks.

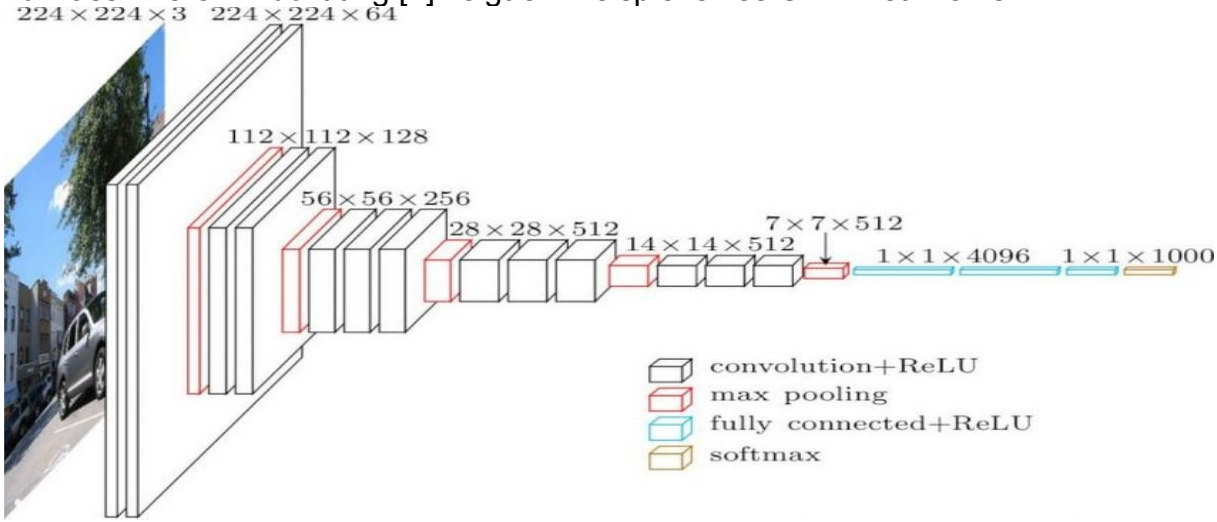

Abbildung 4: Ein Beispiel der Architektur eines Convolutional Neural Network. [9]

Es gibt fünf verschiedene Schichten in einem CNN:

- Eingabeschicht
- Convolution-Layer (Faltungsschicht)
- Pooling-Layer (Bündelungsschicht)
- Vollständig verbundene (FC) Schicht
- Softmax / Logistikschicht
- Ausgangsschicht

2.2 R-CNN (regionales CNN)

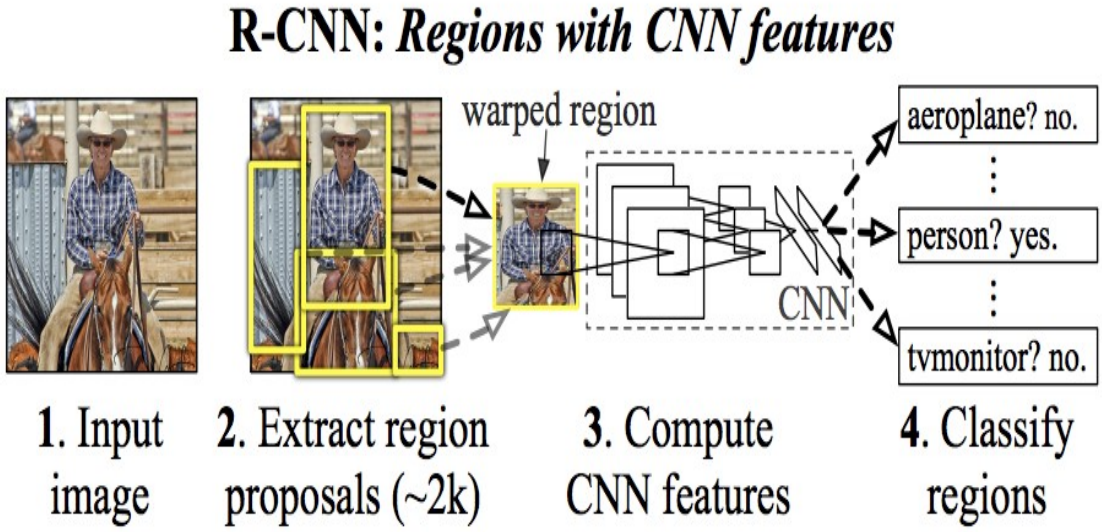

Abbildung 5: R-CNN-Netzwerkarchitektur [6]

Das R-CNN, deren Netzwerkarchitekturen in Abbildung [5] zu sehen ist, wurde 2014 von Ross Girshick vorgeschlagen, um das Problem der effizienten Objektlokalisierung bei der Objekterkennung zu lösen. In R-CNN wird das Bild in etwa 2000 Regionsempfehlungen (Regionsvorschläge) aufgeteilt und ein CNN (ConvNet) wird der Reihe nach auf jede Region angewendet. Die Größe der Regionen wird bestimmt und die richtige Region in das neuronale Netz platziert. Da jede Region im Bild das CNN separat durchgeführt wird, ist der Betrag der Trainingszeit sehr hoch.
Nachteile des R-CNN

- Das Training des Netzwerks nimmt viel Zeit in Anspruch, da jedes Bild 2000 Regionsvorschläge klassifizieren muss.
- Außerdem wird viel Speicherplatz benötigt.

2.3 Fast R-CNN

Da das Bild in R-CNN in 2000 Regionsempfehlungen unterteilt ist, würde das Training in R-CNN viel Zeit in Anspruch nehmen und die Kosten wären hoch. Fast R-CNN wird vorgeschlagen, um dieses Problem von R-CNN zu lösen.Es nimmt alle Bild- und Regionsempfehlungen in einer einzigen Vorwärtsausbreitung als Eingabe in die CNN-Architektur. (Erstens trennt es sich nicht nach offiziellen Regionsempfehlungen.) Es kombiniert auch verschiedene Teile von Architekturen (wie ConvNet, RoI-Pool und Klassifizierungsschicht) zu einer vollständigen Architektur. Dadurch entfällt auch die Notwendigkeit, eine Feature-Map zu speichern, und es wird Speicherplatz gespart. Es verwendet auch Softmax-Layer anstelle von SVM (Support Vector Machine) bei der Zonenempfehlungsklassifizierung, die sich als schneller er-

wiesen hat und eine bessere Genauigkeit bietet als SVM. Fast R-CNN schließt das Training in weniger Zeit ab, und verbessert die Erkennungszeit gegenüber R-CNN erheblich. Es verbessert auch die durchschnittliche Empfindlichkeit (mAP) im Vergleich zu R-CNN.
Abbildung [6] zeigt die Fast R-CNN-Netzwerkarchitektur.

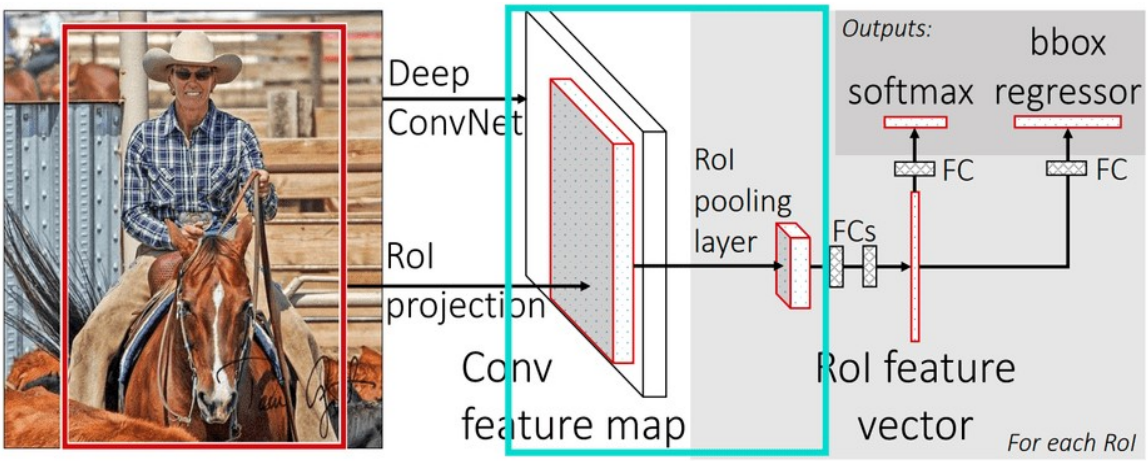

Abbildung 6: Fast R-CNN-Netzwerkarchitektur [7]

3 Grundlagen Maschinelles Lernen

In Kapitel 3.1 werden die Komponenten des Maschinellen Lernens nahegebracht. Abschließend in Kapitel 3.2 erfolgt die Aufführung von den Hauptschritten im maschinellen Lernprozess.

3.1 Komponenten des maschinellen Lernens

Nahezu alle Aufgaben, die mit maschinellem Lernen gelöst werden, umfassen drei Hauptkomponenten:

- Ein maschinelles Lernmodell
- Ein Modelltrainingsalgorithmus
- Ein Modellinferenzalgorithmus

Um die Beziehungen zwischen diesen Komponenten besser verstehen zu können, stellt man sich die Phasen, wie in Abbildung 7 dargestellt ist, vor, in denen eine Teekanne aus einem Klumpen Ton hergestellt wird.

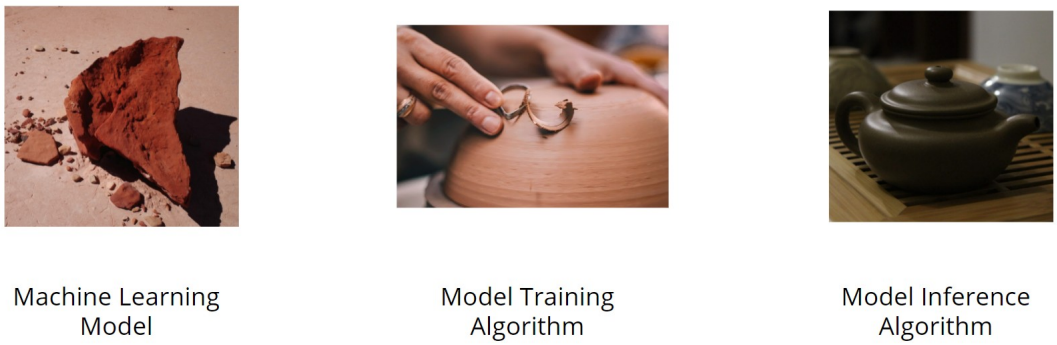

Abbildung 7: Clay-Analogie des maschinellen Lernens [10]

Begonnen wird mit einem Block aus Rohton. In dieser Phase, kann der Ton in viele verschiedene Formen gebracht und für viele verschiedene Zwecke verwendet werden. Man beschließt, aus diesem Tonklumpen eine Teekanne herzustellen.
Man inspiziert und analysiert den rohen Ton und entscheidet, wie man ihn ändern kann, damit er mehr wie die Teekanne aussieht, die man sich vorgestellt hat.
Im letzten Phase wird der Ton geformt, damit er mehr wie die Teekanne aussieht, die das Ziel ist.
Ein maschinelles Lernmodell kann wie ein Stück Ton in viele verschiedene Formen gebracht werden und vielen verschiedenen Zwecken dienen. Eine eher technische Definition wäre, dass ein maschinelles Lernmodell ein Codeblock oder Framework ist, das modifiziert werden kann, um verschiedene, aber verwandte Probleme basierend auf den bereitgestellten Daten zu lösen.
Ein Modell ist ein extrem generisches Programm (oder ein Codeblock), das durch die zum Trainieren verwendeten Daten spezifisch gemacht wird. Es wird verwendet, um verschiedene Probleme zu lösen.
Modelltrainingsalgorithmen durchlaufen einen interaktiven Prozess, bei dem die aktuelle Modelliteration analysiert wird, um festzustellen, welche Änderungen vorgenommen werden können, um dem Ziel näher zu kommen. Diese Änderungen werden vorgenommen und die Iteration wird fortgesetzt, bis das Modell evaluiert wurde, um die Ziele zu erreichen.
Bei der Modellinferenz wird das trainierte Modell zum Generieren von Vorhersagen verwendet.

3.2 Haupt Schritte im maschinellen Lernprozess

In Abbildung [8] ist ein Überblick über die Haupt Schritte des maschinellen Lernprozesses zu sehen. Abgesehen von dem verwendeten spezifischen Modell oder Trainingsalgorithmus geben Machine-Learning-Experten einen gemeinsamen Workflow vor, um Machine-Learning-Aufgaben zu erfüllen.

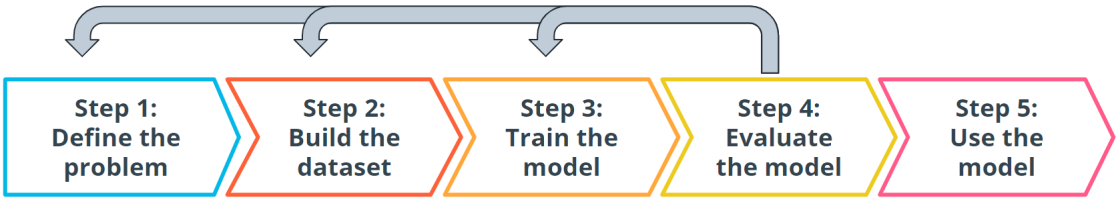

Abbildung 8: Hauptschritte im maschinellen Lernprozess [10]

Diese Schritte sind iterativ. In der Praxis bedeutet dies, dass bei jedem Schritt des Prozesses überprüft wird, wie der Prozess verläuft. Sollten die Dinge nicht Funktionieren, wie es erwartet ist, wird zurückgegangen und aktueller Schritt oder vorherige Schritte überprüft , um zu versuchen, die Aufschlüsselung zu identifizieren.

3.2.1 Das Problem definieren

Alle Modelltrainingsalgorithmen und die Modelle selbst verwenden Daten als Eingabe. Ihre Ausgaben können sehr unterschiedlich sein und werden basierend auf der zu lösenden Aufgabe, in einige verschiedene Gruppen eingeteilt. Häufig wird die Art von Daten verwendet, die zum Trainieren eines Modells als Teil der Definition einer maschinellen Lernen Aufgabe erforderlich sind. In dieser Arbeit, wird auf einer der gängigen Aufgaben des maschinellen Lernens eingegangen, welche sich ‚Überwachtes Lernen' nennt.
Das Vorhandensein oder Fehlen eines Labels (d.h. Kennzeichnung) in Daten wird häufig verwendet, um eine maschinelles Lernen Aufgabe zu identifizieren.
Die Abbildung 9 zeigt den Unterschied zwischen Überwachtem und Unüberwachtem Lernen.

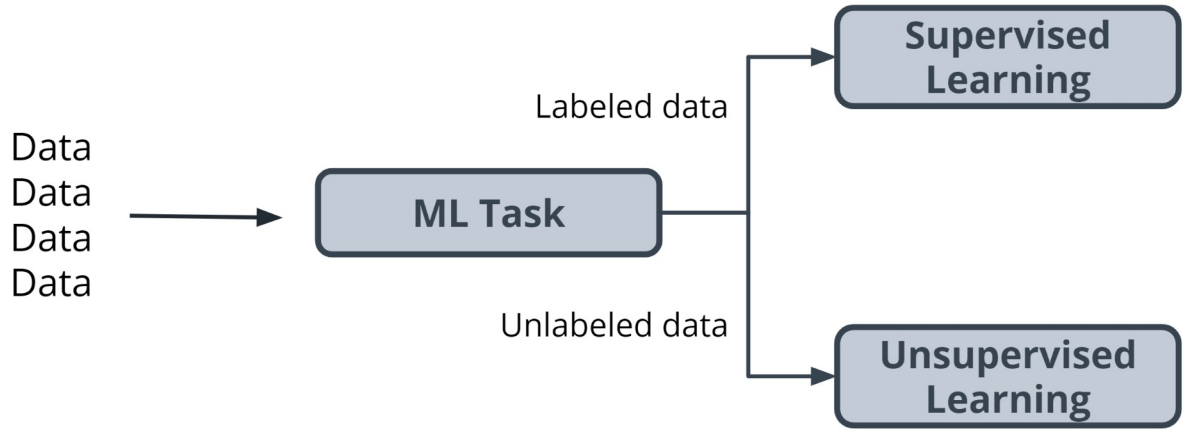

Abbildung 9: Überwachtes und unüberwachtes Lernen [10]

Eine Aufgabe ist überwacht, wenn labeled Daten (d.h. beschriftete Daten) verwendet werden. Der Begriff Labeled wird verwendet, um auf Daten zu verweisen, die bereits die Lösungen enthalten, die als Labels bezeichnet werden.
Ein kategorisches Label hat einen diskreten Satz möglicher Werte. Bei einem maschinellen Lernen Problem, bei dem die Art der Flurförderzeuge anhand eines Bildes identifiziert soll, würde das Modell mit Bildern trainiert werden, die mit den Kategorien der Flurförderzeugen gekennzeichnet sind, die identifiziert sollen. Wenn mit kategori-

alen Labels gearbeitet wird, führt man außerdem häufig Klassifizierungsaufgaben durch, die Teil der Familie des überwachten Lernens sind.

3.2.2 Erstellen eines Datensatzes

Die Arbeit mit Daten ist der wichtigste Schritt des maschinellen Lernprozesses. Im Jahr 2017 zeigte eine O'Reilly-Studie, dass Praktiker des maschinellen Lernens 80 % ihrer Zeit damit verbringen, mit ihren Daten zu arbeiten.
Die vier Aspekte der Arbeit mit Daten sind in Abbildung [10] dargestellt

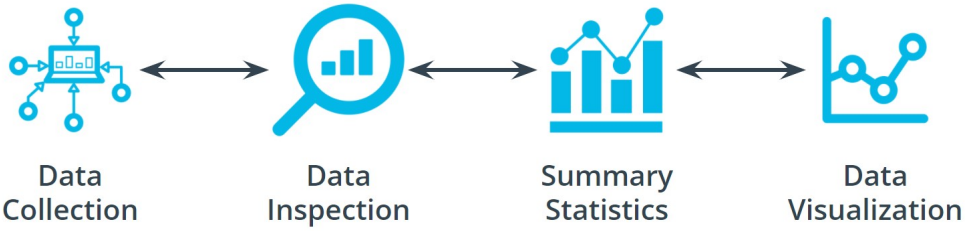

Abbildung 10: Die vier Aspekte der Arbeit mit Daten [10]

3.2.2.1 Datenerfassung

Die Datenerfassung kann so unkompliziert sein wie das Ausführen der entsprechenden SQL-Abfragen oder so kompliziert wie das Erstellen benutzerdefinierter Web-Scraper-Anwendungen zum Sammeln von Daten für das eigene Projekt. Möglicherweise muss ein Modell über die Daten laufen gelassen, um benötigte Labels zu generieren. Hier ist die grundlegende Frage: Stimmen die gesammelten Daten mit der maschinellen Lernaufgabe und dem definierten Problem überein?

3.2.2.2 Datenprüfung

Die Qualität der Daten ist letztendlich der größte Faktor, der sich darauf auswirkt, wie gut man von dem Modell erwarten kann, dass es funktioniert. Beim Überprüfen der Daten auf Folgendes geachtet werden:

- Ausreißer
- Fehlende oder unvollständige Werte
- Daten, die transformiert oder vorverarbeitet werden müssen, damit das richtige Format im Modell verwendet wird.

3.2.2.3 Zusammengefasste Statistiken

Modelle können Annahmen darüber treffen, wie die Daten strukturiert sind. Nachdem man einige Daten zur Hand hat, empfiehlt es sich, zu überprüfen, ob die Daten mit den zugrunde liegenden Annahmen des gewählten Modells für maschinelles Lernen übereinstimmen. Mithilfe statistischer Tools , können Dinge wie der Mittelwert, den Bereich des inneren Quantils (IQR) und die Standardabweichung berechnet werden. Diese Tools können Einblicke in Umfang und Form eines Datensatzes geben.

3.2.2.4 Datenvisualisierung

Datenvisualisierung kann verwenden werden, um Ausreißer und Trends im Datensatz zu sehen.

3.2.3 Das Model trainieren

Trainieren des Modells ist ein Prozess, bei dem die Parameter des Modells iterativ aktualisiert werden, um eine zuvor definierte Verlustfunktion zu minimieren.
Modellparameter sind Einstellungen oder Konfigurationen, die der Trainingsalgorithmus aktualisieren kann, um das Verhalten des Modells zu ändern. Je nach Kontext

werden spezifische Begriffe zur Beschreibung von Modellparametern wie Gewichtungen und Bias verwendet. Gewichtungen, bei denen es sich um Werte handelt, die sich ändern, wenn das Modell lernt, sind spezifischer für neuronale Netze.
Eine Verlustfunktion wird verwendet, um die Entfernung des Modells von einem Ziel zu kodifizieren.
Der erste Schritt beim Trainieren des Modells besteht darin, den Datensatz zufällig aufzuteilen. Auf diese Weise können einige Daten während des Trainings verborgen gehalten, sodass die Daten zur Bewertung des Modells verwendet werden können, bevor es in Produktion zu nehmen. Insbesondere wird dies getan, um gegen den Bias-Varianz-Trade-off zu testen. Nachdem aufteilen des Datensatzes, erhalten man zwei Datensätze:

- Trainingsdatensatz: Die Daten, mit denen das Modell trainiert wird. Sie sind 80% der gesamten Daten.
- Testdatensatz: Die Daten, die dem Modell während des Trainings vorenthalten werden, um zu testen, wie gut sich das Modell auf neue Daten verallgemeinern lässt.

Zusammengefasst ist das Trainingsprozess:

- Eingabe der Trainingsdaten in das Modell.
- Berechnen der Verlustfunktion auf den Ergebnissen.
- Aktualisieren der Modellparameter in eine Richtung, die Verluste reduziert.

Es wird fortgefahren, diese Schritte zu durchlaufen, bis eine vordefinierte Stoppbedingung erreicht ist. Dies kann auf der Trainingszeit, der Anzahl der Trainingszyklen oder einem noch intelligenteren Mechanismus basieren.

3.2.4 Auswerten eines trainierten Modells

Nachdem die Daten gesammelt sind und ein Modell trainiert ist, kann damit begonnen werden, die Leistung des Modells zu bewerten. Die für die Bewertung verwendeten Metriken sind wahrscheinlich sehr spezifisch für das definierte Problem. Die Modellgenauigkeit ist eine ziemlich verbreitete Bewertungsmetrik. Genauigkeit ist der Bruchteil der Vorhersagen, die ein Modell richtig macht.
Jeder Schritt des maschinellen Lernprozesses, welcher in Abbildung 8 dargestellt sind, den durchgelaufen wird, ist hochgradig iterativ und kann im Laufe eines Projekts geändert oder neu festgelegt werden. Bei jedem Schritt wird möglicherweise festgestellt, dass man zurückgeht und einige Annahmen, die man in den vorherigen Schritten hatte, neu bewerten muss.
Der Log Loss (d.h. Protokollverlust) könnte verwendet werden, um die Unsicherheit des Modells in Bezug auf eine bestimmte Vorhersage zu verstehen. In einem einzigen Fall könnte das Modell mit 5 prozentiger Sicherheit vorhersagen, dass ein Bild ein Flurförderzeug enthalten wird. In einem anderen Fall könnte das Modell mit 80 prozentiger Sicherheit vorhersagen, dass ein Bild ein Flurförderzeug enthalten wird. Mit dem Protokollverlust kann gemessen werden, wie stark das Modell glaubt, dass seine Vorhersage richtig ist. In beiden Fällen sagt das Modell voraus, dass ein Bild ein Flurförderzeug enthalten wird, aber die Gewissheit des Modells bezüglich dieser Vorhersage kann sich ändern.

3.2.5 Inferenz des Modells

In diesem Schritt kann das Modell bereitgestellt werden. Sobald das Modell trainiert und seine Effektivität bewertet ist sowie befriedigende Ergebnisse hat, können Vorhersagen zu realen Problemen erstellt werden, indem unsichtbare Daten im Feld verwenden werde. Beim maschinellen Lernen wird dieser Vorgang oft als Inferenz bezeichnet.

4 Trainieren eines benutzerdefinierten Modells

In Kapitel 3 wird beschrieben wie, das benutzerdefinierte Modell trainiert wird. Dazu wird in 4.1 auf die Vorbereitung des Datensatzes eingegangen, und in 4.2 die Visualisierung gezeigt. Anschließend in 4.3 erfolgt die Auswertung sowie Inferenz des trainiertem Modells in Kapitel 4.4. Abschließend darf das trainierte Modell in Kapitel 4.5 ausgewertet.

4.1 Vorbereiten & Registrieren des Datensatzes

Erstens müssen die benötigten Modulen Importieren werden, dazu gehören die bekannten Bibliotheken wie: os, numpy, json, random und cv2.
Außerdem müssen einiger bekannten detectron2 utilities wie: model_zoo, DefaultPredictor, MetadataCatalog, DatasetCatalog und viele weitere Modulen die im Dokumentation von Detectron2 [5] zu finden sind.
Wenn ein benutzerdefinierter Datensatz verwendet ist und gleichzeitig die Datenlader von detectron2 wiederverwendet wird, muss der Datensatz registriert werden (d.h. detectron2 mitteilen, wie es den Datensatz erhält). Verwendet wird der Flurförderzeugserkennungs-Datensatz, der eine einzige Klasse hat: “Flurförderzeug”.
Trainiert wird ein Flurförderzeugs-erkennungsmodell aus einem bestehenden Modell, das auf dem COCO-Datensatz vortrainiert wurde, der im Modellzoo von detectron2 verfügbar ist.
Es gibt bestimmte Formate, wie Daten in ein Modell eingespeist werden können, z. B. ein YOLO-Format, ein PASCAL-VOC-Format, ein COCO-Format und so weiter. Detectron2 akzeptiert das COCO-Format des Datensatzes. Das COCO-Format des Datensatzes besteht aus einer JSON-Datei, die alle Details eines Bildes wie Größe, Annotationen (d.h. Bounding Box Koordinaten), Beschriftungen, die dem Bounding Box entsprechen, usw. enthält. Tabelle 1 zeigt ein Beispiel des verwendetem JSON-Dateien des Datensatzes für ein einziges Bild.
Es gibt verschiedene Arten von Formaten für die Bounding-Box-Darstellung. Für Detectron2 muss es Mitglied von structure.BoxMode sein. Es gibt 5 solcher Formate. Derzeit sind sind BoxMode.XYXY_ABS und BoxMode.XYWH_ABS unterstützt. Verwendet wird das zweite Format. (X, Y) repräsentiert eine Koordinate des Bounding Box, und W, H repräsentiert Breite und Höhe dieses Bounding Box’es. Die category_id bezieht sich auf die Klasse, zu der die Box gehört.
Dann muss der Datensatz registriert werden.
Um zu überprüfen, ob das Laden der Daten korrekt ist, werden die Annotationen zufällig ausgewählter Stichproben im Trainingssatz visualisiert.

	images	annotations	categories
id	9526	136	9526
dataset_id	12		
path	0		
width	1920		
height	1080		
file_name	0		
image_id ≙ id		9526	
category_id		7	
segmentation		912.6, 798.0, 912.6, 830.3, 874.0, 830.3, 874.0, 798.0	
area		1248	
bbox		0	
iscrowd		0	
isbbox		0	
color		0	0
metadata			
Name			0
Supercategory			0
metadata			0

Tabelle 1: Das Datensatzformat von Detectron2

4.2 Visualisieren des Trainingssatzes

Aus dem train Ordner unseres Datensatzes wird zufällig 1 Bild ausgewählt. dann wird beobachtet, wie die Bounding Boxen aussehen.
Die Ausgabe sieht aus, wie sie in Abbildung 11 dargestellt ist:

Abbildung 11: Visualisieren des Trainingssatzes inkl. Annotationen

4.3 Trainieren des Modells

Dies ist der Schritt, in dem die Konfigurationen vorgenommen und das Modell für das Training vorbereitet wird. Technisch wird der Modell nur anhand des Datensatzes optimiert, da das Modell bereits auf COCO Dataset vortrainiert ist.
Im “Model Zoo” von Detectron2 stehen unzählige Modelle zur Objekterkennung zur Verfügung. Hier wird das Modell 'faster_rcnn_R_50_FPN_3x' verwendet, das auf hohem Niveau , wie in Abbildung 12 dargestellt ist, aussieht.

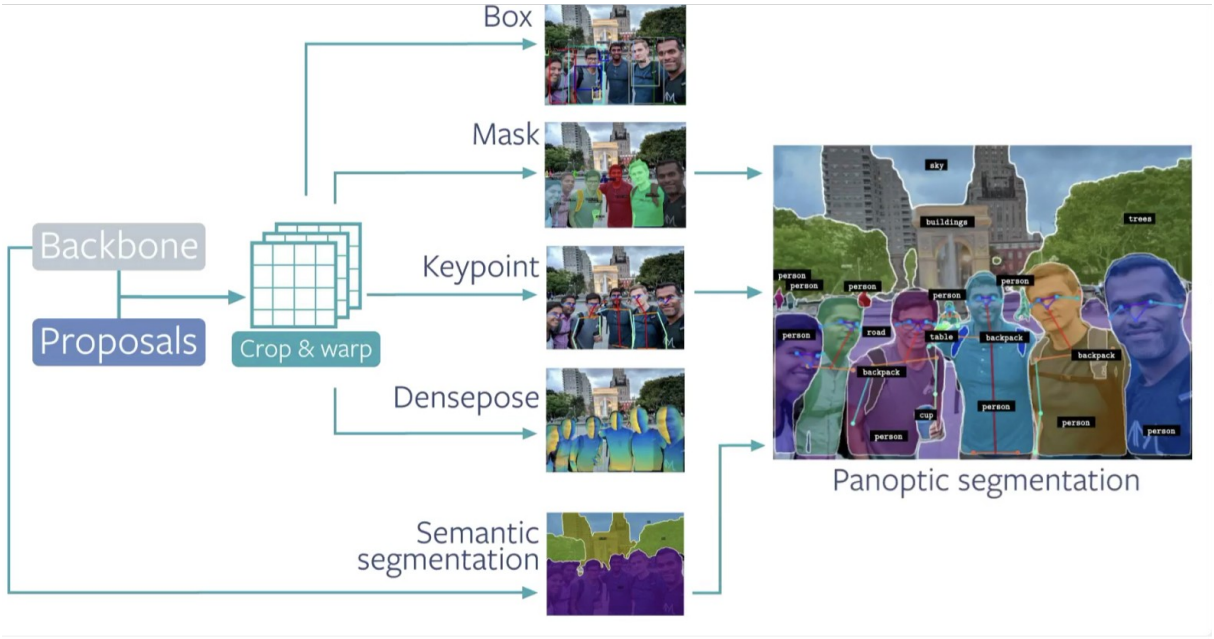

Abbildung 12: 'faster_rcnn_R_50_FPN_3x' Modell [1]

Es gäbe ein Backbone-Netzwerk (in diesem Fall Resnet), das zum Extrahieren von Merkmalen aus dem Bild verwendet wird, gefolgt von einem Region Proposal Network zum Vorschlagen von Regionsvorschlägen und einem Box Head zum Verengen des Bounding Box’es.
Die Konfiguration für das Training wird festgelegt, wobei das Modell mit ein tausend Iterationen und eine Loss Rate von 0.00025 trainiert wird.
Die Genauigkeit kann auch für andere Konfigurationen verbessert werden. Schließlich kommt es auf die Wahl der richtigen Hyperparameter an.

4.4 Inferenz mit dem trainierten Modell

Es ist an der Zeit, die Ergebnisse abzuleiten, indem das Modell auf dem Validierungsset getestet wird. Nach erfolgreichem Abschluss des Trainings wird ein Output Ordner im lokalen Speicher gespeichert, in dem die endgültigen Gewichte gespeichert werden. Für zukünftige Rückschlüsse aus diesem Modell kann dieser Ordner gespeichert werden.
Die Ausgabe sieht aus, wie sie in Abbildung 2 dargestellt ist.

4.5 Auswertung des trainierten Modells

Normalerweise wird das Modell gemäß den COCO-Bewertungsstandards bewertet. Die mittlere durchschnittliche Genauigkeit (mAP) wird verwendet, um die Leistung des Modells zu bewerten. Außerdem ist IoU-Schwellenwert eine bedeutsamer Größe. Es steht für „Schnittpunkt über Union“, welcher ein Wert ist, der bei der Objekterkennung verwendet wird, um die Überlappung eines vorhergesagten mit einem tatsächlichen Begrenzungsrahmen für ein Objekt zu messen. Je näher die vorhergesagten Bounding-Box-Werte an den tatsächlichen Bounding-Box-Werten liegen, des-

to größer ist die Schnittmenge und desto größer ist der IoU-Wert. Der allgemeine Schwellenwert für die IOU kann 0,5 betragen. Dies kann von Problem zu Problem unterschiedlich sein. Normalerweise wird IOU>0,5 als gute Vorhersage angesehen.
Zuerst muss der COCO Evaluator importiert werden, um die COCO Metriken benutzen zu können. dann wird die COCO Evaluator Funktion aufgerufen und ihr der Validierungsdatensatz übergeben. Danach wird das im vorherigen Schritt erstellte vorhergesagte Modell verwendet.

AP	AP50	AP75	APs	APm	AP1
34.877	82.739	38.022	0.000	27.229	47.864

Tabelle 2: Bewertungsmetrik

Wie in Tabelle 2 zu sehen ist, erhält man eine Genauigkeit von etwa 82.739% für einen IoU von 0,5, was nicht so schlecht ist. Sie kann sicherlich erhöht werden, indem die Parameter ein wenig angepasst und die Anzahl der Iterationen erhöht werden. jedoch soll ein ausgiebiges Training im Auge behalten, da das Modell möglicherweise den Trainingssatz überpasst.
Um eine Vorstellung zu geben, wie nützlich das TensorBoard-Diagramm sein kann, soll die Abbildung 13 beachtet werden.

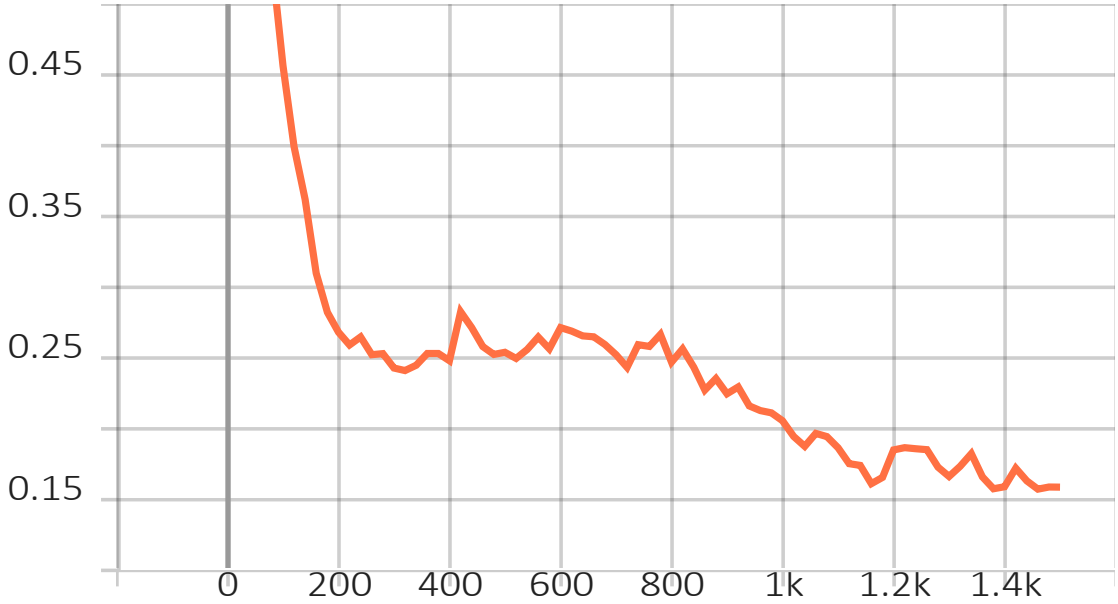

Abbildung 13: TensorBoard Graph: Verlustfunktion

Diese Metrik wird als Verlustfunktion bezeichnet. Ziel ist es, die Verlustfunktion zu minimieren. Mit anderen Worten bedeutet dies, dass das Modell weniger Fehler macht. Alle maschinellen Lernalgorithmen wiederholen viele Male die Berechnungen, bis der Verlust eine flachere Linie erreicht. Um diese Verlustfunktion zu minimieren, muss eine Lernrate definiert werden. Es ist die Geschwindigkeit, die das Modell lernen soll. Wenn die Lernrate zu hoch eingestellt ist, hat das Modell keine Zeit, etwas zu lernen. Dies ist im dargestellten Bild (siehe. Abbildung 13) nicht der Fall. Das Bild zeigt, dass der Verlust über die Iteration abnimmt, bis die Kurve abgeflacht ist, was bedeutet, dass das Modell eine Lösung gefunden hat!.
TensorBoard ist ein großartiges Tool, um solche Metriken zu visualisieren und potenzielle Probleme aufzuzeigen. Das neuronale Netzwerk kann Stunden bis Wochen brauchen, bevor es eine Lösung findet. TensorBoard aktualisiert die Metriken sehr oft. In diesem Fall muss nicht bis zum Ende gewartet werden, um zu sehen, ob das Modell richtig trainiert. TensorBoard überprüft, wie das Training verläuft, und hilft dabei, die entsprechenden Änderungen vorzunehmen.

5 Fazit

Detectron2 ist die neue Vision-Bibliothek von Facebook, die es ermöglicht, Objekterkennung, Instanzsegmentierung, Keypoint-Erkennung und panoptische Segmentierungsmodelle einfach zu verwenden und zu erstellen. Es hat ein einfaches, modulares Design, das es einfach macht, ein Skript für einen anderen Datensatz neu zu schreiben.
Es gibt in der Bibliothek von Detectron2 vieles zu entdecken. Es sind noch gute Menge an Optimierungsparametern, die weiter optimiert werden können, um eine höhere Genauigkeit zu erreichen, was vollständig von dem eigenen Datensatz abhängt.

6 Ausblick

Auf dem Gebiet der angewandten Forschung und Entwicklung aus der Logistikindustrie kann die Effizienz von intralogistischen Prozessen durch kooperative autonome Flurförderzeuge mithilfe des maschinellen Lernens Verbessert werden.
Das trainierte Modell könnte die Daten der bordeigenen Sensorik einzelner Flurförderzeugen, wie zum Beispiel Daten aus Umfeldsensoren sowie Positionsinformationen, auswerten und für die Detektion anderer Flurförderzeugen dienen. Somit trägt das Modell durch einen kooperativen Ansatz zur Automatisierung innerbetrieblicher Flurförderzeug-Flotten bei.

Literatur- und Quellenverzeichnis

[1] Yuxin Wu,Alexander Kirillov, Francisco Massa, Wan-Yen Lo, und Ross Girshick „What's in Detectron2: Generalized R-CNN Models," in: facebookresearch, 2019. [online]. Available: https://research.fb.com/wp-content/uploads/2019/12/4.-detectron2.pdf
(abgerufen am 29.03.2022)

[2] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick: Mask R-CNN, in: ICCV, 2017.

[3] Mayershofer, C., Holm, D.-M., Molter, B., Fottner, und J.: „IEEE International Conference on Machine Learning and Applications (ICMLA)," in: LOCO: Logistics Objects in Context, 2020. [online]. https://github.com/tum-fml/loco
(abgerufen am 29.03.2022)

[4] Yuxin Wu,Alexander Kirillov, Francisco Massa, Wan-Yen Lo, und Ross Girshick „Getting Started with Detectron2, and the Colab Notebook to learn about basic usage," in: facebookresearch, 2019. [online]. https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5
(abgerufen am 29.03.2022)

[5] Yuxin Wu,Alexander Kirillov, Francisco Massa, Wan-Yen Lo, und Ross Girshick „Welcome to detectron2's documentation!," in: facebookresearch, 2019. [online]. Available: https://detectron2.readthedocs.io/en/latest/
(abgerufen am 24.04.2022)

[6] Girshick Ross, Donahue Jeff, Darrell Trevor, und Malik Jitendra „Rich feature hierarchies for accurate object detection and semantic segmentation," in: Computer Vision and Pattern Recognition, 2014. [online]. Available: https://github.com/rbgirshick/rcnn
(abgerufen am 24.04.2022)

[7] Girshick Ross, „Fast R-CNN," in: International Conference on Computer Vision ({ICCV}), 2015. [online]. Available: https://arxiv.org/pdf/1504.08083.pdf
(abgerufen am 24.04.2022)

[8] „Deep learning target detection (object detection) series (3) Fast R-CNN," in: programmersought, [online]. Available: https://www.programmersought.com/article/89804159559/
(abgerufen am 24.04.2022)

[9] I. E. Poletaev, Konstantin S. Pervunin, und M. P. Tokarev, „Artificial neural network for bubbles pattern recognition on the images," in: Scientific Figure on ResearchGate, 2016. [online]. Available: https://arxiv.org/pdf/1504.08083.pdf
(abgerufen am 24.04.2022)

[10] „ AWS DeepRacer Student," in: Amazon, [online]. Available: https://aws.amazon.com/deepracer/student/
(abgerufen am 14.05.2022)

[11] Prof. Dr. Hans-Georg Stark, „KanIS: Kooperative Autonome Intralogistik Systeme," in: th-ab, 2020 [online]. Available: https://www.th-ab.de/transfer/projekte/kanis
(abgerufen am 18.07.2022)

Anlagen

Anlagenverzeichnis