Automatisierte Generierung von Trainingsdaten für YOLO

Yannick Knapp
B.Sc
Robotiklabor
Informatik, Fakultät E
Hochschule Ravensburg-Weingarten

Yannick Richter
B.Sc
Robotiklabor
Informatik, Fakultät E
Hochschule Ravensburg-Weingarten

Jonathan Fish
B.Sc
Robotiklabor
Informatik, Fakultät E
Hochschule Ravensburg-Weingarten

YOLO ist ein Objekterkennungs-Framework, das in den letzten Jahren stark an Aufmerksamkeit gewonnen hat. Grund dafür ist dessen Fähigkeit sowohl in Echtzeit zu klassifizieren als auch eine solide Erkennungsrate zu bieten. Das zugrunde liegende Lernprinzip Deep Learning benötigt jedoch eine Vielzahl an Trainingsdaten. Die Erstellung solcher gelabelten Daten ist deshalb mit erheblichem Aufwand verbunden. Durch den Einsatz einer farbbasierten Bildfreistellung in Kombination mit der Anwendung verschiedener Bildverarbeitungsfiltern soll dieser Aufwand erheblich reduziert werden.
Die Applikation ***YOLO-Train-Data-Generator*** *wurde für diesen Zweck entwickelt. In diesem Research-Paper wird die Vorgehensweise aufgezeigt, wie aus wenigen Objektbildern viele Trainingsbilder generiert werden. Für die Bestimmung der Qualität dieser Trainingsdaten findet eine Auswertung mit YOLOv2 statt.*

1 Einleitung

Der Aufschwung von künstlichen Lernverfahren geht immer weiter und hat bereits viele Bereiche revolutioniert, vor allem im Bereich der Bild-, Sequenz- und Spracherkennung [1]. Dabei fällt immer wieder das Schlagwort *Deep-Learning*.
Deep-Learning-Modelle entwickelten sich in Anlehnung an den Erkenntnissen der Neurowissenschaft. Diese beschäftigt sich mit der Funktionalität des menschlichen und tierischen Gehirns, welches aus einem neuronalen Netz besteht.
Ein künstliches neuronales Netz nutzt die Art und Weise seines biologischen Gegenstücks, um Informationen zu verarbeiten. Wie biologische Netze müssen auch künstliche vorweg durch überwachtes Lernen auf bestimmte Objekte trainiert werden. Mit dieser Technik lässt sich Objekterkennung verwirklichen, mit der bestimmte Objekte oder Muster voneinander unterschieden werden können. Zur Objekterkennung gehört die Einteilung von Objekten in Klassen (Klassifikation) und die Bestimmung ihrer Positionen innerhalb des Bildes (Detektion). Grundlegend gilt das Prinzip, dass je genauer ein Objekten beschrieben werden kann und je mehr auswertbare Informationen vorhanden sind, desto höher ist die Qualität der Objekterkennung.

Ein Objekt-Erkennungs-Framework, das eine beliebige Anzahl an Objekten in einem Bild klassifizieren und deren Position finden kann, nennt man Detektoren. Davon gibt es zwei Arten: Singleshot und 2-Stage Detektoren.
Beim 2-Stage Detektor werden zuerst die Regionen der Objekte im Bild gefunden. Anschließend wird jedes gefundene Objekt via Convolutional Neural Network (CNN) klassifiziert. Der Vorteil ist die hohe Genauigkeit der Klassifikation, welche aber im Gegenzug mit einem Ressourcen-intensiven Aufwand einher geht [2].
Singleshot Detektoren hingegen, wozu auch YOLO gehört, verwenden einen völlig anderen Ansatz. Sie benötigen lediglich einen einzelnen Schritt, um das gesamte Bild auszuwerten. Das Netzwerk unterteilt das Bild in Bereiche und sagt Begrenzungsrahmen und Wahrscheinlichkeiten für jede Region voraus. Diese Begrenzungsfelder werden mit den vorhergesagten Wahrscheinlichkeiten gewichtet [2] [3]. Auf diese Weise können Objekte in Echtzeit auch mit schlecht auflösenden Kameras erkannt werden [4], was für die Robotik von großem Interesse ist. Der Vorteil der Singleshot Detektoren ist der vergleichsweise geringe Rechenaufwand, wohingegen die Klassifikation nicht an die der 2-Stage Detektoren heranreicht.

Unabhängig davon, um welche Art von Detektor es sich handelt, ist die Qualität der Objekterkennung von der

Vielfältigkeit der Trainingsdaten abhängig. Aus diesem Grund ist für das Training eine Vielzahl an Bildern nötig. Je mehr unterschiedliche Bilder aus verschiedenen Blickwinkeln und Lichtverhältnissen verwendet werden, desto besser die resultierende Objekterkennung. Problematisch ist jedoch der benötigte Zeitaufwand, um diese Trainingsdaten zu erstellen. Erschwerend kommt hinzu, dass zu jedem einzelnen Bild eine korrespondierende Datei erstellt werden muss, welche Angaben über die Objektposition innerhalb des Bildes sowie dessen Klasse beinhaltet.

Ziel der automatischen Generierung von Trainingsdaten ist es, diese zeitaufwändige Prozedur erheblich zu reduzieren, ohne dabei an Qualität der resultierenden Objekterkennung zu verlieren.

2 YOLO

You Only Look Once (YOLO) [5] ist die Bezeichnung eines auf CNN basierenden Objekterkennung-Frameworks. Wie bereits in der Einführung erwähnt, besteht die Objekterkennungsaufgabe darin, die Region der vorhandenen Objekte innerhalb des Bildes zu bestimmen und diese zu klassifizieren.
Im Gegensatz zu früheren CNN's benutzt YOLO ein einziges neuronales Netz für die Begrenzungskoordinaten und die Klassenwahrscheinlichkeiten. Als Eingabe wird ein Bild benötigt, welches durch das neuronale Netz zu einem Ausgabevektor von Begrenzungsrahmen und Klassenvorhersagen führt.
Der erste Schritt zum Verständnis von YOLO ist die Kodierung seiner Ausgabe. Das Eingangsbild ist in ein SxS-Raster von Zellen unterteilt. Jede Rasterzelle gibt b Bounding-Boxes sowie c Klassenwahrscheinlichkeiten an. Eine Bounding Box besteht aus fünf Komponenten. Vier sind für die Positionierung und Dimensionen und eine für die Wahrscheinlichkeit, dass die Bounding-Box ein Objekt beinhaltet. Für jedes auf dem Bild vorhandene Objekt übernimmt lediglich eine Rasterzelle die Klassifierung. Dabei handelt es sich um die Rasterzelle, welche sich im Zentrum des Objekts befindet.
Sowohl YOLO als auch andere CNN's bestehen beide aus Faltungs- und Subsampling-Schichten. Der Unterschied zwischen beiden ist die letzte Schicht. Während die von CNN's aus einer eindimensionalen Klassifikation besteht, ist YOLO's letzte Schicht ein drei dimensonaler Würfel, wie in Abbildung 1 zu sehen. Das bedeutet, dass ein SxS-Raster auf das zu verarbeitende Bild gelegt wird.

Mittlerweile gibt es drei Versionen von YOLO. Die erste Version leidet an einer Reihe von Mängeln in Bezug auf moderne Objekterkennungssysteme. Tests haben ergeben, dass YOLO im Vergleich zu Fast R-CNN eine erhebliche Anzahl an Lokalisierungsfehlern enthält. Zudem hat YOLO eine relativ geringe Genauigkeit im Vergleich zu Region Proposal Networks (RPN). Daher konzentrierte sich die Weiterentwicklung hauptsächlich auf die Verbesserung der Genauigkeit und der Lokalisierung unter Beibehaltung der Klassifikationsqualität [3]. Daraus entstand YOLOv2.
Mit einer 30-Layer-Architektur hatte YOLOv2 oft mit der Erkennung kleiner Objekte zu kämpfen. Dies ist auf den Verlust feinkörniger Strukturen zurückzuführen, da die Ebenen die Auflösung der Eingabebilder herunterrechnen. Darüber hinaus fehlte der Architektur von YOLOv2 noch immer einige der wichtigsten Elemente, die heute in den meisten modernen Algorithmen verwendet werden: Sie besitzt weder Restblöcke, Skip-Verbindungen, noch Upsampling. YOLOv3 beinhaltet all diese Funktionen, jedoch büßt es gegebenüber YOLOv2 einiges an Geschwindigkeit ein [6].

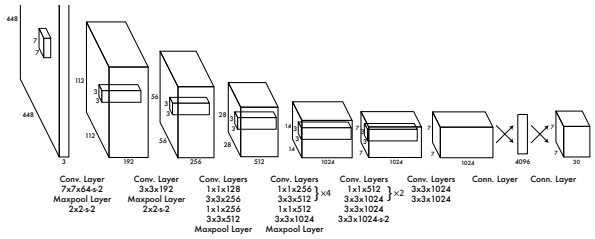

Fig. 1. CNN des YOLO-Frameworks [5]

3 Erzeugen von Trainingsdaten

Ziel der automatischen Generierung Da beim Deep-Learning eine Unmenge an gelabelten Trainingsbildern benötigt wird, ist dies auch einer der aufwändigsten Teile des gesamten Lernprozesses. Für jedes zu erlernendende Objekt muss dazu eine Vielzahl an möglichst unterschiedlichen Bildern geschossen werden. Anschließend müssen diese Bilder gelabelt werden. Für das Training mit YOLO sind dazu alle Objekte in den Bildern mit Bounding Boxes einzurahmen und sie jeweils einer Klasse zuzuweisen.
Dieser zeitaufwändige Prozess soll nun optimiert werden. Ziel ist es hierbei, die Anzahl an Fotografien zu reduzieren und aus dieser kleineren Menge an Bildern eine weitaus größere zu generieren. Ein solches Aufplustern von Trainingsdaten soll durch die Kombination verschiedener Vorgehensweisen erreicht werden: Als erstes werden die fotografierten Objekte mit verschiedenen Hintergründen kombiniert. Durch den Einsatz affiner Transformationen wie Translation, Skalierung und Rotation werden die Bilder weiter verändert. Zusätzlich erzeugt die Anwendung von diversen Bildverarbeitungsfiltern eine weitere große Varianz unter den Daten.

Freistellen von Objekten Sollen Objekte mit Hintergründen kombiniert werden, müssen sie freigestellt werden. Um den Aufwand gering zu halten, soll die Freistellung automatisiert von Statten gehen. Damit Objekte selbständig vom Hintergrund unterschieden werden können, müssen diese vor einem Greenscreen platziert und fotografiert werden. Für ein gutes und zuverlässiges

Ergebnis ist die Beleuchtung der Szene sehr wichtig. Alle Trainingsfotos werden deshalb im Foto-Labor mit einer 5-Punkt-Beleuchtung geschossen (Abbildung 2). Das zu fotografierende Trainings-Objekt steht im Vordergrund auf einem grünen Sockel, der Greenscreen 3 Meter dahinter entfernt. Dieses Setup sorgt dafür, dass Reflexionen von der Leinwand auf das Objekt minimiert werden. Um möglichst wenig Schattenbildung auf dem Objekt zu erzeugen, wird es sowohl rechts und links als auch von oben von jeweils einem Scheinwerfer angestrahlt. Dies bewirken auch zwei auf die Leinwand gerichtete Strahler. Um einen direkten Lichteinfall zu verhindern, werden auf alle Scheinwerfern Softboxen befestigt. Diese sorgen für eine gleichmäßige Ausleuchtung und ein weicheres Licht und vermindern ebenso eine Schattenbildung.

Die automatische Freistellung und anschließende Platzierung des Objekts auf einen Hintergrund besitzt noch einen weiteren Vorteil: Da die genaue Position des Trainings-Objekts auf dem Hintergrund bekannt ist, kann das Setzen von Bounding-Boxen ebenfalls leicht automatisiert werden.

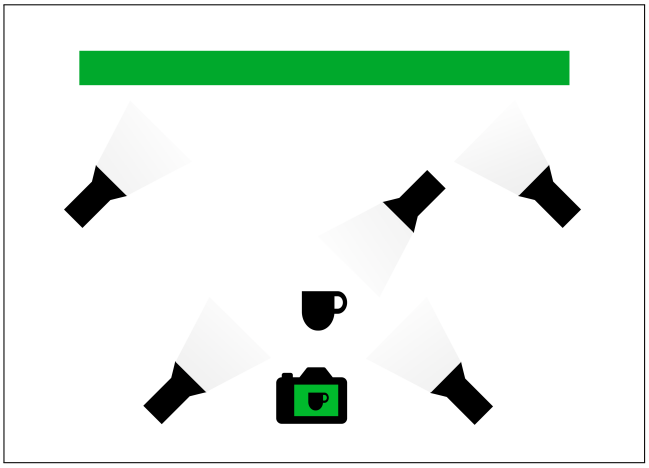

Fig. 2. 5-Punkt-Beleuchtung

automatisch beim Erzeugen von Trainingsdaten verwendet. Das Programm ist auch ein hilfreiches Werkzeug, um den optimalen Licht-Aufbau beim Fotografieren eines Trainings-Objekts zu finden.

YOLO-Train-Data-Generator Für das Generieren von YOLO-spezifischen Trainingsdaten wurde die YOLO-Train-Data-Generator Anwendung entwickelt. Über diese Applikation lassen sich genaue Einstellungen darüber bestimmen, in welcher Weise Eingangsbilder manipuliert und damit Ausgangsbilder generiert werden sollen. Um eine möglichst übersichtliche Bedienung über die große Anzahl an Einstellungsmöglichkeiten zu gewährleisten, besitzt das Programm eine grafische Bedienoberfläche (siehe Abbildung 3).
Die Anwendung umfasst folgende Features:

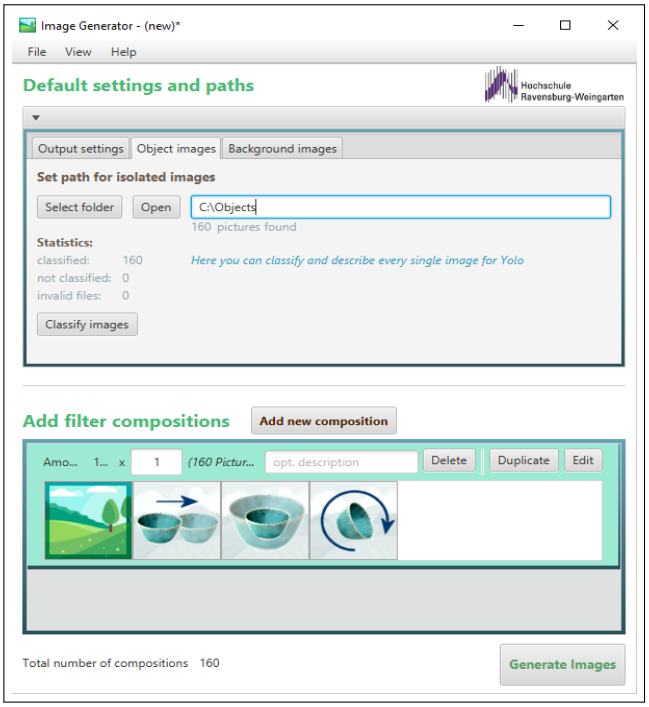

Fig. 3. Grafische Bedienoberfläche der Applikation YOLO-Train-Data-Generator

4 Werkzeuge

HSV-Finder Um ein Objekt vom Greenscreen-Hintergrund zu trennen, muss das Objekt-Bild vom RGB- in den HSV-Farbraum konvertiert werden. Durch das Setzen von bestimmten Bereichen von Farbwerten (Hue), Farbsättigungen (Saturation) und Hellwerten (Value) lassen sich alle grünen Pixel im Bild maskieren und können so weggeschnitten werden. Diese Bereiche hängen jedoch stark von den Fotografien ab. Um eine optimale Maskierung zu erhalten, ist die Verwendung einer speziell dafür entwickelten Anwendung (HSV-Finder) notwendig. Diese zeigt die direkten Auswirkungen von gesetzten HSV-Bereichen und die daraus resultierende Bounding-Box um das Objekt in Echtzeit an. Die hier gesetzten Werte werden später

- Zuweisen von Objekten zu Objektklassen
- Freistellung der Greenscreen-Objekt-Bilder über gesetzte HSV-Werte
- Anwendung von affinen Transformationen auf Objekte: Translation, Skalierung, Rotation
- Anwendung von Bildverarbeitungs-Filtern: Farbwert, Farbsättigung, Hellwert, Helligkeit, Kontrast, Gamma, Verwischen
- Hinzufügen von Objekten weiterer Objektklassen
- Kombinieren des Objekts mit einem Hintergrundbild
- Generierung einer Bounding-Box um das Objekt
- Generieren von YOLO-spezifischen Konfigurations-Dateien, die für das spätere Training von Nöten sind

Eine stark vereinfachte Darstellung des Programmablaufs wird in Abbildung 4 dargestellt. Möchte man Trainings-Daten für eine bestimmte Objektklasse generieren lassen, werden Greenscreen-Objekt-Bilder sowie Hintergrund-Bilder benötigt. Als Ausgabe entsteht ein Mehrfaches der Eingangsbilder und korrespondierende YOLO-Konfigurations-Dateien.

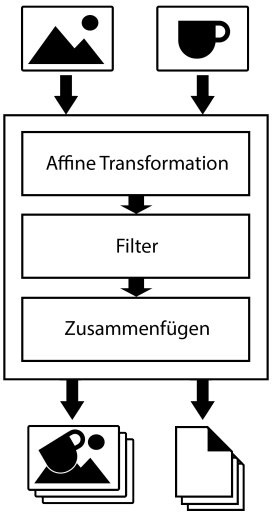

Fig. 4. Vereinfachter Programmablauf des YOLO-Train-Data-Generators

5 Ergebnisse

Um herauszufinden wie gut die automatisierte Generierung von Trainingsdaten wirklich funktioniert, sind Messungen verschiedener Art notwendig. Um die Objekterkennung in der Realität möglichst gut abbilden zu können, wird die Evaluation in voneinander unabhängige Axiome unterteilt. Diese decken die Anzahl an Trainingsiterationen, die Anzahl an verschiedenen Objekten derselben Klasse, verschiedene Blickwinkel, diverse Filter und unterschiedliche Objektklassen ab.
Die Ergebnisse zu den einzelnen Axiomen sind in den Schaubildern 5 bis 9 illustriert. Alle Abbildungen messen die Güte der Objekterkennung anhand der mAP. Die genauen Messwerte können aus dem Anhang entnommen werden.

Anzahl an Trainingsiterationen Abbildung 5 zeigt, wie sich die Anzahl der Trainingsiterationen auf die mAP auswirkt. Wie zu sehen ist, erhöht sich vor allem innerhalb der ersten 3000 Iterationen die Rate signifikant. Ab circa 7000 Iterationen stabilisieren sich die Werte aller Kurven. Hier lässt sich feststellen, dass eine höhere Anzahl an Iterationen nicht zwangsläufig zu besseren Erkennungsraten führt.

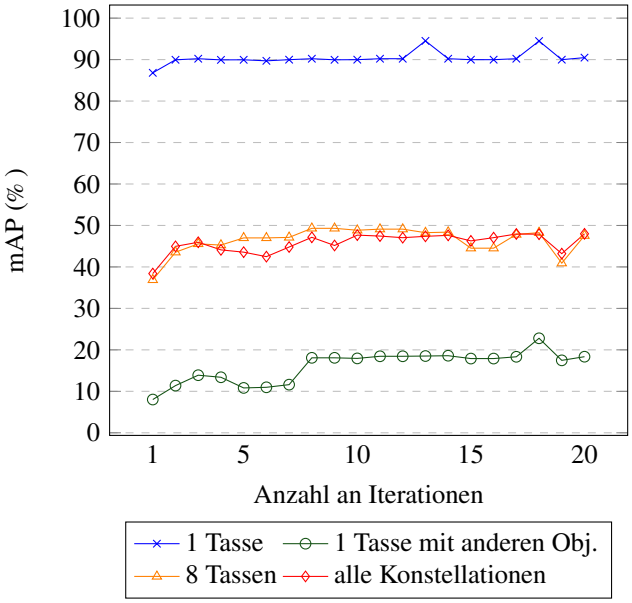

Fig. 5. Anzahl an Trainingsiterationen (x1000

Anzahl an verschiedenen Objekten der selben Klasse Abbildung 6 zeigt, wie sich die Anzahl verschiedener Tassen auf die mAP auswirkt. Hier gibt es auffällige Unterschiede zwischen den drei Konstellationen. Während die Messungen bezüglich einer Tasse beinahe konstant bleiben, verbessert sich die Erkennungsrate bei 8 Tassen mit zunehmender Anzahl an Tassen beim Training. Die Verbesserung ist darauf zurückzuführen, dass eine größere Varianz an Tassen beim Trainieren zu einem robusteren Ergebnis bei der Erkennung mehrerer unterschiedlicher Tassen führt. Die grüne Kurve hingegen weist eine Verringerung der Erkennungsrate auf. Diese Entwicklung ist gegenwärtig nicht erklärbar.

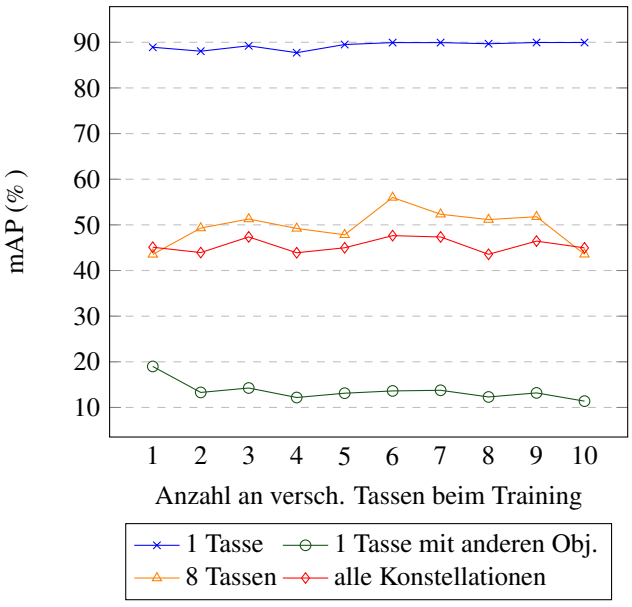

Fig. 6. Anzahl an Eingangsbildern einer Objektklasse

Verschiedene Blickwinkel Abbildung 7 zeigt, wie sich verschiedene Blickwinkel beim Training von Tassen auf die mAP auswirkt. Bei den Messungen 1 Tasse und 8 Tassen lassen sich keine signifikanten Unterschiede ausmachen. Bei 1 Tasse mit anderen Objekten jedoch liefern 22,5° und 45° sowie deren Kombination die besten Ergebnisse. Die Ergebnisse spiegeln wider, dass die genutzten Testbilder tendentiell aus dem Winkelbereich einer natürlichen Haltung geschossen wurden.

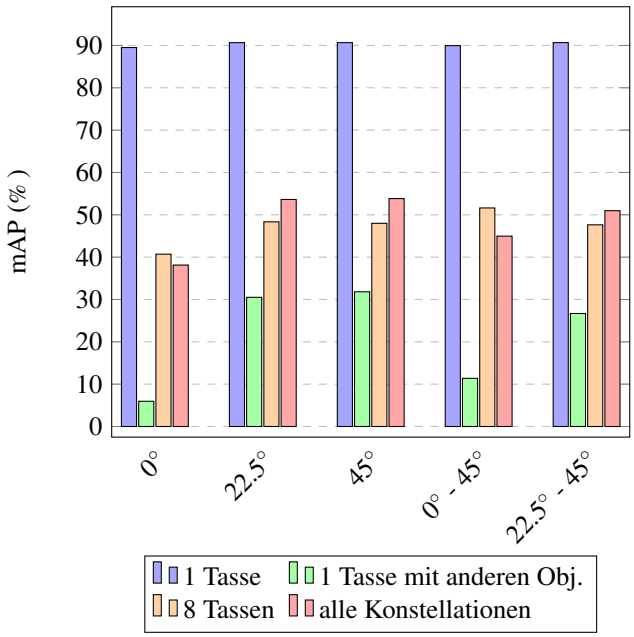

Fig. 7. Verschiedene Blickwinkel

Verschiedene Filter Abbildung 8 zeigt, wie sich die Anwendung verschiedener Filter auf die mAP auswirkt. Die Filter Translation und Skalierung stechen durchweg positiv heraus. Der einfarbige Hintergrund und die Rotation fallen hingegen nicht gut aus. Dies ist nicht verwunderlich, da der einbarbige Hintergrund wenig Varianz bietet. Die Objekte auf den Testbildern haben wenn überhaupt nur eine geringe Neigung, daher ist eine Rotation von 0° bis 360° nur teilweise zutreffend.
Vergleicht man die Ergebnisse ohne Filter mit denen mit überlappendem Objekt, lässt sich besonders im Bereich der grünen Messung eine signifikante Steigerung feststellen.

Verschiedene Objektklassen Abbildung 9 zeigt, wie sich unterschiedlich komplexe Objekte auf die mAP auswirkt. Aus den Ergebnissen lässt sich die allgemeine Tendenz ableiten, dass die Komplexität der Form und Textur eines Objekts mit der Erkennungsrate korreliert. Die schlechten Ergebnisse der Gabel sind auf mehrere Gründe zurückzuführen. Die einfache Oberflächenstruktur (einfarbig) bietet nicht genügend Merkmale für eine Wiedererkennung. Zudem ist es schwierig mit dem Greenscreen-Verfahren (siehe Erzeugen von Trainingsdaten) ein stark reflektierendes Objekt einzufangen. Ein weiterer Grund könnte die Schwäche von YOLOv2 bezüglich der Erkennung kleiner Objekte sein.

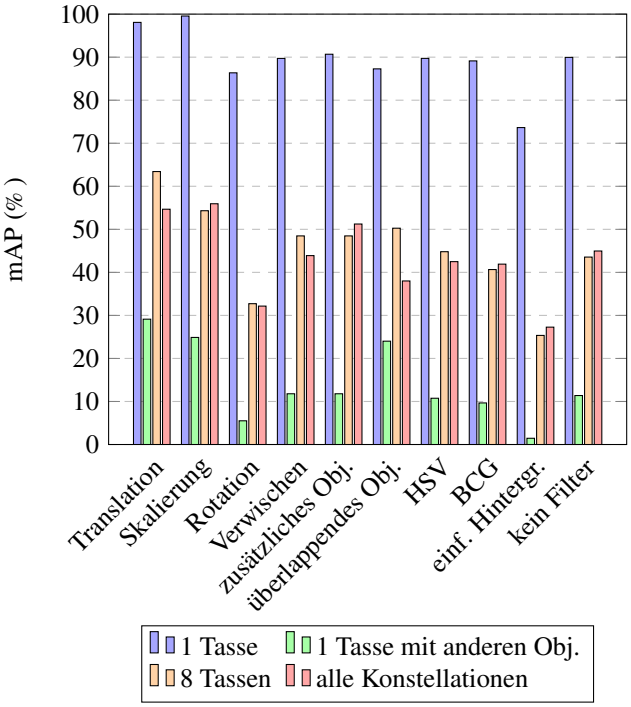

Fig. 8. Verschiedene Filter

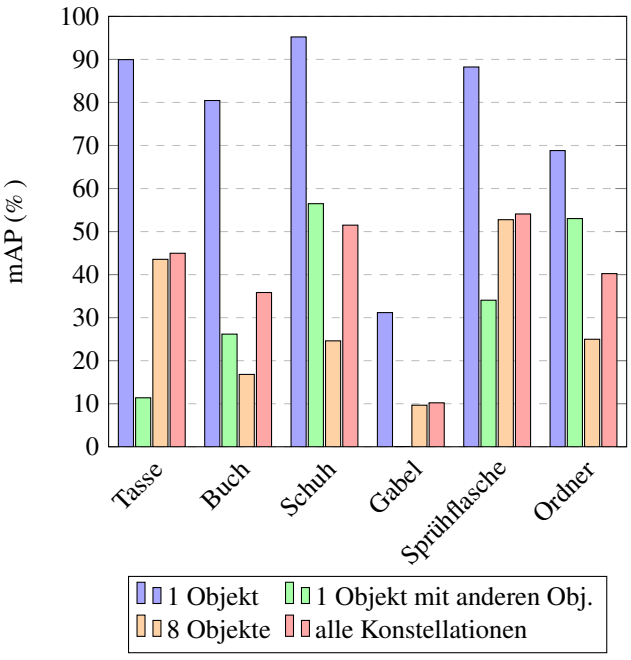

Fig. 9. Verschiedene Objektklassen

6 Evaluation

Um die generierten Bilder korrekt auswerten zu können, müssen einige Grundsätze geklärt werden.

YOLO Zuerst ist die Wahl der YOLO Version von Bedeutung. Wie in 2 bereits beschrieben, gibt es drei Versionen. YOLOv2 basiert auf dem Deep-Learning Netz *Darknet-19*, ein neuronales Netz mit 19 Schichten. YOLOv3 basiert hingegen auf *Darknet-53*, einer tieferen Struktur mit 53 Schichten. Weiterhin besitzt v3 für die Erkennung weitere 53 Schichten, die auf die ursprünglichen gestapelt werden, was zu einer Grundstruktur mit 106 Schichten führt. Somit gilt v3 zwar als bedeutend leistungsfähiger bezüglich kleineren Objekten, jedoch wurde die Steigerung der Genauigkeit für die Geschwindigkeit eingetauscht. Dies ist auch der Grund für die Langsamkeit von v3 im Gegensatz zu v2. Da die Größe der zu lernenden Trainings-Objekte relativ groß sind, wurde zu Gunsten des Zeitaufwands YOLOv2 eingesetzt.

Für die Steigerung der Lernfähigkeit benötigt YOLO ein Start-Gewicht. Dieses darf im Vorhinein nicht auf die Trainings-Objekte trainiert worden sein, da die Ergebnisse ansonsten stark verfälscht und daher nicht repräsentativ wären. Aufgrund dessen wurde das Gewicht der YOLOv2 Pascal-VOC Datensätze gewählt. Dieses Gewicht enthält 20 weitere Klassen, jedoch überschneidet sich keines dieser mit den hier trainierten Objekten.
Zum Trainieren wie auch zum Testen ist es notwendig die Auflösung des neuronalen Netzes anzugeben. Dabei ist die Auflösung mit der das Start-Gewicht trainiert worden ist von entscheidender Bedeutung. Sie sollte für das weitere Training beibehalten werden, um möglichst optimale Ergebnisse zu erreichen. Für das Testen kann hingegen eine beliebige Auflösung gewählt werden. Eine höhere Auflösung als die auf der trainiert wurde kann zu einer höheren Detektion führen, jedoch auch zu mehr falsch-positiven.

Trainingsbilder Um reale Einsatzbedingungen zu simulieren, umfasst die Evaluation drei verschiedene Konstellationen, wie Objekte in einem Bild auftreten können:

1. Ein Objekt eines Typs pro Bild (40 Bilder pro Klasse)
2. Sechs Objekte unterschiedlicher Typen pro Bild (40 Bilder pro Klasse)
3. Acht Objekte desselben Typs pro Bild (5 Bilder pro Klasse)

Die Anzahl an zu detektierenden Objekte ist in jeder der drei Konstellationen dieselbe. Dies führt dazu, dass der Einfluss aller Konstellationen auf das Gesamtergebnis gleich groß ist. Des Weiteren wurden alle Messungen, falls nicht ausdrücklich angegeben, mit 8 unterschiedlichen Tassen, 2000 Trainingsiterationen, den Winkeln 0°, 22,5°, 45° und echten Hintergrundbildern durchgeführt.
Die Auflösung der Trainingsbilder ist von der Auflösung der Netze gänzlich unabhängig. Die für die Evaluation gewählte Auflösung (1920x1080) richtetete sich ausschließlich nach dem zu Grunde liegenden Verwendungszweck, der daraus besteht, mit Hilfe einer Microsoft Kinect2 Objekte klassifizieren zu können.
Das Labeln von Test-Daten wurde mit Hilfe YOLO-Mark [7] durchgeführt, einer Applikation die für die Markierung und Klassifizierung von Objekten speziell für YOLO entwickelt wurde.

Mean Average Precision Die Mean Average Precision (mAP) ist das Maß, mit dem die Messungen bewertet wurden. Es setzt sich aus der Präzision und der Genauigkeit zusammen. Die Präzision wiederum gibt an, wie viele der erkannten Objekte tatsächlich korrekt klassifiziert worden sind. Die Genauigkeit gibt an, wie viele der im Bild vorhandenen Objekte erkannt wurden.

7 Zusammenfassung

Die Messungen zeigen, dass die automatisierte Generierung von Trainingsdaten im Allgemeinen zu einer erfolgreichen Objekterkennung mit YOLOv2 führt. Die Erkennungsrate lässt sich durch eine höhere Anzahl an Trainingsiterationen sowie einer höheren Anzahl an verschiedenen Objekten derselben Klasse steigern. Der Blickwinkel spielt dafür ebenfalls eine entscheidende Rolle. Je ähnlicher die Winkel der Trainingsbilder zu denen der klassifizierenden Bilder sind, desto besser sind die Resultate.
Beim Einsatz von Bildverarbeitungsfiltern lässt sich ebenfalls vermuten, dass sich diejenigen Filter positiv auswirken, welche die Trainingsdaten möglichst gut auf die Klassifizierungsdaten abbildet.
Die Güte der Klassifizierung hängt auch stark von der Art des Objektes selbst ab. Form, Oberflächenbeschaffenheit und die Stärke der Reflexion sind entscheidende Faktoren.

References

[1] Heinz, S., 2017. Deep Learning Teil1: Einführung. Website. https://www.statworx.com/de/blog/deep-learning-teil-1-einfuehrung.

[2] Zimmermann, B., 2018. "Individualisierte Gesichtsdetektion auf Embedded Systemen". *Paper*.

[3] Redmon, J., and Farhadi, A., 2016. "Yolo9000: Better, faster, stronger". *arXiv preprint arXiv:1612.08242*.

[4] Redmon, J., and Farhadi, A., 2018. "Yolov3: An incremental improvement". *arXiv*.

[5] Redmon, Joseph aan Divvala, S., Girshick, R., and Farhadi, A., 2014. "You only look once: Unified, real-time object detection". *arXiv*.

[6] Kathuria, A., 2018. What's new in yolo v3? Website. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b.

[7] Bezlepkin, A., 2018. Yolo mark. GitHub. https://github.com/AlexeyAB/Yolo_mark.

A Appendix

Test	#Detektionen	#Objekte	Präzision	Genauigkeit	F1	TP	FP	FN	ø IoU	**mAP**
1	6545	120	0,5	0,37	0,42	58	59	99	35,8	38,4
2	1196	120	0,52	0,5	0,51	78	73	79	36,61	44,96
3	902	120	0,55	0,45	0,5	71	58	86	43,84	45,95
4	883	120	0,52	0,44	0,48	69	63	88	40,82	44,11
5	731	120	0,54	0,47	0,5	74	63	83	43,09	43,56
6	731	120	0,52	0,43	0,47	67	63	90	40,62	42,48
7	665	120	0,55	0,48	0,51	75	62	82	43,36	44,80
8	611	120	0,56	0,49	0,52	77	60	80	44,65	47,14
9	632	120	0,54	0,48	0,51	75	65	82	42,75	45,17
10	594	120	0,57	0,5	0,54	79	59	78	46,13	47,67
11	540	120	0,59	0,48	0,53	75	52	82	46,76	47,44
12	577	120	0,57	0,5	0,53	78	60	79	44,94	47,05
13	553	120	0,56	0,5	0,53	78	62	79	44,41	47,38
14	521	120	0,58	0,51	0,54	80	57	77	46,1	47,61
15	516	120	0,55	0,5	0,5	78	64	79	43,95	46,28
16	512	120	0,57	0,49	0,53	77	59	80	45,35	47,06
17	489	120	0,59	0,52	0,55	82	58	75	46,79	47,94
18	485	120	0,59	0,51	0,55	80	56	77	46,84	47,89
19	490	120	0,56	0,46	0,51	73	57	84	44,75	43,20
20	486	120	0,58	0,51	0,52	81	58	76	45,89	47,23

Test	#Detektionen	#Objekte	Präzision	Genauigkeit	F1	TP	FP	FN	ø IoU	**mAP**
1	1288	120	0,64	0,45	0,53	70	39	87	46,79	45,1
2	1099	120	0,59	0,45	0,51	71	49	86	41,87	43,97
3	1170	120	0,57	0,52	0,54	81	62	76	40,45	47,36
4	1352	120	0,56	0,43	0,48	67	53	90	41,73	43,89
5	1222	120	0,57	0,5	0,53	78	60	79	42,33	44,98
6	1274	120	0,53	0,52	0,53	82	73	75	39,28	47,64
7	1308	120	0,52	0,51	0,51	80	75	77	35,37	47,35
8	1491	120	0,49	0,50	0,5	78	80	79	35,56	43,55
9	1303	120	0,5	0,52	0,51	81	81	76	34,16	46,44
10	1196	120	0,52	0,5	0,51	78	73	79	36,61	44,96

Test	#Detektionen	#Objekte	Präzision	Genauigkeit	F1	TP	FP	FN	ø IoU	**mAP**
0°	1226	120	0,48	0,38	0,45	60	66	97	36,05	38,13
22.5°	1427	120	0,67	0,47	0,55	74	36	83	51,27	53,62
45°	1546	120	0,73	0,49	0,59	77	29	80	55,38	53,83
0°-45°	1196	120	0,52	0,5	0,51	78	73	79	36,61	44,96
22.5°-45°	1308	120	0,68	0,48	0,57	76	36	81	52,09	50,97

Test	#Detektionen	#Objekte	Präzision	Genauigkeit	F1	TP	FP	FN	ø IoU	**mAP**
Translation	1583	120	0,48	0,66	0,56	103	110	54	37,43	54,67
Skalierung	2189	120	0,54	0,59	0,56	92	77	65	40,03	55,94
Rotation	1590	120	0,4	0,35	0,38	55	81	102	31,05	32,16
Verwischen	1318	120	0,52	0,48	0,50	75	68	82	40,64	43,89
zus. Obj.	1294	120	0,58	0,56	0,57	88	63	69	45,14	51,23
Überlappen	1683	120	0,51	0,39	0,44	61	58	96	38,98	38,00
HSV	1216	120	0,52	0,38	0,44	59	55	98	40,63	42,49
BCG	1782	120	0,54	0,45	0,48	70	65	87	40,31	41,91
einf. Hintergr.	2372	120	0,61	0,19	0,29	30	19	127	42,96	27,27
kein Filter	1196	120	0,52	0,5	0,51	78	73	79	36,61	44,96

Test	#Detektionen	#Objekte	Präzision	Genauigkeit	F1	TP	FP	FN	ø IoU	**mAP**
Tasse	1196	120	0,52	0,5	0,51	78	73	79	36,61	44,96
Buch	1607	120	0,55	0,35	0,43	56	46	103	41,12	35,84
Schuh	1446	120	0,71	0,47	0,57	75	31	84	53,72	51,49
Gabel	2300	120	0,63	0,06	0,11	10	6	148	39,73	10,21
Sprühflasche	1647	120	0,63	0,58	0,61	93	54	67	43,44	54,09
Ordner	1607	120	0,56	0,35	0,43	55	44	103	41,46	40,23