

Final Assignment

Study Project: Machine Learning meets Insect Monitoring

University of Münster
Institute for Geoinformatics

First lecturer:
Prof. Dr. Benjamin Risse

Second lecturer:
Sebastian Thiele

Semester:
WS 2023/2024

Submitted by:
Maximilian Elfers
??? ???

Timo Lietmeyer
459 169

Hendrik Lüning
514 910

Münster, March 2024

Contents

1	Introduction	2
2	Related Work	3
3	Fundamentals	4
3.1	Yolo	4
3.1.1	Der YOLO Algorithmus	4
3.1.2	YOLOv8 von Ultralytics	6
3.2	OpenCV	8
3.2.1	Computer Vision 2	8
3.3	Palma	8
3.4	The camera trap	8
4	Methods	9
4.1	Converting Bounding boxes	9
4.2	computer vision methods	10
4.2.1	Background subtraction	10
4.2.2	RGB to HSV	10
4.2.3	Image Subtraction	10
4.2.4	Time offset	10
4.2.5	Combination of Methods	10
5	Results	11
5.1	Training with RGB data	11
5.2	Training with Background subtraction	11
5.3	Training with Time offset	11
6	Discussion	12
6.1	Comparison of Models	12
7	Contributions by group members	13
7.1	Maximilian Elfers	13

7.2	Timo Lietmeyer	13
7.3	Hendrik Lüning	13
List of Figures		14
List of Tables		15

1 | Introduction

This final assignment of the course "Study Project: Machine Learning meets Insect Monitoring" at the Institute for Geoinformatics at the University of Münster held by Prof. Dr. Benjamin Risse and Sebastian Thiele is about the integration and development of a machine learning model for the detection of insects in videos captured by a in-house constructed camera trap. The camera trap is part of the Computer Vision and Machine Learning Systems Group of Prof. Dr. Benjamin Risse and it delivered the data for this project [1].

Insects are an important part of the ecosystem and are essential for the pollination of plants, the food chain and pest control. However, the populations of insects are drastically declining [2]. To understand the causes of the decline and to develop strategies to protect insect populations, monitoring is important. Monitoring is challenging because insects are small and difficult to observe and identify. Valide data and time series are needed to understand the causes of the decline and to understand the actual state of the insect populations. Current monitoring methods are time-consuming and expensive, e.g. malaise traps or light traps. Therefore, the development of a camera trap for insects is a promising approach to monitor insect populations.

The study project goal is to train models, which can detect insects in videos, with different strategies, to evaluate the performance of the models and find the best methods for insect detection. For this endeavour, this group used the RGB stream of the camera trap and developed different approaches to preprocess the videos to get a better detection of the tiny insects.

After outlining the motivation and goal of this project in the first chapter, the current state of research in the field of insect detection and monitoring is presented in chapter 2. The fundamentals of the machine learning models used and the computer vision library are introduced in chapter 3. In chapter 4 the methods used in this project are explained. The results are discussed in the chapter 6. The contributions of the group members are summarized in the chapter 7.

2 | Related Work

Insect monitoring is a field that invokes a lot of interest in the scientific community. Therefore a lot of studies were conducted in the past.

A study on automated monitoring was able to achieve good results using only one camera that was connected to a computer and a deep learning algorithm. In this setup the images were captured and delivered to the computer for preprocessing. After that the deep learning algorithm was used to analyze these images. The results were promising, achieving a confidence level of at least 70% [3].

Therefore this model seems to be an effective low-cost solution for monitoring insects. The study also showed concerns for the model to be used in the field, as the manual focus of the camera would lead to massive data losses [3].

Another study detecting small traffic signs achieved great results using the YOLOv7 algorithm. It was shown that the model was able to achieve a $\text{mAP}@0.5$ of 88.7% for the improved YOLOv7 algorithms. But also the basic algorithm achieved a $\text{mAP}@0.5$ of 83.2% [4].

This shows that there were good results with simple models for insect detection and also that the YOLO algorithm is a good choice for object detection, therefore it will also be used in this study.

3 | Fundamentals

3.1 Yolo

3.1.1 Der YOLO Algorithmus

Da der Blick des Menschen Objekterkennung, -einordnung und -wirkung intuitiv ermöglicht, ist es unserem Gehirn im Zusammenspiel mit unseren Augen möglich, schnell und genau zu sehen. Durch diese Fähigkeiten können wir mit nur wenig bewussten Gedanken komplexe Aufgaben, wie Fahrradfahren bewältigen, bei denen gleichzeitig mehrere Sinne beansprucht werden. [5]. Dem Computer kann dies mit schnellen und genauen Algorithmen zur Objekterkennung beigebracht werden. Aktuelle Systeme nutzen Klassifikatoren zur Objekterkennung. Dieser wird an verschiedenen Stellen in variablen Skalierungen im Testbild angewendet, um eine Klassifizierung eines Objektes zu ermöglichen [5].

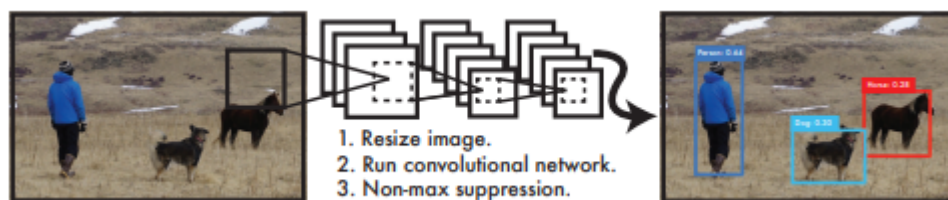


Figure 3.1: Das YOLO Objekterkennungssystem [5]

„You Only Look Once“ (YOLO) betrachtet Objekterkennung als einzelnes Regressionsproblem, indem direkt von Bildpixel zu Boundingbox Koordinaten und Klassenwahrscheinlichkeiten berechnet wird. Dieser Algorithmus analysiert nur einmal ein Bild und sagt direkt vorher, welche Objekte wo vorhanden sind. Dadurch ist die Komplexität des Aufbaus von YOLO sehr gering, wie in Abb. 3.1 zu sehen [5].

Die Performance zur Objekterkennung wird durch das Training von YOLO mit vollständigen Bildern gesteigert. Durch dieses vereinheitlichte Modell entstehen mehrere Vorteile gegenüber den traditionellen Objekterkennungssystemen [5].

Der erste Vorteil von YOLO ist die gesteigerte Performance. Dies wird dadurch ermöglicht, dass Objekterkennung auf Bildern als Regressionproblem betrachtet wird und deshalb keine komplexe Pipeline die Verarbeitung eines Bildes verlangsamt. [5]. Zweitens analysiert YOLO ein Bild global mit Vorhersagen zur Objekterkennung. Dadurch kann YOLO den Fehler beim Verwechseln von Hintergrund und Objekten im Vordergrund um die Hälfte im Vergleich zu Fast R-CNN verringern. Dies geschieht vor allem durch den größeren Kontext, den YOLO durch die Gesamtbildanalyse gewinnt [5].

Der dritte Vorteil ist das YOLO mit generalisierten Repräsentationen von Objekten trainiert wurde um die Fehlertoleranz bei der Anwendung auf neue Bereiche und unerwartete Eingaben zu vergrößern, aufgrund der Möglichkeit der hohen Verallgemeinerung [5].

Ein Nachteil von YOLO liegt in der Genauigkeit. Der Algorithmus hat Schwierigkeiten einige, insbesondere kleine, Objekte genau zu lokalisieren [5].

Da der Quellcode, mehrere vortrainierte Modelle und die Trainingsdaten von YOLO Open-Source sind und zum Download bereitstehen, ist dieser Algorithmus für den Rahmen dieser Arbeit leicht zugänglich und anwendbar [5].

YOLO unterteilt in Bild in $S \times S$ Rasterzellen. Wenn der Mittelpunkt eines Objektes in eine Rasterzelle fällt, ist diese für die Erkennung des Objektes zuständig. Boundingboxen und ihre jeweiligen Confidence Scores werden für jede Rasterzelle vorhergesagt [5]. Der Confidence Score beschreibt, wie sicher sich das Modell ist, dass die Boundingbox ein Objekt dieser Klasse enthält und für wie genau das Modell diese Vorhersage hält.

Dieser Wert enthält nicht nur die Wahrscheinlichkeit, dass diese Klasse in der Boundingbox vorkommt, sondern auch wie gut die vorhergesagte Box mit dem detektierten Objekt übereinstimmt. Ein Beispielablauf ist in Abb. 3.2 zu sehen.

Da jede Rasterzelle nur 2 Boundingboxenvorhersagen und eine Klasse haben kann, unterliegt YOLO einer räumlichen Einschränkung. Dies begrenzt die Anzahl der benachbarten Objekte, die das Modell vorhersagen kann. Außerdem ist es für das Modell schwierig, kleine Objekte, die in Gruppen auftreten, zu detektieren [5].

Eine weitere Herausforderung ist, dass Objekte mit neuen oder ungewöhnlichen Formen auftreten können und dadurch die Vorhersage erschwert wird. Da die Netzwerkarchitektur aus mehreren „Downsampling“-Schichten besteht, benutzt das Modell relativ grobe Features zur Vorhersage der Boundingboxen [5].

Außerdem sorgt das Training mit der YOLO spezifischen Verlustfunktion (für eine Erklärung s. Formel ??, S. ?? und Abb. ??, S. ??), die die Erkennungsleistung annähert, dafür dass Fehler bei kleinen Boundingboxen genauso wie bei großen Boundingboxen behandelt werden. Dies ist ein Nachteil, weil ein kleiner Fehler in einer großen Boundingbox meistens wenig Auswirkungen hat, aber ein kleiner Fehler in einer kleinen Boundingbox eine sehr viel größer Auswirkung auf die IOU

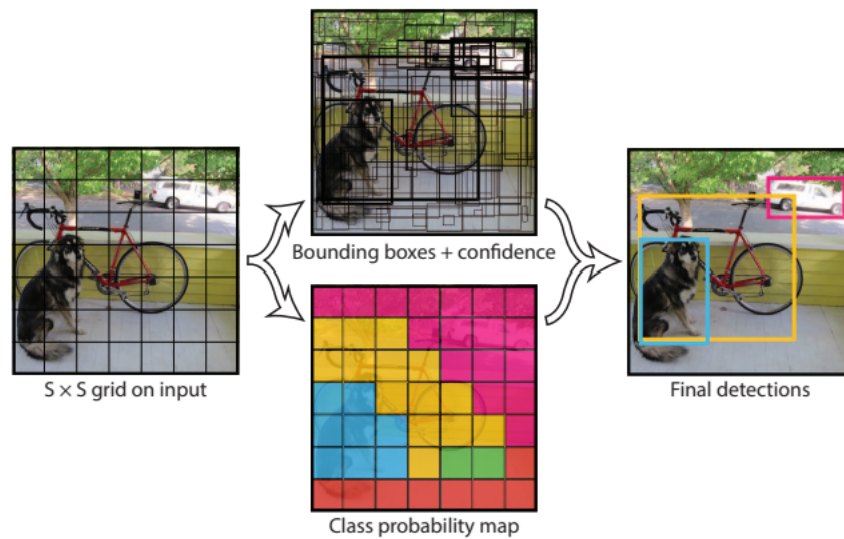


Figure 3.2: Das YOLO Modell[5]

hat. Falsche Lokalisierungen sind eine weitere Hauptfehlerquelle [5].

YOLO ist für den Anwendungszweck dieser Arbeit geeignet, weil der Algorithmus die entsprechende Performance und einfache Verfügbarkeit von trainierten Modellen bietet. Außerdem existieren mehrere weitere Versionen (s. Abb. ??, S. ??) von YOLO, die Vorteile in einzelnen Aspekten bieten [6].

Im Rahmen dieser Arbeit wird YOLOv8 von Ultralytics verwendet. Diese bietet Vorteile in der Performance und Genauigkeit der Objektdetektion und wird im nächsten Kapitel genauer erläutert.

3.1.2 YOLOv8 von Ultralytics

Im Januar 2023 wurde von der Firma Ultralytics YOLOv8 veröffentlicht, welches auf YOLOv5 basiert. Diese Version beinhaltet 5 verschiedene Modelle (YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), YOLOv8x (extra large)), die mit unterschiedlich großen Datensätzen trainiert wurden [6].

Ein Vorteil dieser YOLO Implementierung ist, dass verschiedene Varianten für Objektdetektion, -segmentierung und -verfolgung, sowie -klassifizierung existieren. In dieser Arbeit wird hauptsächlich die „-seg“-Variante verwendet, welches bereits eine Segmentierung der Umrisse der detektierten Objekte integriert hat.

Die Architektur dieser Algorithmen kann man in 3 Teile aufteilen (s. Abb. 3.3). Diese sind der Backbone, der Neck und der Head [6].

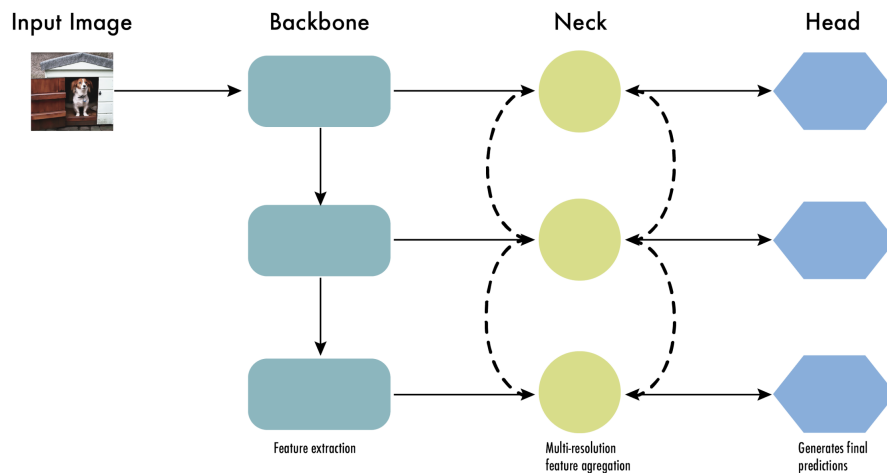


Figure 3.3: Architektur modernen Objektdetektoren [6]

Das Detektieren nützlicher Features vom Eingabebild geschieht im Backbone, welcher meist als CNN implementiert ist [6].

Zwischen Backbone und Head wird der Neck eingesetzt, um die Features, die der Backbone ausgibt, zu aggregieren und zu verfeinern. Der Fokus liegt auf der Verbesserung der räumlichen und semantischen Informationen über die unterschiedlichen Skalierungen hinweg [6].

Die letzte Komponente ist der Head, welcher die Vorhersagen, aufgrund der von dem Backbone und Neck gelieferten Features, trifft. Hier werden meistens aufgabenspezifische Teilnetze eingesetzt, um Klassifizierung, Lokalisierung und auch sofortige Segmentierung durchzuführen. Aus den Features, die der Neck liefert, erstellt der Head Vorhersagen für jeden Objektkandidaten. Ein Post-Processing Schritt, wie die Non-Maximum-Supression (NMS), filtert überlappende Vorhersagen heraus, sodass nur die sichersten Detektionen genutzt werden [6].

Da YOLOv8 auf YOLOv5 basiert, wird in diesem ein ähnlicher Backbone genutzt. Für die Architektur von YOLOv8 s. Abb. ?? (S. ??).

Um die Performance insbesondere bei der Objekterkennung von kleineren Objekten zu verbessern, nutzt YOLOv8 CloU [7] und DFL [8] Verlustfunktionen für Boundingboxloss und binäre Kreuzentropie für den Klassifizierungsloss [6].

Mit dem YOLOv8-seg Modell wird auch eine Variante angeboten, die semantische Segmentierung ermöglicht. Dieses Modell wird in dieser Arbeit genutzt, da es den Anforderungen an Präzision und Performance entspricht. An den Einstellungen des YOLO-Modells, wie bspw. den Werten des Ausgabensensors, wird nichts geändert. Es werden die von Ultralytics vorgegebenen Standardeinstellungen und Modelle genutzt.

3.2 OpenCV

3.2.1 Computer Vision 2

Computer Vision 2 (CV2) ist ein Teil der „Open Source Computer Vision“ Bibliothek [9]. Die aktuelle Version 4.8.0 wurde am 02.07.2023 veröffentlicht [10].

Diese Bibliothek beinhaltet Algorithmen zur Bild- und Videobearbeitung. In dieser Arbeit wird diese Bibliothek zur Manipulation von Frames in Videos genutzt, die zuvor mit YOLO analysiert wurden.

3.3 Palma

3.4 The camera trap

- what is a camera trap
- how does it work
- the two cameras (rgb and event stream(?))

4 | Methods

4.1 Converting Bounding boxes

The process of adapting insect labels from DVS streams to RGB data involved several steps to ensure accurate alignment. First, bounding boxes were applied to the DVS data only. However, due to differences in image capture between the event-based camera and the RGB camera, as well as differences in resolution (1920x1200 for RGB versus 1280x720 for DVS), a conversion process was required.

Several iterations of conversion were attempted before success was achieved. The first attempt involved a simple translation of the bounding boxes by one unit in the x and y axes. Subsequent analysis revealed that the DVS stream captured an enlarged section of the scene. To address this discrepancy, a second version of the conversion was devised that scaled the box coordinates relative to their proximity to the centre of the image.

Despite improvements, problems persisted, particularly at the edge of the image, where distortion was more pronounced. As a result, a new approach was adopted using a homographic transformation to adjust for image distortion. This transformation produced a homography matrix that provided insight into the distortion between the RGB and DVS images.

Ultimately, the final conversion method used the homography matrix to adjust the data, with additional scaling along the x and y axes to ensure alignment. This approach effectively harmonised the insect labels between the DVS and RGB streams.

4.2 computer vision methods

4.2.1 Background subtraction

4.2.2 RGB to HSV

4.2.3 Image Subtraction

4.2.4 Time offset

4.2.5 Combination of Methods

5 | Results

5.1 Training with RGB data

5.2 Training with Background subtraction

5.3 Training with Time offset

6 | Discussion

6.1 Comparison of Models

7 | Contributions by group members

7.1 Maximilian Elfers

7.2 Timo Lietmeyer

7.3 Hendrik Lüning

List of Figures

3.1	Das YOLO Objekterkennungssystem	4
3.2	Das YOLO Modell	6
3.3	Architektur modernen Objektdetektoren	7

List of Tables

Bibliography

- [1] Institute for Geoinformatics, *Computer vision and machine learning systems*. [Online]. Available: <https://www.uni-muenster.de/Geoinformatics.cvm/ls/> (cit. on p. 2).
- [2] D. L. Wagner, E. M. Grames, M. L. Forister, M. R. Berenbaum, and D. Stopak, "Insect decline in the anthropocene: Death by a thousand cuts," *Proceedings of the National Academy of Sciences*, vol. 118, no. 2, e2023989118, 2021 (cit. on p. 2).
- [3] Q. A. Mendoza, L. O. Pordesimo, M. Neilsen, P. R. Armstrong, J. F. Campbell, and P. T. Mendoza, "Application of machine learning for insect monitoring in grain facilities," *AI*, vol. 4, no. 1, pp. 348–360, Mar. 22, 2023. DOI: 10.3390/ai4010017. [Online]. Available: <https://doi.org/10.3390/ai4010017> (cit. on p. 3).
- [4] S. Li, S. Wang, and P. Wang, "A small object detection algorithm for traffic signs based on improved yolov7," *Sensors*, vol. 23, no. 16, p. 7145, Aug. 13, 2023. DOI: 10.3390/s23167145. [Online]. Available: <https://doi.org/10.3390/s23167145> (cit. on p. 3).
- [5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 779–788, ISBN: 978-1-4673-8852-8. DOI: 10.1109/CVPR.2016.91 (cit. on pp. 4–6).
- [6] J. Terven and D. Cordova-Esparza, "A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond," pp. 1–33, 2023. arXiv: 2304.00501. [Online]. Available: <http://arxiv.org/abs/2304.00501> (cit. on pp. 6, 7).
- [7] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, no. 2, pp. 12 993–13 000, 2020, ISSN: 2159-5399. DOI: 10.1609/aaai.v34i07.6999. arXiv: 1911.08287 (cit. on p. 7).
- [8] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 2020-December, pp. 1–14, 2020, ISSN: 10495258. arXiv: 2006.04388 (cit. on p. 7).

- [9] *About - OpenCV*. [Online]. Available: <https://opencv.org/about/> (visited on 07/11/2023) (cit. on p. 8).
- [10] *Releases - OpenCV*. [Online]. Available: <https://opencv.org/releases/> (visited on 07/11/2023) (cit. on p. 8).

Declaration of Academic Integrity

We hereby confirm that this paper on

Final Assignment

Study Project: Machine Learning meets Insect Monitoring

is solely our own work and that we have used no sources or aids other than the ones stated. All passages in our paper for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

Maximilian Elfers, Münster, March 24, 2024

Timo Lietmeyer, Münster, March 24, 2024

Hendrik Lüning, Münster, March 24, 2024