Entwicklung eines Algorithmus zur Erkennung von Bienen

Valentino Golob
Hochschule München
80335, München, Deutschland
valentino.golob@hm.edu

***Zusammenfassung*—Dieser Bericht beschreibt den Modellaufbau eines modifizierten YOLO Algorithmus zur Detektion von Bienen. Insbesondere wird erörtert, wie sich die Wahl der Hyperparameter auf die Erkennungsleistung auswirkt.**

***Index Terms*—mod. YOLO, Convolutional Neuronal Network, Bee Monitoring**

I. Einführung

Gegenwärtig ist allein eine subjektive Aussage zu einem Bienenstand und dessen Korrelation zu diversen Umweltfaktoren möglich. [7] Um zukünftig auch objektive Korrelationen ableiten zu können, werden die einzelnen Bienen mithilfe eines Convolutional Neuronal Networks detektiert.

II. Konzept

Im Rahmen der Modularbeit wird eine angepasste Variante des YOLO Networks von Redmon [2] angewendet. Bei der Bearbeitung wurde das GitHub Repository von Persson [8] herangezogen und an erforderlichen Stellen angepasst und erweitert.

A. Datensatz

Der Datensatz umfasst insgesamt 2000 gelabelte Bilder. Die Positionsbestimmung der Bounding Boxes (BB) in dem unbearbeiteten Datensatz erfolgt in absoluten Pixel Koordinaten. Um die Anwendung und Implementierung bestehender Algorithmen zu erleichtern, werden die Werte zur Bestimmung der lokalen Position und Größe der BB in einem Preprocessing Schritt in relative Koordinaten umgewandelt. Zudem werden x und y Koordinaten zur Bestimmung der linken oberen Ecke der BB so umgewandelt, dass diese die Position der mittleren Koordinaten der BB festlegen. Der gesamte Datensatz wird nicht wie von Goodfellow empfohlen in einen Train-, Validation- und Testdatensatz unterteilt. Aufgrund der geringeren Größe des Datensatzes beschränken wir uns auf eine Unterteilung in lediglich zwei disjunkte zufällig bestimmte Untermengen. Die erste Untermenge umfasst 1600 Trainingsdaten. Mit den Traingsdaten werden die Parameter des neuronalen Netzes gelernt. Die zweite Untermenge mit insgesamt 400 Bildern ist unser Testdatensatz. Mithilfe des Testdatensatzes schätzen wir den Generalisierungsfehler ein und versuchen so durch eine entsprechende Wahl der Hyperparameter eine Überanpassung des neuronalen Netzes auf die Trainingsdaten zu verringern. Da in der Unterteilung der Daten lediglich zwei statt drei disjunkte Untermengen des gesamten Datensatzes verwendet wurden, wird der Generalisierungsfehler auf dem Testdatensatz voraussichtlich unterschätzt. Weiter gilt es zu Beachten, dass der Testdatensatz eine relativ geringe Anzahl an Bildern enthält. Folglich ergibt sich eine statistische Unsicherheit bezüglich der entsprechenden Leistungsmetriken und somit gilt es die Leistungsmetriken der verschiedenen Algorithmen mit einer entsprechenden Vorsicht einzuordnen. K-Fold-Crossvalidation ist eine Möglichkeit diese Unsicherheit zu minimieren. Die Anwendung dieser Methode bedarf jedoch einen relativ hohen zusätzlichen Rechenaufwand und wird im Rahmen dieser Arbeit nicht verwendet. [1]

B. Architektur

Im Rahmen der Arbeit verwenden wir ein Convolutional Neuronal Network, welches sich stark an dem YOLO Network von Redmom [2] orientiert. In der angepassten Version des Netzwerks verwenden wir nicht wie in der ursprünglichen Version 24, sondern lediglich 15 Convolutional Layer, gefolgt von zwei Fully-Connected Layern (siehe Abbildung 1).

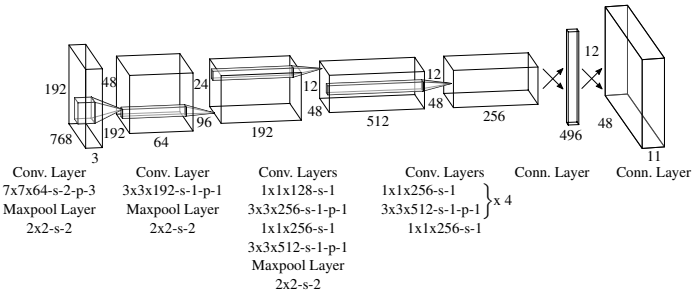

Abbildung 1. Architektur des Neuronalen Netzes zur Objekterkennung

Die YOLO Architektur ist in ihrer Vorgehensweise relativ einfach. Die Grundidee der Methode ist es, ein eingelesenes Bild in ein $S_H \times S_W$ Gitter zu unterteilen. Befindet sich die Mitte eines Objektes in einer Gitterzelle, ist die entsprechende Gitterzelle für die Detektion des Objektes verantwortlich. Die einzelnen Gitterzellen treffen eine Vorhersage über jeweils B Bounding Boxen (x_c, y_c, h, w) und der zugehörigen Klassenwahrscheinlichkeit (c) der entsprechenden BB. Die Vorhersage erfolgt demnach mithilfe eines $S_H \times S_W \times (B * 5 + C)$ Tensors. In der originalen YOLO Version werden die Prognosen mit Hilfe eines 7×7 Gitter (Split-Size) getroffen. Jedoch ist bekannt, dass die ursprüngliche Auslegung des YOLO Algorithmus, Probleme hinsichtlich einer präzisen Lokalisierung

von kleinen Objekten aufweist. Um dieser Eigenschaft entgegenzuwirken, ermöglichen wir eine feinere Diskretisierung der Gitterzellen und verwenden in der modifizierten YOLO Ausführung ein 12×48 Gitter. Die finale Prognose erfolgt in der angepassten Architektur dementsprechend mittels einer 12×48×11 Matrix. Zudem werden die eingelesenen Bilder auf eine Größe von 192×768 (H×W) angepasst. Auf eine Umwandlung der ursprünglichen rechteckigen Bilder mit einer Größe von 371×1613 auf eine quadratische Form wird verzichtet. Die entsprechende Transformation würde eine erhebliche Verzerrung der Bilder bewirken. Eine Möglichkeit, diesen Verzerrungen entgegenzuwirken ist es, die Bilder unter Beibehaltung der ursprünglichen Seitenverhältnisse zu einer quadratischen Form aufzufüllen. Ein Nachteil dieser Methode ist es, dass die aufgefüllten Bereiche keinen Informationsgehalt besitzen. [2] Um eine Ausgangslage für eine nachstehende Optimierung der Hyperparameter zu gewährleisten, werden in einem ersten Schritt wenige Variationen der Input-Size der eingelesenen Bilder, der Hidden Size der Fully-Connected Layer und der Learningrate des Optimiereralgorithmus (Adam) berechnet (siehe Abbildung 2).

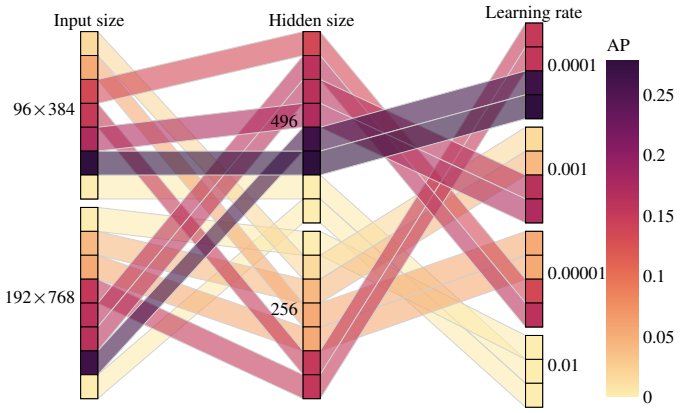

Abbildung 2. Einfluss der der Inputsize, Hidden size und Learningrate auf die Average Precision (AP_{50}) auf den Testdaten

Objekt Detektion benötigt weitgehend fein aufgelöste Bilder. Diese Erfahrung wird auch in der Abbildung 2 bestätigt. Demnach korreliert eine Vergrößerung der Input-Size mit einer erhöhten Average Precision. Die Auswahl einer größeren Anzahl and Hidden-Units erhöht die Kapazität eines neuronalen Netzwerkes. Neben einer gesteigerten Modellkomplexität erhöht eine größere Zahl an Hidden-Units die Trainings- und Inferenzzeit sowie den Speicherbedarf. Eine feinere Auflösung der Bilder oder die Wahl einer größeren Hidden-Size führte im Rahmen der Arbeit vermehrt zu einer »out of memory« Fehlermeldung. Folglich wurde bei der nachstehenden Optimierung der Hyperparameter eine Input-Size von 192×768 und ein Hidden-Size von 496 herangezogen. Eine Learningrate von 0.0001 führte bei den zugehörigen Vorhersagen zu den höchsten Werten für die Average Precision.

C. Regularisierung

In die bisherigen Berechnungen wurde bereits eine vereinfachte Variante der Earlystopping Methode implementiert. Der Algorithmus wird abgebrochen, sobald sich in 50 aufeinanderfolgenden Epochen keine weitere Minimierung des Testfehlers ergibt. Die Average Precision wird anschließend mit der Gewichtung der Modellparameter bestimmt, für welche sich der minimale Testfehler ergeben hat. Das bisherige Modell ist stark überangepasst. Es liefert einen geringen Trainingsfehler bei einem zeitgleich vergleichsweise hohen Generalisierungsfehler.

a) Data Augmentation: Die starke Überanpassung wird besonders durch den kleinen Datensatz begünstigt. Die Trainingsdaten werden hierbei teilweise auswendig gelernt. Eine wirksame Methode der Überanpassung entgegenzuwirken ist es, den Datensatz zu erhöhen. In der Praxis steht jedoch oft lediglich ein begrenzter Datensatz zur Verfügung. Eine Möglichkeit, den Generalisierungsfehler dennoch zu minimieren ist es, den Datensatz durch geometrische Transformationen künstlich zu vergrößern. [3], [4] In dieser Arbeit beschränken wir uns auf horizontale und vertikale Spiegelungen. Zudem wird untersucht, welchen Einfluss es mit sich bringt, wenn die farbig eingelesenen Bilder sowohl im Trainingsmodus als auch in der Auswertung in Graustufen Bilder umgewandelt werden (siehe Abbildung 3)

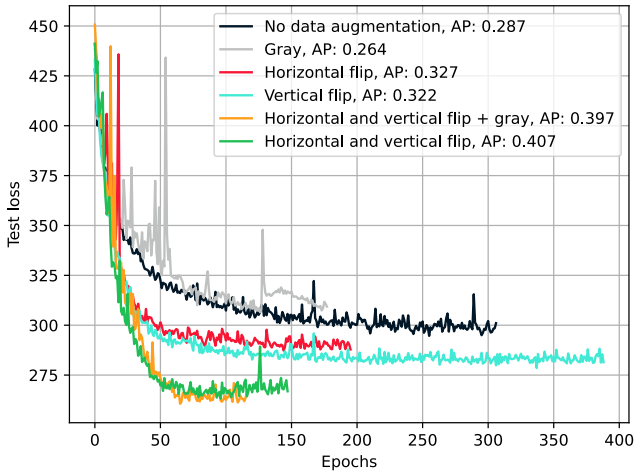

Abbildung 3. Einfluss der künstlichen Datenerhöhung auf den Testfehler und die Average Precision (AP_{50})

In der Abbildung 3 ist zu erkennen, dass die horizontale und vertikale Spiegelung den Generalisierungsfehler in einem ähnlichen Maße beeinflussen. Die Kombination der beiden Transformationen liefert den höchsten Wert für die Average Precision. Die Spiegelungen werden je Batch Size jeweils mit einer Wahrscheinlichkeit von 50% auf dem Trainingsdatensatz angewendet. Die Umwandlung der farbigen Bilder in Graustufen bewirkt keine Reduzierung der Überanpassung. Die Anwendung weiterer Techniken der Data Augmentation im Bereich der geometrischen Transformationen (rotieren, zuschneiden, horizontal oder vertikal verschieben) ist vielversprechend. Bei der Anwendung der entsprechenden Trans-

formationen müssen die Bounding Boxen teilweise manuell nachbearbeitet werden. Somit wird im Rahmen dieser Arbeit auf den Einsatz weiterführender Transformationen verzichtet. In unserer Analyse bewirkt die zufällige Kombination horizontaler und vertikaler Spiegelungen der Trainingsdaten die größte Reduzierung des Testfehlers. Die vertikalen und horizontalen Spiegelungen verringern den Generalisierungsfehler jeweils im gleichen Maße.

b) Dropout und Weight Decay: Um den Generalisierungsfehler weiter zu minimieren, werden verschiedene Parametervariationen der Dropout-Rate an den Fully-Connected Layern und der L2-Regularisierung des Adam Optimierungsalgorithmus untersucht. [5] Die L2-Regularisierung des Adam Optimierungsalgorithmus erfolgt entsprechend der empfohlen Veränderungen in ≫Decoupled Weight Decay≪ [6]. In einem ersten Schritt wurde mithilfe eines Gridsearchverfahren erörtert, dass die Anwendung der Dropout Methode den Testfehler respektive die Überanpassung nicht weiter reduziert. Indessen konnte der Testfehler unter Anwendung der modifizierte L2-Regularisierung [6] weiter minimiert werden (Erhöhung der AP_{50}).

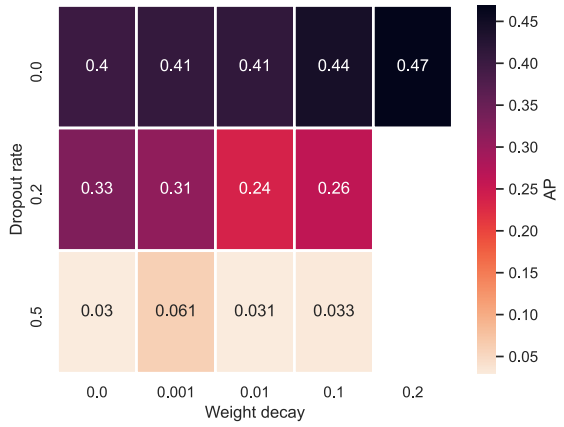

Abbildung 4. Dropout und Weight Decay, (AP_{50})

Mithilfe einer Bayes'schen Hyperparameteroptimierung wurde der optimale Parameter für die angepasste L2-Regularisierung in einem Bereich von 0.1 bis 10 untersucht. Folglich erhalten wir einen maximalen Wert der Average Precision für einen Weight-Decay Parameter von 0.2 (siehe Abbildung 4).

c) Early Stopping: In den vorangegangenen Berechnungen wurde bereits eine Variante der Earlystoppingmethode implementiert. In der Abbildung 5 ist dargestellt, wie sich der Trainings- und Testfehler ohne die Implementierung einer Earlystoppingmethode entwickelt.

D. Evaluation

In den vorangegangenen Betrachtungen wurde der Einfluss verschiedener Hyperparameter ermittelt. Insbesondere eine geeignete Anpassung der Learningrate, der Hidden-Size und des Weight-Decay Parameters ist wesentlich für die Optimierung der modifizierten YOLO Netzwerkarchitektur. Auf eine

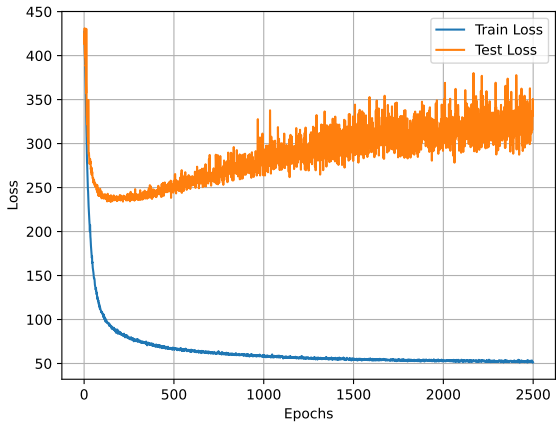

Abbildung 5. Early Stopping

Anwendung der Dropout Methode in den Fully-Connected-Layern sollte verzichtet werden. Eine Implementation des Earlystopping Verfahrens sollte verwendet werden. Unter Anwendung dieser Methode ist es möglich, den Zeitaufwand der Berechnungen deutlich zu senken. Besonders vielversprechend ist der Einsatz zusätzlicher geometrischer Transformationen, die den Datensatz künstlich erhöhen. Die nachfolgende Tabelle stellt die Parameter des abgestimmten modifizierten YOLO Netzwerkes dar.

Tabelle I
PARAMETER MOD. YOLO NETZWERK

Input-Size	192×768	**Data Augmentation**	H+V Spiegelung
Hidden-Size	496	**Weight-Decay**	0.2
Learningrate	0.0001	**Dropout-Rate**	0.0
Split-Size	12×48	**Earlystopping**	50 Iterationen

Mit dem entsprechend optimierten Modell erhalten wir für eine Intersection-Over-Union (IoU) von 50% und 75% folgende Precision-Recall Kurven.

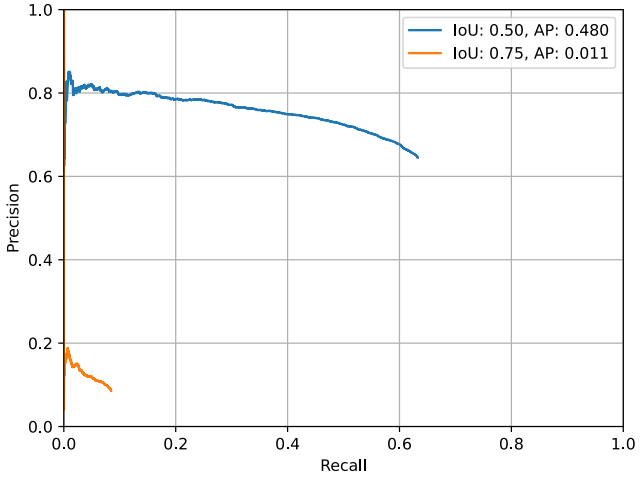

Abbildung 6. Average Precision A_{50}, A_{75}

Mit den in der Tabelle I beschriebenen Parameter erhalten wir folgenden Werten für die Average Precision bei einer Intersection-Over-Union von 50% und 75% und die Ausführgeschwindigkeit des Modells ≫frames per second≪ (FPS).

Tabelle II
PERFORMANCE UND GESCHWINDIGKEIT DES MOD. YOLO NETWERKES

Method	AP_{50} [%]	AP_{75} [%]	FPS
mod. YOLO	48.0	1.1	2.3

Der Einfluss der Anzahl an Convolutional Layer, der Convolutional-Kernel-Width, der vorhergesagten Bounding Boxen je Gitterzelle und der Split-Size wurde im Rahmen der Projektarbeit nicht erfasst.

E. Einschränkungen des YOLO Algorithmus

Da jede Gitterzelle für sich lediglich zwei Bounding Boxen vorhersagt, unterliegt der YOLO Algorithmus in gewisser Hinsicht einer ≫räumlichen≪ Zwangsbedingung. Angesicht dessen wird die mögliche Anzahl an detektierbaren Objekten in nächster Nähe zueinander beschränkt. [2] Diese Einschränkung hat einen negativen Einfluss auf die Detektion von kleineren Objekte, die sich relativ nah beieinander an einem gemeinsamen Ort ansammeln (siehe Abbildung 7).

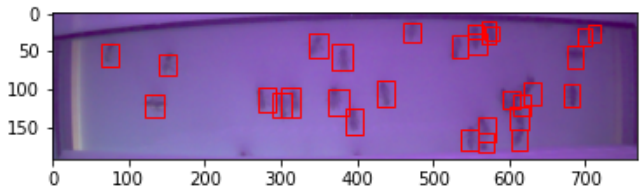

Abbildung 7. Vorhersage der Bounding Boxen

Da in der Netzwerkarchitektur verhältnismäßig viele ≫downsampling Layer≪ eingesetzt werden, verwendet das Modell grobe Features, um die Bounding Boxen vorherzusagen. [2]

III. AUSBLICK

Es ist bekannt, dass der YOLO Algorithmus erhebliche Schwierigkeiten bei der Lokalisierung von kleineren Objekten hat. Die Fast R-CNN Methode hingegen führt zu einer geringeren Fehlerrate bei der Lokalisierung von kleineren Objekten (siehe Abbildung 8). Allerdings verzeichnet das Fast R-CNN Verfahren eine höhere Fehlerrate in Bezug auf den Hintergrund des zugrunde liegenden Bildes (siehe Abbildug 8). [2] Die im Rahmen der Studienarbeit verarbeiteten Datensätze haben relativ homogene Hintergrundbedingungen. Somit ist davon auszugehen, dass eine höhere Fehlerrate in Bezug auf den Hintergrund der zugrunde liegenden Bilder bei der uns vorliegenden Problemstellung zu keinen großen Auswirkungen in der Fehlerrate führen würde. Infolgedessen scheint in zukünftigen Analysen eine Untersuchung der vorgegebenen Problemstellung mithilfe der Fast R-CNN Methode sinnvoll.

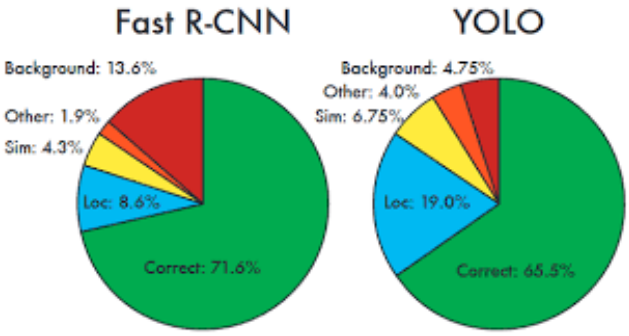

Abbildung 8. Error Analysis: Fast R-CNN vs. YOLO [2]

LITERATUR

[1] Ian J. Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, Cambridge, MA, USA, 2016.
[2] Joseph Redmon and Santosh Divvala and Ross Girshick and Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection, 2015
[3] François Chollet, Deep Learning with Python, Manning, 2017
[4] Aurelien Geron, Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. Sebastopol, CA : O Reilly Media, 2017
[5] Srivastava, Nitish and Hinton, Geoffrey E. and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, Dropout: a simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 2014
[6] Ilya Loshchilov and Frank Hutter, Fixing Weight Decay Regularization in Adam, dblp computer science bibliography, 2018
[7] Lorin Arndt, Entwicklung eines Pareto-Optimalen Algorithmus zur Erkennung von Bienen, Hochschule München, 2020
[8] Aladdin Persson, YOLO, https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/object_detection/YOLO, 2020