Detektion von Bienen im Projekt EnBeeMo

Stefan Schulz, Markus Gross und Patrick Haidinger
Hochschule für Angewandte Wissenschaften
München, Deutschland

***Zusammenfassung*—Diese Seminararbeit entstand im Rahmen der Lehrveranstaltung Deep Learning und befasst sich mit der Entwicklung eines Objektdetektors zur Erkennung von Bienen. Hierfür wird die Architektur "Mask R-CNN" [1] mit unterschiedlichen Regularisierungstechniken verwendet. Dabei werden wesentliche Hyperparameter analysiert und modifiziert. Die jeweiligen Auswirkungen auf die Performanz und Inferenzzeit werden anschließend untersucht und evaluiert. Mit der besten Konfiguration wird ein mAP@50 von 0.854 erreicht. Die Inferenzzeit beträgt unter den gegebenen Hardware-Voraussetzungen 265ms bei 44.662.942 Parametern.**

I. Einleitung

Diese Studienarbeit befasst sich mit der Entwicklung und Evaluierung eines Neuronalen Netzes zur Objekterkennung mit Hilfe des Transferlernens. Es wird eine alternative Detektion von Bienen im Projekt EnBeeMo aufgezeigt, mit dem Ziel die bisher verwendete U-Net Architektur [2] zu ersetzen. Hierfür wird eine Mask R-CNN Architektur auf Basis der Matterport - TensorFlow - Implementierung [3] verwendet.

A. Projekt EnBeeMo

Das Projekt Environment and Bee Monitoring (EnBeeMo) der Hochschule München unterstützt Imker dabei, Korrelationen zwischen dem Bienenbestand und Umweltfaktoren zu erkennen. Umgesetzt wird dies durch die Entwicklung eines Pareto-optimalen Algorithmus, welcher Bienen beim Betreten und Verlassen eines Bienenstocks detektiert, verfolgt und zählt.

B. Datensatz

Abbildung 1: Auszug aus dem bereitgestellten Datensatz

Der für diese Arbeit bereitgestellte Datensatz besteht aus 2000 Infrarot-Bildern mit identischer Auflösung. Die entsprechenden Label liegen in Form von Bounding Box - Koordinaten der einzelnen Bienen vor (Abbildung 1). In einer Strichprobenuntersuchung wurde festgestellt, dass die Bilder über einen Tagesverlauf erstellt wurden, was in einer sich verändernden Darstellung des Hintergrunds durch die Infrarotkamera resultierte. Um homogene Teildatensätz zu erzeugen und somit eine bessere Generalisierung des Modells zu gewährleisten, werden die Daten normalverteilt und über einen reproduzierbaren Zufall in Trainings-, Validierungs- und Testdaten aufgeteilt. Diese Aufteilung erfolgt in 80% Trainingsdaten und 20% Testdaten, wobei wiederum 80% der Trainingsdaten für das eigentliche Training und 20% für die Validierung des Modells während des Trainings verwendet werden.

C. Zur Bewertung genutzte Metriken

1) Inferenzzeit: Die Inferenzzeit beschreibt die im Mittel erforderliche Zeit einer Modellvorhersage, die in dieser Arbeit über 100 Iterationen bestimmt wird. Die Komplexität des Modells kann ermittelt werden, indem die Gesamtmenge der lernbaren Parameter gezählt wird. Insbesondere kann die Größe der Parameterdatei analysiert werden, um die minimale Menge an GPU-Speicher abzuschätzen, die für das Modell benötigt wird.

2) Performanz: Zur Bewertung der Genauigkeit von Objekterkennungsmodellen wird die mittlere durchschnittliche Genauigkeit (mean average precision, mAP) verwendet. In der Praxis wird hierfür zunächst die Intersection over Union (IoU) zwischen vorhergesagter und tatsächlicher Bounding Box berechnet. Mit diesen IoU-Werten und einem zusätzlichen IoU-Schwellwert ist es möglich, die Performanz-Metriken Precision und Recall zu errechnen. Die durchschnittliche Genauigkeit (AP) einer Klasse ist dabei definiert als die Fläche unter der Precision-Recall-Kurve. Der mAP ist der durchschnittliche Wert des für alle Klassen berechneten AP. Die IoU-Schwellwerte für diese Arbeit werden auf 0.5, 0.75, 0.9 und einer Mittlung über diesen drei Werten festgelegt.

II. Mask R-CNN Architektur

Im Folgenden wird ein kurzer Überblick über die Entwicklung vom R-CNN hin zum Mask R-CNN gegeben.

A. R-CNN [4]

Mit einer selektiven Suche werden zu Beginn Bereiche von Interesse im Bild bestimmt, sog. Regions of Interest (ROIs). Anschließend werden diese durch ein vortrainiertes Faltungsnetz (CNN) propagiert, um schließlich die extrahierten Merkmale mithilfe einer Support Vektor Maschine (SVM) zu klassifizieren. Die für die Detektion benötigten Bounding Boxes werden mittels linearer Regression bestimmt. Aufgrund der Propagierung von 2000 ROIs ist die Detektion im R-CNN sehr rechenintensiv. Mit dem Fast R-CNN wird dem entgegengewirkt.

B. Fast R-CNN [5]

Statt der einzelnen ROIs wird das Bild direkt durch das CNN propagiert. Die ROIs werden weiterhin durch eine selektive

Suche auf dem Eingabebild erzeugt, nun jedoch auf die Feature Maps des CNN projiziert. Die Klassifikation erfolgt statt einer SVM durch eine Softmax-Schicht. Aufgrund des hohen Rechenaufwands soll die selektive Suche mit ihren zahlreichen, rekursiven Iterationen im Faster R-CNN ersetzt werden.

C. Faster R-CNN [6]

Die Bestimmung der ROIs erfolgt nun über ein eigenständiges Netzwerk, das Region Proposal Network (RPN), welches die erzeugten Feature Maps des CNN als Eingabe erhält. Der Vorteil ist, neben einer effizienteren Berechnung, die Einbeziehung des RPN in den Lernprozess. Dies resultiert in einer verbesserten Performanz der Objekterkennung.

D. Mask R-CNN [1]

Das Mask R-CNN erweitert die Architektur des Faster R-CNN um die Instanz-Segmentierung (Abbildung 2). Parallel zur Klassifikation und Bestimmung der Bounding Boxes wird über zusätzliche Faltungsschichten eine Vorhersage der Objektmaske berechnet. Diese Maske dient der pixelweisen Zuordnung zwischen Ein- und Ausgabebild.

Im Sinne des Transferlernens besteht das Backbone aus einer Kombination zwischen einem ResNet [7] und einem Feature-Pyramid-Network [8] (ResNet-FPN). Während bei der Faltung im ResNet die räumliche Auflösung der Feature Maps abnimmt, nimmt die semantische Information zu, welche ausschlaggebend für die Erkennung der Objekte im Bild ist. Beim Upsampling (Deconvolution mit stride 2) der semantisch starken Feature Maps im FPN sind die Informationen über die Objekt-Positionen jedoch ungenau. Um diesen Nachteil auszugleichen, werden die Feature Maps beider Netzwerke lateral durch Addition verbunden.

Obwohl zunächst keine Bienen-Masken im bereitgestellten Datensatz zur Verfügung stehen, wird statt des Faster R-CNN dennoch das Mask R-CNN verwendet, da der mAP bei dieser Architektur selbst für eine reine Objekterkennung höher ist. Die Mask R-CNN Autoren He et al. schreiben hierzu in [1]:

> *"We compare Mask R-CNN to the state-of-the-art COCO bounding-box object detection in Table 3. For this result, even though the full Mask R-CNN model is trained, only the classification and box outputs are used at inference (the mask output is ignored). Mask R-CNN using ResNet-101-FPN outperforms the base variants of all previous state-of-the-art models."*

Grund hierfür ist, dass der Masken-Fehler ein Teil des Gesamt-Fehlers ist, welcher bei der Backpropagation auch auf das RPN zurückgeführt wird. Der Gesamt-Fehler berücksichtigt neben der Maske auch die Klasse und die Bounding Boxen (Gleichung 1). Da die Maske somit Einfluss auf den Lernprozess des RPN hat, welches wiederum für die Performanz der Bounding Box - Vorhersage mitverantwortlich ist, kann die entsprechende Vorhersage verbessert werden, indem Masken zur Verfügung gestellt werden.

$$L = L_{class} + L_{bbox} + L_{mask} \tag{1}$$

Um diesen Vorteil zu nutzen, muss beim Training eine Maske als Teil des Labels enthalten sein. Da im bereitgestellten Datensatz die Bienen nicht maskiert sind, wird diese Anforderung wie folgt gehandhabt: Statt einer speziellen Maske wird derjenige Bereich als Maske interpretiert, der durch die entsprechende Bounding Box eingeschlossen ist. Die Annahme hierbei ist, dass der Hintergrund der Bienen selbst bei variierenden Lichtverhältnissen stets hinreichend homogen ist (Abbildung 1). Somit kann der Hintergrund als Teil des Objekts betrachtet werden.

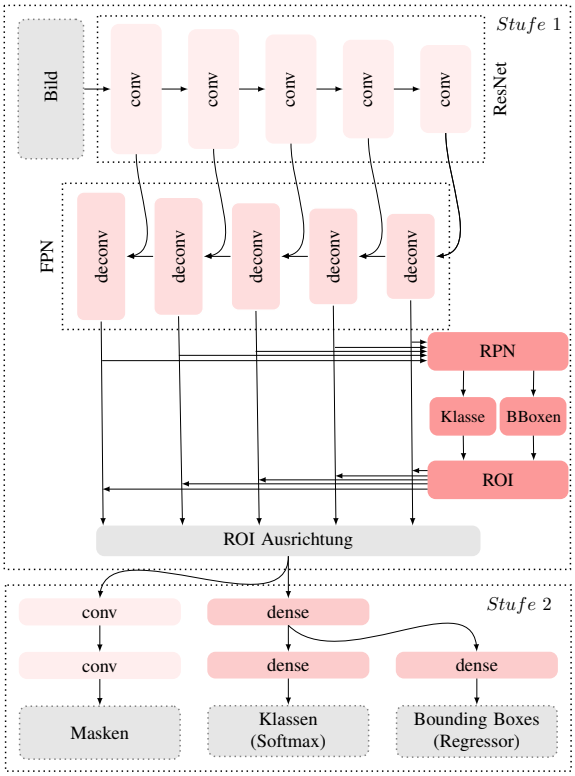

Abbildung 2: Mask R-CNN Schema

III. Regularisierung und Optimierung

Das Ziel der Regularisierung und Optimierung ist es, den Generalisierungsfehler des Modells zu minimieren, indem dessen Komplexität und Kapazität künstlich beschnitten wird. Damit kann sich der optimalen Modellkapazität genähert werden und Über- bzw. Unteranpassung reguliert werden. Hierbei wird die Performanz auf den Trainingsdaten reduziert und auf den Testdaten erhöht, was schlussendlich in einem robusteren Modell resultiert. Das Mask R-CNN bietet einen großen Parameterraum, der im Rahmen dieses Projekts nicht vollständig analysiert werden kann. Daher werden gezielt einzelne, nachfolgend erläuterte Parameter untersucht. Eine alternative Herangehensweise ist in Abschnitt V angeführt.

A. Referenzmodell

Um die Auswirkungen der Hyperparameteranpassungen auf die Performanz zu untersuchen, wird als Referenzmodell das Standard-Modell der Matterport-Implementation genutzt. Lediglich das ResNet101-Backbone wird durch ein ResNet50 ersetzt. Die Gründe für diese Entscheidung werden in Abschnitt IV erläutert. Alle anderen Änderungen werden auf Basis dieses Referenzmodells vorgenommen. Die Standard-Parameter werden im Folgenden bei der Beschreibung der gewählten Änderungen angeführt.

B. Regularisierung

Early Stopping. Die beobachtete Metrik für diese Regularisierung entspricht dem Fehler auf den Validierungsdaten. Das Training wird beendet, sobald der Validierungsfehler über 20 Epochen keine Verbesserung aufzeigt.

Reduktion der Lernrate. Um zu verhindern, dass das Modell auf einem Plateau der Fehlerlandschaft verweilt, wird die initiale Fehlerrate wiederholt um 40% reduziert, wenn keine Verbesserung über 10 Epochen stattfindet.

Data Augmentation. Hierbei werden die Daten entweder rotiert ($15°$, $30°$ oder $45°$), skaliert (70%-130%) oder mit Gausscher Unschärfe versehen ($\mu = \vec{0}$, $\sigma = (3,3)^T$). Bei der Rotation sind die genannten Werte aufgrund der Symmetrie von Bienen und deren Ausrichtungen im Datensatz als ausreichend zu betrachten.

Bestrafung der Parameternorm. Standardmäßig wird in der Matterport-Implementation des Mask R-CNN die L^2-Norm verwendet. Für diesen Hyperparameter wird zusätzlich die L^1-Norm und eine Kombination beider Varianten untersucht.

Batch Normalisierung und Dropout. Diese beiden Regularisierungstechniken werden in den Arbeiten der in Abschnitt II angeführten Architekturen nicht diskutiert [5, 4, 1, 6]. Aufgrund der Komplexität des Modells, in Kombination mit der limitierten Zeit für dieses Projekt, werden diese Techniken daher ebenfalls nicht untersucht. Im Bezug auf das ResNet-Backbone wird Batch Normalisierung nach jeder Faltungsschicht angewendet. Die Dropout-Regularisierung im Backbone wird erst ab einer Modellgröße mit mehr als 1000 Schichten empfohlen, da das Ziel dieser Regularisierung bereits über die tiefe und schmale Architektur des ResNet erzwungen wird [7].

C. Hyperparameteroptimierung

Backbone. In der Matterport-Implementation des Mask R-CNN wird standardmäßig das ResNet101 (101 Schichten) als Backbone verwendet. Zusätzlich wird eine Halbierung der Layer-Anzahl, mithilfe des ResNet50 untersucht.

Lernrate und Optimierer. Zu dem standardmäßig implementierten, stochastischen Gradientenabstiegsverfahren (SGD) mit einer Lernrate von 10^{-3}, wird zusätzlich der Optimierer ADAM analysiert. Beide Verfahren werden mit den Lernraten der Menge $\{10^{-k} \mid k \in \{1,2,3,4,5\}\}$ untersucht. Aufgrund limitierter Hardware-Ressourcen bleibt die Batch Größe über alle Tests bei einem Wert von 2.

ROI Dimensionen. Das Eingabebild des Mask R-CNN wird auf 1024x1024 Pixel skaliert. Dieses Bild wird durch das ResNet-FPN propagiert und dessen Feature Maps an das RPN weitergegeben, wo die ROIs bestimmt werden. Grundlage für die Bestimmung der ROIs sind sogenannte Anker (Anchors), die in Kombination mit unterschiedlichen Seitenverhältnissen und Skalierungen die potentiell zu findenden Objekte repräsentieren. Standardmäßig werden Objekte bis zu einer maximalen Seitenlänge von 512 Pixel gesucht, was der halben Bildgröße entspricht. Jedoch entsprechen die Bienen maximal ca. einem Fünftel der Bildgöße (vertikales Extremum). Um diese a priori Information zu nutzen, werden die Seitenverhältnisse aller Anker halbiert, womit die maximale Ankergröße auf 256 Pixel gesetzt wird. So können mehrere ROIs erzeugt werden, die den tatsächlich zu detektierenden Objekten entsprechen.

Aggregation. Die Konfigurationen der besten Ergebnisse werden zu einer einzigen Konfiguration zusammengeführt, um dessen Auswirkung auf die Performanz zu untersuchen.

Minimale Konfidenz. Nach einer Inferenz werden lediglich diejenigen Objekte ausgegeben, die mit einer Konfidenz von mindestens 70% erkannt werden (Matterport-Standard). Für die beste Konfiguration wird dieser Wert auf 90% gesetzt. Die Annahme ist, dass hierdurch weniger falsch-positive Objekte detektiert werden. Allerdings können dadurch auch korrekt detektierte Objekte ignoriert werden.

IV. Ergebnisse und Evaluation

Die Evaluationsergebnisse werden in Tabelle I aufgeführt.

Konfiguration	mAP@0.5	mAP@0.75	mAP@0.9	$\overline{mAP}$
ResNet50-FPN	0.840	0.271	0.0046	0.3719
ResNet101-FPN	0.818	0.257	0.0044	0.3598
Augmentation	0.837	0.2359	0.0029	0.3587
ROI Dimension *	0.8414	0.2058	0.0018	0.3496
LR 10^{-1} (SGD)	0.000	0.000	0.0000	0.0000
LR 10^{-2} (SGD)	0.837	0.2854	**0.0054**	0.3759
LR 10^{-4} (SGD)	0.7400	0.1441	0.0003	0.2948
LR 10^{-5} (SGD)	0.0211	0.0012	0.0000	0.0075
LR 10^{-1} (Adam)	0.0000	0.0000	0.0000	0.0000
LR 10^{-2} (Adam)	0.839	0.2689	0.0044	0.3707
LR 10^{-3} (Adam) **	**0.854**	**0.292**	0.0038	**0.3834**
LR 10^{-4} (Adam)	0.688	0.1191	0.0005	0.2693
LR 10^{-5} (Adam)	0.222	0.0141	0.00001	0.0788
L1	0.6274	0.0404	0.0001	0.2227
L1 + L2	0.8399	0.2426	0.0036	0.3620
* + **	0.8406	0.2266	0.0033	0.3568
* + ** + Conf 0.9	0,7681	0.2178	0.0033	0.3297

Tabelle I: Ergebnisse der unterschiedlichen Konfigurationen

Backbone. Mit dem ResNet101-Backbone wird ein mAP@0.5 von 0.818 und eine Inferenzzeit von 307ms erreicht. Mit dem ResNet50 hingegen wird ein mAP@0.5 von 0.840 und eine Inferenzzeit von 262ms erzielt. Obwohl das ResNet über Residual-Schichten verfügt, kommt bei der 101-Schichten-Architektur vermutlich das Degradationsproblem zum Tragen, weshalb mit der 50-Schichten-Architektur eine bessere Performanz erreicht werden kann. Eine Validierung dieser Hypothese benötigt jedoch eine genauere Untersuchung, die im Rahmen dieser Arbeit nicht möglich ist. Da das ResNet50 zudem über

eine verringerte Inferenzzeit verfügt, wird diese Variante im Referenzmodell als Backbone verwendet.

Augmentation. Mit den verwendeten Erweiterungstechniken kann keine Verbesserung des mAP erreicht werden. Dies lässt sich darin begründen, dass Bilddaten durch die Erweiterung erzeugt werden, welche nicht im Testdatensatz enthalten sind. Für eine genauere Einschätzung müssen die Erweiterungen im einzelnen betrachtet werden.

ROI Dimension. Obwohl die Performanz des Modells bei dieser Konfiguration abnimmt, zeigt eine manuellen Evaluation, dass hiermit verhältnismäßig kleine Bienen detektiert werden können, die vorher nicht erkannt wurden. Diese Tatsache kann bei einer signifikanten Auswirkung auf den Zähl-Algorithmus näher untersucht werden.

Lernrate und Optimierer. Bei diesen Konfigurationen beweist sich die Kombination aus ADAM und einer Lernrate von 10^{-3} als beste Konfiguration dieser Arbeit. Lediglich für den mAP@0.9 zeigt der SGD mit einer Lernrate von 10^{-2} ein besseres Ergebnis. Somit besitzt dieses Modell $44.662.942$ Parameter, was in einer Speichergröße von 171MB resultiert. Die Inferenzzeit beträgt 265ms. Ein optisches Beispiel dieser Konfiguration ist in Abbildung 3 zu sehen.

Bestrafung der Parameternorm. Eine Verwendung der L^1-, statt der L^2-Norm, hat die Performanz des Netztes signifikant verschlechtert. Eine Kombination der beiden Normen führt zu einem minimal schlechteren Ergebnis.

Aggregation. Da die Konfigurationen "ROI Dimensionen" und "LR 10^{-3} (Adam)" die besten Ergebnisse liefern (Tab. I), wird auch eine Aggregation dieser beiden Konfigurationen untersucht. Die Ergebnisse sind in der vorletzten Zeile der Tabelle I dargestellt. Diese Aggregation übertrifft nicht die "LR 10^{-3} (Adam)" - Konfiguration.

Minimale Konfidenz. Die ausschließliche Berücksichtigung von Objekten mit einer Wahrscheinlichkeit von mind. 90% führt zu keiner Verbesserung der Performanz.

Inferenzzeit. Eine signifikante Veränderung der Inferenzzeit folgt einzig aus der Verwendung unterschiedlicher Backbones. Da diese Zeit abhängig von der berechnenden Hardware ist, geben die hier angeführten Werte lediglich ein relatives Verhältnis zu den Parameter- bzw. Architekturänderungen an.

Um die gesamte Berechnungszeit des EnBeeMo - Algorithmus in Betracht zu ziehen, inkl. Zeit für für Tracking (t_t) und Zählung (t_z), kann a priori Wissen genutzt werden. Unter Berücksichtigung der Abmaße der Bienen-Vorrichtung (100mm vertikal, Abbildung 3) und der mittleren Bewegungsgeschwindigkeit von Bienen ($30\frac{mm}{s}$ [9]), stehen im Extremfall ca. $1,3$s für die gesamte Berechnung zur Verfügung. Somit lässt sich die Anzahl der zu bearbeitenden Frames f bestimmen (Gleichung 2). Falls der Tracking-Algorithmus für diese Zeitspanne jedoch eine Mindestanzahl an Frames benötigt, um dessen Funktionalität zu gewährleisten, lässt sich hieraus die maximale Inferenzzeit $t_{inf,max}$ des neuronalen Netzes bestimmen:

$$f = \frac{1.3s}{\overline{t_{inf}} + \overline{t_t} + \overline{t_z}} \quad \Rightarrow \quad t_{inf,max} = \frac{1.3s}{f_{min}} - (\overline{t_t} + \overline{t_z}) \quad (2)$$

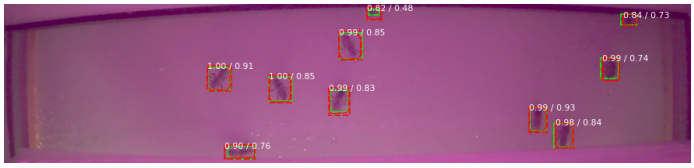

Abbildung 3: Vergleich - ■ Modellvorhersage ■ Annotation

V. Fazit und Ausblick

Abschließend lässt sich feststellen, dass die Performanz des Modells als zufriedenstellend zu betrachten ist. Damit eine absolute Aussage getroffen werden kann, muss die Inferenzzeit auf der Hardware bestimmmt werden, die im EnBeeMo - Projekt genutzt wird.

Um im Rahmen dieser Arbeit den Einfluss der einzelnen Parameter verstehen und nachvollziehen zu können, wurde bewusst auf eine automatisierte Optimierung verzichtet. Im Hinblick auf künftige Optimierungen jedoch, wäre eine automatisierte Optimierung zu empfehlen, bspw. eine Raster- oder Zufallssuche im Parameterraum, wobei diese Herangehensweise, als auch das manuelle Design der Netzwerk-Architektur, zeitaufwändige und fehleranfällige Prozesse sind. Um dieses Problem zu lösen, bieten sich im Bereich des *Neural Architecture Search* (NAS) modernere Verfahren an. Elsken et al. [10] geben einen Überblick über bestehende Arbeiten auf diesem Forschungsgebiet.

Bezüglich der Inferenzzeit, wären ebenfalls Verbesserungen möglich. Denkbar wäre die Verwendung von NVIDIA-TensorRT [11], womit neuronale Netze im Anschluss an das Training optimiert und dadurch deren Inferenzzeit beschleunigt werden können.

Literatur

[1] Kaiming He u. a. „Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[2] Olaf Ronneberger, Philipp Fischer und Thomas Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Hrsg. von Nassir Navab u. a. Cham: Springer International Publishing, 2015, S. 234–241. ISBN: 978-3-319-24574-4.

[3] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN. 2017.

[4] Ross B. Girshick u. a. „Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[5] Ross B. Girshick. „Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[6] Shaoqing Ren u. a. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[7] K. He u. a. „Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, S. 770–778. DOI: 10.1109/CVPR.2016.90.

[8] T. Lin u. a. „Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, S. 936–944. DOI: 10.1109/CVPR.2017.106.

[9] Morgane Nouvian und C. Giovanni Galizia. „Aversive Training of Honey Bees in an Automated Y-Maze". In: *Frontiers in Physiology* 10 (2019), S. 678. ISSN: 1664-042X. DOI: 10.3389/fphys.2019.00678. URL: https://www.frontiersin.org/article/10.3389/fphys.2019.00678.

[10] Thomas Elsken, Jan Hendrik Metzen und Frank Hutter. *Neural Architecture Search: A Survey*. 2019. arXiv: 1808.05377 [stat.ML].

[11] *NVIDIA TensorRT*. Jan. 2021. URL: https://developer.nvidia.com/tensorrt.

Kommentar der Autoren: Die inhaltliche Umsetzung und schriftliche Ausarbeitung wurde von allen Personen zu gleichen Teilen bearbeitet.