



Digitalisierung der Imkerei durch den Einsatz selbstlernender Algorithmen

STUDIENARBEIT

für die Prüfung zum

Bachelor of Science

des Studienganges Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

David Huh, Elisabeth Kletsko, Patrick Müller

Bearbeitungszeitraum

30 Wochen

Abgabedatum

16.05.2022

Matrikelnummern

9062577, 2772676, 1916243

Kurs

TINF19B4

Gutachter*in der Studienakademie

Herr Daniel Lindner

Erklärung

Wir versichern hiermit, dass wir unsere Studienarbeit mit dem Thema: „Digitalisierung der Imkerei durch den Einsatz selbstlernender Algorithmen“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort

Datum

Unterschrift – David Huh

Unterschrift – Elisabeth Kletsko

Unterschrift – Patrick Müller

Zusammenfassung

Diese Studienarbeit analysiert, inwiefern sich verschiedene Faktoren der Bildqualität auf die Erkennungsqualität eines auf YOLOv5 basierenden Objekterkennungs- und trackingalgorithmus für Bienen auswirkt. Um diese Frage zu beantworten, wurde Videomaterial aus Twitch-Streams verwendet. Für dieses wurde bei 12 Aspekten der Bildqualität untersucht, ob sich eine Änderung positiv oder negativ auf die Objekterkennung auswirkt.

Die Ergebnisse zeigen, dass sich vor allem ein höherer Detailgrad positiv auswirkt. Faktoren wie Übersättigung und starkes Rauschen der Bilder verschlechtern dabei die Erkennungsqualität erheblich. Hervorzuheben ist, dass eine Komprimierung der Bilder verhältnismäßig schwache Auswirkungen zeigt.

Die gewonnenen Erkenntnisse dienten als Basis für den daraufhin entwickelten Algorithmus zur Auswertung der An- und Abflugrate von Bienen. Diese können als Grundlage für weitergehende Arbeiten auf diesem Gebiet verwendet werden. Aus den Ergebnissen der Studienarbeit lässt sich folgern, dass mit modernen Technologien selbst bei komprimiertem Videomaterial in geringer Auflösung informative Auswertungen für Freizeitimker*innen erstellt werden können.

Abstract

This research project analyses how different factors of image quality impact the detection quality of a YOLOv5-based object detection and -tracking algorithm for honey bees. Videomaterial from twitch streams has been used to answer this question. 12 aspects of image-quality have been changed for the video-material and the impact on object detection has been examined.

The results show that especially a higher detail rate results in better detection. Factors like supersaturation and high image noise strongly reduce the detection quality. It is worth highlighting that a strong compression of the images does not have an especially strong impact.

The findings obtained were used as a basis for the development of an algorithm that can be used to analyse the arrival and departure rate of honey bees. Both of these can be used as a foundation for further work in this field. From the results of the research project it can be concluded that with modern technologies, informative evaluations for leisure beekeepers can be created even with compressed video material in low resolution.

Inhaltsverzeichnis

Erklärung	ii
<hr/>	
Inhaltsverzeichnis	iv
Abbildungsverzeichnis	ix
Tabellenverzeichnis	x
Codeverzeichnis	xi
Abkürzungsverzeichnis	xii
Glossar	xiii
<hr/>	
I Einleitung	1
1 Überblick	2
2 Problemstellung	4
3 Zielsetzung	5
4 Verwandte Projekte	6
5 Methodik	8
II Grundlagen	10
6 Imkerei	11
6.1 Der Jahreszyklus eines Bienenvolkes	11
6.2 Stockkarten - Frühere Analyse von Bienenstöcken	12
6.3 Aufgabenbereich von Imker*innen	12

6.4	Bisheriger Stand der Digitalisierung in der Imkerei	13
6.5	Häufige Fehler von Imkerei-Anfänger*innen	13
6.6	Merkmale von Bienen	13
6.7	Gründe für das Bienenersterben	14
7	Künstliche Intelligenz	15
7.1	Historischer Kontext	15
7.2	Definition Künstliche Intelligenz	17
7.3	Maschinelles Lernen	17
7.4	Künstliche Neuronale Netze	23
8	Objekterkennung mit künstlicher Intelligenz	32
8.1	Objekterkennung vs. Bildklassifizierung	32
8.2	Vergleich verschiedener Algorithmen	33
8.3	Vergleich verschiedener Objekt-Tracking-Lösungen	36
8.4	Herausforderungen bei der Bildverarbeitung mittels künstlicher Intelligenz	38
8.5	Vorbereitung der Trainingsdaten	39
9	Metriken zur Evaluation	41
9.1	Metriken zur Evaluation des Algorithmus zur Bienenerkennung	41
9.2	Metriken zur Evaluation des Tracking-Algorithmus	42
9.3	Zielsetzung für die Metriken	42
III	Umsetzung	44
10	Verwendete Algorithmen	45
10.1	Funktionsweise	45
10.2	Implementierungskomplexität	50
10.3	Begründung der Entscheidung zu YOLOv5	51
10.4	Begründung der Entscheidung zu Norfair	51
11	Herausforderungen bei der Implementierung	52
11.1	Datenquellen	52
11.2	Labeling der Daten	52
11.3	Jahreszeit	53
11.4	Bessere Datenqualität durch eigenen Bienenstock	53
11.5	Rechenleistung	54
11.6	Validierung	55
12	Bildqualität	56
12.1	Auswirkungen schlechter Bildqualität auf das Training	56
12.2	Aufnahmegeräte	58
12.3	Auswirkungen von Farbräumen	59
12.4	Schärfe der Aufnahmen	61
12.5	Auswirkungen Auflösung	63
12.6	Auswirkungen Komprimierung	64
12.7	Blickwinkel	65

12.8 Lichtverhältnisse	65
12.9 Vergleich aller Aspekte der Bildqualität	67
12.10 Fazit	71
13 Videoqualität	72
13.1 Bildwiederholungsrate	72
14 Auswertung der Daten	74
14.1 Wie ist die Auswertung umgesetzt?	74
15 Visualisierung der Daten	76
15.1 Möglichkeiten zur Visualisierung	76
IV Diskussion	77
16 Zusammenfassung der Ergebnisse	78
17 Evaluation der Ergebnisse	81
17.1 Bewertung	81
18 Ausblick	85
18.1 Optimierung der Auswertung	85
18.2 Live-Analyse	85
18.3 Weitere Optimierung des Objekterkennungsmodells	85
18.4 Deployment-Möglichkeiten	86
<hr/>	
V Anhang	88
A Anhang	I
A.1 Projekt-Links	I
A.2 Bildvariationen	I
Index	VII
Literaturverzeichnis	VIII

Abbildungsverzeichnis

1.1	Anzahl der Freizeitimker*innen in Deutschland seit 1995, aus [DEUTSCHER IMKerbund e.V. 2022]	3
6.1	Stockkarte, aus [PLANTOPEDIA o. D.[b], S. 2]	12
7.1	Evolution des Interesses am Forschungsgebiet Künstliche Intelligenz (KI), aus [GARVEY 2018]	16
7.2	Darstellung der verschiedenen KI-Ausrichtungen Die Breite der Balken stellt die Ausprägung dar (vgl. [ERTEL 2016, S. 8])	17
7.3	Traditioneller Ablauf (vgl. [GÉRON 2019, Why Use Machine Learning? Figure 1-1])	18
7.4	ML Ablauf (vgl. [GÉRON 2019, Why Use Machine Learning? Figure 1-2])	18
7.5	Labeling von Bienen, Screenshot aus LabelImg mit Ausschnitt aus Twitch-Stream von [JUNG 2021]	20
7.6	Darstellung eines Regressionsproblems mit einem einem Eingabewert (x -Achse) und einem vorhergesagten Wert (y -Achse) (vgl. [GÉRON 2019, Supervised learning Figure 1-6])	20
7.7	Nicht gelabelte Tranings-Datenmenge für ein unüberwachtes Lernsystem (vgl. [GÉRON 2019, Unsupervised learning Figure 1-7])	20
7.8	Beispielhaftes Ergebnis eines Clustering-Verfahrens bei einem unüberwachten Lernsystem (vgl. [GÉRON 2019, Unsupervised learning Figure 1-8])	20
7.9	Visualisierung des Over- und Underfittings (vgl. [KOEHRSEN 2018])	22
7.10	Auswahl an Möglichkeiten von Data Augmentation, nach [AMEISEN 2019, Fig. 6-12]	23
7.11	Vereinfachte Darstellung eines Künstliche Neuronale Netze (KNN)s und einzelnen Neurons, aus [RUSSELL 2012, S. 846]	24
7.12	Sigmoid Aktivierungsfunktion (blau) und deren Ableitung (rot), aus [WANG 2019, Image 1: The sigmoid function and its derivative]	25
7.13	Vergleich der ursprünglichen Rectified Linear Unit (ReLU) und Leaky ReLU, aus [SHARMA 2017, ReLU (Rectified Linear Unit) Activation Function]	26
7.14	Darstellung einer Funktion höheren Grades und Visualisierung des damit verbundenen Problems beim Gradientenabstieg [GÉRON 2019, Training Fig. Models 4-3]	27
7.15	Visualisierung eines möglichen Gradientenabstiegverfahrens, bei welchem iterativ zum globalen Minimum hingearbeitet wird, aus [GÉRON 2019, Training Models Fig. 4-6]	27

7.16 Visualisierung der k-fachen-Kreuzvalidierung (eigene Abbildung). Die gelben Abschnitte stellen jeweils die Testdatensätze dar	28
7.17 Architektur von ConvNet, nach [KIM 2017, S. 123]	30
8.1 Vergleich der drei Arten der Bildverarbeitung, aus [KUZNETSOVA u. a. 2020, S. 2]	32
8.2 Beispielhafte Darstellung der Verwendung von Convoltional Layers in RCNN, nach [KIM 2017, S. 125]	34
8.3 Vergleich der Modelle von YOLOv5, aus [MAINDOLA 2021]	35
8.4 Vergleich von Objekterkennung und Objekttracking, aus [MEEL o. D.]	37
8.5 Vergleich von Objekterkennung und Objekttracking, verarbeitete Screenshots aus YouTube-Video von [DUNN 2021]	38
8.6 Vorbereitung von Trainingsdaten: Original, Graustufen und Kanten, aus [JUNG 2021]	39
8.7 Vorbereitung der Trainingsdaten mit YOLOv5	40
10.1 YOLOv1 Schema, aus [REDMON u. a. 2015, S. 1]	45
10.2 YOLOv1 Layers, aus [REDMON u. a. 2015, S. 3]	46
10.3 Vergleich von YOLOv2 mit anderen Algorithmen, aus [REDMON und FARHADI 2016, S. 4]	48
10.4 Vergleich von YOLOv3 mit anderen Algorithmen, aus [REDMON und FARHADI 2018, S. 4]	49
10.5 Vergleich von YOLOv4 mit anderen Algorithmen, aus [BOCHKOVSKIY, WANG und LIAO 2020, S. 1]	50
11.1 Labeling von Bienen, Screenshot aus LabelImg mit Ausschnitt aus Twitch-Stream von [JUNG 2021]	53
12.1 Vergleich der verschiedenen Trainingsdaten, Screenshot aus Twitch-Stream von [JUNG 2021] mit nachträglicher Bearbeitung	57
12.2 Auswirkungen verschiedener Farbräume auf die Verlustrate	59
12.3 Auswirkungen Kontraste auf Verlustrate	61
12.4 Auswirkungen Tiefenschärfe auf Verlustrate	62
12.5 Auswirkungen Auflösung auf Verlustrate	63
12.6 Auswirkungen Komprimierung auf Verlustrate	65
12.7 Auswirkungen Bildrauschen auf Verlustrate	67
12.8 Vergleich aller Aspekte auf Verlustrate	68
12.9 Vergleich aller Aspekte auf Präzision	69
12.10 Vergleich aller Aspekte auf Trefferquote	70
13.1 Funktionsweise von Interpolation, bearbeitete Screenshots aus Twitch-Stream von [JUNG 2021]	73
15.1 Durch KI-Modell ermittelte An-/Abflugrate (grün - Anflüge; gelb - Abflüge; blau - Differenz)	76
16.1 Verlustrate des Modells und erkannte Bienen, siehe Abschnitt 8.3.2	78
16.2 Tracking und Numerierung der Bienen, siehe Abschnitt 8.3.2	79
16.3 Visuelle Aufbereitung der Flugpfade, mit Daten aus [DUNN 2021]	80

16.4 Visuelle Darstellung der An- und Abflugraten der Bienen, siehe Abschnitt 15.1.1	80
17.1 Verlustrate des Modells	82
17.2 Präzision des Modells	82
17.3 Trefferquote des Modells	82
17.4 Mehrere Bienen werden als eine einzelne Biene erkannt, bearbeiteter Screenshot aus [JUNG 2021]	83
17.5 Analyse der Genauigkeit des Tracking (grün - Anzahl getrackter Bienen; gelb - Anzahl detekтирter Bienen; blau - Differenz)	84
A.5 Vergleich der verschiedenen Trainingsdaten, Screenshot aus Twitch-Stream von [JUNG 2021] mit nachträglicher Bearbeitung	VI

Tabellenverzeichnis

8.1 Vergleich der Objekterkennungs-Algorithmen	36
12.1 Auswirkungen Farbräume auf Verlustrate	60
12.2 Auswirkungen Kontraste auf Verlustrate	60
12.3 Auswirkungen Tiefenschärfe auf Verlustrate	63
12.4 Auswirkungen Auflösung auf Verlustrate	64
12.5 Auswirkungen Komprimierung auf Verlustrate	64
12.6 Auswirkungen Blickwinkel zum Bienenstock	65
12.7 Auswirkungen Bildrauschen auf Verlustrate	67
12.8 Vergleich der Verlustrate	68
12.9 Vergleich der Präzision	69
12.10 Vergleich der Trefferquote	70
14.1 Benötigte Spalten zur Speicherung in einer Datenbank, mit Informationen aus [ORACLE CORPORATION o. D.]	75
14.2 Benötigte Speicherkapazität für verschiedene Videolängen und 60-Frames per second (FPS), mit Informationen aus [ORACLE CORPORATION o. D.]	75

Liste der Algorithmen

7.1	Code nach traditionellem Regelwerk	18
7.2	Pseudo Code mit Machine Learning	18

Abkürzungsverzeichnis

CNN	Convolutional Neural Network	25
FNN	Feedforward Neural Network	29
ID	Identifikationsnummer	36
KI	Künstliche Intelligenz	vii
KNN	Künstliche Neuronale Netze	vii
ReLU	Rectified Linear Unit	vii
pip	Pip Installs Packages	50
ML	Machine Learning	26
SGD	Stochastic Gradient Descent	26
FPS	Frames per second	x
YOLO	You Only Look Once	35
HOG	Histogram of Oriented Gradients	33
RCNN	Region-based Convolutional Neural Network	33
RPN	Region Proposal Network	34
RFCN	Region-based Fully Convolutional Networks	34
RoI	Region of Interest	34
SSD	Single Shot Multibox Detector	36

Glossar

Bare-Metal-Hypervisor Bare-Metal-Hypervisor werden zur Virtualisierung direkt auf der Computer-Hardware verwendet ohne das eine vorherige Installation eines Betriebssystems Voraussetzung ist. 87

Dimensionsreduktion Dimensionsreduktion ist eine Gruppe unüberwachter Lernalgorithmen, welche das Ziel haben die Dimension von benötigten Features für eine Datenrepräsentation zu reduzieren, dabei dennoch die integralen Features zu behalten (vgl. [VANDERPLAS 2016]). Ein Dimensionsreduktionsalgorithmus ist bspw. die Principal Component Analyse (PCA). Es ist jedoch zu beachten, dass eine Dimensionsreduktion zwar mit einer Geschwindigkeitsverbesserung einhergeht, das Modell tendenziell schlechter performed. . 20

Feature In der Objekterkennung werden Merkmale eines Objekts als Feature bezeichnet. 30

Support Vector Machine Durch eine Support Vector Machine kann eine Menge von Objekten in Klassen einteilen, sodass ein möglichst großer Bereich um die Klassengrenzen herum frei bleibt. 33

YAML YAML wird zur Serialisierung von Daten in Form einer Auszeichnungssprache verwendet. Dateien enden auf auf .yaml/.yml. 52

Teil I

Einleitung

1. Überblick

Künstliche Intelligenz ist allgegenwärtig. 2021 erzielte sie einen globalen Marktwert von über 93.5 Milliarden USD (vgl. [GRAND VIEW RESEARCH 2022]), 2022 sollte dieser bereits auf 136.6 Milliarden USD ansteigen. Selbst in Bereichen, in denen sie nicht erwartet wird ist sie präsent – so auch in der Ökologie, genauer gesagt der Imkerei.

Beispielsweise lebensbedrohliche Herausforderungen, wie das jährliche (Wild-)Bienensterben sollen durch künstliche Intelligenz minimiert werden. Ein weniger negatives Beispiel sind Projekte, bei welchen KI eingesetzt wird, um den Honigertrag zu steigern oder das Wohlergehen von Honigbienenkolonien sicherzustellen.

Ein Beispiel für eines der ersten KI-Projekte im Bereich Imkerei ist „The World Bee Project“ (vgl. [ORACLE CORPORATION 2019], [ORACLE CORPORATION 2018]). Dabei werden Honigstöcke mit umfassenden Sensoren ausgestattet, um Messwerte aus den Sensoren durchgehend aufzunehmen und mit der künstlichen Intelligenz daraus Prognosen zu erstellen. Mögliche Auswertungen umfassen zum Beispiel, wann Bienen sich zum Schwarmflug bereit machen oder ob es Signale gibt, sobald eine Hornisse im Anflug ist.

Weitere Projekte auf diesem Gebiet (näher dargestellt in Kapitel 4) sind jedoch meist auf professionelle Imker*innen ausgerichtet und nicht auf Freizeitimker*innen. Dabei steigt die Anzahl dieser stetig (siehe Abb. 1.1).

Eine IoT-Software, welche einige Freizeitimker*innen benutzen (rund 700 in Deutschland, vgl. [HONEYPI 2021]) ist HoneyPi ([HONEYPI o. D.]) in Kombination mit einem Raspberry Pi. Das Projekt selbst bezeichnet sich als intelligente „Stockwaage“.

Bei HoneyPi sind immer noch Ausstattungsarbeiten notwendig und die Ersteinrichtung kostet ca. 150-200 € (je nachdem, wie viele Daten erhoben werden sollen).

HoneyPi ist jedoch nicht in der Lage, die An- und Abflugrate von Bienenkolonien in einem Stock zu bestimmen oder eine Vorhersage zu machen, wie viele Bienen möglicherweise zurückkommen. Eine Aggregation von Daten findet ebenfalls nicht statt, es können lediglich die Daten über eine App abgerufen und in einem Diagramm unbewertet eingesehen werden.

An diesem Punkt setzt diese Studienarbeit an. Sie soll die Frage beantworten, ob es mithilfe von künstlicher Intelligenz und einer Kamera möglich ist, eine einfache und intuitive Lösung für Freizeitimker*innen anzubieten, mit welcher die An- und Abflugrate von Bienen in Echtzeit bestimmt und die Daten automatisch gespeichert und über ein Daten-Analyse-Plattform jederzeit abgerufen können.

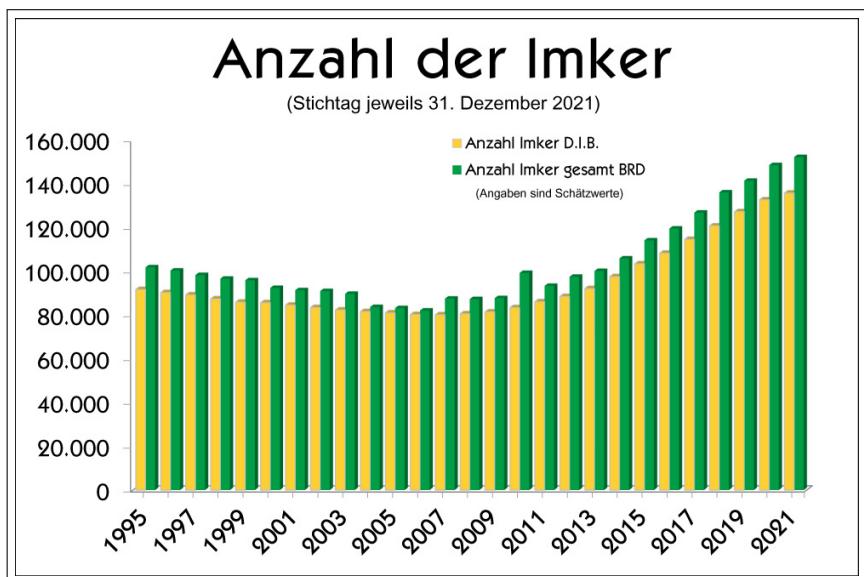


Abbildung 1.1: Anzahl der Freizeitimker*innen in Deutschland seit 1995, aus [DEUTSCHER IMKERBUND E.V. 2022]

2. Problemstellung

Die Freizeitimkerei erfreut sich seit einigen Jahren immer größerer Beliebtheit (vgl. [DEUTSCHER IMKERBUND E.V. 2022]). Darunter befinden sich auch einige technikaffine Freizeitimker*innen, die ihre Bienenstöcke mit einer Kamera aufnehmen und diese Videoaufnahmen auf Streaming-Plattformen wie beispielsweise „Twitch“ bereitstellen.

Oftmals ist es für Freizeitimker*innen allerdings schwer, einen Überblick über das Wohlbefinden und die Aktivität der Bienen zu erhalten. Historisch etablierte Mittel wie Stockkarten sind zeitaufwändig und professionelle Hardware, die diesen Job übernimmt, ist häufig zu teuer.

Dieses Problem soll in der vorliegenden Arbeit behandelt werden. Es soll eine Anwendung entwickelt werden, die mit kostengünstiger Hardware, wie einer handelsüblicher Webcam, Auswertungen über die Bienenaktivität erzeugen kann. Diese Lösung soll Freizeitimker*innen helfen, die Aktivität des Bienenstocks in Zahlen zu erfassen und ggf. Trends daraus zu schließen.

Der Fokus soll dabei auf der Fragestellung liegen: „Welche Mindestanforderungen an die Qualität des Videomaterials bestehen, um für Freizeitimker*innen wertvolle Auswertungen über einen Bienenstock erstellen zu können“.

3. Zielsetzung

Das übergeordnete Ziel dieser Studienarbeit ist die Entwicklung einer Anwendung, welche es erlaubt auf passendem Videomaterial die Bienenaktivität zu untersuchen in Form der **An- und Abflugrate**.

Das Ziel entspricht den folgenden Teilzielen:

1. Entwicklung eines Algorithmus zur Objekterkennung von Bienen
2. Entwicklung eines Algorithmus für das Tracking von Bienen und welcher erkennen, ob eine Biene sich im An- oder Abflug befindet
3. Verwendung einer Software, welche die Ergebnisse visualisieren kann

Darüber hinaus soll untersucht werden, welche Faktoren der Bildqualität (bspw. Kontrast oder Übersättigung) sich negativ auf die Genauigkeit des Algorithmus auswirken.

4. Verwandte Projekte

Das folgende Kapitel soll einen Überblick über bestehende Arbeiten und Projekte zum selben Thema geben.

apic.ai

apic.ai ist ein 2018 in Karlsruhe gegründetes Startup, das sowohl Hard- als auch Software zur Analyse von Bienen und Hummeln herstellt. Das Startup erstellt mit ihrem Produkt Auswertungen über die An- und Abflugrate, die Körpergröße der Bienen, die Menge der Pollen, die mit in den Stock gebracht werden, die Art einzelner Bienen (Arbeiter, Drone, Königin) und mehr. Dabei wird der komplette Einrichtungsprozess und die Auswertung der aufgezeichneten Rohdaten von apic übernommen. Typische Projekte kosten Interessierte zwischen 5000€ und 40000€. Die Projekte richten sich damit vor allem an größere Imker*innen oder groß angelegte Studien zur Verhaltensforschung von Bienen.

(vgl. [APIC.AI GMBH 2022])

Xailient

Die australischen Firmen Bega und Xailient haben sich zusammengeschlossen, um das „Purple Hive Project“ umzusetzen. Dabei geht es um die Erkennung von Bienen, die mit Varroa-Milben befallen sind. Diese können schnell ein ganzes Bienenvolk auslöschen, weshalb eine frühzeitige Erkennung sehr wichtig ist. Zur Erkennung werden Kameras verwendet, die am Eingang des Bienenstocks platziert sind. Xailient erkennt dabei mit einer Genauigkeit von 98% einen Befall von einer von 1000 Bienen in weniger als einer Stunde. Dank Edge Computing und Solarstrom benötigt die Anwendung dazu keine Internetverbindung, da die Daten direkt lokal ausgewertet werden. Ein Einsatz in abgelegenen Gebieten ist damit problemlos möglich. (vgl. [XAILIENT INC 2022])

metaflow-ai Hive

Hive ist ein sehr kleines Projekt eines französischen Programmierers, das Deep Learning mit TensorFlow einsetzt, um Bilder von Bienenlarven erkennen zu können. (vgl. [GIRAUD 2016])

Abgrenzung

Die im Rahmen dieser Arbeit entwickelte Anwendung unterscheidet sich in einem Punkt klar von beschriebenen Arbeiten. Sie richtet sich an Freizeitimker*innen mit stark begrenztem Budget. Die Anwendung soll also keine vollumfänglichen und perfekten Daten liefern, sondern stattdessen kostengünstig und einfach einsetzbar sein. Xailient und Hive sind zudem nur Anwendung zur Erkennung von Objekten (mit bestimmten Eigenschaften). Diese Objekte werden allerdings nicht getracked. Lediglich apic.ai führt auch ein Tracking der Bienen durch.

5. Methodik

Da sich diese Arbeit primär mit der Forschungsfrage:

„Welche Mindestanforderungen an die Qualität des *Bild- bzw. Videomaterials* bestehen, damit Bienenvideos wertvoll ausgewertet können“

wird ein *quantitativer* Forschungsansatz gewählt.

Für die Datenakquise wurden bereits vorhandene Daten eines Twitch-Streams verwendet. Diese wurden im Zeitraum August 2021 bis Oktober 2021 erhoben. Dabei entstanden über 60 GB Videomaterial.

Aus diesem Videomaterial wurden Test- und Validierungsdaten generiert.

Es wurde bewusst kein eigenes Kamera-Setup oder Bienenstock verwendet, da dies einerseits einen erhöhten Zeitaufwand bedeuten würde (da keine Erfahrung mit der Imkerei vorliegt) und andererseits ein „Unconscious Bias“ vorliegen könnte (bspw. explizites Verwenden einer besseren Kamera, um bessere Ergebnisse zu erzielen).

Für das Training des Objekterkennungsmodells wurden **Teilmengen** der Daten vorher mit der Software LabelImg [TZUTALIN 2022a] annotiert. Dies wurde von drei Personen durchgeführt, um den unterbewussten Bias zu minimieren.

Um einen passenden Algorithmus für das Trainieren des Modells zu finden, wurden verschiedenen Objekterkennungsalgorithmen und speziell YOLO-Versionen miteinander verglichen. Die Entscheidung ist aufgrund verschiedener Faktoren auf YOLOv5 gefallen.

Zur Beantwortung der Forschungsfrage wurde ein experimenteller Ansatz gewählt. Dabei sollte ermittelt werden, welche Faktoren der Bildqualität die Verlustrate des Objekterkennungsmodells maßgeblich beeinflussen (siehe Kapitel 12). Hierbei wurden für alle Modelle die gleichen Ursprungsbilder verwendet, jedoch mit unterschiedlichen Ausprägungen verschiedener Einflussfaktoren (bspw. ein zu hoher oder zu niedriger Kontrast). Die Genauigkeit und Trefferquote wurde dabei mit dem Kreuzvalidierungsverfahren gemessen. Die verschiedenen Modelle wurden jeweils mit dem Modell verglichen, das mit den Ursprungsbildern trainiert wurde. Zum Vergleich wird dabei die Verlustrate als Parameter verwendet. Betrachtet wurde dieser über die Trainingsrunden hinweg in Diagrammen. Die Ergebnisse nach 300 Epochen wurden zudem tabellarisch festgehalten.

Diese Methodik eignet sich für die vorliegende Problemstellung, da sie die realen Umstände der Anwendung repliziert. Es wird ein Ökosystem bereitgestellt, welches keine Laborbedingungen mit hochwertiger Hardware¹, sondern realistisch vorkommende Umstände² darstellt.

¹Hochwertige Hardware wäre z.B. eine digitale Spiegelreflexkamera oder eine professionelle Filmkamera wie eine Sony Alpha 7 III

²Bspw. eine Webcam für unter 50€

Teil II

Grundlagen

6. Imkerei

Das folgende Kapitel stellt ausgewählte Grundlagen der Imkerei, welche für diese Arbeit relevant sind, dar. Dies ist wichtig für das Verständnis und die Auswertung der Daten. Auch für das Trainieren der KI mit einem möglichst geringen Bias (dt. Voreingenommenheit) kann das Verständnis z.B. des Bienenjahres essentiell sein.

6.1 Der Jahreszyklus eines Bienenvolkes

Das Bienenjahr besteht aus vier verschiedenen Phasen, die stark mit den vier Jahreszeiten korrelieren. Dabei beeinflussen Witterungsumstände in unmittelbarer Nähe des Bienenstocks die Längen der einzelnen Phasen maßgeblich. Der*die jeweilige Imker*in sollte den Bienenstock stets beobachten, um etwaige Änderungen im Zyklus direkt wahrnehmen zu können.

6.1.1 August bis September

Die erste Phase beginnt zur Hochzeit des Sommers. In dieser Zeit finden bereits Vorbereitungen statt, um den Winter zu überstehen, da nur ein Bruchteil des Bienenvolkes den Winter überleben wird. Zu Beginn dieser Phase schlüpfen Winterbienen, die wiederum von den übrigen Bienen bestens für den Winter gepflegt werden. Je mehr Bienen im Stock über den Winter verharren, desto leichter fällt es, den Bienenstock aufzuwärmen. Der übrige Honig wird für den Winter gelagert.

6.1.2 Oktober bis Februar

Sobald es im Oktober zu kalt wird, verlassen die Bienen den Stock nicht mehr und rücken innerhalb des Stockes um die Bienenkönigin herum dichter zusammen. Dadurch wird das Oberhaupt des Bienenvolkes am besten vor der bevorstehenden Kälte beschützt. Fällt die Außentemperatur unter 10°C, nutzen die Bienen ihre Flügelmuskulatur, um ein Muskelzittern zu erzeugen. Dadurch kann sich der Bienenstock auf bis zu 30°C erwärmen. Wenn die Temperatur wieder über 10°C beträgt fliegen manche Arbeiterinnen aus, um ihren Darm zu entleeren. Gleichzeitig werden die Ausflüge zu Erkundungszwecken ausgenutzt.

6.1.3 März bis April

Die speziell für den Winter herangewachsenen Winterbienen sterben Anfang März und werden durch neue Sommerbienen ausgetauscht. Hauptziel für das Bienenvolk ist es, die Bienenkönigin

ausreichend mit Nahrungsmitteln versorgen zu können, damit diese möglichst viele Eier legen kann.

6.1.4 Mai bis Juli

Zum Ende des Frühlings beginnt die Entwicklung der neuen Königin und neuer Drohnen. Währenddessen werden nach neuen Trachtquellen Ausschau gehalten, da sich immer mehr Bienen innerhalb eines Stockes befinden und mehr Platz benötigen. Dabei zieht die bestehende Bienenkönigin mit einem Teil des Volkes aus dem Bienenstock aus und sucht sich mit diesem gemeinsam einen neuen Aufenthaltsort für das kommende Bienenjahr (vgl. [CAREFUL o. D.]).

6.2 Stockkarten - Frühere Analyse von Bienenstöcken

Zur Dokumentation und weiteren Analyse wurden lange Zeit Stockkarten verwendet. Auch heute noch befinden sich Stockkarten im Einsatz, jedoch in einem etwas neueren Format als Handy-App für jede*n Freizeitimker*in. Auf solch einer Stockkarte werden neben Jahr, Stand, Bienenvolk und Alter der Königin in regelmäßigen Abständen Werte zur Stärke, Schwarmtrieb, Wabensitz, Temperament und Futtermenge vermerkt. Diese Werte (ausgenommen der Futtermenge) werden in einer subjektiven Skala von 1 (schwach) bis 4 (stark) gemessen. Schließlich wird noch die Ertragsmenge am Ende des Frühlings und am Ende des Sommers notiert (vgl. [PLANTOPEDIA o. D.[a]]).

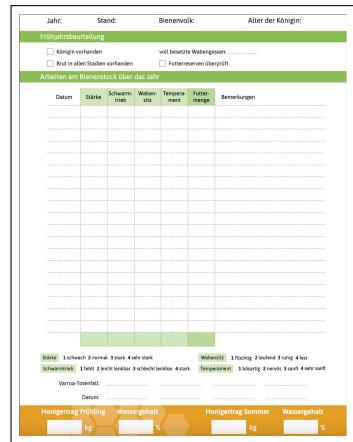


Abbildung 6.1: Stockkarte, aus [PLANTOPEDIA o. D.[b], S. 2]

6.3 Aufgabenbereich von Imker*innen

Zum Hauptaufgabenbereich von Imker*innen gehört vor allem die Züchtung der Königinnen und damit auch die einhergehende Betreuung der Bienen. Da die Bienen einen geeigneten Wohnort benötigen, wird auch nach handwerklichem Geschick gefragt, um Bienenstöcke aufzubauen. Bis auf wenige Ausnahmen ist der Arbeitsplatz einer*eines Imker*in zu großen Teilen in der freien Natur. Um jedoch überhaupt Imker*in zu werden, sollte zunächst eine meist dreijährige Ausbildung zum „Tierwirt – Fachrichtung Bienenhaltung“ absolviert werden (vgl. [BERNBURG o. D.]).

6.4 Bisheriger Stand der Digitalisierung in der Imkerei

Heutzutage besteht immer mehr auch die Chance für Freizeitimker*innen, ihre Völker maschinell durch zum Beispiel einen HoneyPi überwachen zu lassen. HoneyPi misst unterbrechungsfrei das Ertragsgewicht, die Luftfeuchte in und außerhalb des Stocks, die Lautstärke des Bienenvolks und die Temperatur innerhalb / außerhalb des Stocks.

6.5 Häufige Fehler von Imkerei-Anfänger*innen

Zu den häufigsten Fehlern von Anfänger*innen in der Imkerei zählt an erster Stelle der Wissensmangel über die Imkerei selbst. Daher ist es von besonderer Wichtigkeit, nicht nur die Grundlagen zu erlernen, sondern sich stetig in diesem Bereich weiterbilden zu lassen.

Häufig sterben Bienen auch an Nahrungsmangel, weshalb es hilfreich sein kann, diese selbst zu füttern.

Die Varroa sind eine Art von Milben, die die meisten Milben-Befälle bei Bienenstöcken ausmachen. Auch hiergegen muss ein*e Imker*in stets gewappnet sein.

Der erste erntereife Honig braucht seine Zeit. Daher ist es ratsam, die erste Honigernte im zweiten Bienenjahr zu ernten, da die Bienen dann an Reife gewonnen haben. Hierbei sollte den Bienen natürlich nicht ihre ganze Ernte entnommen werden, da sie selbst davon leben müssen und anderenfalls sterben (vgl. [MONITORING o. D.]).

6.6 Merkmale von Bienen

Allgemein werden Bienen in ihrem Leben zwischen 7 und 19 Millimeter groß. Ihr Körper ist dabei eher lang-gestreckt und moderat dicht behaart. Genauso sind ihre Komplexaugen behaart. Unterschieden werden verschiedene Bienenarten anhand ihrer Körpergröße, Flügelgeäder und Färbungsmerkmale. Innerhalb eines Bienenvolkes gibt es drei verschiedene Arten von Bienen, welche selbst auch unterschiedliche Merkmale aufweisen.

6.6.1 Drohnen

Die Drohnen besitzen keinen Stachel und haben ein schwach skelortisiertes Genital. Sie entstehen dabei durch nicht-befruchtete Eier und haben auch keinen Stachel. Ihr Lebenssinn ist die Begattung der Königin, woraufhin sie unmittelbar sterben. Drohnen machen im Bienenvolk einen Gesamtanteil von maximal 10% aus.

6.6.2 Arbeiterinnen

Die nächsten in der Rangfolge sind die Arbeiterinnen. Deren Mandibeln tragen weder Zähne noch Kiele. Dafür haben Arbeiterinnen jedoch Körbchen zur Sammlung von Pollen. Sie entstehen durch befruchtete Eier und haben wiederum einen Stachel. Ihr Anteil stellt knapp 90% des Bienenvolks dar. Arbeiterinnen üben zudem eine Vielzahl an Berufen in Abhängigkeit ihres Alters aus. Dazu zählen folgende:

- Putzerin
- Ammenbiene

- Baubiene
- Wächterbiene
- Heizerbiene
- Tankerbienen
- Belüfterin
- Zofen
- Honigbereiterin
- Kundschafterin
- Sammlerin
- Herstellerin von Winterfutter
- Reserve an Arbeitslosen, die bei unvorhergesehenen Ereignissen zum Einsatz kommen

6.6.3 Königin

Ganz oben in der Rangfolge steht die Königin. Die Königin besitzt gespaltene Klauen. Dazwischen sitzt ein Arolium als Haftpolster auf glatten Oberflächen. Die Tibien der Hinterbeine besitzen keine Sporne (vgl. [WIKIPEDIA 2021],[SIMPLYSCIENCE.CH 2013], [BEEBETTER o. D.[b]]).

6.7 Gründe für das Bienensterben

Der Lebensraum von Bienen wird nach und nach mehr zerstört. Beim Bau von Straßen beispielsweise werden große Flächen versiegelt und unbrauchbar für Bienen. Auch Monokulturen von Pflanzen, wie es auf vielen Ackerfeldern ausgeübt wird, sind schädlich für Bienen. Auf vielen Feldern werden darüber hinaus auch Pestizide eingesetzt, um Schädlinge fernzuhalten beziehungsweise zu töten. Natürlich sind auch Fressfeinde und Schädlinge für Bienen suboptimal. Aus all diesen Ursachen folgen eine negativ beeinflusste Nahrungsgrundlage, Orientierungslosigkeit und eine allgemeine Schwächung (vgl. [BEEBETTER o. D.[a]]).

7. Künstliche Intelligenz

Dieses Kapitel definiert Begriffe aus dem Gebiet der KI, welche für das Verständnis dieser Arbeit notwendig sind.

7.1 Historischer Kontext

Der Begriff KI

John McCarthy münzte den Begriff „künstliche Intelligenz“ das erste Mal auf der Konferenz in Dartmouth 1956 [MCCARTHY 1955]. Diese hatte das Ziel, sich mehr in die Thematik einzuarbeiten und Verfahren zu entwickeln, welche „[...] Maschinen dazu bringen, Sprache zu verwenden, Abstraktionen und Konzepte zu bilden, Probleme zu lösen, die derzeit dem Menschen vorbehalten sind, und sich selbst zu verbessern“ (vgl. [MCCARTHY 1955, Abschnitt 1]).

Doch schon bevor der genaue Begriff feststand, gab es Arbeiten, welche sich in den Bereich künstliche Intelligenz klassifizieren lassen. Dazu zählen das *McCulloch-Pitts Neuron* (dies war der erste Versuch, ein mathematisch Modell eines Neurons dazustellen) oder der von Alan Turing entwickelte Turing-Test, über welchen sich definieren lässt, wann eine Maschine als intelligent gilt (vgl. [RUSSELL 2012, 39f]).

KI-Winter

In den Jahren nach der Konferenz wurde mit hoher Erwartung entwickelt und geforscht. Jedoch basierten viele der verzeichneten Erfolge größtenteils auf *syntaktischen Manipulationen* oder einem *einfachen Regelwerk*. Ferner probierten die intelligenten Programme *schrittweise* Lösungen durch, bis eine richtige gefunden worden ist.

Sobald das Aufgabenfeld bzw. das Problem komplexer wurde, scheiterten diese „intelligenten“ Programme. Die Forschung unterlag der fälschlichen Annahme, dass sich Probleme mit erhöhter Komplexität durch stärkere Rechenleistung lösen ließen (vgl. [RUSSELL 2012, S. 45]).

Die andauernden Misserfolge führten zu einem Ausbleiben anfänglich bereitgestellter Fördergelder. Damit trat auch der erste „KI Winter“ (engl. AI-Winter) ein. Dieser Zeitabschnitt beschreibt das stagnierende Interesse an der KI-Forschung. Die Historie weist zwei davon auf.

Der erste KI-Winter führte zu einem Umdenken in der KI-Forschung und dem Verlagern des Fokus auf Systeme, welche sich auf einen bestimmten Bereich konzentrierten. Davor wurde versucht, Systeme zu entwickeln, welche der „breiten“ Masse dienen konnten bzw. Probleme mit einer *allgemeinen* Wissensbasis lösen konnten (vgl. [RUSSELL 2012, S. 46]).

Abb. 7.1 zeigt, dass dennoch ein zweiter KI-Winter einbrach. Viele in der Zeit entwickelten Expertensysteme konnten den anspruchsvollen Versprechen ihrer Firmen nicht standhalten (vgl. [RUSSELL 2012, S. 48]).

Andauernder Boom

Durch die Wiederentdeckung des erstmalig 1969 angesprochenen **Backpropagation**-Algorithmus (vgl. Abschnitt 7.4.4) entwickelte sich wieder ein breites Interesse an der Forschung. Der KI-Bereich etablierte sich als Wissenschaft, in dem auf bereits bestehenden Modellen aufgebaut worden ist. Darüber hinaus ermöglichen einheitliche Bibliotheken, Experimente zu replizieren und zu validieren. Im Vergleich dazu wurde in frühester Vergangenheit wahllos probiert.

In der Wissenschaft entwickelte sich zunehmend die Einstellung, „Expertensysteme [zu entwickeln], die rational gemäß der Gesetze der Entscheidungstheorie agieren und nicht versuchen, die Gedankenschritte menschlicher Experten zu imitieren.“ (vgl. [RUSSELL 2012, S. 50]). Das maschinelle Lernen (vgl. Abschnitt 7.3) gewinnt immer mehr an Bedeutung.

Durch Entwicklungen im technologischen Bereich und der hohen Verfügbarkeit sämtlicher Daten und Verbesserung der Hardware ist das Gebiet der (angewandten) KI in der Gegenwart ein hochaktuelles Thema.

Viele Unternehmen setzten Anwendungen mit KI für diverse Zwecke ein, weshalb vorerst nicht mit einem weiteren KI-Winter zu rechnen ist (vgl. [RUSSELL 2012, S. 52]).

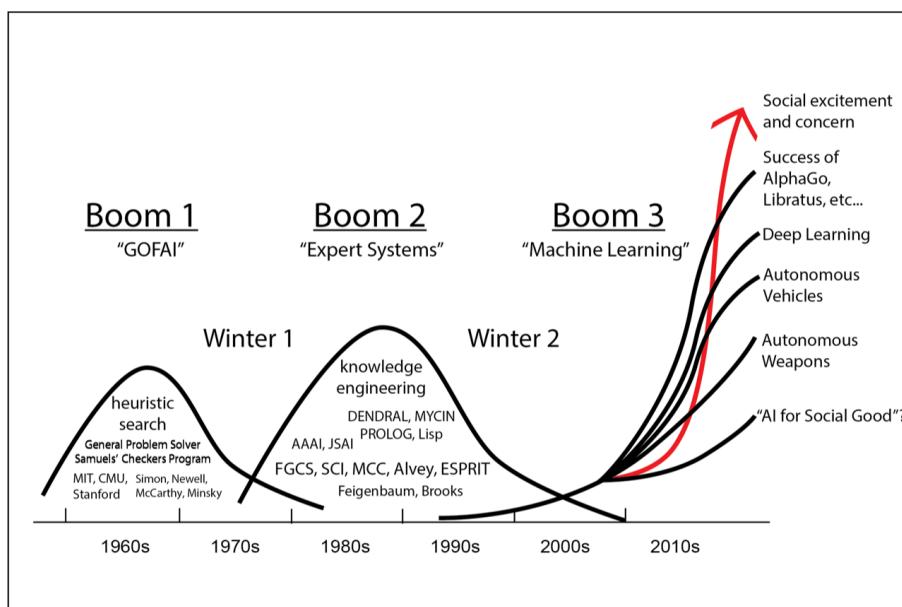


Abbildung 7.1: Evolution des Interesses am Forschungsgebiet KI, aus [GARVEY 2018]

7.2 Definition Künstliche Intelligenz

Abb. 7.2 zeigt die verschiedenen Ausprägungen der KI-Forschung über die Zeit. Neben des jungen Alters der KI-Wissenschaft ist auch die Diversität einer der Gründe, weshalb es schwierig ist, den Begriff „KI“ einheitlich für das gesamte Forschungsgebiet zu definieren (vgl. [ERTEL 2016, S. 1–3]).

Diese Arbeit wird deshalb mit der Definition von Elain Rich arbeiten, welche nicht explizit die *KI* definiert hat, jedoch das Ziel des Forschungsgebiets.

„Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.“ (vgl. [ERTEL 2016, S. 2]).

Aus der Definition von E. Rich lässt sich ableiten, dass maschinelles Lernen (vgl. Abschnitt 7.3) ein zentraler Bestandteil des Forschungsgebiet KI ist, da dieses den Ansatz verfolgt Maschinen *adaptives Lernen* näher zu bringen.

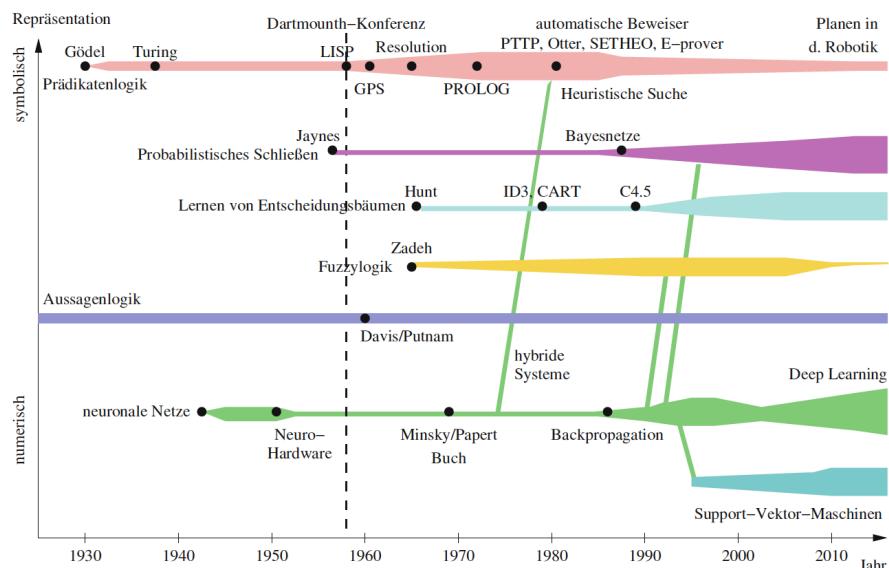


Abbildung 7.2: Darstellung der verschiedenen KI-Ausrichtungen Die Breite der Balken stellt die Ausprägung dar (vgl. [ERTEL 2016, S. 8])

7.3 Maschinelles Lernen

Tom Mitchell definiert 1997 **maschinelles Lernen** als „das Forschungsbiet von Computer-Algorithmen, welche durch Erfahrung *selbstständig* lernen und sich *verbessern*“ (vgl. [GÉRON 2019, What is Machine Learning?]).

Maschinelles Lernen findet vor allem dort Einsatz, wo traditionelle Algorithmen ihre Grenzen erreichen, beispielsweise bei einem zu umfangreichen Regelwerk für ein Problem.

Das „Auswendiglernen“ steht hierbei nicht im Fokus. Denn hier sind Computer uns mit ihrer Rechenleistung und *theoretisch* unbegrenzten Speicherplatz überlegen. Deshalb wird beim maschinellen Lernen die Form des *adaptiven* Lernens betrachtet.

Menschen können sich an neue Situationen und Umgebungen anpassen und lernen. Maschinen können dies mit dem Auswendiglernen nie erreichen, da das Regelwerk zu *komplex* werden würde. (vgl. [ERTEL 2016, S. 192]).

Für die Differenzierung beider Lernformen soll ein E-Mail Spamfilter betrachtet werden: Bei einem *traditionellen* Algorithmus müssten die Entwickler initial Regeln definieren welche Buchstaben oder Symbole eine Spam-Email charakterisieren und diese immer wieder anpassen (vgl. Listing 7.1). Ein Beispiel dafür wäre eine Regel, welche eine E-Mail mit dem Betreff „4U“ als „Spam“ kategorisiert. Würde dieses „4U“ jedoch durch ein „ForU“ ersetzt wird, filtert der Algorithmus diese E-Mail nicht heraus, da keine Regel dafür existiert. Ein Mensch hingegen würde diesen Zusammenhang eher verstehen und sich „adaptieren“ (vgl. [GÉRON 2019, Why Use Machine Learning?]).

Beim Einsatz von *maschinellem* Lernen kann ein Entwickler einmalig den Algorithmus definieren und diesen durch Parameter und Daten auf eigene Bedürfnisse anpassen.

Entscheidender Vorteil daran ist, dass dieser Prozess automatisiert werden kann und demnach weniger zeitaufwändig ist als die Implementierung eines traditionellen Algorithmus **traditionelle** Algorithmus. Abb. 7.3 und 7.4 vergleichen den „traditionellen“ Ansatz mit dem „Machine Learning“ Ansatz.

```

1 if 'Get rich with Bitcoin' in email:
2     email.isSpam = True
3 if 'Loose 10 pounds in two days' in email:
4     email.isSpam = True
5 ... Deklaration weiterer Regeln

```

Algorithmus 7.1: Code nach traditionellem Regelwerk

```

1 try to classify some emails;
2 change self to reduce errors;
3 repeat;

```

Algorithmus 7.2: Pseudo Code mit Machine Learning

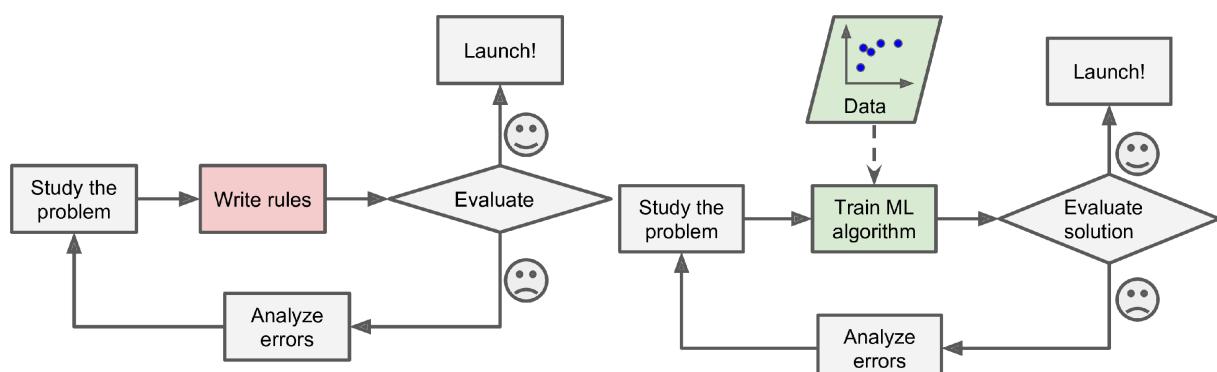


Abbildung 7.3: Traditioneller Ablauf (vgl. [GÉRON 2019, Why Use Machine Learning? Figure 1-1])

Abbildung 7.4: ML Ablauf (vgl. [GÉRON 2019, Why Use Machine Learning? Figure 1-2])

7.3.1 Lernprobleme

Maschinelles Lernen bzw. selbstlernende Algorithmen können sogenannte „Lernprobleme“ lösen. Diese lassen sich in **Such-**, **Klassifikations-** und **Regressionsprobleme** untergliedern. Zum Trainieren der jeweiligen Algorithmen werden Methoden des **unüberwachten**, **semi-überwachten**, **überwachten** oder des **bestärkenden** Lernens verwendet. Diese Methoden unterscheiden sich in der Art und dem Umfang des Feedbacks, welche der selbstlernende Algorithmus bei seinen Aktionen erhält (vgl. [RUSSELL 2012, S. 811]).

Überwachtes Lernen

Beim überwachten Lernen erhält der Algorithmus im Trainingsverfahren zu den Eingaben die gewünschten Ausgaben und soll anhand dieser Basis für neue Daten die richtigen Ausgaben vorhersagen.

Auch diese Arbeit arbeitet mit dem überwachten Lernen, da Insekten in den Ausgangsdaten mit dem Label „Biene“ gelabelt werden (vgl. Abb. 7.5). Der entwickelte Algorithmus soll dann auf neuem Bildmaterial Bienen erfolgreich erkennen können. Lernprobleme, welche Algorithmen des überwachten Lernen lösen können sind unter anderem Klassifikations- oder Regressionsprobleme.

Bei Klassifikationsproblemen soll ein Algorithmus in der Lage sein, neue Instanzen zu klassifizieren. Beispielsweise ob ein gezeigtes Insekt eine Biene oder keine Biene ist.

Regressionsprobleme widmen sich einer anderen Fragestellung, nämlich: „Wie hängt eine oder mehrere abhängige Variable(n) y von der unabhängigen Variable x ab“ (vgl. Abb. 7.6). Ein konkretes Beispiel:

„Wie hängt der Honigertrag mit der An-und Abflugrate von Bienen an einem Honigstock zusammen.“

Manche Regressionsalgorithmen, wie die logistische Regression, können auch für Klassifikationsprobleme verwendet werden können. Bei dieser kann ein Algorithmus in der Ausgabe die Wahrscheinlichkeit einer Zugehörigkeit vorhersagen.

Zu den Algorithmen des überwachten Lernens zählen: *k-Nearest Neighbors*, *lineare und logistische Regression*, *neuronale Netze*, *Support Vector Machines*, *Entscheidungsbäume* und *Random Forests* (vgl. [GÉRON 2019, Supervised Learning]).



Abbildung 7.5: Labeling von Bienen, Screenshot aus LabelImg mit Ausschnitt aus Twitch-Stream von [JUNG 2021]

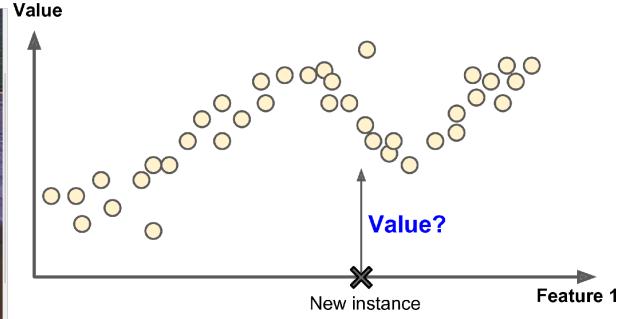


Abbildung 7.6: Darstellung eines Regressionsproblems mit einem Eingabewert (x -Achse) und einem vorhergesagten Wert (y -Achse) (vgl. [GÉRON 2019, Supervised learning Figure 1-6])

7.3.2 Unüberwachtes Lernen

Beim unüberwachten Lernen versucht der Algorithmus durch Exploration der Daten Muster und Zusammenhänge zu erkennen – ohne menschlichen Einfluss. Demnach erhält der Algorithmus nur die Eingabedaten, jedoch keine dazugehörigen Ausgaben. Dies kann nützlich sein, um Anomalien in Datensätzen zu erkennen, Cluster zu bilden, Daten zu visualisieren, Assoziationen zu bilden oder Dimensionsreduktion durchzuführen (vgl. [GÉRON 2019, Unsupervised Learning]).

Unüberwachtes Lernen eignet sich diese für die Ziele dieser Arbeit, da keine Muster oder Zusammenhänge erkannt werden müssen. Darauf hinaus existiert durch die minimale menschliche Intervention in diesem Verfahren wenig Einfluss auf die Ergebnisse [DELUA 2021, Other key differences between supervised and unsupervised learning].

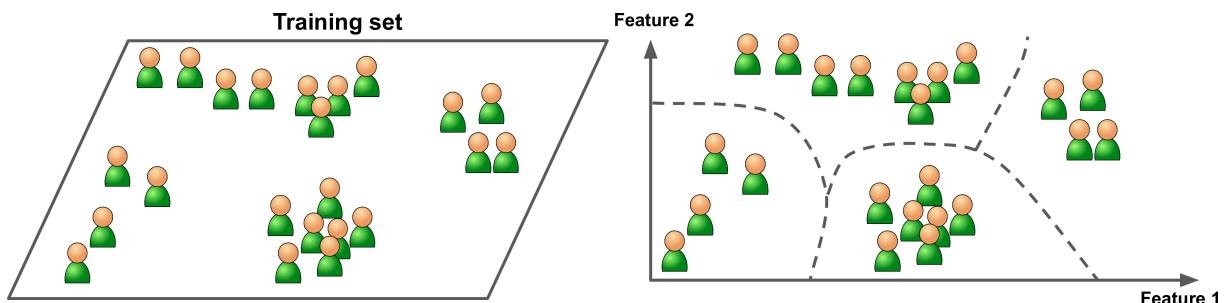


Abbildung 7.7: Nicht gelabelte Trainings-Abbildung 7.8: Beispielhaftes Ergebnis eines Datenmenge für ein unüberwachtes Lernsystems Clustering-Verfahrens bei einem unüberwachten Lernsystems (vgl. [GÉRON 2019, Unsupervised learning Figure 1-7])

7.3.3 Teil-überwachtes Lernen

Die Form des teil-überwachten Lernens ist ein Hybrid aus un- und über-wachtem Lernen. Bei dieser Form werden die Daten nur **teilweise** manuell gelabelt. Dies hat den Vorteil, dass weniger Zeit für das Labelling aufgewendet wird.

Ein Beispiel für eine Anwendung, die teil-überwachtes Lernen einsetzt, ist ein Fotoservice, bei welchem teilweise Personen mit Namen gelabelt werden und das System weitere identische Personen identifiziert.

Ein Algorithmus für das teil-überwachte Lernen sind die „deep belief networks (DBNs)“ (vgl. [GÉRON 2019, 1. The Machine Learning Landscape – Semisupervised Learning]).

Teil-überwachtes Lernen eignet sich für diese Arbeit nicht da, wie beim unüberwachten Lernen, Einbußen in der Genauigkeit des Algorithmus entstehen.

7.3.4 Bestärkendes Lernen

Ein **Agent** ist „[ein] ein System, welches Information verarbeitet und aus einer Eingabe eine Ausgabe produziert“ [ERTEL 2016, S.18].

Bestärkendes Lernen (engl. Reinforcement Learning) ist eine Form des Lernens welche sich von den drei vorherigen unterscheidet in der Hinsicht, dass es einen Agenten gibt, welcher von seiner Umwelt lernt und Aktionen durchführt für die er entweder belohnt oder bestraft wird (nach dem „Zuckerbrot und Peitsche“-Prinzip). Ein Beispiel für einen Agenten, welcher mit Reinforcement Learning gelernt hat war DeepMinds AlphaGo, welcher 2017 den gegen den damaligen Weltmeister Ke Jie im Go-Spiel gewonnen hat. Diese Form des Lernens benötigt ebenfalls viel Rechenleistung (vgl. [GÉRON 2019, 1. The Machine Learning Landscape – Reinforcement Learning]).

Over- und Underfitting

Das Phänomen des Overfittings (Überanpassung) entsteht im Lernprozess dann, wenn ein Modell zu gut auf einen bestimmten Datensatz trainiert ist (man spricht auch von einem *hohen* Bias und *niedriger* Varianz in den Trainingsdaten). Dies geschieht beispielsweise dann, wenn nur eine Kameraeinstellung und immer wieder der selbe Bildausschnitt verwendet wird. Dies tritt auf, wenn das Modell jegliche *Details* lernt, jedoch eine niedrige *Generalisierung* aufweist.

Im Kontrast dazu befindet sich das Underfitting, welches dadurch entsteht, dass das Modell nicht in der Lage ist, aus den gegebenen Daten zu lernen. Beispielsweise ist ein Modell zum Erkennen von Bienen erwünscht. Diesem werden jedoch für das Training ausschließlich verrauchte oder zu wenige Daten gegeben. Hier ist wichtig zu erwähnen, dass beim Underfitting ein *niedriger* Bias, aber eine umso höhere Varianz vorliegt. Der selbstlernende Algorithmus schneidet bei allen Daten gleichermaßen schlecht ab. Abb. 7.9 illustriert beide Phänomene.

Das Ziel in der Entwicklung bzw. beim Trainieren ist es, einen „Kompromiss“ zwischen der Varianz und Bias zu finden (auch Varianz-Verzerrungs-Dilemma genannt)

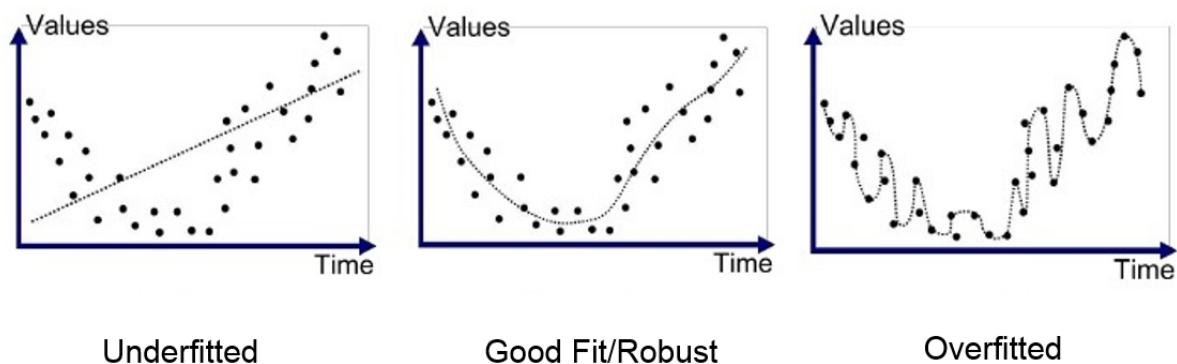


Abbildung 7.9: Visualisierung des Over- und Underfittings (vgl. [KOEHRSEN 2018])

Data Augmentation

Dem Problem des Overfittings (vgl. Abschnitt 7.3.4) kann Data Augmentation entgegenwirken. Darunter wird eine Technik verstanden, bei welcher bereits vorhandene Datensätze in kleinen Maßen alterniert werden. Das Ziel ist es hierbei, *künstlich* neue Daten zu generieren, um das trainierende Modell Variationen an Daten auszusetzen. Dies gestaltet das Training *komplexer* für das Modell.

Welche Strategien unter anderem für Data Augmentation angewandt werden, hängt vom *Typ* der vorhandenen Daten ab.

Ein Nachteil an Data Augmentation ist, dass das Modell im Trainingsdatensatz mit mehr Fehlern performt, im Testdatensatz jedoch mit weniger (vgl. [AMEISEN 2019, Kapitel Data Augmentation]).

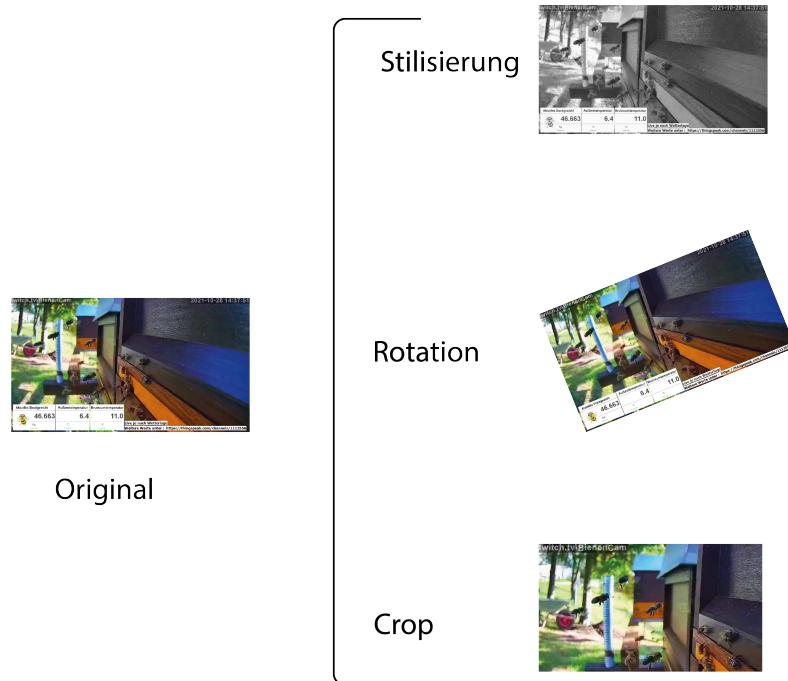


Abbildung 7.10: Auswahl an Möglichkeiten von Data Augmentation, nach [AMEISEN 2019, Fig. 6-12]

7.4 Künstliche Neuronale Netze

Die folgenden Abschnitte geben einen Überblick über künstliche neuronale Netze, da diese unter anderem in dieser Arbeit Einsatz finden. Dabei wird über den Aufbau und über den damit verbundenen Lernprozess geschrieben. Hierbei wird ein Fokus auf Konzepte gelegt, die auch der für diese Arbeit verwendete Algorithmus beinhaltet.

7.4.1 Aufbau eines KNN

Künstliche Neuronale Netze sind den *biologischen* nachempfunden. Sie bestehen dabei aus einer Anzahl verknüpften *Neuronen* (die *Kreise* in Abb. 7.11), welche den dazugehörigen *Schichten* zugeordnet sind. Mathematisch gesehen handelt es bei neuronalen Netzen um *gerichtete Graphen*. Die *erste* Schicht eines KNN wird *Eingangsschicht* genannt, die letzte *Ausgabeschicht*. Die *Eingangsneuronen* erhalten die *ursprünglichen* Daten, welche dann in den *versteckten* Schichten modifiziert werden. Das Ausgabeneuron hält das Resultat des Lernprozesses fest. Die *mittleren*

Schichten heißen *versteckte* Schichten, da die Neuronen keine ursprünglichen Daten entgegennehmen oder finale Ausgaben produzieren.

Ein neuronales Netz besteht in seiner primitivsten Form aus einer Eingabe und direkter Ausgabeschicht, dies ist dann ein sogenanntes Perzepron. Das andere Extrem sind neuronale Netze mit tausenden von Schichten, welche sich dann in den Bereich des Deep Learnings klassifizieren lassen.

7.4.2 Aufbau eines künstlichen Neurons

Die Bestandteile eines künstlichen Neurons sind *Eingabedaten*, *Übertragungs-* und die *Aktivierungsfunktion*. Die *Aufgabe* eines künstlichen Neurons ist die Annahme von *Eingabedaten* a_i , diese optional mit einem (Bias)-Gewicht w_i zu multiplizieren und die *Summe* (vgl. Gleichung (7.1)) dieser Werte als *Argument* in eine Aktivierungsfunktion zu übergeben (vgl. Gleichung (7.2)) (vgl. [RUSSELL 2012, S. 847]).

Die *Gewichte* w_i den Eingabedaten a_i definieren die *Stärke* zwischen einzelnen Neuronenverbindungen. Durch Optimierungsverfahren wie *Backpropagation* (siehe Abschnitt 7.4.4) lassen sich die Werte für diese Gewichte anpassen. Gewichte sind in neuronalen Netzen insofern wichtig als dass sie Phänomene wie *Vanishing* oder *Exploding* Gradients beeinflussen können (vgl. [MENDELS 2019]).

$$in_j = \sum_n^{i=0} w_{i,j} a_i \quad (7.1)$$

$$a_j = g(in_j) = g(\sum_n^{i=0} w_{i,j} a_i) \quad (7.2)$$

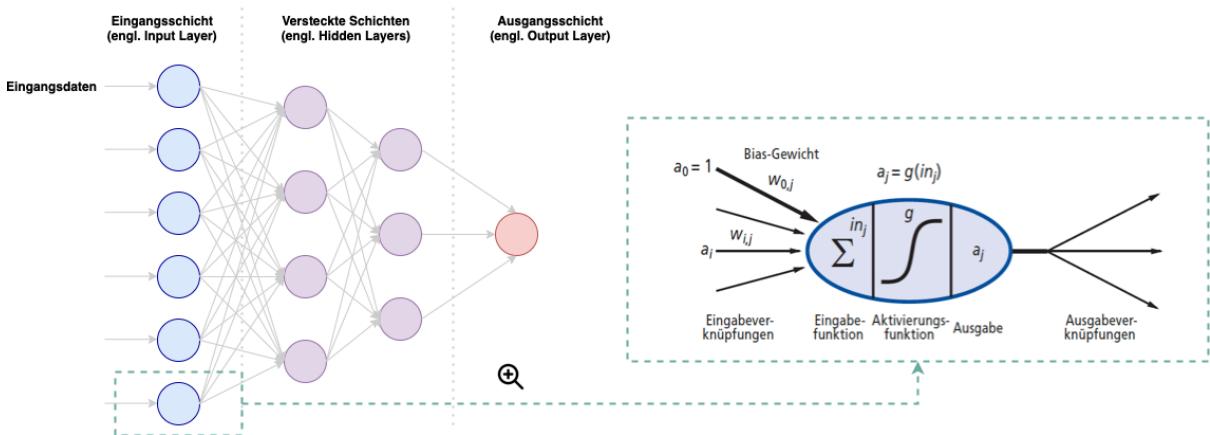


Abbildung 7.11: Vereinfachte Darstellung eines KNNs und einzelnen Neurons, aus [RUSSELL 2012, S. 846]

7.4.3 Aktivierungsfunktionen

Damit künstliche Neuronen *aktiviert* werden muss, so wie bei biologischen Neuronen, ein ausreichender *Reiz* bzw. *Aktivierung* vorliegen.

Hierfür werden die gewichteten Verknüpfungen aufsummiert und das Ergebnis in einer *Aktivierungsfunktion* verarbeitet (vgl. Gleichung (7.2)). Diese entscheidet darüber, ob ein Neuron *aktiviert* wird oder nicht (vgl. [ERTEL 2016, 268f]).

Im Allgemeinen werden *nicht-lineare* Aktivierungsfunktionen verwendet, da im Optimierungsprozess mit Ableitungen der jeweiligen Funktion gearbeitet wird (vgl. ??).

Im Folgenden wird nur die nicht-lineare Sigmoid und (Leaky) ReLU Aktivierungsfunktion erläutert, da YOLOv5 mit diesen arbeitet.

Dabei verwenden die versteckten Schichten des neuronalen Netzes bei YOLOv5 die Leaky ReLU Aktivierungsfunktion. Die Ausgabeschicht verwendet die Sigmoid Aktivierungsfunktion (vgl. [RAJPUT 2020]).

Sigmoid Aktivierungsfunktion

Die Sigmoid Aktivierungsfunktion (vgl. Abb. 7.12) wird überwiegend für Modelle verwendet, welche *Wahrscheinlichkeiten* prognostizieren. Der Grund ist, dass der Wertebereich der Sigmoidfunktion zwischen **0** und **1** liegt (vgl. Abb. 7.12).

Die Sigmoid Aktivierungsfunktion weist jedoch einen Nachteil auf. Zu große oder zu kleine Eingabewerte resultieren in **0** oder **1** → es kommt zum Problem des *verschwindenden Gradienten*. Die Ableitungskurve (blau) in Abb. 7.12 visualisiert das Problem.

Das Problem des verschwindenden Gradienten führt zu einer *Verlangsamung* des Lernfortschritts eines Modells, dies kostet Rechenleistung und Zeit.

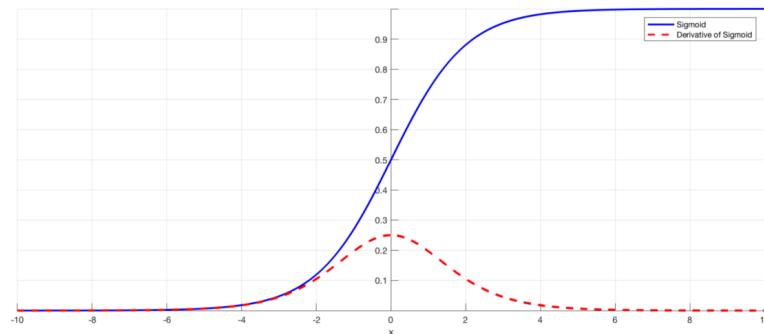


Abbildung 7.12: Sigmoid Aktivierungsfunktion (blau) und deren Ableitung (rot), aus [WANG 2019, Image 1: The sigmoid function and its derivative]

ReLU-Aktivierungsfunktionen

ReLU behebt den Nachteil der Sigmoid Aktivierungsfunktion, das Problem der „verschwindenden“ Gradienten. Sie ist die derzeit am häufigsten genutzte Aktivierungsfunktion, da sie sich aufgrund ihres Aufbaus für Convolutional Neural Networks (CNNs) und Deep Learning eignet. Der Wertebereich der ReLU-Funktion liegt im Intervall $[0, \infty)$

Damit geht der Nachteil einher, dass alle Werte < 0 nicht „beachtet“ bzw. 0 werden und dass das Modell nicht vollständig ist [SHARMA 2017].

Leaky ReLU-Aktivierungsfunktion

Um das Problem der *ursprünglichen* ReLU-Funktion zu lösen, wurde eine Erweiterung eingeführt, die *Leaky* (dt. „durchlässige“) Aktivierungsfunktion, vgl. Abb. 7.13.

Der Parameter a stellt das *Leak* dar. Der Wert hierfür ist normalerweise **0.01**. Bei anderen Werten wird von *Randomized* ReLU gesprochen.

Daraus folgt, dass der Wertebereich der Leaky ReLU theoretisch von $-\infty$ bis ∞ gehen kann [SHARMA 2017].

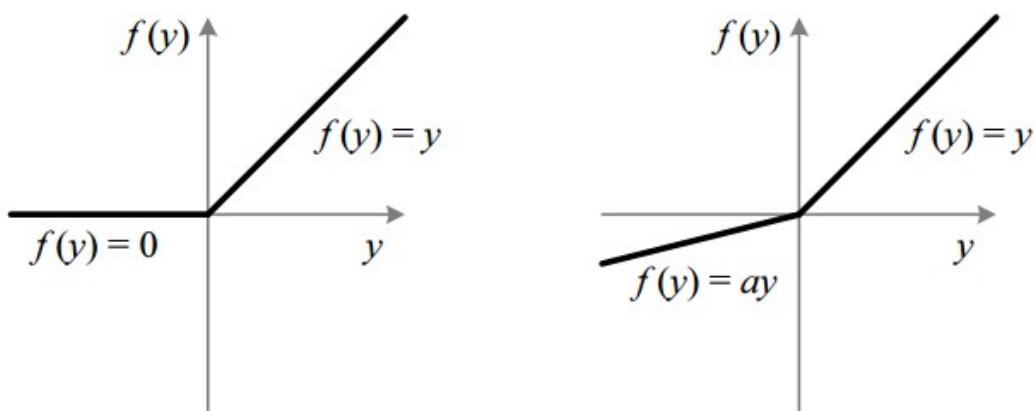


Abbildung 7.13: Vergleich der ursprünglichen ReLU und Leaky ReLU, aus [SHARMA 2017, ReLU (Rectified Linear Unit) Activation Function]

7.4.4 Optimierung neuronaler Netze

Durch eine Kostenfunktion kann die Performance eines Machine Learning (ML)-Modells bewertet werden. Die Regel dabei lautet, je höher die *Generalisierung* ist, desto fehlerfreier ist der Algorithmus. Der Wert der Generalisierung sagt aus, wie gut die Qualität eines Algorithmus auf ungesesehenen Daten ist (vgl. [ERTEL 2016, S. 192]).

Darüber hinaus können bereits entwickelte Algorithmen verbessert werden. Dies geschieht unter anderem durch das *Gradientenabstiegsverfahren* (engl. „Gradient Descent“). Hierbei findet das Verfahren vor allem im Backpropagatio-Algorithmus Einsatz (vgl. Abschnitt 7.4.4).

Bei der Verwendung von YOLOv5 kann entweder der Adam-Algorithmus verwendet werden oder der Stochastic Gradient Descent (SGD). Da diese Arbeit SGD verwendet, wird auch dieser im Folgenden erläutert nach der Grundidee von Gradientenabstiegsverfahren.

Gradient Descent

„Gradient Descent“ ist ein Algorithmus zur Verlustminimierung. Die Idee dabei ist, Parameter einer Zielfunktion iterativ anzupassen, bis ein minimaler Wert für eine Kostenfunktion vorliegt

(vgl. [GÉRON 2019, 4. Training Models – Gradient Descent])

Eine Herausforderung beim Finden des globalen Minimums stellen Funktionen höheren Grades dar. Denn diese können Terrassen enthalten, bei welchen sich der Gradient über längeren Zeitraum nicht verändert. Bei zu geringer Lernrate könnte das Verfahren nicht konvergieren (vgl. Abb. 7.14).

Es gibt zwei Arten von Gradient Descent Verfahren: Batch-Gradient Descent und den Stochastic Gradient Descent. Letzterer wird in YOLOv5 angewandt (vgl. [RUSSELL 2012, S. 838]). Beim ersten ist die Konvergenz zum globalen Minimum garantiert, da die gesamten Trainingsdaten verwendet werden – sofern die Lernrate klein genug ist. Der Nachteil an diesem Verfahren ist, dass es zeitintensiv werden kann.

Stochastic Gradient Descent

Stochastic Gradient Descent wählt immer einen zufälligen Trainingsdatensatz und berechnet daraufhin den Gradienten und versucht das Minimum der Funktion zu finden. Der Nachteil ist, dass bei Stochastic Gradient Descent eine Konvergenz bei einem Minimum nicht garantiert ist und kein kontinuierlicher Abstieg geschieht. Jedoch ist dieser Algorithmus im Vergleich zum Batch Gradient Descent schneller, aufgrund der geringeren Datensatzanzahl (vgl. [GÉRON 2019, 4. Training Models – Stochastic Gradient Descent]).

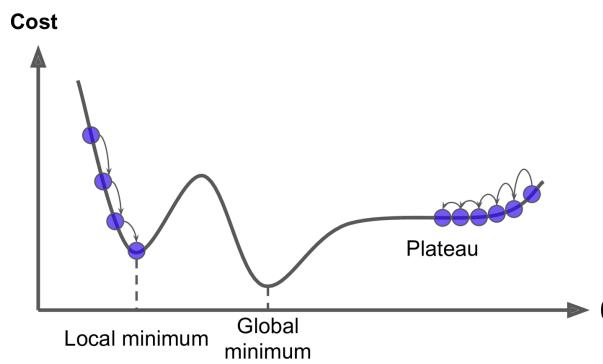


Abbildung 7.14: Darstellung einer Funktion höheren Grades und Visualisierung des damit verbundenen Problems beim Gradientenabstieg [GÉRON 2019, Training Fig. Models 4-3]

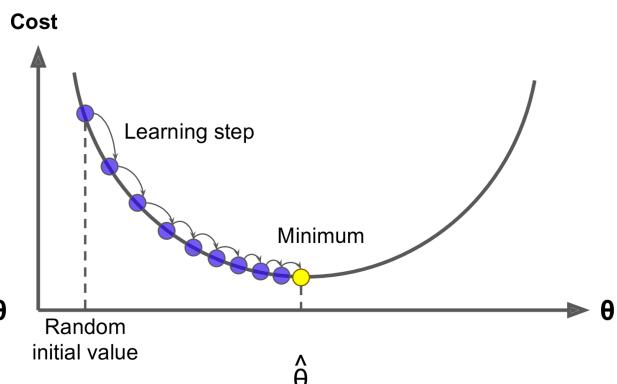


Abbildung 7.15: Visualisierung eines möglichen Gradientenabstiegverfahrens, bei welchem iterativ zum globalen Minimum hingearbeitet wird, aus [GÉRON 2019, Training Models Fig. 4-6]

Backpropagation Algorithmus

Der Backpropagation Algorithmus stellt eine Trainingsmethode für mehrschichtige neuronale Netze unter Verwendung des Gradientenabstiegs dar. Dabei sind die Schritte wie folgt:

1. Die Daten in sogenannte „Batches“ einteilen und diese dann in das neuronale Netz übergeben (engl. „forward pass“)
2. Den Fehler messen (engl. „error measurement“)

3. Von der letzten Schicht bis in die Eingabeschicht durchgehen und dabei Berechnen wie mit welchem Anteil die versteckte Schicht zum Fehler beträgt (engl. „reverse pass“)
4. Gradient Descent Schritt, bei welchem das Gewicht der Neuronenverbindung angepasst wird um den Fehler zu minimieren

(vgl. [GÉRON 2019, Introduction to Artificial Neural Networks – Multi-Layer Perceptron and Backpropagation])

7.4.5 Evaluation eines ML-Modells

Kreuzvalidierung ist ein Verfahrensgruppe, welche ermöglicht die Performance eines ML-Algorithmus (die Accuracy) zu bestimmen. Diese Methode eignet sich auch gut, wenn wenige Daten vorhanden sind (vgl. [RUSSELL 2012, 825ff]).

Bei der k -fachen Kreuzvalidierung werden die *gesamten* vorhandenen Daten auf k gleichgroße Mengen aufgeteilt (vgl. Abb. 7.16). Danach durchläuft das Modell k Testrunden, bei welchen eine Teilmenge als *Testsatz* und $k - 1$ Teilmengen als Trainingssätze verwendet werden. Über die Resultate der k Testrunden wird ein *Mittelwert* gebildet, welcher das Modell bewertet (vgl. [ERTEL 2016, 232f]).

Doch nicht nur eine geringe Fehlerrate ist bei einem Modell entscheidend, sondern auch dessen Komplexität.

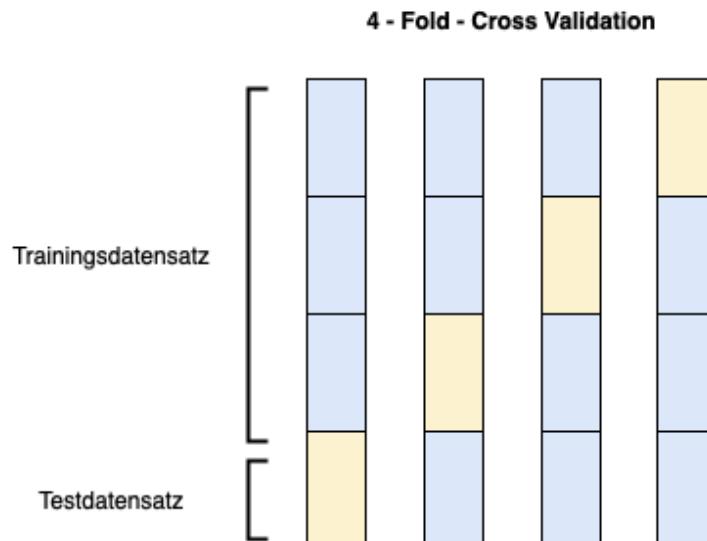


Abbildung 7.16: Visualisierung der k -fachen-Kreuzvalidierung (eigene Abbildung). Die gelben Abschnitte stellen jeweils die Testdatensätze dar

Precision, Recall und Accuracy

Metriken für die Bewertung eines Klassifikations-Modells sind unter anderem Precision (dt. „Genauigkeit“), Recall (dt. „Trefferquote“) und Accuracy (dt. „Richtigkeit“).

Recall versucht die Frage zu beantworten: „Welcher Anteil der tatsächlich positiven Klassifikationen wurde erkannt?“ (vgl. Gleichung (7.3))

Ein Nachteil an Recall ist, dass sich diese Metrik manipulieren lässt, indem die Mehrheit der Dateninstanzen als positiv bzw. true markiert wird (vgl. [GOOGLE 2020b, Recall]).

Precision versucht die Frage zu beantworten: „Welcher Anteil der positiven Klassifikationen war tatsächlich positiv?“ (vgl. Gleichung (7.4), [GOOGLE 2020b, Precision]).

Accuracy beschreibt den Anteil der tatsächlich positiv klassifizierten Objekte, gerechnet auf die Gesamtheit der klassifizierten Objekte (vgl. Gleichung (7.5)) [GOOGLE 2020a].

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (7.3)$$

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (7.4)$$

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (7.5)$$

Beispiel

Wir analysieren Insekten-Bilder, um herauszufinden ob ein Insekt eine Biene ist (true) oder nicht (false). 100 Insektenbilder, 10 Bilder davon sind Bienen.

- Precision: Wenn wir nun ein Bild von 100 als true identifizieren und es tatsächlich true ist, dann ist die Precision: 100%. Jedoch haben wir 9 weitere Bienen verpasst.
- Recall: Wenn wir nur ein Bild von 100 als true identifiziert haben, und das korrekt ist, dann ist die Precision 100% und Recall 10%, da 9 weitere (als positiv klassifizierte) Bilder fehlen.
- Accuracy: Wenn wir nun 10 Bilder von 100 als true klassifizieren und diese 10 Bilder auch korrekt sind, dann liegt die Accuracy bei 100%.

7.4.6 Architekturen Neuronaler Netze

Es gibt keine *allgemeine* „one-size-fits-all“ Architektur für neuronale Netze. Eine Basis-Architektur ist das Feedforward Neural Network (FNN). FNNs kennzeichnet sich dadurch, dass die Neuronen *keine* Zyklen untereinander aufweisen (vgl. [VAN VEEN 2016, Abb. „The Neural Network Zoo“]).

Da diese Arbeit mit CNN (vgl. Abschnitt 7.4.7) arbeitet, werden diese im Folgenden genauer erläutert.

7.4.7 Convolutional Neural Networks

CNNs sind eine spezielle Art neuronaler Netze. Genauer sind sie eine Unterart der Deep Neural Networks. CNNs eignen sich dabei besonders zur Bilderkennung (vgl. [KIM 2017, S. 121]).

ConvNet ist ein CNN, das den Bilderkennungsvorgang im menschlichen Gehirn rekonstruieren soll. Die Eigenschaft, sich am menschlichen Bilderkennungsvorgang zu orientieren, unterscheidet

ConvNet von den bis dahin entwickelten neuronalen Netzen. Gleichzeitig steigert diese Ähnlichkeit jedoch auch stark die Komplexität des Netzes, da ConvNet auf viele versteckte Schichten setzt (vgl. [KIM 2017, S. 121–123]).

Abb. 7.17 zeigt die Grundarchitektur von ConvNet. Besonders zu beachten ist dabei, dass die letzte Schicht eine Klassifizierungsschicht ist. Der Grund hierfür ist, dass Objekterkennung und

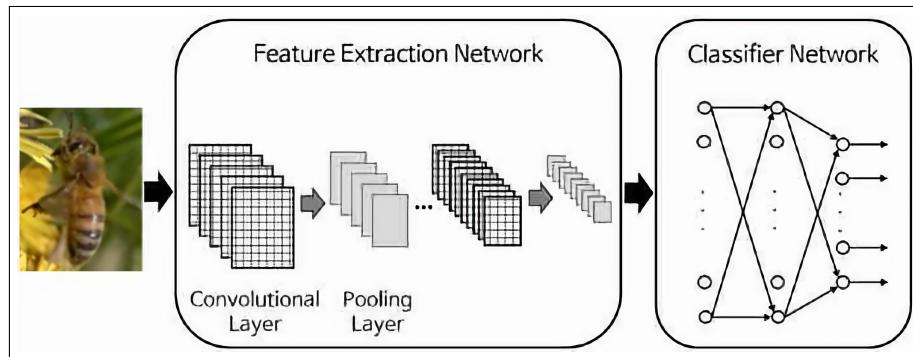


Abbildung 7.17: Architektur von ConvNet, nach [KIM 2017, S. 123]

Bildklassifizierung grundsätzlich den gleichen Prozess darstellen, sobald die einzelnen Features extrahiert sind. Die Erkennung eines Objektes als Hund stellt zum Beispiel den selben Vorgang dar wie die Klassifizierung eines Bildes oder Features in die Klasse der Hunde.

Ein Problem bei der Objekterkennung stellt dabei die Einheitlichkeit der Features dar. Um einheitliche Features bezüglich Faktoren wie Belichtung, Farben etc. zu erhalten, wird die sog. Feature Extraction verwendet. Dies geschah vor ConvNet noch meist manuell auf Grundlage der jeweiligen fachlichen Anforderung des neuronalen Netzes. Dieser Prozess war zeit- und kostenaufwändig, das resultierende Netz war nur für einen sehr speziellen Bereich einsetzbar und es konnte nur eine inkonsistente Leistungsfähigkeit erreicht werden.

ConvNet verfolgt für die Feature Extraction einen grundlegend anderen Ansatz. Hier ist dieser Prozess nicht unabhängig vom eigentlichen machine learning, sondern verwendet selbst ein neuronales Netz. Die Gewichte dieses Netzes werden während des Trainings angepasst, sodass die Features für den jeweiligen Anwendungsbereich bestmöglich extrahiert werden können.

Die Ergebnisqualität von ConvNet steigt dabei mit der Anzahl der Schichten im Feature Extraction Network. Jedoch sorgen mehr Schichten auch für einen höheren Trainings- und somit Zeitaufwand. Hier muss also je nach Anwendungsfall ein passendes Mittelmaß gefunden werden. Das Feature Extraction Network (dt. Netz zur Merkmalsextraktion) besteht dabei aus Convolutional Layers (dt. Faltungsschichten), daher auch der Name Convolutional Neural Network, und Pooling Layers (dt. Bündelungsschichten).

Eine Convolutional Layer unterscheidet sich stark von „klassischen“ Schichten eines neuronalen Netzes (Abschnitt 7.4.1). Sie verfügt nicht über Gewichte und Neuronen, die am Ende eine gewichtete Summe ausgeben, sondern enthält Filter. Wird ein Bild mittels solcher Filter verarbeitet, wird am Ende die Feature Map ausgegeben.

Die Anzahl der von einer Convolutional Layer erzeugten Feature Maps entspricht dabei genau der Anzahl der Filter, die die Schicht enthält (siehe Abb. 8.2).

Die Convolutional Filter sind dabei zweidimensionale Matrizen, meist 5x5 oder 3x3, wobei auch 1x1-Matrizen teils Verwendung in Convolutional Filtern finden. Die Werte dieser Matrizen

werden dabei während des Trainings festgelegt.

Der Convolution-Prozess addiert dabei die entsprechenden Felder der Bildmatrix mit denen der Filtermatrix. Dadurch lässt sich z.B. eine 4x4-Bildmatrix mittels einer 2x2 Filter-Matrix zu einer 3x3-Feature-Map reduzieren (zu sehen in Gleichung (7.6), vgl. [KIM 2017, S. 128]). Die Werte der einzelnen Filter-Matrizen sorgen dabei für unterschiedliche Gewichtungen der Werte in der Bild-Matrix, wodurch verschiedene Filter am Ende auch sehr verschiedene Feature-Maps aus dem gleichen Bild generieren können.

$$\begin{bmatrix} 1 & 1 & 1 & 3 \\ 4 & 6 & 4 & 8 \\ 30 & 0 & 1 & 5 \\ 0 & 2 & 2 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 9 \\ 4 & 7 & 9 \\ 32 & 2 & 5 \end{bmatrix} \quad (7.6)$$

Bevor die Feature Maps der jeweiligen Convolutional Layer ausgegeben werden, werden diese noch von der Aktivierungsfunktion (siehe Abschnitt 7.4.3) verarbeitet. Dies kann wieder eine für neuronale Netze typische Funktion wie (Leaky) ReLU oder Sigmoid sein (vgl. [KIM 2017, S. 122–130]).

Die zweite Schicht-Art in CNNs sind Pooling Layers. Sie reduzieren die Bildgröße, indem sie benachbarte Pixel zusammenfassen. Diese Vorgehensweise wird nicht erst seit CNNs oder ConvNet verwendet, sondern bereits in früheren Implementierungen neuronaler Netze zur Bildverarbeitung.

Wie genau das Pooling jeweils umgesetzt wird, hängt vom Anwendungsfall ab. Variable Parameter sind dabei die Anzahl der benachbarten Pixel, die zusammengefasst werden, sowie die Auswertungsfunktion. Hierzu kann z.B. der Durchschnitt der Pixel oder das Maximum im fraglichen Bereich verwendet werden.

Durch das Pooling wird dabei nicht nur aufgrund der hinterher kleineren Bildgröße die Performance des neuronalen Netzes erhöht. Der Prozess kompensiert auch in einem gewissen Maß für leichte Drehungen oder Verschiebungen von Objekten im Erkennungsbereich, wodurch das Modell am Ende weniger anfällig für Overfitting wird. (vgl. [KIM 2017, 130f])

8. Objekterkennung mit künstlicher Intelligenz

Dieses Kapitel befasst sich mit den Grundlagen der Objekterkennung in Bildern mittels künstlicher Intelligenz.

8.1 Objekterkennung vs. Bildklassifizierung

Die Verarbeitung von Objekten in einem Bild mittels künstlicher Intelligenz lässt sich in drei Arten aufteilen. (vgl. [KUZNETSOVA u. a. 2020, S. 1])

Das einfachste Vorgehen ist die **Bildklassifizierung**. Hierbei ist das Ziel, zu erkennen, welche Objekte in einem Bild vorhanden sind. Die Position der Objekte im Bild wird dabei nicht erfasst. Bei der **Objekterkennung** wird wiederum die Position der Objekte im Bild mitbestimmt. Der Prozess besteht hierbei aus zwei Schritten. Im ersten Schritt werden Objekte mit der Position im Bild erkannt. Im zweiten Schritt werden die erkannten Objekte klassifiziert.

Die verbleibende Verarbeitungsart ist die sogenannte **Visual Relationship Detection** (dt. Visuelle Beziehungserkennung). Hierbei werden zusätzlich, wie in Abb. 8.1 dargestellt, zu den erkannten Objekten noch die Beziehungen untereinander erkannt.

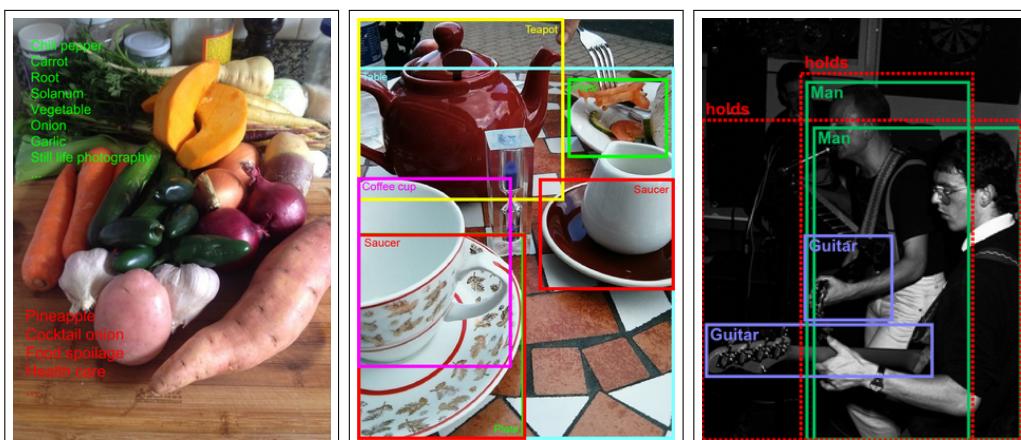


Abbildung 8.1: Vergleich der drei Arten der Bildverarbeitung, aus [KUZNETSOVA u. a. 2020, S. 2]

Für das im Rahmen dieser Arbeit zu behandelnde Problem wird ein Algorithmus zur Objekterkennung benötigt. Einfache Bildklassifizierung reicht nicht aus, da die Positionen und die Anzahl der Bienen auf dem Bild erkannt werden müssen. Visual Relationship Detection wird hingegen nicht benötigt, da nicht erkannt werden muss, was die Bienen konkret machen. Für die Erkennung der Bewegungsrichtung der Bienen reicht jeweils die aktuelle Position pro Bild aus.

8.2 Vergleich verschiedener Algorithmen

Die Entwicklung eines eigenen Algorithmus zur Objekterkennung in Bildern würde den Rahmen dieser Arbeit übersteigen. Daher werden nachfolgend einige der bekannteren, bereits existierenden Algorithmen miteinander verglichen, um bestimmen zu können, welcher dieser Algorithmen für die vorliegende Problemstellung am besten geeignet ist.

8.2.1 HOG

Histogram of Oriented Gradients (HOG) ist einer der ersten Algorithmen zur Objekterkennung in Bildern. Er wurde 2005 von N. Dalal und B. Triggs entwickelt und stellt eine wichtige Verbesserung bisher vorhandener Algorithmen dar, da er wesentlich besser mit invarianten Features wie Helligkeit und Skalierung umgehen kann.

HOG zerlegt dabei das Bild in ein Raster aus Zellen mit gleichen Abständen. Sollen verschiedene große Objekte erkannt werden, so wird das Bild mehrmals skaliert, wobei das Erkennungsfenster seine Größe nicht ändert.

HOG war für mehrere Jahre eine wichtige Grundlage für weitere Objekterkennungsalgorithmen und für Anwendungen, die auf Objekterkennung basierten (vgl. [ZOU u. a. 2019, S. 3]).

8.2.2 RCNN

Region-based Convolutional Neural Network (RCNN) wurde 2014 von R. Girshick et al entwickelt. Als einer der ersten Algorithmen zur Objekterkennung, die auf GPU-Architekturen optimiert wurden, verfolgt dieser Algorithmus einen grundlegend anderen Ansatz als HOG.

RCNN sucht mittels selektiver Suche alle Objektkandidaten im Bild. Jeder dieser Kandidaten wird dann auf eine einheitliche Bildgröße skaliert und in ein Convolutional Neural Network (CNN), wie in Abb. 8.2, gegeben, um Features zu extrahieren (siehe Abb. 8.2). Abschließend werden lineare Support Vector Machine (SVM) Klassifikatoren verwendet. Mit diesen wird bestimmt, ob in der jeweiligen Region ein Objekt ist und zu welcher Objektklasse dieses gehört. Das Problem dieses Ansatzes ist, dass viele überlappende Objektkandidaten entstehen, für die das CNN redundant die Features bestimmt. Dadurch steigt die Verarbeitungszeit drastisch an, wodurch RCNN nicht für die Live-Analyse von Videos verwendbar ist (vgl. [ZOU u. a. 2019, 3f]).

Fast RCNN

2015 stellte Girshick **Fast RCNN** als Verbesserung von RCNN und *Spatial Pyramid Pooling Networks* (SPPNet) vor. Es kombiniert dabei die Vorteile der beiden Objekterkennungsalgorithmen, wodurch der Detektor und Bounding-Box-Regressor gleichzeitig trainiert werden können. Dieser Algorithmus ist jedoch immer noch von der Geschwindigkeit der Erkennung von Objektkandidaten limitiert, weshalb sich auch Fast RCNN nur eingeschränkt für Live-Auswertungen eignet (vgl. [ZOU u. a. 2019, S. 4]).

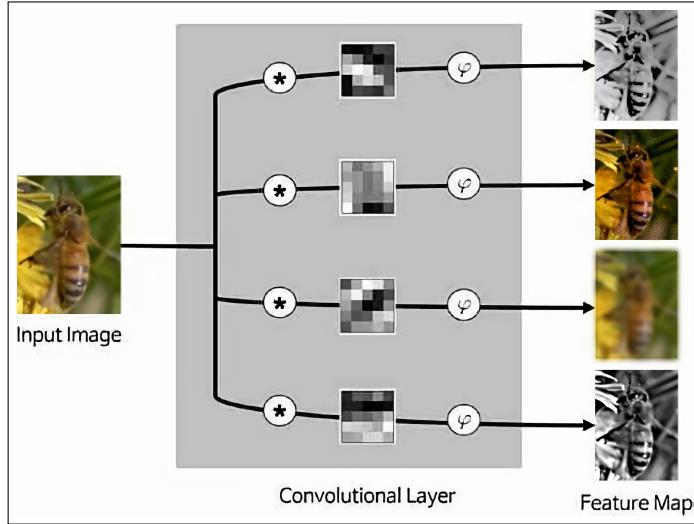


Abbildung 8.2: Beispielhafte Darstellung der Verwendung von Convolutional Layers in RCNN, nach [KIM 2017, S. 125]

Faster RCNN

Kurz nach Fast RCNN, auch im Jahr 2015, wurde dieses Problem von S. Ren et al mit **Faster RCNN** behoben, indem auch für die Erkennung von Objektkandidaten Convolutional Neural Networks anstelle von selektiver Suche verwendet werden. Genauer gesagt verwendet Faster RCNN ein sog. Region Proposal Network (RPN) zur Erzeugung der Objektkandidaten. Dadurch lässt sich mit Faster RCNN eine nahezu Echtzeit-Analyse von Videodaten erreichen. Der Algorithmus besitzt durch die redundante Auswertung von Features allerdings immer noch Performance-Einbußen. Dieses Problem wurde später unter anderem von RFCN in Angriff genommen (vgl. [ZOU u. a. 2019, S. 4]).

8.2.3 RFCN

Region-based Fully Convolutional Networks (RFCN) beheben das Problem von RCNN, dass Regionen des Bildes teilweise mehrfach ausgewertet bzw. klassifiziert werden, wodurch die Performance stark leidet. Dadurch ist RFCN 2,5-20x schneller als Faster RCNN und kann daher auch für Live-Analysen von Videos verwendet werden.

Während bei den verschiedenen RCNN-Ansätzen die Features als Ganzes zur Auswertung verwendet werden, wird bei RFCN jedes Feature in sog. Bins unterteilt. Für jeden dieser Bins wird dann die Wahrscheinlichkeit des Vorkommens einer bestimmten Objektklasse bestimmt. Durch diese Vorgehensweise kann die Klassifizierung der Bins einmal mit dem ganzen Bild durchgeführt werden. Die Auswertung, ob ein fragliches Objekt tatsächlich in einer Region of Interest (RoI) vorhanden ist, kann dann über die Bins stattfinden. Sind überlappende RoIs vorhanden, werden die Bins trotzdem nur einmal klassifiziert. Im Gegensatz dazu würde bei RCNN der sich überlappendende Bereich mehrfach ausgewertet werden (vgl. [DAI u. a. 2016]).

8.2.4 YOLO

Auch You Only Look Once (YOLO) ist ein Algorithmus zur Objekterkennung, der auf Convolutional Neural Networks setzt. YOLO wurde entwickelt, da selbst die schnellsten bisherigen Ansätze mit CNNs noch nicht schnell genug sind, um Videomaterial in der heute typischen Qualität und Bildrate auszuwerten.

YOLO wurde 2015 von Redmon et al entwickelt und erreicht Geschwindigkeiten von bis zu 155 Bildern pro Sekunde. Verschiedene Implementierungen von YOLO erreichen bei höherer Erkennungsqualität etwas weniger Bilder pro Sekunde, sind aber dennoch deutlich schneller als bisherige Ansätze.

Die Funktionsweise von YOLO unterscheidet sich dabei hauptsächlich in einem Punkt, der auch im Namen schon ersichtlich ist: Der bisherige Ansatz, erst Objektkandidaten zu finden und diese dann zu verifizieren, wird durch einen Ansatz ersetzt, der beide Schritte zu einem kombiniert und das ganze Bild daher „nur einmal anschaut“.

Das Bild wird dabei als ganzes in ein neuronales Netz gegeben, das das Bild in Regionen unterteilt und zeitgleich Bounding-Boxes und deren Wahrscheinlichkeiten bestimmt.

Die erste Version von YOLO hat dabei noch einige Probleme mit der Genauigkeit der Objektpositionen, die mit den folgenden Versionen bei gleich bleibender Geschwindigkeit verbessert werden.

2020 ist Redmon aufgrund von ethischen Problemen im Bereich der computergesteuerten Objekterkennung aus dem Projekt ausgestiegen. YOLO wurde noch im gleichen Jahr von Bochkovskiy unter dem Namen YOLOv4 verbessert.

Fast zeitgleich veröffentlichte die Firma Ultralytics YOLOv5, eine Erweiterung von YOLOv3 auf das Machine-Learning-Framework **PyTorch**. Während YOLOv5 daher keine wirkliche Verbesserung des Algorithmus an sich darstellt, bedeutet es eine wesentlich einfachere Verwendung. YOLOv5 hat zudem, wie bereits seine Vorgängerversionen, verschiedene Modellgrößen, die unterschiedliche Erkennungsqualität erlauben. Die Geschwindigkeit der Modelle nimmt dabei antiproportional zur Erkennungsqualität ab, wie in Abb. 8.3 ersichtlich (vgl. [ZOU u. a. 2019, S. 4], [MAINOLA 2021]).

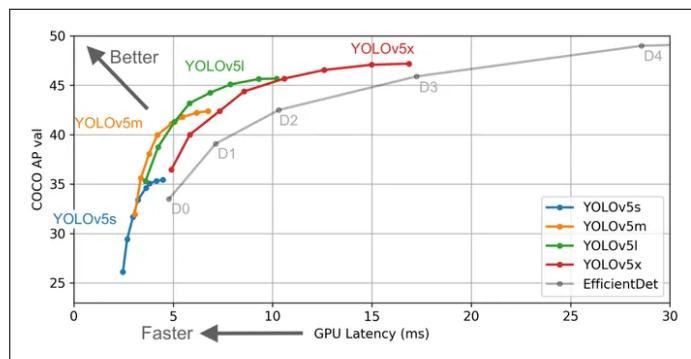


Abbildung 8.3: Vergleich der Modelle von YOLOv5, aus [MAINOLA 2021]

8.2.5 SSD

Single Shot Multibox Detector (SSD) wurde im Jahr 2015 von W. Liu et al vorgestellt. Wie YOLO ist auch SSD ein sogenannter **One Stage Detector**, es werden also zeitgleich in einem einzigen neuronalen Netz sowohl die Objektkandidaten als auch deren Wahrscheinlichkeiten bestimmt. Schnelle Versionen von SSD können dabei mit ca. 60 Bildern pro Sekunde arbeiten. Der Unterschied zwischen anderen Algorithmen und SSD ist, dass SSD die Erkennung verschieden großer Objekte auf unterschiedlichen Schichten im neuronalen Netz durchführt. Frühere Algorithmen führen die Objekterkennung ausschließlich auf der letzten Schicht durch. Dadurch hat SSD den anderen Algorithmen gegenüber einen Vorteil, sofern es um die Erkennung verschieden großer oder sehr kleiner Objekte geht (vgl. [ZOU u. a. 2019, 4f]).

8.2.6 Detectron

Detectron ist ein von Meta Research entwickeltes Framework. Ziel von Detectron ist es, die Forschung an Objekterkennungsalgorithmen zu erleichtern. Es enthält unter anderem die Algorithmen Fast RCNN, Faster RCNN und RFCN, welche in der Implementierung von Detectron unter anderem auf ResNet und ResNeXt als neuronale Netze setzen. Die Implementierung weiterer Arten von neuronalen Netzen ist dabei so einfach wie möglich gestaltet, sodass problemlos mit verschiedenen Kombinationen geforscht werden kann.

Detectron setzt auf Caffe2, ein Deep Learning Framework für Python. Sein Nachfolger, Detectron2, wurde aufbauend auf PyTorch von Grund auf neu geschrieben. Für Detectron2 ist dabei wesentlich weniger Trainingsaufwand nötig, während gleichzeitig neue Features wie Rotating-Bounding-Boxes unterstützt werden (vgl. [GIRSHICK u. a. 2018], [WU u. a. 2019]).

8.2.7 Vergleich

Tabelle 8.1 gibt einen Überblick über die für diese Arbeit wichtigen Parameter der in diesem Kapitel behandelten Objekterkennungs-Algorithmen. Der Vergleich ist dabei ein relativer Vergleich der verschiedenen Algorithmen und soll keine absoluten Aussagen treffen.

Algorithmus	Erkennungsqualität	Geschwindigkeit
HOG	schlecht	schlecht
RCNN	mittel	schlecht
Fast RCNN	mittel	schlecht - mittel
Faster RCNN	mittel	mittel
RFCN	mittel	gut
YOLO	gut	sehr gut
SSD	sehr gut	gut

Tabelle 8.1: Vergleich der Objekterkennungs-Algorithmen

8.3 Vergleich verschiedener Objekt-Tracking-Lösungen

8.3.1 Objekt-Tracking

Objekt-Tracking ist der Prozess, bei dem mehrere, bereits detektierte Objekte eine Identifikationsnummer (ID) zugewiesen bekommen. Diese ID bleibt während der ganzen Videoaufzeichnung dem jewei-

ligen Objekt zugewiesen. Bei der Objekterkennung hingegen bekommen die Objekte lediglich ihre Klassennummer zugewiesen. Sofern nur ein Objekt getrackt werden muss, genügt eine reine Objektdetektion. Bei mehreren Objekten ist es unter anderem möglich, dass die Informationen bestimmten Objekten zugewiesen müssen, weshalb Tracking nützlich ist (vgl. [PAPERSWITHCODE o. D.]).

In Abbildung Abb. 8.4 zu sehen, wird im Falle der Objektdetektion jedem Auto die gleiche Nummer vergeben. Eine Möglichkeit Tracking zu integrieren wäre hierbei, verschiedene farbige Autos verschiedenen Klassen zuzuordnen, um sich eine bessere Übersicht der Fahrzeuge zu schaffen. Dabei wird zwar nicht jedes Auto separat getrackt, aber bereits in drei Gruppen aufgeteilt.

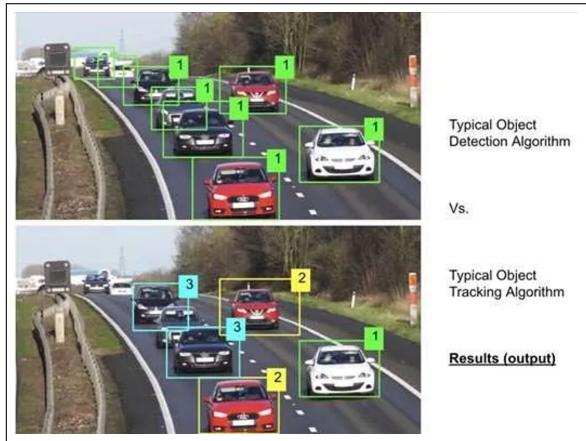


Abbildung 8.4: Vergleich von Objekterkennung und Objekttracking, aus [MEEL o. D.]

Ablauf des Objekt-Tracking

Das Vorgehen beim Objekt-Tracking verläuft in drei Phasen. Zunächst erfolgt die **Objekterkennung**. Hierbei werden die Objektklasse und die jeweiligen Koordinaten in Form einer Liste gespeichert. Im zweiten Schritt wird die **Bewegung analysiert** und zukünftige Standorte versucht vorherzusagen. In der letzten Phase werden die Vorhersagen mit den realen Koordinaten der Objekte **verglichen**. Das Objekt am nächsten zur Vorhersage ist dann voraussichtlich das vorausgesagte Objekt (vgl. [MEEL o. D.]).

8.3.2 Tryolabs - Norfair

Norfair ist eine Python-Bibliothek für Objekt-Tracking in Echtzeit des Entwicklerteams Tryolabs. Voraussetzungen für Norfair ist die vorangehende Objekt-Detektion mittels eines beliebigen Detektors. Zum Vorgehen des Algorithmus werden die Koordinaten der einzelnen Detektionen verwendet. Mithilfe der vorangehenden Koordinaten wird der nächste Standort des Objekts vorausgesagt. Die Vorhersage wird mit den echten Daten verglichen und daraufhin der am besten geeigneten Detektion zugewiesen (vgl. [TRYOLABS 2022a]).

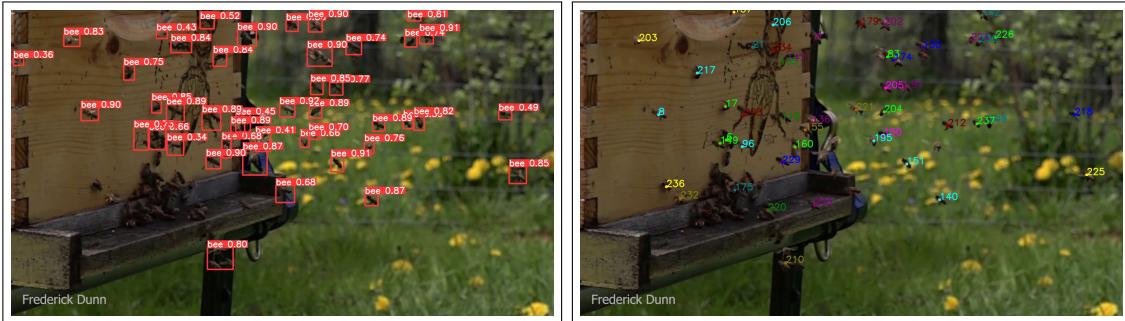


Abbildung 8.5: Vergleich von Objekterkennung und Objekttracking, verarbeitete Screenshots aus YouTube-Video von [DUNN 2021]

Nachdem der Norfair-Algorithmus angewendet wurde, besitzt jedes erkannte Objekt eine eigene ID und kann somit getrackt werden, wie in Abbildung Abb. 8.5 zu sehen.

8.4 Herausforderungen bei der Bildverarbeitung mittels künstlicher Intelligenz

Gerade bei der automatisierten Verarbeitung von Bildern und Videos mittels selbstlernender Algorithmen ergeben sich besondere Herausforderungen, auf die im folgenden Kapitel eingegangen werden soll.

Während Bilder und Videos für Menschen wesentlich leichter als einfache Zahlen zur Informationsgewinnung verwendet werden können, ist für die Verarbeitung dieser mittels Computer zusätzlicher Aufwand notwendig. Das ist darauf zurückzuführen, dass ein Bild zwar für einen Menschen sehr viele Informationen auf geringem „Raum“ zur Verfügung stellen kann. Diese Daten sind jedoch nicht klar strukturiert. Computer sind aber in erster Linie und mit klassischen Vorgehensweisen primär für die Verarbeitung strukturierter Daten geeignet.

Besonders bei der Arbeit mit künstlicher Intelligenz ergeben sich zusätzlich zu den ohnehin vorhandenen Herausforderungen weitere Probleme. Die Vorgehensweise bei der Objekterkennung beinhaltet, wie in vorangehenden Kapiteln behandelt, in jedem Fall eine **Klassifizierung** von Bildern oder Objektkandidaten. Dieser Vorgang funktioniert besser, je ähnlicher sich die Objekte sehen.

Dabei können schon minimal andere Farben der Objekte in verschiedenen Szenarien einen exponentiell größeren Trainingsaufwand bedeuten, da das zugrundeliegende neuronale Netz bspw. nicht nur die Form des Objekts, sondern auch das mögliche Farbspektrum erlernen muss.

Auch ein anderer Kamerawinkel oder eine Verzerrung des Bildes durch eine andere Objektiv-Art sind fatal für die Erkennung eines Objektes und führen zu einem deutlich höheren Trainingsaufwand.

Nicht zuletzt stellt auch eine teilweise Verdeckung eines Objektes ein Problem dar, da ein neuronales Netz bei der Objekterkennung nach dem Vorhandensein bestimmter Merkmale „sucht“. Sind relevante Merkmale eines Objektes verdeckt, führt dies zu schlechteren Erkennungsraten. Dieses Problem lässt sich nur bedingt durch längeres Trainieren oder mehr Trainingsdaten lösen. Wird zu lange auf bestehenden Daten trainiert, könnte es zu Overfitting kommen. Ein zu großer Trainingsdatensatz mit vielen teilweise verdeckten Objekten stellt zudem nicht nur einen sehr

großen Aufwand dar, sondern führt im schlimmsten Fall auch dazu, dass das neuronale Netz nicht lernt, da es keine sinnvollen Objektklassen erarbeiten kann.

Die aufgeführten Herausforderungen sind speziell im Kontext dieser Arbeit relevant. Da sich die zu entwickelnde Anwendung vor allem an Freizeitimker*innen mit begrenzten finanziellen Mitteln richtet, muss die Anwendung mit Rohdaten verschiedenster Kameras arbeiten. Diese sind gerade im Fall von Webcams oder Actioncams oft von geringer Qualität. Günstige Kameras haben meist eine geringe Auflösung und verhältnismäßig schlechte Sensoren, wodurch Farben schlecht aufgenommen werden, Kontraste geringer ausfallen und Änderungen an den Lichtverhältnissen große Auswirkungen auf das entstehende Bild haben. Da die Kamera einen Bienenstock aufzeichnen muss, ist anzunehmen, dass sich die Lichtverhältnisse im Laufe eines Tages ändern können. Auch der Winkel der Kamera auf den Bienenstock kann nicht als konstante Größe betrachtet werden, da dieser oftmals von den örtlichen Gegebenheiten abhängig ist.

8.5 Vorbereitung der Trainingsdaten

Um mit den im vorangehenden Abschnitt beschriebenen Herausforderungen möglichst unkompliziert umgehen zu können, bietet es sich an, die Trainingsdaten entsprechend Vor- und Aufzubereiten. Dieser Vorgang wird auch als **Data Augmentation** (dt. „Datenzuwachs“) bezeichnet.

Da für den Anwendungsfall der Erkennung von Bienen die Farben nicht wichtig sind, ist ein erster Schritt, die Farben in Graustufen umzuwandeln. Weitergehend kann zur weiteren Reduzierung der Komplexität eine Kantenerkennung durchgeführt werden, wie in Abb. 8.6 zu sehen. Dieser Prozess ist in Abb. 8.6 ersichtlich. In diesem Beispiel wird die Kantenerkennung durch den sog. „Ant colony algorithm“ durchgeführt (vgl. [ZHANG und DAHU 2019, 43f]).

Da die Anwendungen Bienen aus verschiedenen Winkeln, in verschiedenen Größen und mit verschiedenen Lichtbedingungen erkennen soll, können die Trainingsdaten zudem weitergehend aufbereitet werden. Es bietet sich an, die Bilder zu drehen und zu spiegeln, um den Einfluss von geänderten Kamerawinkeln oder -positionen auf die Erkennung von Objekten zu minimieren. Zudem kann die Sättigung der Bilder erhöht und verringert werden. Dadurch werden Effekte durch unterschiedliche Lichtbedingungen abgeschwächt, da z.B. bei einer Verringerung der Sättigung auch die Kontraste im Bild abnehmen. Eine der Stärken von YOLO als Bilderkennungsalgorithmus ist, dass die Daten hierbei im Rahmen des Trainierens von YOLO automatisch vorbereitet werden, siehe Abb. 8.7.

Dabei muss ein Mittelmaß gefunden werden, um möglichst viele der tatsächlichen Objekte im Bild zu erkennen, andererseits aber auch, um nicht zu viele falsche Treffer zu erzeugen.

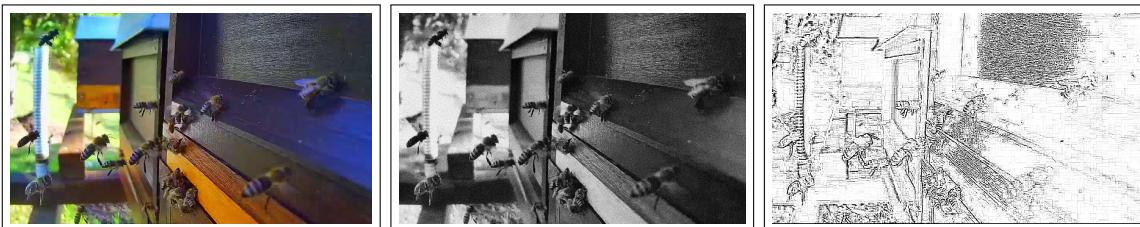


Abbildung 8.6: Vorbereitung von Trainingsdaten: Original, Graustufen und Kanten, aus [JUNG 2021]

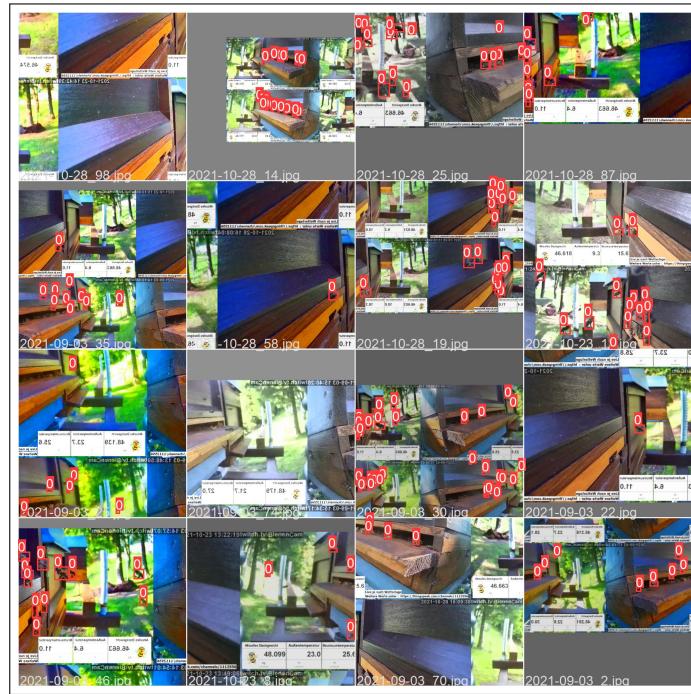


Abbildung 8.7: Vorbereitung der Trainingsdaten mit YOLOv5

9. Metriken zur Evaluation

Um die entwickelten Algorithmen zur Erkennung und dem Tracking der Bienen bewerten zu können, erarbeitet das folgende Kapitel passende Metriken.

Dabei muss beachtet werden, dass zum Evaluieren sowohl für den Objekterkennungs- als auch den Trackingalgorithmus die selbe Datenbasis verwendet werden muss. Liegt dies nicht vor, so können die Algorithmen nicht bewertet werden, da der Trackingalgorithmus auf dem Objekterkennungsalgorithmus aufbaut. Die Evaluation soll anhand der produzierten Ergebnisse stattfinden. Dabei sollen Performance (quantitativ) und Qualität betrachtet werden.

Die Performance kann hierbei in den überwiegenden Fällen mit numerischen Werten bewertet werden, beispielsweise der Anzahl der Bilder pro Sekunde.

Die Ergebnisqualität ist schwieriger zu vergleichen, da hierfür unter Umständen eine manuelle Prüfung der Ergebnisse benötigt wird.

Dies kann zum Beispiel beim Trackingalgorithmus der Fall sein, da die Validierung eines solchen Algorithmus kaum automatisiert werden kann. Zum Validieren müssten sowohl die Bienen manuell gezählt als auch die An- und Abflugrate bestimmt werden.

Im besten Fall lässt sich hierfür eine quantitative Metrik finden, welche die Vergleichbarkeit steigert (vgl. [GODIL u. a. 2014, S. 3]).

9.1 Metriken zur Evaluation des Algorithmus zur Bienenerkennung

Zeitliche Performance

Die zeitliche Performance bei der Objekterkennung lässt sich je nach Implementierung des verwendeten Algorithmus relativ leicht messen, da meist eine optionale Ausgabe der **Berechnungszeit pro Bild** über die Konsole ausgegeben werden kann. Mit YOLOv5, der PyTorch-Implementierung des YOLO-Algorithmus, ist dies möglich. Für die Bewertung kann ein „Schwellenwert“ angegeben werden. Wird dieser überschritten, gilt die zeitliche Performance eines Algorithmus in Anbetracht dieser Arbeit als ausreichend und umgekehrt.

Qualität

Die Ergebnisqualität lässt sich bei der reinen Objekterkennung mit quantitativen Metriken bestimmen. Eine der Metriken kann Accuracy sein (vgl. Abschnitt 7.4.5). Accuracy wird unter

anderem bei der Kreuzvalidierung (vgl. Abschnitt 7.4.5) gemessen, welche auch in dieser Arbeit verwendet wird zur Evaluation. Anhand der Accuracy können Algorithmen untereinander verglichen werden.

9.2 Metriken zur Evaluation des Tracking-Algorithmus

Zeitliche Performance

Beim Objekttracking ist die zeitlichen Performance über die Anzahl der berechneten Bilder pro Sekunde zu bestimmen.

Qualität

Die Bestimmung der Ergebnisqualität ist beim Objekttracking komplexer als bei der Objekterkennung. Denn hierbei kann die Auswertung nicht auf Basis einzelner Bilder erfolgen. Damit fällt die Möglichkeit weg, die Ergebnisse direkt anhand der ohnehin schon gelabelten Bilder zu bewerten.

Hier ergeben sich nun zwei mögliche Ansätze zur Evaluation. Die erste Möglichkeit ist ein kleiner Ausschnitt aus einem Video, der zu Benchmark-Zwecken manuell gelabelt wird. Dies ist eine recht problemlose Option, da hierfür lediglich etwas Zeit für das Labeling investiert werden muss. Es können jedoch menschliche Fehler auftreten. So kann es zum Beispiel passieren, dass ein Algorithmus besser funktioniert als das manuelle Labeln und so eine Fehlerrate angezeigt wird, die nicht der Wirklichkeit entspricht. Zudem ist es in einem solchen Benchmark schwer, die verschiedenen Lichtverhältnisse etc., die in der Realität auftreten können, abzubilden. Hierfür wäre ein erheblicher Zeitaufwand nötig, da verschiedene Videoausschnitte identifiziert und gelabelt werden müssen.

Die zweite Möglichkeit ist ein Vergleich der Tracking-Daten mit den Daten des Objekterkennungs-Algorithmus. Dazu wird für jedes Bild im Video die Anzahl der erkannten Bienen gespeichert. Nun werden die Bienen mittels des Tracking-Algorithmus getrackt. Für jede getrackte Biene existiert nun eine „Lebensdauer“. Daraus kann abgeleitet werden, wie viele Bienen in jedem Bild erfolgreich getrackt wurden. Diese beiden Zahlen lassen sich nun numerisch vergleichen. Es ergibt sich eine quantitative Bewertungsmöglichkeit für die Tracking-Qualität relativ zur Erkennungsqualität.

Zu beachten ist bei der zweiten Methode, dass es nur eine relative Bewertung ist. Werden alle erkannten Bienen auch getrackt, bedeutet das nicht automatisch, dass die Anwendung insgesamt gut performt. Liegt die Erkennungsrate nur bei 70%, bedeutet folglich eine Tracking-Qualität von 100% letztlich auch nur eine Gesamtqualität von 70%. Bei dieser Vorgehensweise müssen die beiden Werte also miteinander verrechnet werden.

9.3 Zielsetzung für die Metriken

Zeitliche Performance

Das zeitliche Performance-Ziel für den zu entwickelnden Algorithmus ist leicht definierbar. Das Optimum wäre, dass die Anwendung das Bildmaterial in Echtzeit auswerten kann. Dazu müssen die Verarbeiteten Bilder pro Sekunde größer oder gleich der Bilder pro Sekunde des Quell-Videos oder Video-Streams sein.

Generell ist eine bessere Performance wünschenswert, da die Anwendung auch auf schlechterer Hardware¹ laufen soll. Auf überdurchschnittlich guter Hardware muss also auch ein überdurchschnittlich gutes Ergebnis erzielt werden, um die Anwendung auch auf durchschnittlicher Hardware sinnvoll einsetzen zu können.

Soll ein Video-Stream aus dem Internet analysiert werden, sind grundsätzlich FPS im Bereich von 24-60 zu erwarten (vgl. [TWITCH.TV 2022]). Die Anwendung sollte also auf moderater Hardware mindestens 24 Bilder pro Sekunde, im Optimalfall bis zu 60 Bilder pro Sekunde verarbeiten können. Eine Erhöhung auf bis zu 120 FPS kann in Zukunft nötig sein, zum Zeitpunkt des Schreibens dieser Arbeit nehmen jedoch die wenigsten handelsüblichen Kameras mit einer solchen Anzahl an Bildern pro Sekunde auf. Aufgrund der Ausrichtung der Anwendung auf das Anwendungsgebiet der Freizeitimker*innen ist es unwahrscheinlich, dass eine derartige Performance in naher Zeit benötigt wird.

Qualität

Die tatsächliche Tracking-Qualität ist für die gewünschte Auswertung weniger relevant, als es auf den ersten Blick erscheint. Da das Ziel der Anwendung nicht ein perfekter Blick auf den aktuellen Stand sondern eher ein qualitativer Vergleich über die Zeit ist, muss die Qualität nicht unbedingt bei 100% liegen.

Eine zu geringe Qualität würde jedoch auch einen qualitativen Vergleich stark negativ beeinflussen. Die Quote der korrekt erkannten Bienen sollte daher bei mindestens 90-95% liegen, wobei hier eine höhere Quote erstrebenswert, wenn auch nicht notwendig ist. Ebenso sollten die erkannten Bienen mit einer Genauigkeit von mehr als 90% korrekt getrackt werden, da die Auswertung maßgeblich von den Flugpfaden der einzelnen Bienen abhängt.

Mit der im vorherigen Abschnitt beschriebenen Bewertungsmethode für das Tracking ergibt sich daraus also eine erwartete Gesamtgenauigkeit von 81%-85,5%.

¹Beispielsweise Rechner ohne dedizierte Grafikkarte oder auch Einplatinencomputer wie Raspberry Pis

Teil III

Umsetzung

10. Verwendete Algorithmen

Die verschiedenen Algorithmen, die für die Objekterkennung in Frage kommen, wurden bereits in Abschnitt 8.2 verglichen. Aus Tabelle 8.1 geht hervor, dass die YOLO-Algorithmenfamilie aufgrund ihrer Geschwindigkeit und hoher Erkennungsgenauigkeit am besten für die vorliegende Problemstellung geeignet ist. Im folgenden Kapitel soll daher genauer auf die verschiedenen Versionen des YOLO-Algorithmus und die jeweils zugrundeliegende Funktionsweise eingegangen werden.

Da die verschiedenen Versionen aufeinander aufbauen, werden dabei auch die älteren Versionen detailliert beschrieben, um ein grundlegendes Verständnis für die Funktionsweise der YOLO-Familie aufzubauen.

10.1 Funktionsweise

YOLOv1

Wie bereits in Abschnitt 8.2.4 beschrieben, werden Bilder von YOLO von einem einzigen neuronalen Netz verarbeitet, das sowohl die Bounding-Boxes als auch die Klassenwahrscheinlichkeiten bestimmt. YOLO behandelt die Objekterkennung dabei als **Regressionsproblem**.

YOLO wird dabei in den drei in Abb. 10.1 ersichtlichen Schritten ausgeführt.

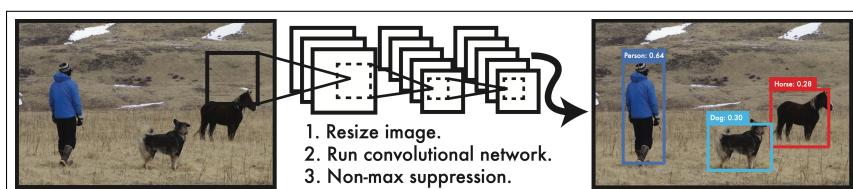


Abbildung 10.1: YOLOv1 Schema, aus [REDMON u. a. 2015, S. 1]

Zuerst werden die Bilder auf eine einheitliche Größe skaliert. Danach wird das Convolutional Network verwendet, um die Bounding-Boxes für jede Klasse zu bestimmen. Im letzten Schritt werden mittels Non-max suppression die erkannten Bounding-Boxes nach der vom Netzwerk bestimmten Wahrscheinlichkeit gefiltert, um nur die Objekte anzuzeigen, für die das Netzwerk eine hohe Wahrscheinlichkeit bestimmt hat.

Durch die Verwendung eines einzigen neuronalen Netzes hat YOLO dabei den Vorteil, dass es

wesentlich leichter trainiert und optimiert werden kann. Bei mehreren hintereinander geschalteten neuronalen Netzen müssten zur Erkennungsoptimierung alle vorhandenen neuronalen Netze verschieden trainiert und kombiniert werden, wodurch ein hoher Zeitaufwand generiert wird. Bei YOLO hingegen kann das Netz bereits während des Trainings leicht optimiert werden, da Änderungen an Gewichten direkt ausgewertet werden können.

Durch die Analyse des gesamten Bildes während des Trainings hat YOLO zudem den Vorteil, dass der Kontext von Objekten erkannt werden kann, wodurch z.B. Visual Relationship Detection möglich wird. Zudem kann das Netz dadurch erkennen, welcher Teil des Bildes zum Hintergrund gehört. „Flecken“ auf dem Hintergrund werden von YOLO daher wesentlich seltener fälschlicherweise für Objekte gehalten als bei anderen Algorithmen. Dieser Umstand ist gerade für die Erkennung kleiner Objekte wie Bienen auf eventuell unscharfem Bildmaterial von großem Vorteil.

Zuletzt kann YOLO durch seine Funktionsweise auch besser generalisieren als andere Algorithmen, wodurch z.B. ein mit natürlichen Bildern trainiertes Netz auch Objekte auf Kunstwerken erkennen kann. Dies unterstützt auch die Erkennungsqualität bei sich verändernder Bildqualität oder bei verschiedenen Lichtverhältnissen.

Im Detail geht YOLO dabei wie folgt vor:

Zuerst wird das Bild in ein Raster eingeteilt. Verantwortlich für die Erkennung eines Objektes ist dabei die Zelle, auf die das Zentrum des zu erkennenden Objektes fällt. Ist in einer Zelle kein Objekt, so soll die Wahrscheinlichkeit für diese Zelle 0 sein. Andernfalls soll die Wahrscheinlichkeit der Intersection Over Union zwischen der Vorhersage und dem Ground Truth entsprechen.

Jede Bounding-Box besteht dann aus fünf Vorhersagen: (x, y) als Koordinaten des Objektzentrums relativ zur Raster-Zelle, (w, h) für die Höhe und Breite der Box und der Wahrscheinlichkeit. Zum Testen der Ergebnisse werden sowohl die bestimmte Wahrscheinlichkeit als auch die Maße der bestimmten Bounding-Box mit einbezogen.

Das von YOLO verwendete Convolutional Neural Network besteht aus 24 Convolutional Layers, gefolgt von 2 Fully Connected Layers. Die Convolutional Layers des Netzes sind dabei für die Feature-Extraktion verantwortlich. Die Bestimmung der Klassenwahrscheinlichkeiten und die Bounding Boxes werden allein in den Fully Connected Layers durchgeführt. Der genaue Aufbau ist in Abb. 10.2 ersichtlich.

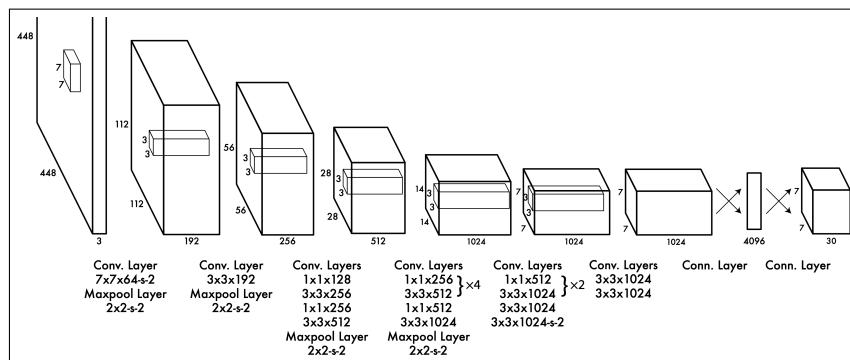


Abbildung 10.2: YOLOv1 Layers, aus [REDMON u. a. 2015, S. 3]

YOLOv1 hat durch die Eigenschaft, dass jede Zelle nur zwei Bounding-Boxes und jeweils nur eine Klasse bestimmen kann allerdings die Einschränkung, dass kleine Objekte und vor al-

lem Ansammlungen von kleinen Objekten schlechter erkannt werden. Durch die Bestimmung der Bounding-Boxes aus den Trainingsdaten im selben Schritt wie der Bestimmung der Klassenwahrscheinlichkeiten hat YOLOv1 zudem Schwierigkeiten, wenn neue oder ungewöhnliche Seitenverhältnisse verwendet werden. Zuletzt sorgt die Art, wie YOLOv1 Fehler bestimmt, dazu, dass vor allem Verortungsfehler auftreten, da ein kleiner Fehler in einer kleinen Bounding-Box wesentlich stärker gewichtet wird, als ein kleiner Fehler in einer großen Bounding-Box (vgl. [REDMON u. a. 2015, 1ff]).

YOLOv2

YOLOv2 ist das zugrundeliegende Modell für YOLO9000, der Erweiterung des originalen YOLO-Algorithmus. Der Name „YOLO9000“ stammt daher, dass die verbesserte Variante über 9000 Objektkategorien erkennen kann.

YOLOv2 hat dabei verschiedene Modellgrößen, wodurch es selbst Algorithmen wie Faster R-CNN und SSD schlägt.

Einer der wesentlichen Unterschiede im Vergleich zu YOLOv1 ist dabei, dass zum Training von YOLO9000 verschiedene Datensätze verwendet werden können. Ein Problem beim Trainieren von Objekterkennungsalgorithmen ist, dass Trainingsdatensätze sehr teuer sind, da die Bilder manuell gelabelt werden müssen. Dahingegen gibt es sehr viel mehr Datensätze, in denen die Bilder nur klassifiziert sind. YOLO9000 kann beide Arten von Datensätzen verwenden. Die Erkennungs-Datensätze werden verwendet, um die Lokalisierung der Objekte zu trainieren und die Klassifizierungsdatensätze, um den Algorithmus für mehr Objektklassen zu trainieren.

Das zugrundeliegende Modell, YOLOv2, soll dabei vor allem die Lokalisierungsfehler und den geringen Recall von YOLOv1 beheben. Um die Performance hoch zu halten, soll das verwendete neuronale Netz dabei nicht vergrößert werden.

Für die Entwicklung von YOLOv2 wurden folgende Techniken geprüft, um bei einem kleineren neuronalen Netzwerk höhere Geschwindigkeiten und bessere Erkennungsraten zu erreichen:

Batch Normalization Dabei wird die Normalisierung der Bilder, die auf jeder Convolutional Layer des neuronalen Netzes stattfindet, als Batch ausgeführt. Es wird also in jedem Durchgang ein Batch (dt. „Stapel“) an Bildern verarbeitet. Das erhöht die Erkennungspräzision um ca. 2% (vgl. auch [IOFFE und SZEGEDY 2015]).

High Resolution Classifier Durch eine höhere Auflösung der Bilder, die zum Trainieren des Classifier-Networks verwendet werden, kann das Netzwerk besser mit höheren Auflösungen umgehen. Die Präzision steigt dadurch um ca. 4 %.

Convolutional with Anchor Boxes Die Fully Connected Layers werden durch Anchor-Boxes zur Bestimmung der Bounding-Boxes ersetzt. Dadurch verliert das Modell zwar leicht an Präzision (69,5 mAP auf 69,2 mAP), gewinnt dafür allerdings an Recall (81% zu 88%) und kann mehr als 10x so viele Bounding-Boxes pro Bild bestimmen. Dadurch öffnen sich wesentlich mehr Möglichkeiten für weitere Verbesserungen.

Dimension Clusters Die Anchor-Boxes bringen beim Einsatz in YOLO noch Probleme mit sich, die mit den Dimension Clusters behoben werden, indem die Anfangsdimensionen für die Anchor-Boxes automatisch bestimmt werden.

Direct Location Prediction Durch die Verwendung von Anchor-Boxes wird das Modell, gerade in den ersten Iterationen, auch Instabil. Dies hängt mit der Berechnung der Positionen der Bounding-Boxes relativ zur Zelle zusammen, bei der die Anchor-Box überall auf dem Bild enden kann. Um das Problem zu lösen, werden die Positionen relativ zur Zelle so bestimmt, dass der Ground Truth nur zwischen 0 und 1 fallen kann, wodurch die Boxen maximal um ihre eigene Größe vom Zentrum der Zelle verschoben werden können. Zusammen mit den Dimension Clusters bedeutet das eine Verbesserung von 5% gegenüber der alleinigen Verwendung von Anchor-Boxes.

Fine-grained Features Da YOLO mit seiner normalen 13x13 Feature-Map Probleme mit dem Erkennen kleiner Objekte hat, wird hier eine 26x26 Pass-Through Feature-Map hinzugefügt. Diese 26x26x512 Feature-Map wird am Ende zu einer 13x13x2048 Feature-Map transformiert, wodurch sie mit der normalen Feature-Map konkateniert werden kann. Die Performance des Modells steigt durch diese Verbesserung um ca. 1%.

Multi-Scale Training Alle 10 Batches wird die Bildauflösung zufällig aus einem Vielfachen von 32 zwischen 320x320 und 608x608 gewählt. Dadurch wird das Netz trainiert, gut mit verschiedenen Bildauflösungen umgehen zu können.

Von diesen Techniken wurden letztendlich alle bis auf die Anchor Boxes tatsächlich in YOLOv2 verwendet (vgl. [REDMON und FARHADI 2016, S. 1–5]).

Die Geschwindigkeit und Erkennungspräzision von YOLOv2 im Vergleich zu YOLOv1 und weiteren bekannten Algorithmen sind in Abb. 10.3 ersichtlich.

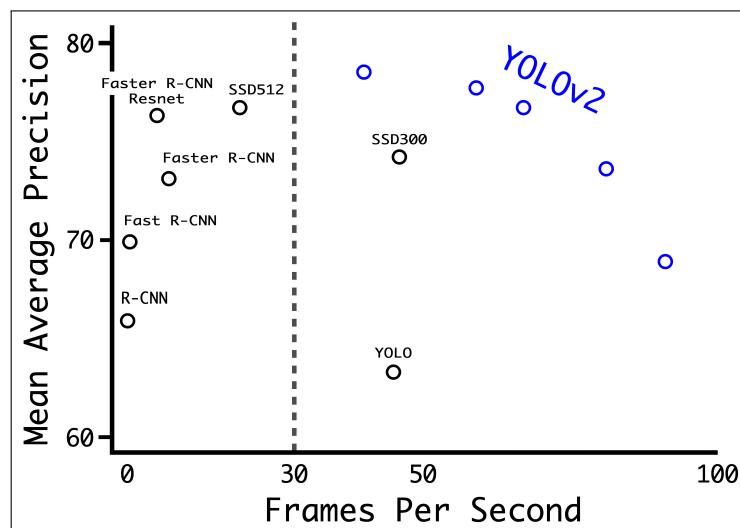


Abbildung 10.3: Vergleich von YOLOv2 mit anderen Algorithmen, aus [REDMON und FARHADI 2016, S. 4]

YOLOv3

YOLOv3 stellt gegenüber YOLOv2 nur eine marginale Verbesserung dar, baut jedoch nochmal auf dem Leistungsgewinn auf und erreicht daher z.B. bei gleicher Erkennungspräzision eine dreifach schnellere Erkennungszeit als SSD.

Die Verbesserungen sind dabei bei der Bestimmung der Bounding-Boxes sowie der Klassenwahrscheinlichkeiten. Zudem wurde der Umgang mit verschiedenen Bildauflösungen verbessert und ein verbessertes Netzwerk zur Feature Extraction implementiert (vgl. [REDMON und FARHADI 2018]).

In Abb. 10.4 ist ersichtlich, wie sich YOLOv3 hinsichtlich der Erkennungspräzision und Geschwindigkeit gegenüber vergleichbaren Algorithmen verhält.

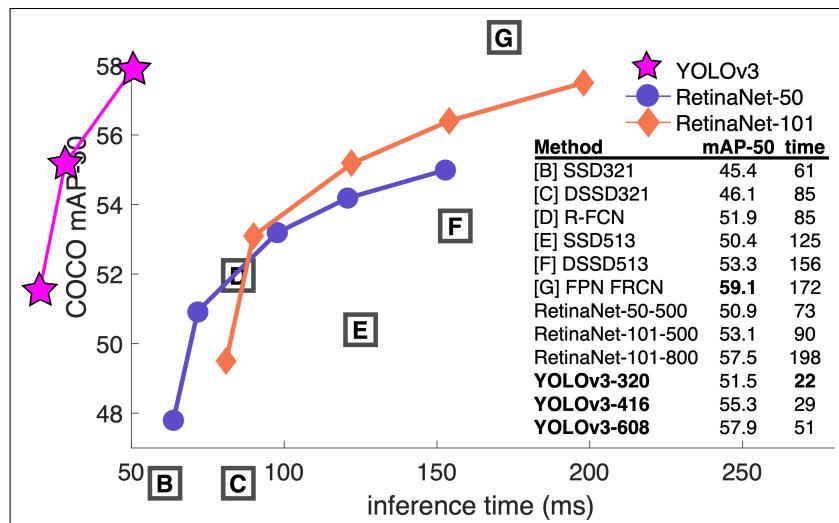


Abbildung 10.4: Vergleich von YOLOv3 mit anderen Algorithmen, aus [REDMON und FARHADI 2018, S. 4]

YOLOv4

Im Gegensatz zu den vorangehenden YOLO-Versionen wurde YOLOv4 nicht mehr von Redmon und seinem Team, sondern vom russischen Informatiker Alexey Bochkovskiy und seinem Team entwickelt.

Dabei stellt auch YOLOv4 keine wesentlichen Verbesserungen gegenüber YOLOv3 dar, sondern verbessert wiederum leicht die Performance und Erkennungsqualität mit neuen Technologien. YOLOv4 ist zudem derart entwickelt, dass es möglichst einfach ist, das Modell auf einer handelsüblichen Verbraucher-GPU wie einer NVIDIA GeForce GTX 1080 Ti oder 2080 Ti zu trainieren. Damit soll die Einstiegshürde in die Entwicklung von Anwendungen auf Basis des Algorithmus gesenkt werden.

Der Hauptfokus bei der Entwicklung liegt dabei auf der Effizienz. Der Algorithmus soll schneller und besser laufen als vergleichbare Alternativen, dabei aber auch auf schlechterer Hardware trainierbar und vor allem für die Erkennung einsetzbar sein. Dazu wird unter anderem ein Fokus auf parallele Verarbeitung gelegt. Zudem soll die optimale Balance zwischen der Eingangsauflösung, Anzahl der Convolutional Layers, Anzahl der Parameter und der Anzahl der Ausgangsschichten bzw. Filter gefunden werden.

Gegenüber YOLOv3 wird dabei eine Steigerung der mAP von 10% bei einer gleichzeitigen Steigerung der FPS um 12% erreicht. Dabei ist auch YOLOv4 in verschiedene Modellgrößen unterteilt, die jeweils auf das Einsatzgebiet angepasste Verhältnisse aus Erkennungsqualität und Geschwindigkeit bieten. Der Vergleich zu YOLOv3 und anderen aktuellen Algorithmen ist in Abb. 10.5 zu sehen (vgl. [BOCHKOVSKIY, WANG und LIAO 2020, S. 1–6]).

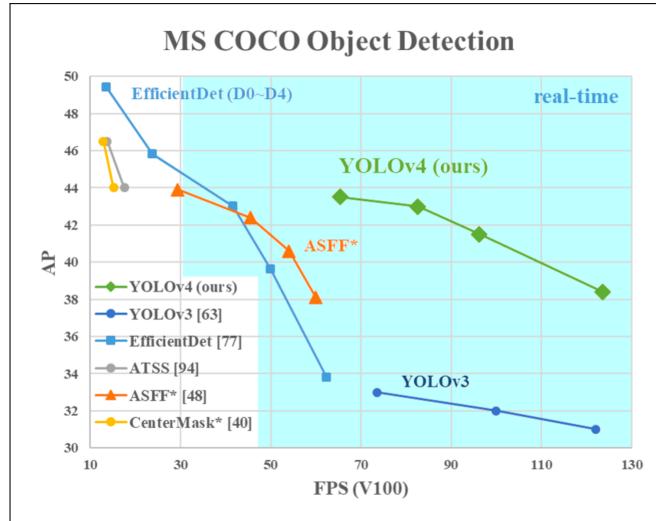


Abbildung 10.5: Vergleich von YOLOv4 mit anderen Algorithmen, aus [BOCHKOVSKIY, WANG und LIAO 2020, S. 1]

YOLOv5

Wie bereits in Abschnitt 8.2.4 erläutert, stellt YOLOv5 lediglich eine Erweiterung von YOLOv3 im Sinne einer leichteren Verwendbarkeit dar. Im Rahmen dieser Arbeit wird daher davon ausgegangen, dass die Funktionsweise identisch zu YOLOv3 ist. Im Gegensatz zu den vorangegangenen YOLO-Versionen ist YOLOv5 dabei allein auf dem Machine-Learning-Framework PyTorch aufgebaut und nicht auf anderen Frameworks wie Darknet (vgl. [MAINOLA 2021]).

10.2 Implementierungskomplexität

Durch die Portierung auf das PyTorch-Framework hat YOLOv5 gegenüber den vorangehenden Versionen einen kleinen Vorteil. Um die auf dem Darknet-Framework basierenden Vorgängerversionen zu nutzen, musste neben YOLO auch Darknet installiert und ggf. selbst kompiliert werden. Zur Verwendung von YOLOv5 müssen lediglich die benötigten Abhängigkeiten mittels eines Python-Paket-Managers wie Pip Installs Packages (pip) installiert werden.

Zur Verwendung von NVIDIA-Grafikkarten müssen in beiden Fällen CUDA-Treiber manuell auf dem System installiert werden. Da das für diese Arbeit benötigte Modell jedoch verhältnismäßig klein ist, wird dieser Schritt nur zum schnelleren Trainieren des Netzes benötigt und nicht zur Verwendung des trainierten Modells.

10.3 Begründung der Entscheidung zu YOLOv5

Die Entscheidung zu YOLOv5 als Algorithmus für das vorliegende Problem wurde primär aufgrund der unkomplizierteren Implementierung getroffen.

YOLOv4 erzielt zwar je nach Datenset etwas bessere Ergebnisse, da die Erkennung von Bienen jedoch aufgrund mehrerer Faktoren ein relativ einfaches Erkennungsproblem darstellt, spielt das für die Qualität der Ergebnisse kaum eine Rolle.

Um den entwickelten Algorithmus später problemloser einsetzen und weiterentwickeln zu können, ist es daher sinnvoll, die Abhängigkeiten möglichst gering zu halten und eine gewisse Plattformunabhängigkeit zu bewahren. Der daraus resultierende Nutzen ist größer als der Nutzen durch minimal bessere Erkennungsergebnisse.

10.4 Begründung der Entscheidung zu Norfair

Aufbauend auf YOLO als Objekterkennungsalgorithmus wird Norfair (Abschnitt 8.3.2) als Algorithmus zum Objekttracking verwendet. Norfair eignet sich besonders gut für den vorliegenden Anwendungsfall, da es sich leicht mit einem bestehenden Detektor kombinieren lässt. Der Algorithmus läuft effizienter als manche der Alternativen und eignet sich auch für die gleichzeitige Erkennung vieler Objekte in einem Bild (vgl.[TRYOLABS 2022b]).

11. Herausforderungen bei der Implementierung

11.1 Datenquellen

Ein Problem beim Training neuronaler Netze sind oftmals einseitige Datenquellen, da diese schnell zu Overfitting führen können. Der Algorithmus erkennt also Bienen im Trainingsmaterial sehr gut, kann dieses Wissen allerdings nicht auf anderes Bildmaterial übertragen.

Das Problem sind dabei nicht nur geographisch einseitige Trainingsdaten, also z.B. vom gleichen Bienenstock, sondern auch zeitlich einseitige Daten, wenn sie z.B. alle aus der selben Jahreszeit stammen.

Der Algorithmus kann bei einseitigen Datenquellen schlechter generalisieren. Es wäre zum Beispiel denkbar, dass eine Biene nur erkannt wird, wenn der Hintergrund grün ist (Rasen) und Bienen auf einem anderen Hintergrund schlechter oder gar nicht erkannt werden.

Dieses Problem lässt sich durch die Verwendung verschiedener Datenquellen für die Trainingsdaten lösen. Dabei müssen nicht zwangsläufig Videos von verschiedenen Bienenstöcken verwendet werden. Auch ein Wechsel der Kameraperspektive und Aufnahmen von verschiedenen Tageszeiten können ausreichen, um das benötigte Maß an Generalisierung zu erreichen.

11.2 Labeling der Daten

Es werden annotierte Bilddaten von Bienen benötigt, um ein neuronales Netz einer künstlichen Intelligenz für Bienen trainieren zu können. Mithilfe von Programmen wie LabelImg [TZUTALIN 2022b] werden mehrere Bilder von menschlicher Hand, wie in Abb. 11.1, annotiert. Das bedeutet, dass die Bienen auf den Bildern durch Rechtecke markiert werden. In einer YAML-Datei werden die Eckpunkte der Rechtecke anschließend gespeichert, um daraufhin zum Training der KI verwendet zu werden. Hierbei ist die Datensetze, die für das erfolgreiche Trainieren der KI notwendig ist, ausschlaggebend auf die Zeitdauer, die zum Annotieren benötigt wird. Nach eigenen Erkenntnissen werden mindestens 200-300 annotierte Bilder benötigt, sodass das trainierte Modell in der Lage ist zuverlässig Bienen zu erkennen. Hierbei hängt die Mindestanzahl von der Anzahl an Bienen pro annotiertem Bild ab, da sich somit unterschiedlich viele Informationen innerhalb der Bilder befinden. Um Overfitting zu vermeiden, wurde sich dazu entschieden Bildausschnitte von verschiedenen Tagen und Tageszeiten zum Training zu annotieren, da somit die Datenvielfalt ausgeweitet wird.



Abbildung 11.1: Labeling von Bienen, Screenshot aus LabelImg mit Ausschnitt aus Twitch-Stream von [JUNG 2021]

Je genauer die Bilder annotiert werden, desto schneller und besser kann das Modell lernen. Werden jedoch Bilder annotiert, in denen mehrere Bienen auf einem Haufen sitzen, stellt sich dies als schwer heraus. Mehrere Bienen als eine einzige Biene zu annotieren führt zu Erkennungsproblemen. Besser ist es in diesem Fall also, nur eindeutig erkennbare Bienen zu markieren.

11.3 Jahreszeit

Aufgrund des kurzen Zeitraums der Implementierungsphase, konnte nur Videomaterial zum Herbstanfang für das Training des neuronalen Netzes aus dem Twitch-Stream von [JUNG 2021] verwendet werden. Dabei befinden sich die Bienen bereits in der Vorbereitungsphase, um sich für den Winter vorzubereiten. Aufgrund der veränderten Aufgaben der Bienen verändert sich unter anderem auch das Erscheinungsbild der Bienenerbeiter. Die Arbeiterinnen tragen zum Herbst hin immer weniger Pollen mit sich, wodurch die Datengrundlage für die KI für Bienen mit Pollen gering ausfällt. Zudem ist der Schattenwurf in den verschiedenen Jahreszeiten unterschiedlich und sollte für das Training beachtet werden, da die Kontraste von Bienen im Schatten schwächer ausfallen als von Bienen, die sich im Sonnenschein befinden. Optimalerweise sollten Bilddaten zu allen Jahres- und Tageszeiten annotiert für das Training vorhanden sein, damit sich die Genauigkeit des Modells erhöht.

11.4 Bessere Datenqualität durch eigenen Bienenstock

Durch die Verwendung von Videomaterial eines Twitch-Streams entstehen weitere Einschränkungen. Abseits von schnell ersichtlichen Problemen wie der Komprimierung des Videomaterials durch die Streaming-Plattform geht dadurch auch die Möglichkeit verloren, Einfluss auf das erzeugte Videomaterial zu nehmen.

Bei Verwendung eines eigenen Bienenstocks ist es möglich, kurzfristig bestimmte Faktoren anzupassen. Es kann zum Beispiel die Kameraperspektive geändert werden, bei Bedarf auch mehrmals täglich, um einseitigen Trainingsdaten entgegenzuwirken.

Auch wäre es bei Verwendung eines eigenen Bienenstocks denkbar, verschiedene Kameras und Kamera-Einstellungen zu vergleichen. Werden zum Beispiel für optimales Tracking der Bienen mehr Bilder pro Sekunde benötigt, könnte die verwendete Kamera dahingehend angepasst werden.

11.5 Rechenleistung

11.5.1 Training des neuronalen Netzes

Die benötigte Rechenleistung zum Trainieren des neuronalen Netzes hängt von mehreren Faktoren ab:

- Auflösung des Bild-/Videomaterials
- Anzahl der Bilder, mit denen trainiert wird
- Modellgröße
- Batch-Größe
- Trainingsauflösung
- Anzahl der Epochen

Für alle dieser Faktoren gilt jeweils: Je höher der Faktor, desto höher ist die benötigte Rechenleistung. Das im Rahmen dieser Arbeit erstellte Modell zur Erkennung von Bienen wird mit einer Eingangsauflösung von 1280x720px gearbeitet. Es wird jeweils mit ca 300 Bildern trainiert. Das Grundmodell ist YOLOv5s, das zweitkleinste der 5 möglichen Modelle. Es wird mit einer Batch-Größe von 16 bei einer Trainingsauflösung von 640x360px für 300 Epochen trainiert. Das Trainieren des Modells dauert mit den beschriebenen Parametern auf einer NVIDIA RTX 3070 ca. 20 Minuten. Ein testweise ausgeführter Trainingslauf mit einer Eingangsauflösung von 3840x2160px, einer Batch-Größe von 2 und einer Trainingsauflösung von 1920x1080px dauert bei sonst gleichen Einstellungen über 4h. Die Verlustrate sinkt dabei von 1,7% auf 1,4%.

11.5.2 Anwenden des neuronalen Netzes

Die Verwendung des trainierten Modells zur Objekterkennung kann auf einer NVIDIA RTX 3070 mit ca. 140 FPS durchgeführt werden. Zur reinen Objekterkennung auf moderner Hardware eignet sich das Modell also problemlos.

Das darauf aufbauende Tracking der Bienen ist jedoch deutlich zeitaufwändiger. Hier wird auf der selben Hardware nur ein Wert von ca. 10 FPS erreicht. Damit ist der erarbeitet Algorithmus nicht für Live-Auswertungen geeignet, da für aussagekräftiges Tracking in jedem Fall Videomaterial mit mehr als 10 FPS benötigt wird.

11.6 Validierung

11.6.1 Andere Datenquellen

Um das trainierte Modell optimal auf seine Generalisierungsfähigkeiten hin zu prüfen, bietet es sich an, mit Bildmaterial aus anderen Datenquellen zu validieren. Wird dabei festgestellt, dass der durchschnittliche Fehler bei der Validierung wesentlich höher ist als der durchschnittliche Testfehler, kann das Modell nicht ausreichend generalisieren.

Diese Art der Validierung stellt allerdings einen erheblichen Mehraufwand dar, da zusätzlich zu den Trainingsdaten auch separat Daten zur Validierung gelabelt werden müssen. Um diesen Aufwand zu umgehen, kann die im nachfolgenden Abschnitt beschriebene Cross-Validation angewandt werden.

11.6.2 Cross-Validation

Aufgrund des Aufwandes, größere Mengen von Bildern zu annotieren, wird das trainierte Modell mittels Cross-Validation validiert. Das Verfahren ist in Abschnitt 7.4.5 beschrieben.

12. Bildqualität

Bei selbstlernenden und auf Bildmedien basierenden Algorithmen ist ein entscheidendes Kriterium für das erfolgreiche Erlernen von Bildstrukturen die *Bildqualität*. Faktoren, welche die Bildqualität formen sind unter anderem:

- Auflösung
- Farbkontraste
- Tiefenschärfe
- Entfernung zum Zielobjekt

Dieses Kapitel erläutert Bildbearbeitungsverfahren, welche diese Arbeit verwendet hat, um eine passende Bildqualität zum Trainieren zu erreichen. Darüber hinaus werden Kriterien eruiert, welche für ein erfolgreiches Training entscheidend sein können.

12.1 Auswirkungen schlechter Bildqualität auf das Training

Um herauszufinden, welche Faktoren der Bildqualität das trainierte Modell maßgeblich beeinträchtigen, vergleichen die folgenden Abschnitte mehrere trainierte Modelle miteinander. Dabei unterscheiden sich diese lediglich in den *verwendeten Bildvariationen* zum Trainieren.

YOLOv5 validiert die Ergebnisse einer Trainingseinheit mit den bereitgestellten und gelabelten Validierung-Bildern. Die daraus resultierenden Informationen werden innerhalb eines Box-Loss-Diagramms dargestellt. Dieses zeigt, wie hoch die Verlustrate des jeweiligen Modells bei einer bestimmten Anzahl an Iterationen ist. Hierbei gilt: Je niedriger die Verlustrate, desto besser ist das Modell. Zudem gilt auch: Je früher eine niedrige Verlustrate erreicht wird, desto effizienter lernt das Modell. In diesem Vergleich wird jedes Modell mit jeweils 300 Epochen trainiert.

12.1.1 Bildvariationen

Im Folgenden zeigt die Abb. 12.1 Bilder des Original-Modells und dessen Variationen, um die grundlegenden Unterschiede darzustellen. Diese Variationen sind in höherer Auflösung dem Anhang A.5 zu entnehmen.

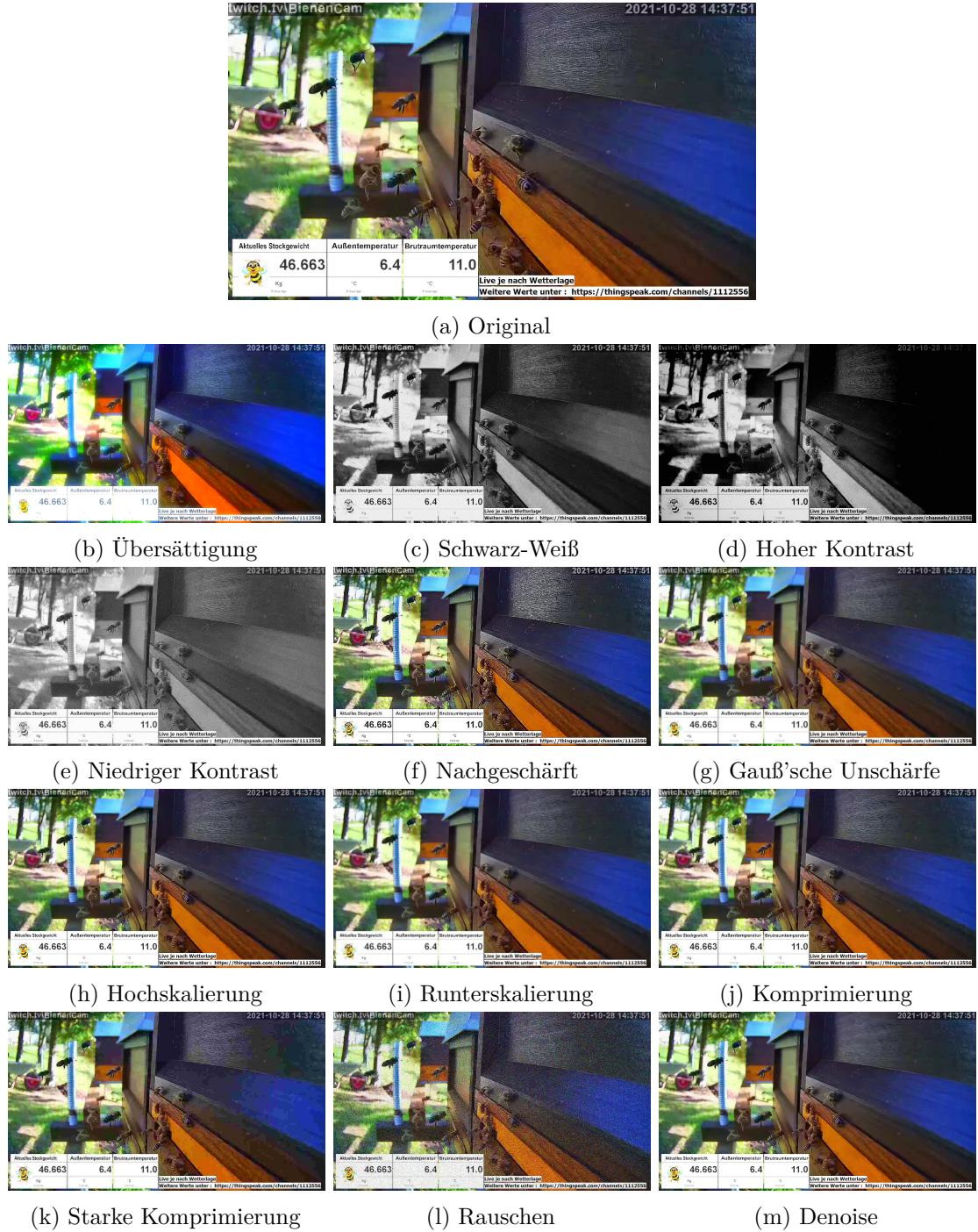


Abbildung 12.1: Vergleich der verschiedenen Trainingsdaten, Screenshot aus Twitch-Stream von [JUNG 2021] mit nachträglicher Bearbeitung

12.2 Aufnahmegeräte

Aufgrund der fehlenden Gewinnabsicht haben Freizeitimker*innen in den meisten Fällen ein recht kleines Budget. Daher sind Webcams als Basis für den Einsatz selbstlernender Algorithmen in der Imkerei vorgesehen. Diese lassen sich auch mit kleinem Budget erwerben. Übliche Auflösungen für einen Großteil von Webcams sind HD (720p) bis Full-HD (1080p). Zudem besitzen sie eine geringe Brennweite¹, da der Einsatz für Objekte direkt vor der Linse bestimmt ist. Auch der Einsatz von Action-Cams ist denkbar. Diese können oftmals mit hohen Auflösungen und Bildraten aufnehmen, dafür ist die Fokussierung und die Farbqualität oftmals schlechter. Diese beiden Faktoren fallen bei typischen Einsatzgebieten von Action-Cams nicht sonderlich ins Gewicht.

¹Die Brennweite einer Kamera bestimmt, wie weit Objekte von der Kamera entfernt sein können/müssen, um fokussiert werden zu können

12.3 Auswirkungen von Farbräumen

Um die Wichtigkeit von Farbräumen zu bestimmen, werden drei Objekterkennungsmodelle trainiert:

1. Das erste Modell wird nicht verändert und erhält unveränderte Farbdaten zu den Bildern zum Lernen → *Kontrollmodell* für die weiteren Modelle.
2. Das zweite Modell erhält durchweg *übersättigte* Bilder, um zu zeigen, ob eine Übersättigung von Bildern eine Änderung der Verlustrate mit sich bringt.
3. Das dritte Modell erhält lediglich *schwarz-weiß*-Bilder. So wird überprüft, ob Farbwerte für das Trainieren eines Bilderkennungsmodells positive oder sogar negative Faktoren sein können.

Ergebnisse

Die verschiedenen Farbräume wirken sich wie folgt auf den Trainingserfolg des Modells aus: Sowohl bei Übersättigung als auch bei Untersättigung (schwarz-weiß) steigt die Verlustrate, wie in Abb. 12.2 und Tabelle 12.1 zu sehen, um knapp 1%.

Daraus folgt, dass eine gute Abmischung der Farben im Bild Voraussetzung für einen höheren Trainingserfolg ist. Generell gilt jedoch, dass Farbinformationen ein positiver Einflussfaktor für das Trainieren einer KI sind, da von den drei verglichenen Modellen das Modell mit schwarz-weißen Bildern am schlechtesten abgeschnitten hat.

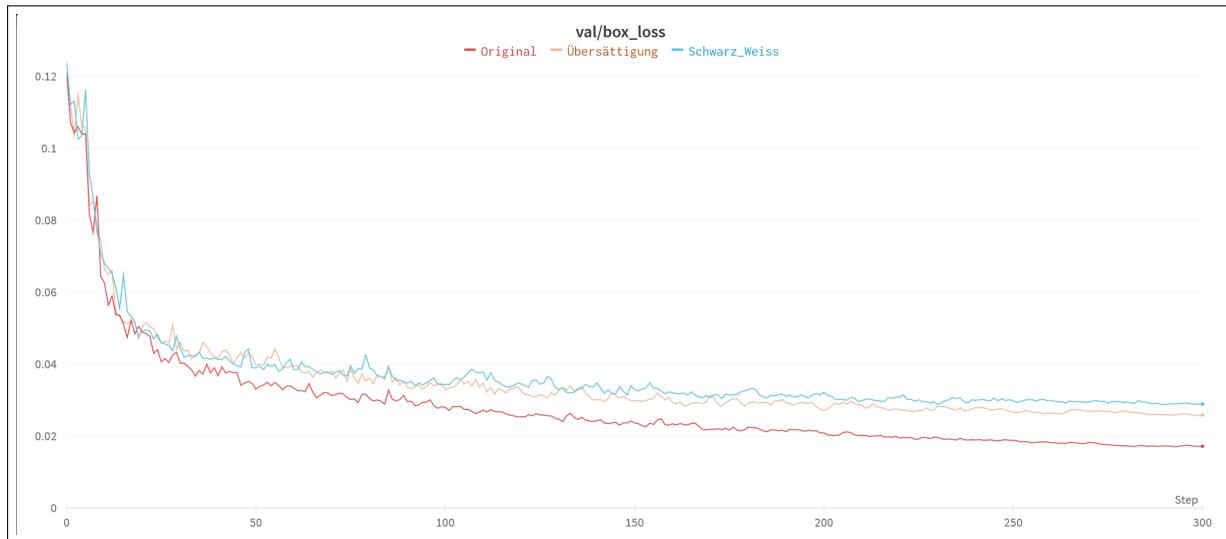


Abbildung 12.2: Auswirkungen verschiedener Farbräume auf die Verlustrate

Modell	Original	Übersättigung	Schwarz-Weiß
Verlustrate	1,7%	2,6%	2,9%

Tabelle 12.1: Auswirkungen Farbräume auf Verlustrate

12.3.1 Auswirkungen Kontraste

Der Kontrast stellt die „Differenz zwischen hellen und dunklen Flächen in einem Bild“ dar. Darüber hinaus determiniert dieser die Anzahl der Farbnuancen in einem Bild (vgl. [ADOBE 2021]).

Für diese Arbeit wird angenommen, dass der Kontrast des verwendeten Bildmaterials sich auf die Performance des Modells auswirken könnte. Um diese Annahme zu verifizieren, werden drei Modelle mit unterschiedlichen Kontrastwerten im Bildmaterial trainiert.

1. Das erste Modell erhält unveränderte Bilder
2. Beim zweiten Modell wurden die Kontrastwerte erhöht (Beispiel: Abb. 12.1d)
3. Beim dritten Modell wurden die Kontrastwerte gesenkt (Beispiel: Abb. 12.1e)

Darüber hinaus erhalten die Bilder zusätzlich einen schwarz-weiß-Filter, um einen besseren Fokus auf die Kontraste zu erreichen.

Resultate

Wenn die Kontraste eines Bildes zu schwach beziehungsweise zu stark ausfallen, kann es zu erheblichen Einbußen bei der Verlustrate kommen. Das Training mit bearbeiteten Bildern erhöht die Verlustrate zur letzten Iteration, wie in Abb. 12.3 und Tabelle 12.2 zu sehen, um mehr als 1%. Bei sehr hohen Kontrasten beträgt die Verlustrate circa 3,0%. Bei niedrigem Kontrast verhält sich die Verlustrate besser als bei reinen schwarz-weißen Bildern (Tabelle 12.1). Daraus lässt sich schließen, dass mehr Kontraste im Bild keinen positiven Einfluss auf die Verlustrate haben. Details bleiben im Bild bei niedrigerem Kontrast wahrnehmbarer, wodurch dunkle Stellen nicht zu einem schwarzen Fleck werden. Das ist relevant im Bezug auf die Bienen, da diese in einigen Bildausschnitten ohnehin schon sehr dunkel erscheinen.

Modell	Original	Hoher Kontrast	Niedriger Kontrast
Verlustrate	1,7%	3,0%	2,7%

Tabelle 12.2: Auswirkungen Kontraste auf Verlustrate

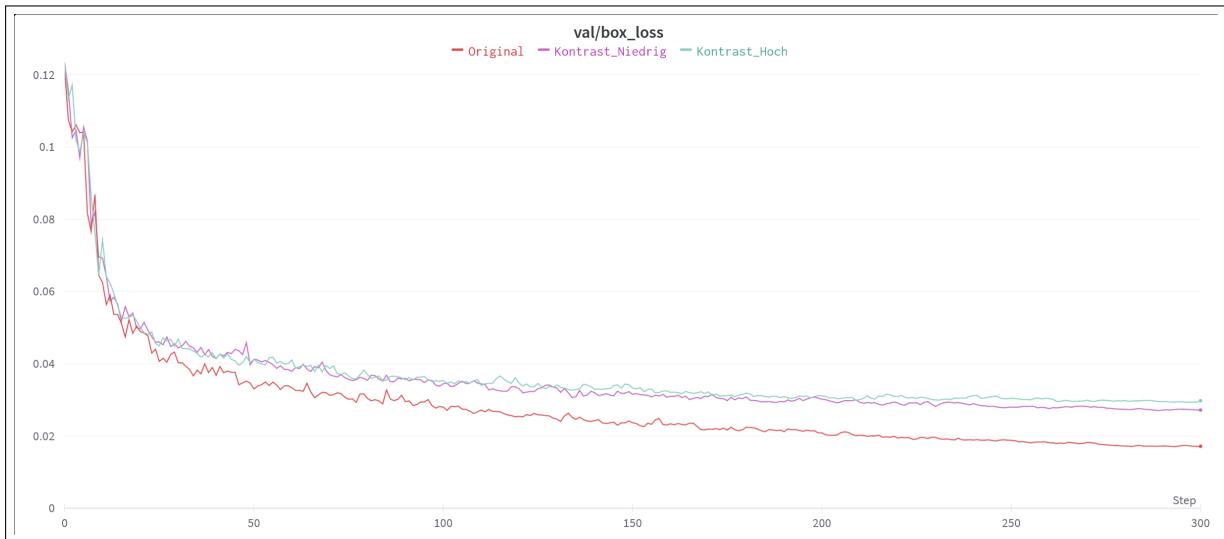


Abbildung 12.3: Auswirkungen Kontraste auf Verlustrate

12.4 Schärfe der Aufnahmen

12.4.1 Brennweite

Bei Veränderung der Brennweite ändert sich nicht nur der Bildausschnitt, sondern auch die Tiefenschärfe. Je höher die Brennweite, desto schwächer wird der Effekt der Tiefenschärfe. Beim Herauszoomen aus dem Bild wird die Tiefenschärfe wiederum prägnanter. Um einen möglichst scharfen Hintergrund beizubehalten, sollte eine Kamera mit geringer Brennweite genutzt werden.

12.4.2 Blende

Eine weit geöffnete Blende lässt mehr Licht auf den Sensor treffen. Hierdurch wird das gesamte Bild zum einen heller, andererseits wird allerdings der Hintergrund unschärfer. Das heißt, dass Tiefenschärfe bei einer größeren Blende verloren geht. Das hat zur Folge, dass nur Objekte unmittelbar um den fokussierten Bereich scharf erscheinen, während alles andere unscharf wird.

12.4.3 Fokus

Auch die Fokussierung hat Auswirkungen auf die restliche Tiefenschärfe. Wird der Fokus beispielsweise direkt auf den Eingang des Bienenstocks gesetzt, können Bienen, die sich ein Stück weiter im Hintergrund oder im Vordergrund ausgehend vom Eingang befinden auch noch scharf wahrgenommen werden. Sobald jedoch der Fokus auf einen näheren oder entfernteren Punkt gesetzt wird, erscheinen mehr Bienen unscharf. Um also eine möglichst große Anzahl an Bienen scharf aufzunehmen, sollte ein Punkt zum Fokussieren gefunden werden, um den herum sich die meisten Bienen befinden.

12.4.4 Auswirkungen Tiefenschärfe

Je nach Blickwinkel der Kamera auf den Bienenstock ist die Tiefenschärfe ein entscheidendes Kriterium. Sofern die Kamera den Stock frontal filmt und der Fokus hierbei korrekt eingestellt

ist, tritt keine Unschärfe im Hintergrund beziehungsweise Vordergrund auf, da sämtliche Bienen in der gleichen Entfernung zur Kamera stehen. Sobald jedoch der Winkel auf das Bienenvolk flacher wird, tritt der Effekt von Tiefenschärfe immer mehr in Erscheinung. Sofern der Eingang des Bienenvocks fokussiert wird, hat das zur Folge, dass nur noch Bienen direkt am Eingang scharf erscheinen. Sonstige Bienen, die entweder weiter hinten oder vorne sind, werden nicht mehr vollständig scharf dargestellt. Um die daraus resultierenden Folgen zu evaluieren, werden drei Bilderkennungsmodelle verglichen. Es gibt ein Kontrollmodell, bei dem keinerlei Änderungen vorgenommen werden. Das zweite Modell erhält zum Trainieren nur künstlich unscharfe Bilder. Diese Bilder sind die ursprünglichen Bilder mit einem Gauß'schen Weichzeichner, um eine geringe Tiefenschärfe zu simulieren. Das dritte Modell bekommt durch den Einsatz von Bildschärfungs-Algorithmen, die auf künstlicher Intelligenz basieren, schärfere Bilder. Hierdurch wird die Auswirkung einer höheren Tiefenschärfe erprobt.

Resultate

Ein falsch gesetzter Fokus hat erhebliche Auswirkungen auf den Lernerfolg, da bereits bei geringer Unschärfe im gesamten Bild die Verlustrate um circa 0,9% ansteigt. Das künstliche Nachschärfen der Bilder ist prinzipiell möglich. Dabei können jedoch auch unerwünschte Nebeneffekte entstehen. So stehen durch ein Nachschärfen Holzmaserungen und andere markante Details mehr hervor, wodurch die Erkennung von Bienen leiden kann. Dies zeigt sich auch in den aufgezeichneten Ergebnissen. Das Modell, welches ausschließlich mit den künstlich nachgeschärften Bildern trainiert hat, hatte eine 0,3% höhere Verlustrate, wie in Abb. 12.4 und Tabelle 12.3 zu sehen. Um also eine möglichst hohe Lernrate zu erreichen, sollte der Fokus auf den Eingang des Bienenvocks ausgerichtet sein, da sich hier die meisten Bienen befinden. Zusätzlich sollte die Tiefenschärfe hoch sein, indem eine relativ kleine Blende verwendet wird. Dadurch erscheinen auch Bienen, die sich nicht direkt im Fokus befinden, relativ scharf.

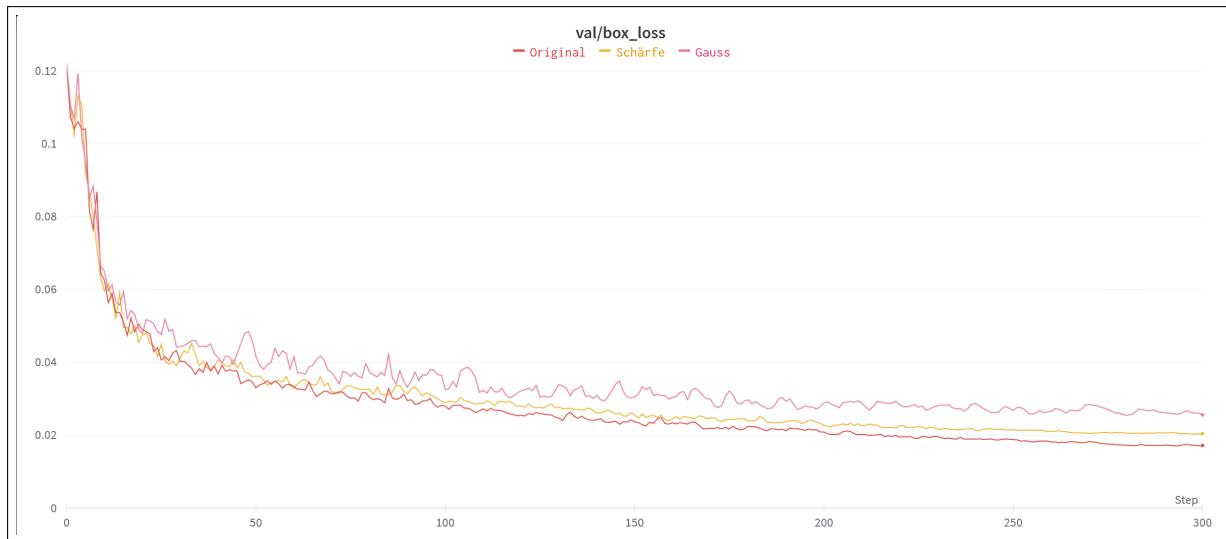


Abbildung 12.4: Auswirkungen Tiefenschärfe auf Verlustrate

Modell	Original	Nachgeschärft	Gauß'sche Unschärfe
Verlustrate	1,7%	2,0%	2,6%

Tabelle 12.3: Auswirkungen Tiefenschärfe auf Verlustrate

12.5 Auswirkungen Auflösung

Aufnahmen mit einer Webcam besitzen im Vergleich zu Vollformat-Kameras meistens eine geringere Auflösung. Die Auswirkungen einer niedrigeren Auflösung sollen auch beurteilt werden. Hierzu werden drei Modelle trainiert, wobei eines wieder als Kontrollmodell eingesetzt wird. Das zweite Modell basiert auf Bildern mit einer niedrigeren Auflösung, während das dritte Modell durch KI-Hochskalierung auf Bildern mit höherer Auflösung basiert.

Resultate

Eine hohe Auflösung ermöglicht im Allgemeinen eine größere Informationsmenge in Bildern. Bei Bildern mit einer sehr geringen Auflösung wie 360p² schafft es das Modell nicht, die Verlustrate unter 3,0% zu halten, wie in Abb. 12.5 und Tabelle 12.4 zu sehen. Bei höheren Auflösungen verbessert sich die Lernrate rapide.

Hierbei gilt jedoch keine lineare Relation zwischen Auflösung und der Senkung der Verlustrate. Die Anzahl der Pixel im Bild beträgt bei 720p die vierfache Anzahl der Pixel von 360p. Die Verbesserung des Modells beträgt hierbei circa 45%. 2160p hat neunmal so viel Pixel wie 720p. Allerdings liegt die Verbesserung hier nur bei knapp 18%. Aufgrund des exponentiellen Anstiegs der Trainingsdauer lässt sich schließen, dass eine höhere Auflösung nicht unbedingt rentabel ist, da die Verbesserung der Verlustrate stetig kleiner wird.

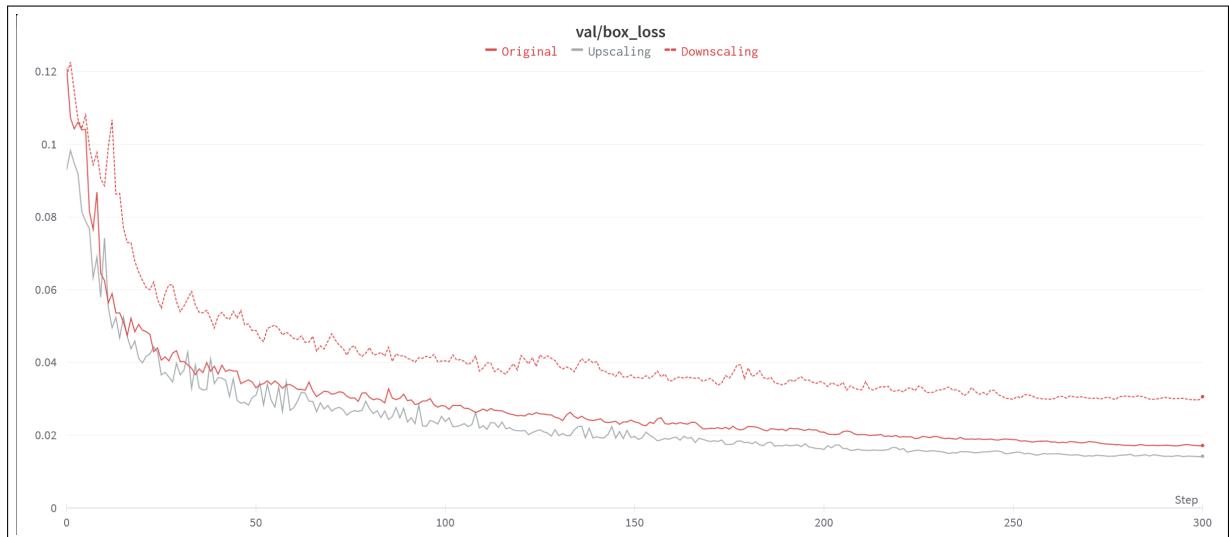


Abbildung 12.5: Auswirkungen Auflösung auf Verlustrate

²Die p-Schreibweise geht von einem 16:9-Format aus und gibt die Anzahl der Pixel auf der y-Achse an. 360p steht also für eine Auflösung von 640x360, 1080p stehen für 1920x1080 usw.

Modell	Original	Hochskaliert	Runterskaliert
Auflösung	1280x720p	3840x2160p	640x360p
Trainingsdauer	20m	4h 20m	15m
Verlustrate	1,7%	1,4%	3,1%

Tabelle 12.4: Auswirkungen Auflösung auf Verlustrate

12.6 Auswirkungen Komprimierung

Eine mögliche Datenquelle für die Trainingsdaten ist die Video-Streaming-Plattform „Twitch“. Auf dieser Webseite können Benutzer*innen Videos in Echtzeit ausstrahlen. Darunter befinden sich auch einige Benutzer*innen, die ihre Bienenstöcke mit einer Kamera aufnehmen und Live auf Twitch hochladen (vgl. [TWITCH 2022]).

Sofern das Einverständnis der*des jeweiligen Video-Eigentümer*in vorliegt, können die bereits ausgestrahlten Videoaufnahmen von Twitch heruntergeladen und für die Verarbeitung zu Trainingsdaten vorbereitet werden.

Um die Bandbreiten-Nutzung möglichst gering zu halten, werden die Streams vor dem Upload komprimiert. Dadurch wird auch sichergestellt, dass Streams über einen Internetanschluss mit geringerer Bandbreite flüssig laufen. Hierfür wird jedoch ein verlustbehaftetes Kompressionsverfahren verwendet. Die Auswirkungen dieser verlustbehafteten Komprimierung sollen im weiteren Verlauf betrachtet werden. Um die Auswirkungen zu evaluieren, sind drei trainierte Modelle notwendig.

1. Ein Modell erhält unkomprimierte Daten
2. Das zweite Modell schwach (90% der ursprünglichen Qualität)
3. Das dritte Modell stark (10% der ursprünglichen Qualität) komprimierte Daten zum Trainieren bekommt

Resultate

Eine geringe Kompression von knapp 10% führt zu keinen nennenswerten Unterschieden in der Verlustrate durch das Training hinweg. Bei einer starken Komprimierung steigt die Verlustrate nur um circa 0,2% an, wie in Abb. 12.6 und Tabelle 12.5 zu sehen. Somit ist es also auch in der Praxis möglich, bereits komprimierte Bilder zum Training eines KI-Modells zu verwenden.

Modell	Original	Komprimierung (90%)	Starke Komprimierung (10%)
Verlustrate	1,7%	1,7%	1,9%

Tabelle 12.5: Auswirkungen Komprimierung auf Verlustrate

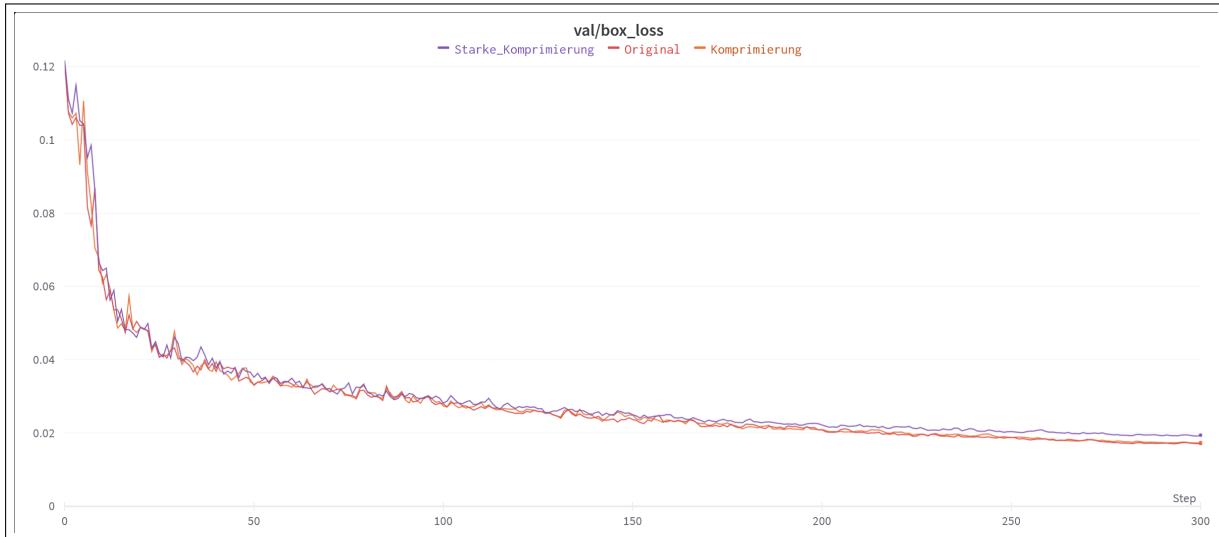


Abbildung 12.6: Auswirkungen Komprimierung auf Verlustrate

12.7 Blickwinkel

Der Blickwinkel auf den Bienenstock kann Änderungen am Erfolg des KI-Modells herbeiführen, da verschiedene Blickwinkel unterschiedliche Schwerpunkte setzen. Diese Auswirkungen werden in Tabelle 12.6 dargelegt.

Blickwinkel	Vorteile	Nachteile
Frontal (90 Grad)	Bienen, die sich auf der Außenseite des Bienenstocks befinden, können gut erkannt und verfolgt werden	Anfliegende und abfliegende Bienen können meist nur von Vorne oder Hinten betrachtet werden
Seitlich (0 Grad)	Fliegende Bienen können besser erkannt und verfolgt werden	Bienen auf der Außenseite des Bienenstocks können schlechter verfolgt werden, da nur Auf- und Ab-Bewegungen erkennbar sind.
Schräg (45 Grad)	Sowohl fliegende Bienen als auch Bienen, die sich auf der Außenseite des Bienenstocks befinden können gut erkannt und verfolgt werden.	-

Tabelle 12.6: Auswirkungen Blickwinkel zum Bienenstock

12.8 Lichtverhältnisse

Die Lichtverhältnisse der Bilder verändern sich stetig aufgrund zweierlei Faktoren. Der erste Faktor ist die Tageszeit, da sich der Sonneneinstrahlungswinkel kontinuierlich ändert. Der zweite Faktor ist aber auch die Jahreszeit, da die Sonne die Erde zu verschiedenen Jahreszeiten in einem unterschiedlichen Winkel trifft.

12.8.1 Tageszeit

Um eine möglichst hohe Bildqualität zu erreichen, sollten die Trainingsdaten nicht nur zu einer bestimmten Tageszeit aufgenommen werden. Dies könnte ein Overfitting auf die Lichtverhältnisse zu dieser Tageszeit zur Folge haben. Daraus resultiert wiederum, dass andere Tageszeiten vom Modell eventuell schlechter verarbeitet werden können. Das Modell erkennt in diesem Fall beispielsweise nur Bienen, die von rechts beleuchtet werden und keine Bienen, die von links beleuchtet werden.

12.8.2 Jahreszeit

Jahreszeiten können auch Einfluss auf den Erfolg des Modells haben, da der Einfallswinkel die Länge der Schatten bestimmt. Im Sommer ist der Sonneneinstrahlungswinkel steiler als im Winter. Das bedeutet, dass Schatten im Sommer generell kürzer ausfallen als im Winter. Um Erkennungsprobleme der künstlichen Intelligenz auszuschließen, sollten von allen Jahreszeiten Bildmaterialien vorhanden sein, von denen später Videomaterial mit diesem Modell ausgewertet werden soll.

Die Unterschiede der Lichtverhältnisse zwischen den verschiedenen Jahreszeiten sind dabei jedoch begrenzt. Der Einsatz eines speziell auf die jeweils aktuelle Jahreszeit hin trainierten Modells erscheint daher wenig sinnvoll. Wird ein einziges Modell trainiert, das mit jeder Jahreszeit umgehen kann, ergeben sich jedoch weitere Vorteile. So kann ein solches Modell meist besser generalisieren, wodurch die Genauigkeit insgesamt steigt.

12.8.3 Helligkeit erhöhen

Die Helligkeit kann zum Einen durch eine größere Blendenöffnung, aber auch durch das Erhöhen des ISO-Wertes erreicht werden. Bei der ersten Möglichkeit geht durch die größere Blende Tiefenschärfe verloren, weshalb es zu weiteren Problemen kommen kann. Eine Erhöhung des ISO-Wertes kann wiederum ein stärkeres Bildrauschen zur Folge haben.

12.8.4 Auswirkungen Bildrauschen

Um die Auswirkungen von Bildrauschen zu evaluieren, werden drei Modelle verglichen. Darunter ist ein Modell, welches unverändert bleibt. Das zweite Modell wird mit Bildern trainiert, die ein höheres Bildrauschen aufweisen. Das letzte Modell basiert auf Bildern, die durch einen Algorithmus zur Entfernung von Bildrauschen bearbeitet wurden.

Resultate

Werden dem KI-Modell zum Training nur Bilddaten gegeben, welche ein erhöhtes Aufkommen von Bildrauschen haben, kann dies die Verlustrate deutlich erhöhen. Hierbei hat die Verlustrate um knapp 0,8% zugenommen, wie in Abb. 12.7 und Tabelle 12.7 zu sehen. Sofern die Bilder nicht ohnehin schon ein erhöhtes Bildrauschen haben, führt ein Denoise-Algorithmus nicht zu einem besser trainiertem Modell. In diesem Fall haben sowohl das Modell mit den originalen Bildern als auch das Modell mit den entrauschten Bildern beinahe gleich gut abgeschnitten. Aus diesen Erkenntnissen lässt sich schließen, dass bei dunkleren Umgebungen der ISO-Wert

nicht unbedingt im größeren Maße erhöht werden sollte, da dadurch ein stärkeres Bildrauschen auftritt. Stattdessen sollte ein Abwägung zwischen Erhöhung des ISO-Wertes und einer größeren Blendenöffnung getroffen werden. Eine Kombination aus beidem lässt das Bild ausreichend scharf sein, begrenzt dabei aber auch das Bildrauschen.

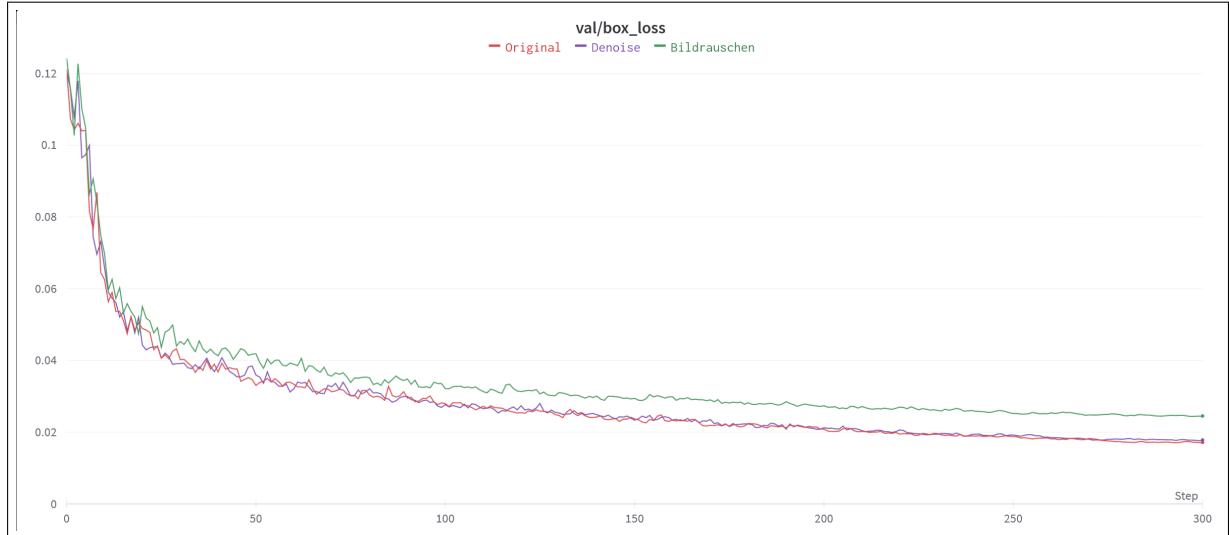


Abbildung 12.7: Auswirkungen Bildrauschen auf Verlustrate

Modell	Original	Erhöhtes Bildrauschen	Denoise
Verlustrate	1,7%	2,5%	1,8%

Tabelle 12.7: Auswirkungen Bildrauschen auf Verlustrate

12.9 Vergleich aller Aspekte der Bildqualität

12.9.1 Auswertung der Graphen

Im weiteren Verlauf werden Graphen zu allen Aspekten der Bildqualität gezeigt, jedoch nur die Aspekte hin ausführlich miteinander verglichen, die keine negativen Auswirkungen auf den Box-Loss-Wert haben. Dazu gehören die Modelle, die auf den originalen (im Folgenden als „originale Modell“ bezeichnet), hochskalierten (i.F. als „hochskaliertes Modell“ bz.) oder komprimierten (i.F. als „komprimiertes Modell“ bz.) Bildern basieren.

Vergleich der Verlustrate

Festzustellen ist, wie in Tabelle 12.8 aufgelistet, dass sowohl das originale als auch das komprimierte Modell eine ähnlich abfallende Verlustrate aufweisen und die finale Verlustrate gleich ist. Das hochskalierte Modell erreicht dazu im Vergleich bereits nach 200 Epochen eine bessere Verlustrate als die beiden anderen Modelle. Die finale Verlustrate ist hierbei mit 1,4% um ungefähr 0,3% besser.

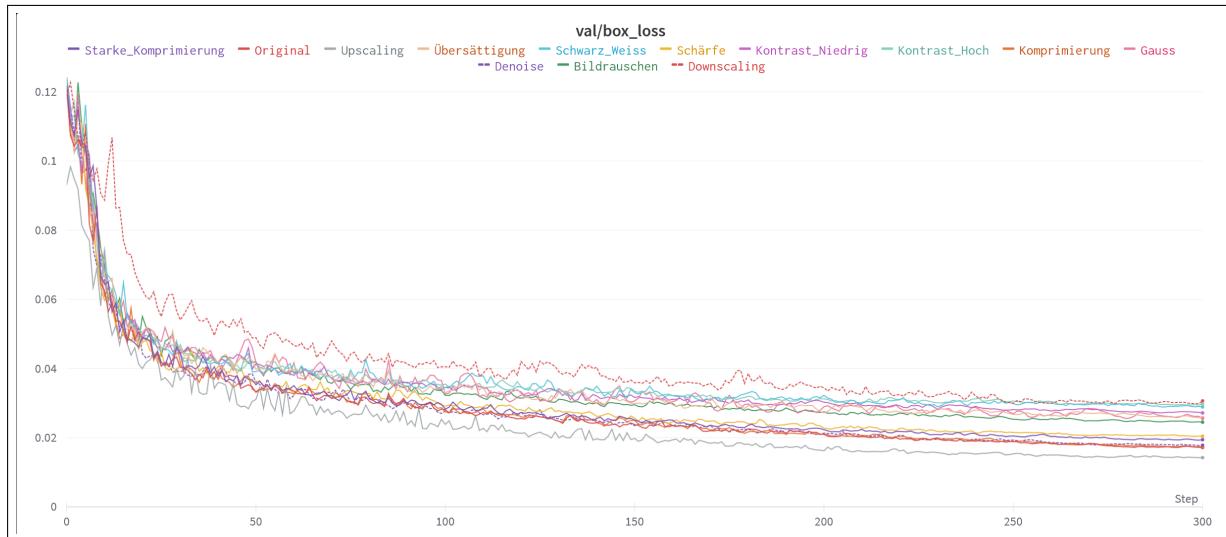


Abbildung 12.8: Vergleich aller Aspekte auf Verlustrate

Epoche	Original	Hochskaliert	Kompression
50	3,3%	3,1%	3,3%
100	2,8%	2,3%	2,7%
150	2,4%	2,0%	2,5%
200	2,1%	1,6%	2,1%
250	1,9%	1,5%	1,9%
300	1,7%	1,4%	1,7%

Tabelle 12.8: Vergleich der Verlustrate

Vergleich der Präzision

Trotz ähnlicher Verlustrate kann das originale Modell schneller an Präzision gewinnen als das komprimierte Modell. Nach bereits 250 Epochen schlägt das originale Modell das Komprimierte um knapp 0,3% an Präzision. Unerwartet schneidet das hochskalierte Modell im Vergleich zu beiden anderen Modellen um circa 1,0% schlechter ab. Eine mögliche Begründung dieses Verhaltens ist, dass das hochskalierte Modell mehr Bienen erkennt als im manuell annotiert wurden. Somit könnten Bienen erkannt werden, die nicht annotiert sind. Obwohl es sich also um tatsächliche Bienen handelt, erhält das Modell hierbei einer höhere Fehlerrate. Solche Fehler in den Trainingsdaten lassen sich jedoch nicht ganz vermeiden (siehe Abschnitt 11.2).

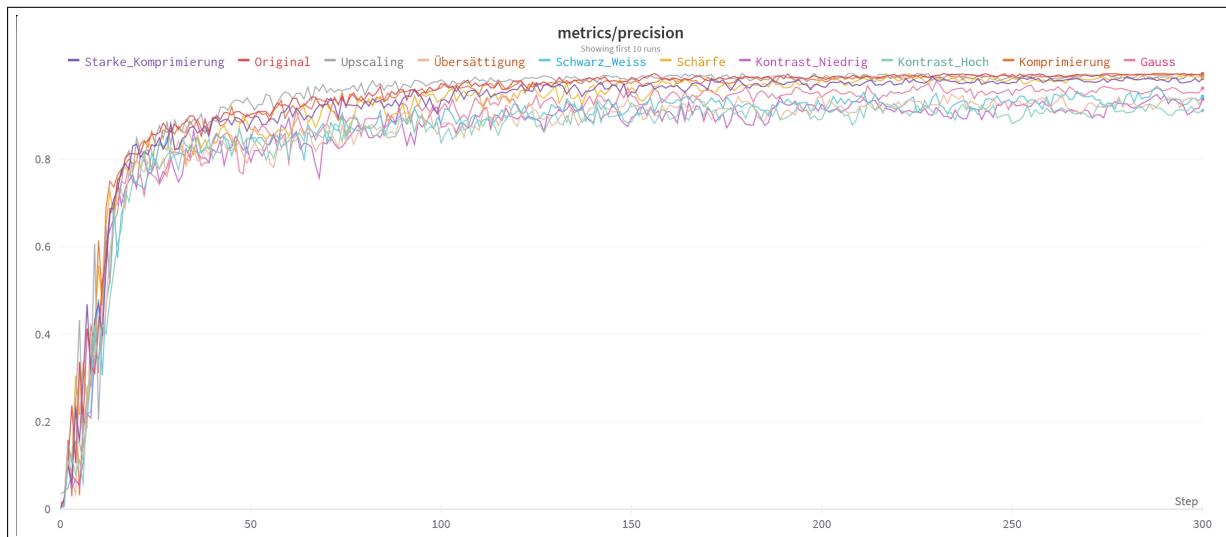


Abbildung 12.9: Vergleich aller Aspekte auf Präzision

Epoche	Original	Hochskaliert	Kompression
50	90,4%	92,8%	89,5%
100	97,4%	98,2%	96,4%
150	98,8%	98,4%	97,5%
200	98,7%	98,6%	98,2%
250	99,5%	98,9%	99,0%
300	99,6%	98,6%	99,3%

Tabelle 12.9: Vergleich der Präzision

Vergleich der Trefferquote

Auch in Bezug auf die Trefferquote verhalten sich das originale und das komprimierte Modell sehr ähnlich. Entgegen den Erwartungen schneidet das hochskalierte Modell ungefähr 3,0% schlechter als die anderen Modelle ab. Das heißt wiederum, dass die KI trotz Hochskalierung nicht alle zuvor annotierten Bienen erkennt.

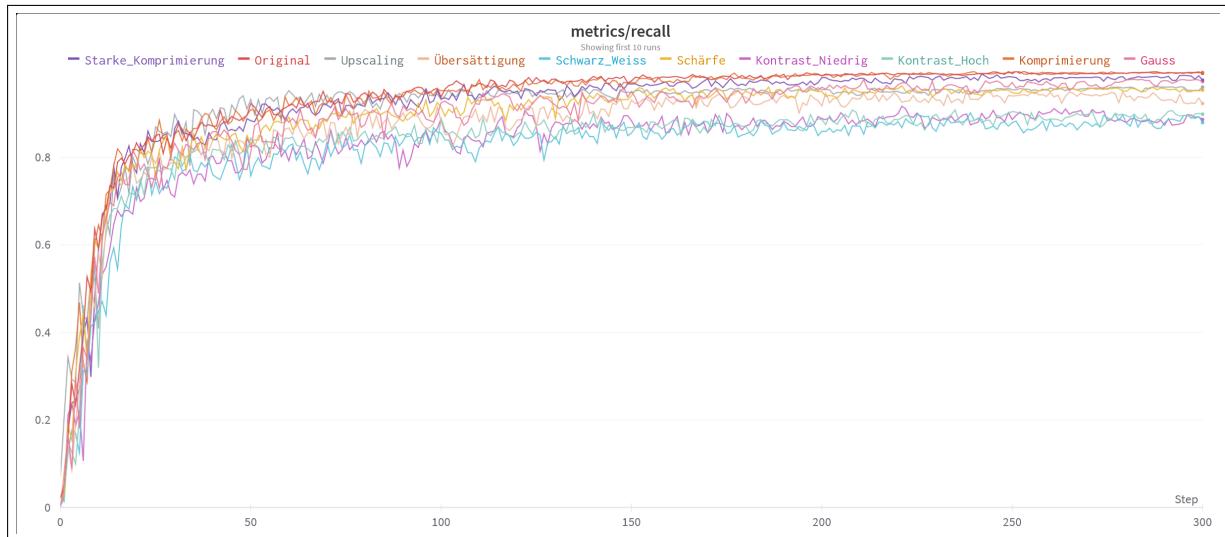


Abbildung 12.10: Vergleich aller Aspekte auf Trefferquote

Epoche	Original	Hochskaliert	Kompression
50	92,3%	93,3%	90,9%
100	93,8%	93,2%	95,2%
150	97,5%	94,8%	97,1%
200	98,6%	95,4%	98,6%
250	99,1%	95,4%	99,5%
300	99,2%	96,0%	99,4%

Tabelle 12.10: Vergleich der Trefferquote

12.10 Fazit

Aus der abschließenden Betrachtung der Aspekte, die die Bildqualität beeinflussen, lässt sich folgern, dass ein höherer Detailgrad durch eine höhere Auflösung das Modell erheblich verbessern kann. Das lässt sich unter anderem dadurch erklären, dass bei einer höheren Auflösung insbesondere kleinere Objekte innerhalb des Bildes klarer voneinander getrennt wahrgenommen werden können.

Durch Makel wie farblose/übersättigte, niedrig-auflösende oder übermäßig rauschende Bilder verschlechtert sich die Verlustrate stark. Den Erkenntnissen zufolge kann schlechte Bildqualität künstlich kaum verbessert werden. Es ist allerdings möglich, die Auflösung durch KI-Hochskalierung zu erhöhen, um einen höheren Detailgrad der Bilder zu erreichen.

Entgegen der Erwartung beeinflusst selbst eine starke Komprimierung der Bilder die Qualität des KI-Modells kaum. Allerdings entstehen durch eine stärkere Komprimierung unter anderem mehr Artefakte im Bild. Demnach kann davon ausgegangen werden, dass Bienen aufgrund dieser Artefakte schlechter vom Hintergrund zu unterscheiden sind.

13. Videoqualität

13.1 Bildwiederholungsrate

Um mit Hilfe eines qualitativ hochwertigen Bienen-Erkennungsmodells Bienen präzise im Video erkennen und verfolgen zu können, ist die Bildwiederholungsrate ein ausschlaggebendes Kriterium. Bienen sind je nach gewähltem Bildausschnitt relativ klein im Vergleich zum Rest der Videokulisse. Daher ist es schwer, Bienen im Flug mit hoher Genauigkeit verfolgen zu können. Auch können abhängig von der Bildwiederholungsfrequenz unterschiedlich große Sprünge zwischen den Positionen der Bienen über Bilder hinweg auftreten. Bienen, die um den Bienenstock herum krabbeln, können dahingegen besser verfolgt werden, da die Bewegungsgeschwindigkeit im Vergleich zur Bildwiederholrate kleiner ist als im Flug. So muss je nach Zielsetzung entschieden werden, wie viele Bilder pro Sekunde für die Auswertung nötig sind. Die entscheidende Frage dabei ist, ob Bienen im Flug oder während des Krabbelns zuverlässig verfolgt werden müssen. Nach eigenen Erkenntnissen können krabbelnde Bienen bereits mit 24 Bildern in der Sekunde erfolgreich verfolgt werden, wobei fliegende Bienen eher mit 60 oder mehr Bildern in der Sekunde aufgenommen werden sollten, um brauchbare Ergebnisse zu ermitteln.

13.1.1 Erhöhung der Bildwiederholungsrate durch Bewegungsinterpolation

Durch Bewegungsinterpolation können Videos, die mit 24 FPS aufgenommen wurden, künstlich auf 60 FPS oder mehr erweitert werden. Dabei werden jeweils zwei aufeinanderfolgende Bilder interpoliert, sodass sich-im-Bild-bewegende Objekte im interpolierten Bild auf halber Strecke projiziert werden (siehe beispielhaft Abb. 13.1). Das bedeutet, dass die Position des jeweiligen Objekts im ersten und zweiten Bild bestimmt und dessen Mittelwert errechnet wird. Dieser Mittelwert ist dann die neue Position für das interpolierte Bild.

Nachteile von Bewegungsinterpolation

Bewegungsinterpolation funktioniert dann sehr gut, wenn sich Objekte kontrolliert und einigermaßen gleichmäßig im Video bewegen. Dadurch können die Objekte selbst gut vom Hintergrund getrennt werden und deren Bewegung besser vorhergesagt werden, als wenn sich die Objekte unkoordiniert und rasant bewegen (vgl. [WIKIPEDIA 2018]). Im Falle der Videoaufnahmen von Bienen führt Interpolation selten zum gewünschten Ergebnis, da sich Bienen vor allem im Flug sehr schnell von bewegen.



Abbildung 13.1: Funktionsweise von Interpolation, bearbeitete Screenshots aus Twitch-Stream von [JUNG 2021]

14. Auswertung der Daten

14.1 Wie ist die Auswertung umgesetzt?

Um die ermittelten Daten auswerten zu können, werden während des Trackings der Bienen Informationen bei jedem Video-Einzelbild gespeichert. In diesem Prozess wird jede Biene einzeln unterschieden und bewertet. Beispielsweise wird bestimmt, ob sich die jeweilige Biene außerhalb des sichtbaren Bildausschnitts bewegt und anhand dessen in Kombination mit der Flugrichtung bestimmt, ob sich die Biene vom Bienenstock entfernt oder auf diesen zugeflogen ist. Die Flugrichtung wird hierbei durch den Startpunkt, an dem die Biene das erste Mal durch das Tracking erkannt wird, und dem Endpunkt, an dem die Biene zum letzten Mal vom Tracking erkannt wird, ermittelt. Vor der Analyse wird durch eine Eingabe der Benutzer*innen angegeben, wo sich der Bienenstock innerhalb des Videos befindet. Durch die Start- und Endkoordinaten kann dann ein Flugwinkel bestimmt werden. Dieser Winkel wird mit der Position des Bienenstocks im Bild verglichen. Da die Biene nur zum Bienenstock oder vom Bienenstock weg fliegen kann, werden beiden Richtungen 180° zugewiesen. Die Richtungsbestimmung erfolgt, indem geprüft wird, in welchem der beiden Winkelintervalle der Flugwinkel liegt. Im Anschluss an diese Analyse werden die Tracking-Objekte der Bienen wieder konvertiert, um die Anzahl der an-/abfliegenden Bienen pro Bild herauszufinden. Diese finalen Daten werden daraufhin in einer Datenbank gespeichert. Dadurch bleiben die Daten abrufbar und die Analyse muss nur einmal pro Video ausgeführt werden.

14.1.1 Optimierung der Trackingdaten

Da sich einige Bienen nur wenig um den Eingang des Bienenstocks herumbewegen, können diese, um die Datenqualität zu verbessern, herausgefiltert werden. Dazu wird die Differenz der Startkoordinaten und Endkoordinaten einer jeweiligen Biene berechnet. Dem Programm kann ein relativ zur Bildgröße bestimmter Schwellwert übergeben werden. Unterschreitet eine Biene diesen Schwellwert, erhält sie den Status „NO_MOVEMENT“. Bienen mit diesem Status werden bei weiteren Berechnungen nicht weiter beachtet, wodurch die An-/Abflugrate präzisiert wird. Bienen, die nur aus dem Bienenstock herauskommen und wieder hineingehen, werden dadurch nicht mit eingerechnet. Auch stationäre Objekte, die fälschlicherweise als Biene erkannt werden, verfälschen so nicht die Ergebnisse der Auswertung.

14.1.2 Speicherung in Datenbank

Um die zu einem Video zugehörigen Daten in einer Datenbank zu speichern, müssen Informationen (siehe Tabelle 14.1) zu jedem Einzelbild im Video gespeichert werden.

Spaltenname	Beschreibung	Datentyp	Speichergröße
data_id	Primärschlüssel auf Datensatz	INTEGER	4 Bytes
hive_id	Fremdschlüssel auf Bienenstock	INTEGER	4 Bytes
video_id	Fremdschlüssel auf Videoinformationen	INTEGER	4 Bytes
timestamp	Datum	TIMESTAMP	8 Bytes
approaches_count	Anzahl der anfliegenden Bienen	INTEGER	4 Bytes
departures_count	Anzahl der abfliegenden Bienen	INTEGER	4 Bytes
no_movement_count	Anzahl der nicht-aktiven Bienen	INTEGER	4 Bytes
Gesamt			32 Bytes

Tabelle 14.1: Benötigte Spalten zur Speicherung in einer Datenbank, mit Informationen aus [ORACLE CORPORATION o. D.]

Pro Tabelleneintrag werden 32 Byte ohne weitere Komprimierung benötigt. Mithilfe dieser Information kann für Videos mit 60 Bildern in der Sekunde und verschiedenen Videolängen der benötigte Speicherplatz berechnet werden (siehe Tabelle 14.2).

Videolänge	Speichergröße in Megabyte
1 Minute	0,12
1 Stunde	6,91
1 Tag	165,89
1 Woche	1161,22
4 Wochen	4644,86
1 Jahr	60549,12

Tabelle 14.2: Benötigte Speicherkapazität für verschiedene Videolängen und 60-FPS, mit Informationen aus [ORACLE CORPORATION o. D.]

Da eine ausführliche Analyse des Bienenstocks nur bei kontinuierlicher Überwachung des selbigen möglich ist, ist von einer Aufnahmedauer mehrerer Kalenderjahre auszugehen. Jährlich entstehen so bei einem 60-FPS Video ungefähr 60,55 Gigabyte an unkomprimierten Daten in der Datenbank, da sekündlich 60 Datensätze mit jeweils 32 Byte gespeichert werden. Bei unterschiedlichen Bildwiederholungsraten fallen unterschiedliche Datenmengen an. Daher kann das Speicherintervall beispielsweise auch auf eine Sekunde oder mehr erhöht werden, sodass die Daten pro Sekunde aggregiert und statt 60,55 nur circa 1,01 Gigabyte benötigt werden. Dadurch ist die benötigte Speicherkapazität unabhängig von der Bildwiederholungsrate. Dieser Schritt empfiehlt sich gerade bei der Verwendung von Videomaterial mit hoher Wiederholungsrate. Der tatsächliche Informationsverlust bei dieser Vorgehensweise ist minimal, gleichzeitig ist die entstehende Datenmenge jedoch um ein vielfaches kleiner.

15. Visualisierung der Daten

15.1 Möglichkeiten zur Visualisierung

Um die gespeicherten Daten zu visualisieren ergeben sich verschiedene Möglichkeiten. Zum Einen könnte eine eigene Web-Applikation geschrieben werden, in der die gesammelten Daten aufgelistet oder in Form von Graphen dargestellt werden. Aufgrund des damit einhergehenden Aufwandes besteht alternativ die Möglichkeit, eine Datenanalyse-Plattform wie Grafana einzusetzen. Mithilfe einer solchen Plattform können Daten aus unterschiedlichsten Quellen visualisiert werden. Der Overhead für die Entwicklung einer ganzen Website mitsamt der Module zur Erstellung von Graphen etc. entfällt dabei. Dadurch kann die Arbeitszeit in die Entwicklung aussagekräftiger Analysen gesteckt werden.

15.1.1 Grafana

In Abb. 15.1 sind die durch das KI-Modell ermittelten Daten durch Grafana in 15 Sekunden-Zeitintervallen visualisiert. Das wäre eine Möglichkeit, um das Potenzial des trainierten Modells zu nutzen. Hierbei kann erkannt werden, ob mehr Bienen an- oder abfliegen. Befindet sich der blaue Graph unterhalb der x-Achse ($y=0$), fliegen mehr Bienen vom Bienestock weg als heran. Bei langfristiger Beobachtung des Bienestocks kann darüber hinaus eine Analyse über die täglichen Differenzen der An- und Abflugraten erstellt werden. Dadurch lässt sich feststellen, ob sich die Größe des Bienenvolkes verringert oder dieses eventuell umzieht.

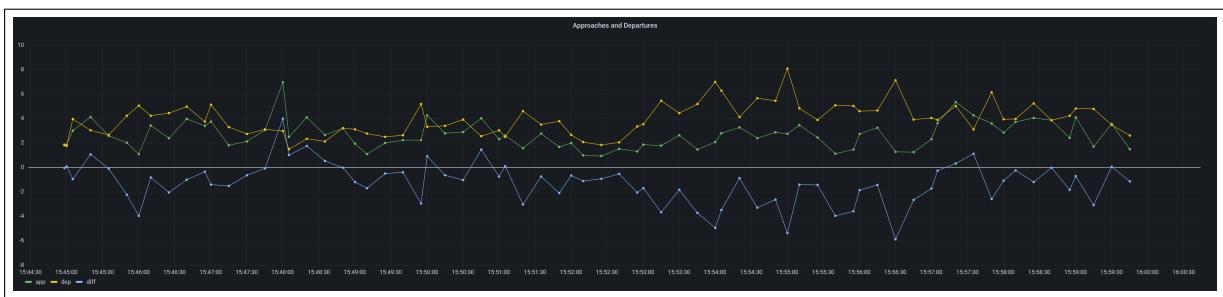


Abbildung 15.1: Durch KI-Modell ermittelte An-/Abflugrate (grün - Anflüge; gelb - Abflüge; blau - Differenz)

Teil IV

Diskussion

16. Zusammenfassung der Ergebnisse

Datenakquise

Der erste Schritt zur Realisierung des Projektes ist die Sammlung verwendbarer Bilddaten für das Training des Objekterkennungsmodells.

Hierfür müssen die Rechte am Bild zur Nutzung innerhalb der Studienarbeit eingeholt werden. Darüber hinaus muss eine ausreichende Bildqualität gegeben sein, damit das Modell effektiv lernen kann (siehe Kapitel 12).

Schließlich können die Bilder manuell annotiert werden, sodass sie im weiteren Verlauf für das Training mit YOLOv5 einsetzbar sind (siehe Abschnitt 11.2).

Training des Modells zur Objekterkennung

Für das Training mit YOLOv5 werden ungefähr 300 Bienen-Bilder annotiert. Nach 300 Trainings-Epochen erreicht das Modell eine kaum noch sinkende Verlustrate von circa 1,7% (siehe u.a. Abschnitt 11.5.1 und Kapitel 12). Mit diesem trainierten Modell können bereits Bienen in Videos in Echtzeit erfolgreich und mit hoher Präzision erkannt werden.

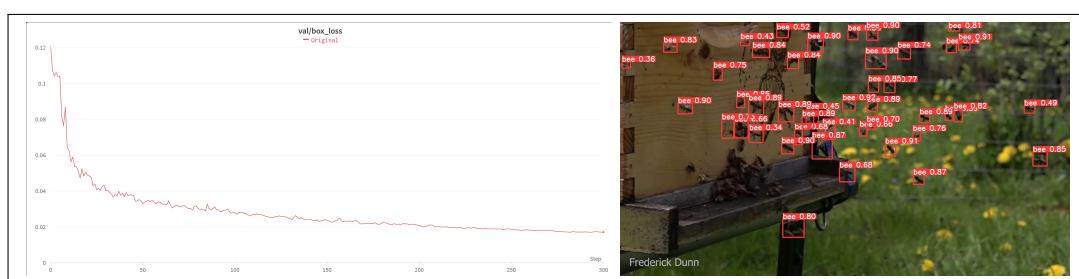


Abbildung 16.1: Verlustrate des Modells und erkannte Bienen, siehe Abschnitt 8.3.2

Einsatz des Tracking-Algorithmus

Für das Tracking der Bienen wird der Norfair-Algorithmus mit dem trainierten YOLOv5-Modell verwendet.

Aus den Erkennungen in den Einzelbildern bestimmt dieser die zusammengehörenden Bienen anhand der Positionen (siehe Abschnitt 8.3.2). Die Erkennungsgenauigkeit hängt hier von der Anzahl der FPS im Quellvideo ab.

Ein Video mit mindestens 60 FPS führt zu einer Erkennungsrate von durchschnittlich ca. 85%. Videos mit 24 FPS führen zu teils schlechteren Ergebnissen, jedoch können diese durch Interpolation leicht verbessert werden.

Beispielvideos, die die Daten des entwickelten Tracking-Algorithmus darstellen, sind in Anhang A.1 verlinkt.



Abbildung 16.2: Tracking und Numerierung der Bienen, siehe Abschnitt 8.3.2

Verarbeitung der Tracking-Daten

Aus den Daten des Tracking-Algorithmus können unter anderem die *Pfade* und *Flugwinkel* der Bienen-Flugbahnen bestimmt werden.

Die Bewegungsdaten werden dabei so weiterverarbeitet, dass zu jedem Bildausschnitt im Video die Anzahl der *an- und abfliegenden* und sich nicht-bewegenden Bienen ermittelt werden. Die implementierte Anwendung speichert die Daten anschließend in einer Datenbank.

Da die Verarbeitung rein mathematisch stattfindet (siehe Abschnitt 14.1), liegt die Genauigkeit hierbei theoretisch bei 100%, da lediglich die Flugrichtung und Distanz aus den vorher ermittelten Daten bestimmt werden und keine der erfassten Daten verloren gehen können.

Allerdings müssen zur Bestimmung von Parametern wie der Flugrichtung bestimmte Annahmen getroffen werden. Sollten diese Annahmen ungenau sein, kann es erneut zu Fehlern kommen. Aufgrund der konkreten Implementierung sollten sich Fehler zur Flugrichtung allerdings statistisch ausgleichen, da im Mittel genauso viele Bienen zum Bienenstock hin- und wegfliegen sollten. Dadurch können diese Fehler vernachlässigt werden.

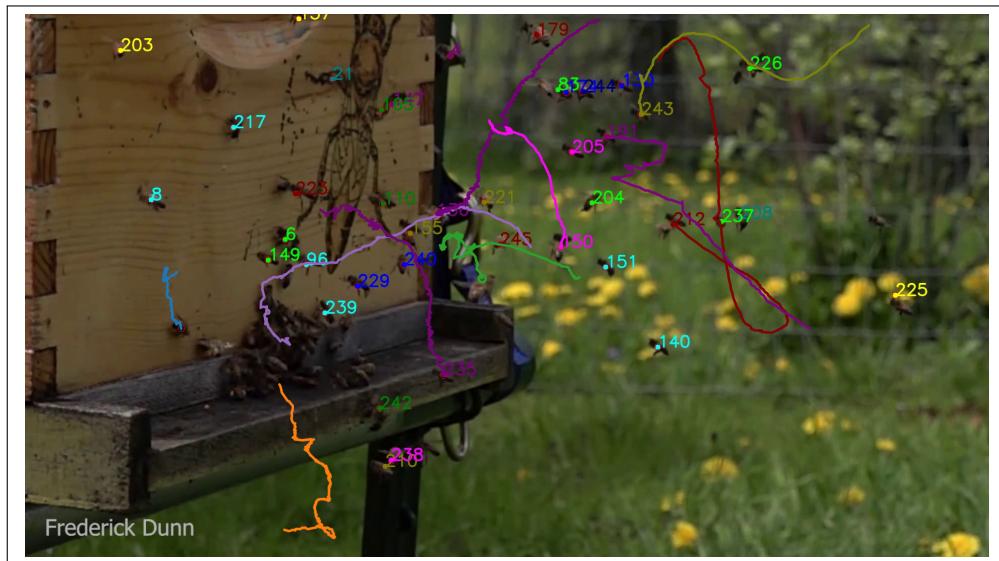


Abbildung 16.3: Visuelle Aufbereitung der Flugpfade, mit Daten aus [DUNN 2021]

Visualisierung der Daten

Die ausgewerteten Daten kann Grafana visualisieren, dadurch können Informationen gewonnen werden, wie die An- und Abflugrate der Bienen über einen Tag. Anwender*innen haben dabei die Möglichkeit, das Verhalten und langfristige Trends bei ihren Bienenvölkern zu erkennen, sofern diese regelmäßig geeignetes Videomaterial bereitstellen.

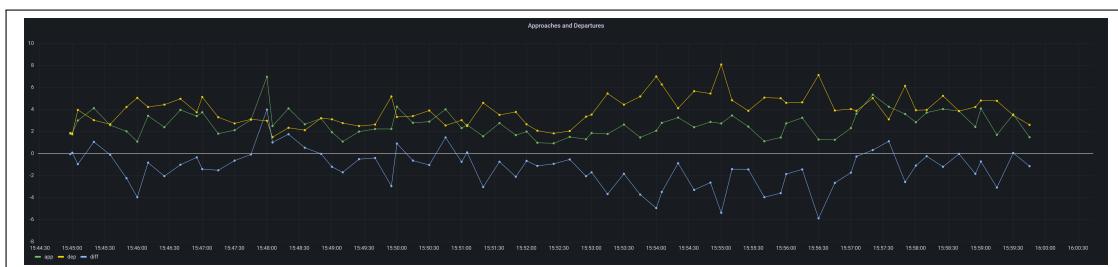


Abbildung 16.4: Visuelle Darstellung der An- und Abflugraten der Bienen, siehe Abschnitt 15.1.1

17. Evaluation der Ergebnisse

17.1 Bewertung

17.1.1 Bewertung der Bienen-Erkennung

Vorgehen

Wie bereits in Kapitel 9 beschrieben werden zur Bewertung der Erkennung von Bienen die Validierungsdaten verwendet. Dabei werden verschiedene Metriken generiert. Unter anderem sind Verlustraten-, Präzisions- und Trefferquote-Diagramme von Bedeutung.

Ergebnis

Das Modell weist eine finale Verlustrate von 1,7% (siehe Abb. 17.1), Präzision von 99,6% (siehe Abb. 17.2) und Trefferquote von 99,2% (siehe Abb. 17.3) auf. Mithilfe der generierten Videos, in denen der YOLO-Algorithmus die Erkennungen visualisiert, können manuelle Prüfungen durchgeführt werden. Festzustellen ist, dass vor allem mehrere Bienen, die einen Haufen bilden (siehe Abb. 17.4), teilweise als nur eine Biene erkannt werden.

Ein Video mit Full-HD-Auflösung, 50 FPS als Bildwiederholungsrate und durchschnittlich 47 Bienen pro Bild kann auf dem Test-Computer mit einer NVIDIA GeForce RTX 3070 mit mindestens 60 FPS als Ausgaberate detektiert werden. Somit könnte die Detektion theoretisch auch bereits auf schlechterer Hardware in Echtzeit ablaufen.

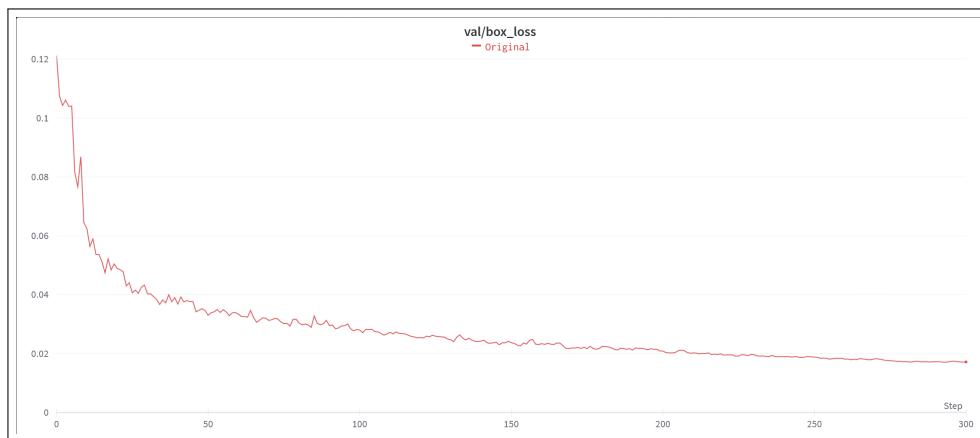


Abbildung 17.1: Verlustrate des Modells

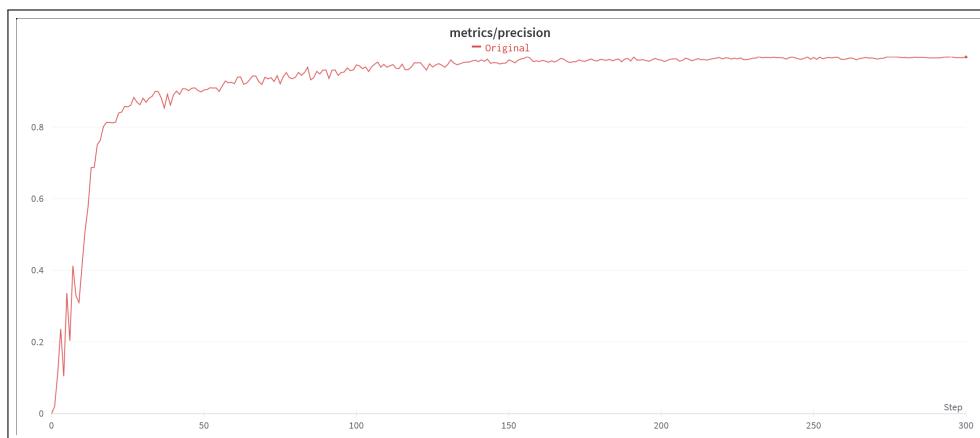


Abbildung 17.2: Präzision des Modells

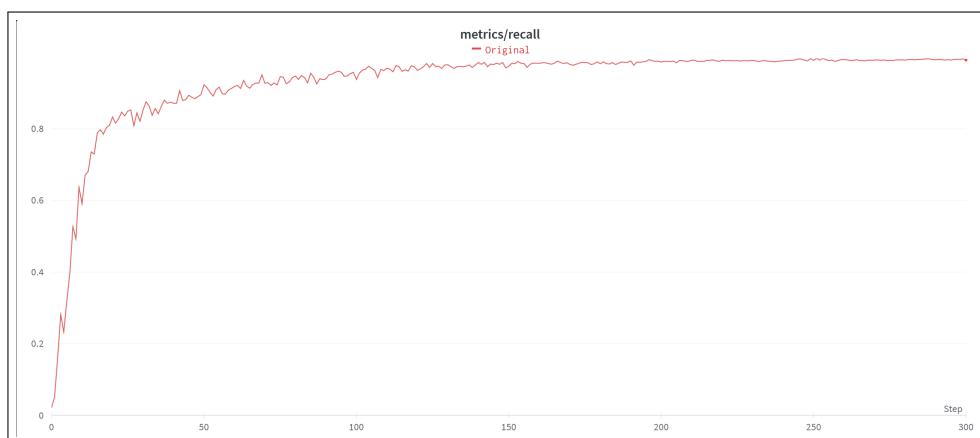


Abbildung 17.3: Trefferquote des Modells



Abbildung 17.4: Mehrere Bienen werden als eine einzelne Biene erkannt, bearbeiteter Screenshot aus [JUNG 2021]

17.1.2 Bewertung des Trackings

Vorgehen

Um die Genauigkeit des Trackings bewerten zu können, werden die bisherigen Daten der Detektionen mit denen des Trackings verglichen. Dazu wird die Differenz zwischen der Anzahl der erkannten und der Anzahl der getrackten Bienen bestimmt. Dieser Wert sollte stets, wie in Abb. 17.5 zu sehen, negativ sein, da jederzeit mehr Detektionen als Trackings ermittelt werden. Dies resultiert aus der Logik des Tracking-Algorithmus, indem mehrere aufeinanderfolgende Detektionen zu einem Tracking zusammengeführt werden. Einzelne Stellen, an denen die Differenz positiv wird, sind durch die Funktionsweise des Trackings erklärbar. Wird eine Biene in einem einzelnen Bild nicht erkannt, im darauf folgenden aber wieder, wird sie trotzdem als eine Biene erkannt. In dem Bild, in dem sie nicht erkannt wurde, fehlt sie also in der Anzahl der Detektionen. Aufgrund ihrer berechneten Lebensdauer aus dem Tracking ist sie aber in den getrackten Bienen enthalten, wodurch die positive Differenz entstehen kann.

Ergebnis

Je größer die Differenz zwischen Detektionen und Trackings, desto schlechter performt der Tracking-Algorithmus im Vergleich zum KI-Erkennungsmodell. Beispielsweise vergrößert sich in Abb. 17.5 die Differenz zum Ende des Videoausschnitts. Dies ist auf das gehäufte Überlappen von Bienen innerhalb des Bildausschnitts gegen Ende des Videos zurückzuführen. Um solche Ausschläge in der Qualität des Trackings leicht erkennen zu können, bietet es sich an, ab einem bestimmten Schwellwert eine Warnung anzuzeigen. So kann schnell auf Missstände reagiert werden, um eine hohe Datenqualität beizubehalten. Ein Video mit Full-HD-Auflösung, 50 FPS als Bildwiederholungsrate und durchschnittlich ungefähr 47 Bienen pro Bild kann auf dem Test-Computer mit einer NVIDIA GeForce RTX 3070 mit circa 10 FPS als Ausgaberate getrackt werden. Das hat zur Folge, dass das Tracking nicht in Echtzeit geschehen kann.



Abbildung 17.5: Analyse der Genauigkeit des Tracking (grün - Anzahl getrackter Bienen; gelb - Anzahl detektierter Bienen; blau - Differenz)

18. Ausblick

18.1 Optimierung der Auswertung

Aufgrund der schnellen Flugbewegungen von Bienen kommt es häufig vor, dass sich mehrere Bienen überlappen. Dies hat den Grund, dass sich mehrere Bienen gleiche x - und y -Koordinaten teilen, jedoch auf unterschiedlichen z -Koordinaten fliegen.

Das führt zu einer Unterbrechung des Trackings und zu unvollständigen Tracking-Instanzen von Bienen, da der zweidimensionale Raum die z -Koordinate vernachlässigt. Um dieses Problem zu beheben, könnten bei allen Bienen die Start- und Endkoordinaten abgeglichen werden. Die Genauigkeit des Tracking-Algorithmus könnte erhöht werden, indem Bienen mit sehr ähnlichen Koordinaten „zusammengeführt“ werden.

18.2 Live-Analyse

Aufgrund *mangelnder Effizienz* des Tracking-Algorithmus und fehlender Rechenleistung scheitert eine Live-Auswertung des Videomaterials. Je nach Bienenanzahl im auszuwertenden Videoausschnitt dauert die Analyse teils die doppelte Zeitspanne des ursprünglichen Ausschnitts. Beispielsweise benötigt die Analyse bei Videoausschnitten mit hoher Bienenanzahl und 50 FPS bei einem 15-minütigem Video ca. 30 Minuten.

18.3 Weitere Optimierung des Objekterkennungsmodells

Das bisher verwendete YOLO-Modell basiert lediglich auf ca. 300 annotierten, teilweise mangelhaften, Bildern von Bienen.

Durch eine weitere Optimierung des YOLO-Modells kann die Genauigkeit des Programms erhöht werden, sodass mehr Bienen richtig und weniger Hintergrundobjekte falsch erkannt werden.

Die Optimierung des Modells kann durch höherwertiges und mehr Bilddaten mit beispielsweise einem eigenen Bienenstock geschehen.

18.4 Deployment-Möglichkeiten

18.4.1 Eigenständige Webapplikation

Grafana könnte durch eine eigene Webapplikation mit einem zielgerichteteren Frontend ersetzt werden.

Hierfür könnten Nutzerprofile angelegt und Bienen-Videos direkt hochgeladen werden. Nach abgeschlossener Analyse des Videos könnten die dazugehörigen Video-Dateien wieder vom Server gelöscht und nur die ermittelten Daten gespeichert werden.

Bereitstellung der Rechenleistung

Nicht jede*r Imker*in besitzt einen rechenstarken PC und kann Videos effizient selbst analysieren. Um dieses Problem zu umgehen, kann die Rechenleistung auf einen Server ausgelagert werden. Denkbar ist hierbei bei starker Nachfrage der Einsatz von mehreren On-Demand-Instanzen. Die benötigte Rechenleistung könnte durch Serverless-Computing-Anbieter gedeckt werden. Dabei wird ausschließlich für die Rechenleistung gezahlt, die auch verwendet wird. Sofern keine Videos analysiert werden, entstehen dabei keine Kosten.

Livestream-Analyse

Denkbar ist auch die Analyse einer Live-Videoübertragung, um eine permanente Datenermittlung zu ermöglichen. So könnte zum Beispiel ein Link zu einem Twitch-Stream angegeben werden, der daraufhin live auf einem Server analysiert wird.

Datenspeicher in der Cloud

Aufgrund der erleichterten Benutzung durch die Weboberfläche könnten die ermittelten Daten aus den Videos auch direkt auf einem zentralen Server gespeichert werden. Somit müssten Anwender*innen diese Daten nicht auf einem eigenen System speichern. Dadurch verbessert sich die User Experience, da User ohne weitere Schritte die Analysen einsehen können. Zudem ergibt sich durch die zentrale Speicherung die Möglichkeit für die Betreiber der Plattform, über alle vorhandenen Daten hinweg Analysen zu erstellen, um beispielsweise längerfristige (globale) Trends frühzeitig erkennen zu können.

Software as a Service

Somit wäre die Weboberfläche ein All-In-One-Tool für den*die Anwender*in, während die benötigte Rechenleistung durch den*die Hoster*in abgedeckt würde. Dabei könnten auch längere Videos einfach hochgeladen werden und zu einem späteren Zeitpunkt betrachtet werden. Daraus ließe sich ein Software-Produkt bauen, das durch ein Abonnement-Modell finanziell abzudecken wäre.

18.4.2 Self-hosted Deployment

Docker-Image

Eine weitere Möglichkeit, die Anwendung zukünftig publik zur Verfügung zu stellen, wäre das Veröffentlichen als Docker-Image. Benutzer*innen könnten die Applikation auf eigenen

Computern/Servern innerhalb eines Docker-Containers ausführen und die Kontrolle über die generierten Daten behalten.

Open-Source

Der gesamte Applikations-Code wird auf GitHub mit der MIT-Lizenz als Open-Source-Anwendung veröffentlicht. So können Benutzer*innen, die über das nötige technische Wissen verfügen, die Anwendung selber auf ihrem PC oder Server installieren und bei Bedarf erweitern.

Datenschutz

Ein Vorteil des selbstständigen Aufsetzen der Anwendung ist, dass sich jegliche Daten zu den bereitgestellten Videos stets auf eigenen Computern/Servern befinden. Damit besteht keine Gefahr für den Schutz der Daten.

Benötigte Rechenleistung

Aufgrund der benötigten Rechenleistung ist die Option einer eigenen Anwendungsinstanz nur bedingt möglich. Durch die Virtualisierung mittels Docker wird ohnehin mehr Rechenleistung im Vergleich zu einem Bare-Metal-Hypervisor benötigt. Das heißt, dass, sofern eine Live-Auswertung des Videomaterials erwünscht ist, die Mindestanforderungen auf einer virtualisierten Plattform erweitert werden müssen.

Teil V

Anhang

A. Anhang

A.1 Projekt-Links

Der gesamte Quellcode der für diese Arbeit entwickelten Algorithmen ist auf GitHub verfügbar: <https://github.com/Plan-Bee>.

Die zum Testen des Tracking-Algorithmus verwendeten Videoausschnitte sind dem YouTube-Video [DUNN 2021] entnommen. Einschließlich der Tracking-Annotationen sind die Videoausschnitte auf YouTube bereitgestellt:

- Video 1: <https://youtu.be/08A6Px-3Enc>
- Video 2: <https://youtu.be/LIW8VoPco9Y>
- Video 3: <https://youtu.be/lilCZmkMcQc>
- Video 4: <https://youtu.be/YDwg2BISZ4k>

Die zugehörigen Rohdaten des Trackings der jeweiligen Videos sind im folgenden GitHub-Repository zur Verfügung gestellt: <https://github.com/Plan-Bee/Publications>.

A.2 Bildvariationen



(a) Original



(b) Übersättigung



(c) Schwarz-Weiß



(a) Hoher Kontrast



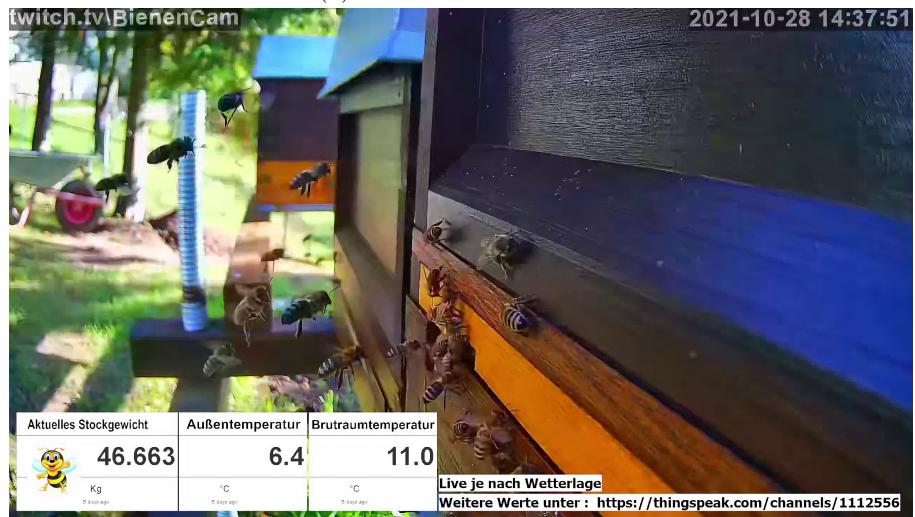
(b) Niedriger Kontrast



(c) Nachgeschärft



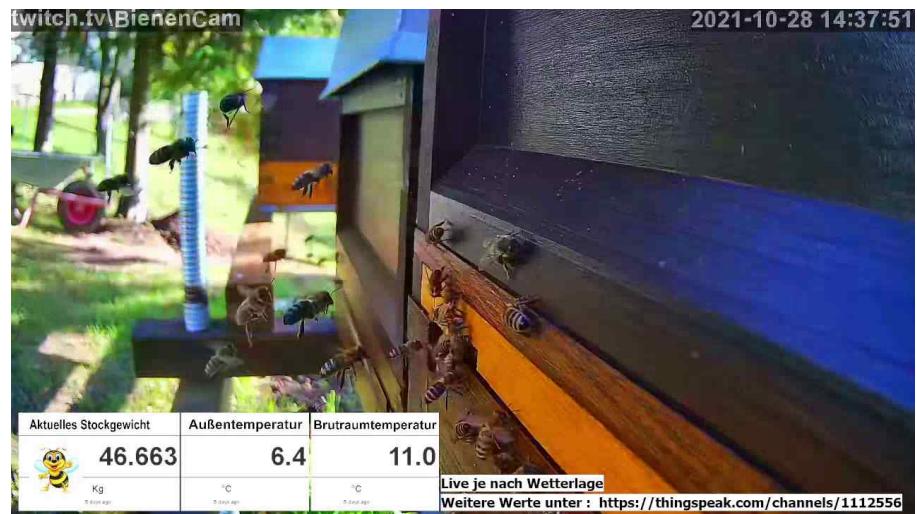
(a) Gauß'sche Unschärfe



(b) Hochskalierung



(c) Runterskalierung



(a) Komprimierung

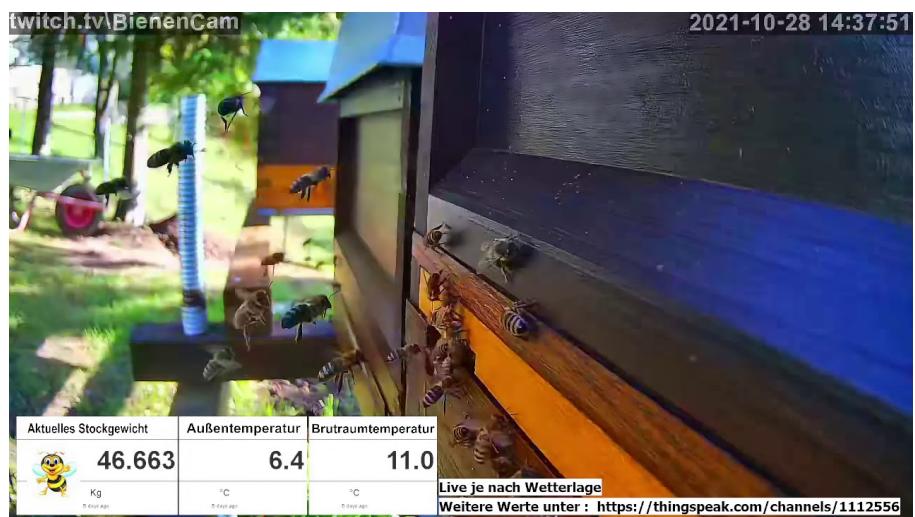


(b) Starke Komprimierung



(c) Rauschen

V



(a) Denoise

Abbildung A.5: Vergleich der verschiedenen Trainingsdaten, Screenshot aus Twitch-Stream von [JUNG 2021] mit nachträglicher Bearbeitung

Index

A

Accuracy, 29
Aktivierungsfunktion, 24

B

Backpropagation, 27
Batch Normalization, 47
Bestarkendes Lernen, 21
Bildklassifizierung, 32
Bounding Box, 35

C

Convolutional Layer, 30
Convolutional Neural Network, 29
Convolutional with Anchor Boxes, 47

D

Data Augmentation, 22, 39
Detectron, 36
Dimension Clusters, 47
Direct Location Prediction, 48

G

Generalisierung, 26
Gradient Descent, 26

H

High Resolution Classifier, 47
Histogram of Oriented Gradients, 33

K

Kostenfunktion, 26
Kreuzvalidierung, 28
Kunstliche Intelligenz, 15
Künstliches Neuron, 24
Künstliches Neuronales Netz, 23

M

Maschinelles Lernen, 17
Multi-Scale Training, 48

O

Objekt-Tracking, 36
Objekterkennung, 32
One Stage Detector, 36
Overfitting, 22

P

Pooling Layer, 31
Precision, 29
PyTorch, 35

R

Recall, 28
Region-based Convolutional Neural Network, 33
ReLU-Aktivierungsfunktion, 25

S

Semiüberwachtes Lernen, 21
Sigmoid-Aktivierungsfunktion, 25
Single Shot Multibox Detector, 36

U

Überwachtes Lernen, 19
Underfitting, 22
Unüberwachtes Lernen, 20

V

Visual Relationship Detection, 32

Y

You Only Look Once, 35

Literatur

- ADOBE [2021]. *Contrast*. URL: <https://helpx.adobe.com/photoshop/key-concepts/contrast.html> [besucht am 29.04.2022] [siehe S. 60].
- AMEISEN, Emmanuel [2019]. *Building Machine Learning Powered Applications*. 1. Edition. Sebastopol: O'Reilly Media, Inc. ISBN: 978-1-492-04511-3 [siehe S. 22, 23].
- APIC.AI GMBH [2022]. *Unser Service*. URL: <https://apic.ai/service.html> [besucht am 23.04.2022] [siehe S. 6].
- BEEBETTER [o. D.[a]]. *Bienensterben - Ursachen und Auswirkungen*. URL: <https://www.beebetter.de/bienensterben-ursachen-und-auswirkungen> [besucht am 26.01.2022] [siehe S. 14].
- [o. D.[b]]. *Bienenstock - Das Zuhause des Bienenvolks*. URL: <https://www.beebetter.de/bienenstock-aufbau-rollenverteilung-im-bienenstaat> [besucht am 26.01.2022] [siehe S. 14].
- BERNBURG, Serumwerk [o. D.] *Imker – Hobby oder Beruf?* URL: <https://www.bienen-gesundheit.com/imker-hobby-oder-beruf/> [besucht am 26.01.2022] [siehe S. 12].
- BOCHKOVSKIY, Alexey, Chien-Yao WANG und Hong-Yuan Mark LIAO [2020]. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. DOI: 10.48550/ARXIV.2004.10934. URL: <https://arxiv.org/abs/2004.10934> [siehe S. 50].
- CAREFUL, Bee [o. D.] *Das Bienenjahr*. URL: <https://www.bee-careful.com/de/initiative/das-bienenjahr/> [besucht am 18.01.2022] [siehe S. 12].
- DAI, Jifeng u. a. [2016]. *R-FCN: Object Detection via Region-based Fully Convolutional Networks*. DOI: Tech. URL: <http://arxiv.org/pdf/1605.06409v2> [siehe S. 34].
- DELUA, Julianna [2021]. *Supervised vs. Unsupervised Learning: What's the Difference?* Zuletzt aufgerufen: 03.05.2022. URL: <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning> [siehe S. 20].
- DEUTSCHER IMKERBUND E.V. [2022]. *Imkerei in Deutschland*. URL: https://deutscherimkerbund.de/161-Imkerei_in_Deutschland_Zahlen_Daten_Fakten [besucht am 06.05.2022] [siehe S. 3, 4].
- DUNN, Frederick J. [2021]. *Honey Bee Slowmotion Goodness Collecting Pollen Returning Home*. URL: <https://youtu.be/ue1EHAndk4w> [besucht am 22.04.2022] [siehe S. 38, 80, I].

- ERTEL, Wolfgang [2016]. *Grundkurs KI*. 4. Auflage. Springer. ISBN: 978-3658135485 [siehe S. 17, 18, 21, 25, 26, 28].
- GARVEY, Shunryu [März 2018]. „Broken Promises and Empty Threats: The Evolution of AI in the USA, 1956-1996“. In: *Technology's Stories*. DOI: 10.15763/jou.ts.2018.03.16.02 [siehe S. 16].
- GIRAUD, Morgan [2016]. *A simple bees larvae detector in Deep learning*. URL: <https://github.com/metaflow-ai/hive> [besucht am 24.04.2022] [siehe S. 6].
- GIRSHICK, Ross u. a. [2018]. *Detectron*. <https://github.com/facebookresearch/detectron> [siehe S. 36].
- GODIL, Afzal u. a. [2014]. *Performance Metrics for Evaluating Object and Human Detection and Tracking Systems*. DOI: 10.6028/NIST.IR.7972. URL: <https://dx.doi.org/10.6028/NIST.IR.7972> [siehe S. 41].
- GOOGLE [2020a]. *Classification: Accuracy*. Zuletzt aufgerufen: 06.05.2022. URL: <https://developers.google.com/machine-learning/crash-course/classification/accuracy> [siehe S. 29].
- [2020b]. *Classification: Precision and Recall*. Zuletzt aufgerufen: 06.05.2022. URL: <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall> [siehe S. 29].
- GRAND VIEW RESEARCH [2022]. *Artificial Intelligence Market Size, Share and Trends Analysis Report*. URL: <https://www.grandviewresearch.com/industry-analysis/artificial-intelligence-ai-market> [besucht am 02.05.2022] [siehe S. 2].
- GÉRON, Aurélien [2019]. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow - Concepts, Tools, and Techniques to Build Intelligent Systems*. 2. Edition. Sebastopol: O'Reilly Media, Inc. ISBN: 978-1-492-03261-8 [siehe S. 17–21, 27, 28].
- HONEYPI [2021]. *Aktuelles über HoneyPi*. URL: <https://honey-pi.de/aktuelles-ueber-honeypi/> [besucht am 11.11.2021] [siehe S. 2].
- [o. D.] *HoneyPi – Bienenstockwaage*. URL: <https://honey-pi.de/> [besucht am 11.11.2021] [siehe S. 2].
- IOFFE, Sergey und Christian SZEGEDY [2015]. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. In: *CoRR* abs/1502.03167. arXiv: 1502.03167. URL: <http://arxiv.org/abs/1502.03167> [siehe S. 47].
- JUNG, Rene [2021]. *Bienencam*. URL: <https://www.twitch.tv/bienencam> [besucht am 23.03.2022] [siehe S. 20, 39, 53, 57, 73, 83, VI].
- KIM, Phil [2017]. *MATLAB Deep Learning*. Berkeley, CA: Apress. ISBN: 978-1-4842-2844-9. DOI: 10.1007/978-1-4842-2845-6 [siehe S. 29–31, 34].
- KOEHRSEN, Will [2018]. *Overfitting vs. Underfitting: A Conceptual Explanation*. Zuletzt aufgerufen: 02.05.2022. URL: <https://towardsdatascience.com/overfitting-vs-underfitting-a-conceptual-explanation-d94ee20ca7f9> [siehe S. 22].
- KUZNETSOVA, Alina u. a. [2020]. „The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale“. In: URL: <https://arxiv.org/abs/1811.00982> [besucht am 17.01.2022] [siehe S. 32].

- MAINDOLA, Gaurav [2021]. *Introduction to YOLOv5 Object Detection with Tutorial*. URL: <https://machinelearningknowledge.ai/introduction-to-yolov5-object-detection-with-tutorial/> [besucht am 08.02.2022] [siehe S. 35, 50].
- MCCARTHY, John [1955]. *A proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. Zuletzt aufgerufen: 22.01.2022. URL: <http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf> [siehe S. 15].
- MEEL, Vidushi [o. D.] *What is Object Tracking? – An Introduction*. URL: <https://viso.ai/deep-learning/object-tracking/> [besucht am 23.02.2022] [siehe S. 37].
- MENDELS, Gideon [2019]. *Selecting the right weight initialization for your deep neural network*. Zuletzt aufgerufen: 24.05.2022. URL: <https://towardsdatascience.com/selecting-the-right-weight-initialization-for-your-deep-neural-network-8ccf8dfcf4c> [siehe S. 24].
- MONITORING, Bee Hive [o. D.] *Häufige Fehler von Imker-Anfängern*. URL: <https://www.beehivemonitoring.com/de/blog/post/haufige-fehler-von-imker-anfangern.html> [besucht am 26.01.2022] [siehe S. 13].
- ORACLE CORPORATION [o. D.] *11.7 Data Type Storage Requirements*. URL: <https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html> [besucht am 25.04.2022] [siehe S. 75].
- [2019]. *How Oracle and The World Bee Project are Using AI to Save Bees*. URL: <https://blogs.oracle.com/ai-and-datasience/post/how-oracle-and-the-world-bee-project-are-using-ai-to-save-bees> [besucht am 11.11.2021] [siehe S. 2].
 - [2018]. *World's first AI smart hives network helps conserve declining global honey bee populations*. URL: <https://www.oracle.com/uk/corporate/pressrelease/ai-smart-hives-network-helps-conserving-global-honey-bee-2018-10-16.html> [besucht am 11.11.2021] [siehe S. 2].
- PAPERSWITHCODE [o. D.] *Object Tracking*. URL: <https://paperswithcode.com/task/object-tracking> [besucht am 23.02.2022] [siehe S. 37].
- PLANTOPEDIA, Katharina von [o. D.[a]]. *Stockkarte für Bienen: Vorlage zum Ausdrucken*. URL: <https://www.plantopedia.de/stockkarte-bienen-vorlage/> [besucht am 26.01.2022] [siehe S. 12].
- [o. D.[b]]. *Stockkarte für Bienen: Vorlage zum Ausdrucken*. URL: https://www.plantopedia.de/wp-content/uploads/2020/03/Plantopedia_Stockkarte.pdf [besucht am 26.01.2022] [siehe S. 12].
- RAJPUT, Mihir [2020]. *YOLOv5-s Model Architecture*. Zuletzt aufgerufen: 27.04.2022. URL: <https://gist.github.com/mihir135/2e5113265515450c8da934e15d97fc6b> [siehe S. 25].
- REDMON, Joseph und Ali FARHADI [2016]. *YOLO9000: Better, Faster, Stronger*. DOI: 10.48550/ARXIV.1612.08242. URL: <https://arxiv.org/abs/1612.08242> [siehe S. 48].
- [2018]. *YOLOv3: An Incremental Improvement*. DOI: 10.48550/ARXIV.1804.02767. URL: <https://arxiv.org/abs/1804.02767> [siehe S. 49].
- REDMON, Joseph u. a. [2015]. *You Only Look Once: Unified, Real-Time Object Detection*. DOI: 10.48550/ARXIV.1506.02640. URL: <https://arxiv.org/abs/1506.02640> [siehe S. 45–47].

- RUSSELL, Stuart [2012]. *Künstliche Intelligenz: Ein moderner Ansatz*. 3., aktualisierte Aufl. Bd. 4098. Pearson Studium / Informatik. München: Pearson Studium. ISBN: 978-3-86894-098-5 [siehe S. 15, 16, 19, 24, 27, 28].
- SHARMA, Sagar [2017]. *Activation Functions in Neural Networks*. Zuletzt aufgerufen: 03.05.2022. URL: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> [siehe S. 26].
- SIMPLYSCIENCE.CH, Redaktion [2013]. *Das Bienenvolk - Königin, Drohnen und Arbeiterinnen*. URL: <https://www.simplyscience.ch/teens/wissen/das-bienenvolk-koenigin-drohnen-und-arbeiterinnen> [besucht am 26.01.2022] [siehe S. 14].
- TRYOLABS [2022a]. *Norfair*. URL: <https://github.com/tryolabs/norfair> [besucht am 28.01.2022] [siehe S. 37].
- [2022b]. *Norfair*. URL: <https://github.com/tryolabs/norfair#comparison-to-other-trackers> [besucht am 29.04.2022] [siehe S. 51].
- TWITCH [2022]. *Twitch 101*. URL: <https://www.twitch.tv/creatorcamp/en/learn-the-basics/twitch-101/> [besucht am 29.04.2022] [siehe S. 64].
- TWITCH.TV [2022]. *Twitch Streamers - Twitch Video Encoding/Bitrates/And Stuff*. URL: <https://stream.twitch.tv/encoding/> [besucht am 18.04.2022] [siehe S. 43].
- TZUTALIN [2022a]. *LabelImg*. URL: <https://github.com/tzutalin/labelImg> [besucht am 29.04.2022] [siehe S. 8].
- [2022b]. *LabelImg*. URL: <https://github.com/tzutalin/labelImg> [besucht am 22.04.2022] [siehe S. 52].
- VAN VEEN, Fjodor [2016]. *The Neural Network Zoo*. Zuletzt aufgerufen: 14.09.2016. URL: <https://www.asimovinstitute.org/neural-network-zoo/> [siehe S. 29].
- VANDERPLAS, Jake [Dez. 2016]. *Python Data Science Handbook*. Sebastopol, CA: O'Reilly Media [siehe S. xiii].
- WANG, Chi-Feng [2019]. *The Vanishing Gradient Problem*. Zuletzt aufgerufen: 06.05.2022. URL: <https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484> [siehe S. 25].
- WIKIPEDIA [2021]. *Honigbienen — Wikipedia, die freie Enzyklopädie*. URL: <https://de.wikipedia.org/w/index.php?title=Honigbienen&oldid=218146216> [besucht am 26.01.2022] [siehe S. 14].
- [2018]. *Motion-Interpolation*. URL: <https://de.wikipedia.org/w/index.php?title=Motion-Interpolation&oldid=176939562> [besucht am 20.04.2022] [siehe S. 72].
- WU, Yuxin u. a. [2019]. *Detectron2*. <https://github.com/facebookresearch/detectron2> [siehe S. 36].
- XAILIENT INC [2022]. *Xailient Protects Australian Bees with Solar-Powered Computer Vision AI*. URL: <https://xailient.com/blog/xailient-protects-bees-with-solar-powered-computer-vision-ai/> [besucht am 25.04.2022] [siehe S. 6].
- ZHANG, Xin und Wang DAHU [2019]. „Application of artificial intelligence algorithms in image processing“. In: *Journal of Visual Communication and Image Representation* 61, S. 42–49. ISSN: 1047-3203. DOI: <https://doi.org/10.1016/j.jvcir.2019.03.004>. URL:

<https://www.sciencedirect.com/science/article/pii/S1047320319300975> [siehe S. 39].

ZOU, Zhengxia u. a. [2019]. *Object Detection in 20 Years: A Survey*. URL: <http://arxiv.org/pdf/1905.05055v2> [siehe S. 33–36].