Universität Münster

ifgi
Institut für Geoinformatik
Universität Münster

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

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 interst in the scientific community. Therefore a lot of studies were conducted in the past.

A study on automated monitoring was able to achive 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, achiving a confidence level of at least 70% [3].

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

Another study detecting small traffic signs achived great results using the YOLOv7 algorithm. It was shown that the model was able to achive a MaP@0.5 of 88.7% for the improved YOLOv7 algorithms. But also the basic algorithm achive a MaP@0.5 of 83.2% [4].

This shows that the 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].

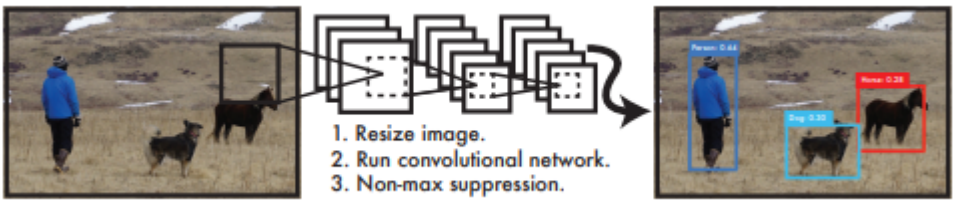

Figure 3.1: Das YOLO Objekterkennungsystem [5]

„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

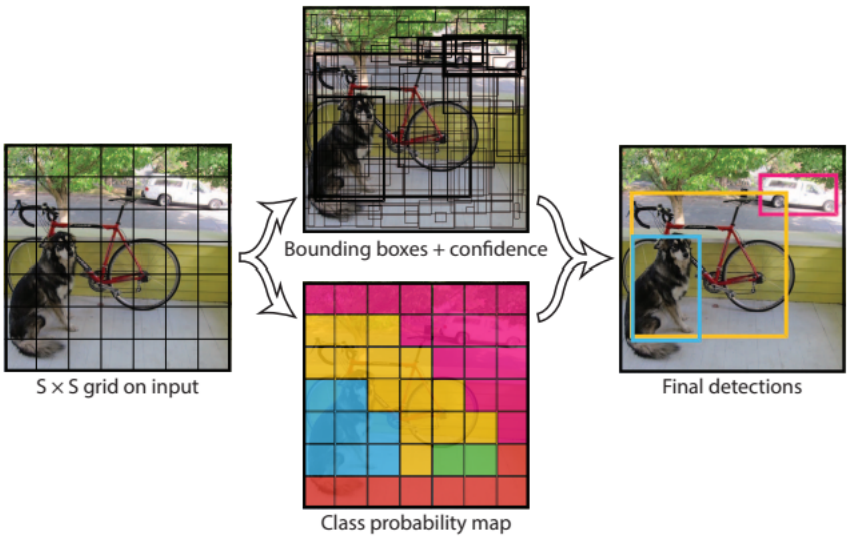

Figure 3.2: Das YOLO Modell[5]

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 Objektdetektierung 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].

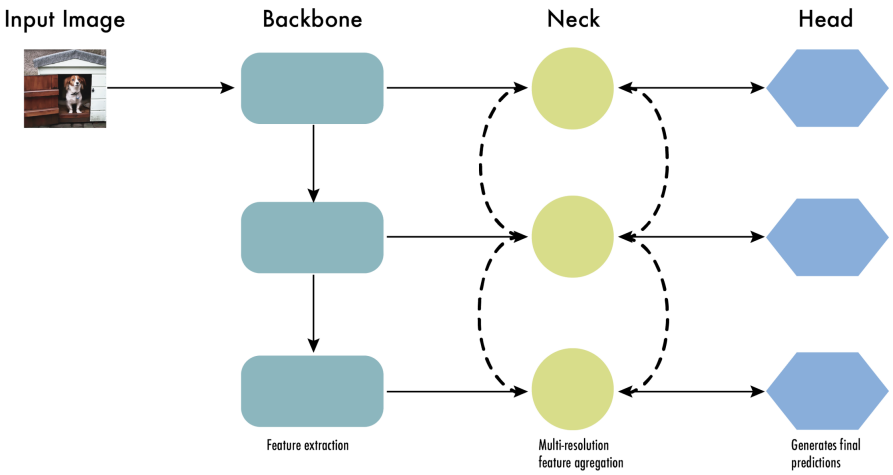

Figure 3.3: Architektur modernen Objektdetektoren [6]

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-Procressing 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 Ausgabetensors, 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

The scope of this project is to create a model which is abale to detect insects in a video stream. To achive this goal, a combination of computer vision methods was used to preprocess the video stream. The following sections will describe the all methods used in this project.

The generell workflow for this project was to first collect (and label) and then to preprocess the data using computer vision methods. After that the data was converted to a typ usable for YOLO and finally a model was trained on the preprocessed data. The preprocessing and training was a iterative process to find the best possible model.

4.1 Data Collection

The collection of the raw video stream, containing the DVS and RGB streams, was done by team CVMLS of Prof. Dr. Benjamin Risse from the Institute of Geoinformatics at the University of Münster.

The provided datasets were then labeled by all students of the study project "Machine learning meets insect monitoring" in the winter semester 2023/2024. The labeling was done using labelbox, a platform for labeling images and videos.

At the end of the data collection phase there were 34 videos, containing both streams stack on top of each other (later individual videos for each stream), with corresponding labels. The labels were in the form of bounding boxes around the insects in the video.

4.2 Data Preprocessing

The proprocessing of the video streams was done using computer vision methods provided by the OpenCV library. Our group was tasked with the preprocessing of the RGB stream.
Note that all of the following methods were not used in the sequence they are presented here and also are not all used in the same preprocessing. These are the basic methods used for different

preprocessings or in combination with each other, but never all at the same time.
To see all used combinations see Chapter **??**.

Substraction
The substraction is the simple method of substracting one image from another. The idea behind this was to subtract the background of the images and only keep the moving parts.
The results were a black and white image that showed the difference between the two images.

Background Subtraction
The background subtraction method is a more advanced version of the substraction method. The idea was again to remove the background from the image. Here you dont substract the two consecutive images but rather have a background model that is subtracted from the image.
This Method is somewhat comparable with the dvs stream, as it also only shows the moving parts of the image.

RGB to HSV
The RGB to HSV method converts the RGB image to a HSV image. The HSV color space is a cylindrical color space containing three components: hue, saturation, and value. The idea behind this method was to have another color space to work with.
This also allows to change each component individually, for example enhance the saturation.

Time offset
Time offset is a method to create each frame from three images (each gets its own channel) which have a time offeset. This offest can range from just one frame to multiple frames. The idea behind this method was to capture the movement of the insects. The more frames are between the iamges the bigger is the captured movement. This also creates a change to capture insects that are sitting still for a certain amount of time.

4.3 Data Processing

After pre-processing the videos, the data was converted into a format that could be used to train a model. The videos are available as mp4 files and the labelbox data is available as an ndjson file. This ndjson contains information about the bounding box and its associated frame. The YOLO training algorithm requires the images and the bounding boxes to be in a specific format, which is performed by the build dataset script. The basic script to perform this data processing was written

by Jakob Marten Danel, a fellow student in the Study Project, and has been adapted to the rgb data and also includes a bounding box conversion.

Dataset building

The build dataset script iterates through every video in the input folder and extracts the frames from the video. The script then checks if the frame is in the labelbox data and if so, it creates a txt file with the bounding box information. The script then moves the frame to the corresponding folder in the output folder. The aim is to have the videos as individual frames and the corresponding bounding boxes per frame in a txt file of the same name. In addition, the videos are randomly divided into training and validation data in a ratio of 80:20. A yaml file describes the folder structure and the locations where yolo finds the images and txt files.

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. 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 tried before the result was satisfying. 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 developed 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 distinct. 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 aligned the insect labels between the DVS and RGB streams.

4.4 Model Training

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

List of Tables

Bibliography

[1] Institute for Geoinformatics, *Computer vision and machine learning systems*. [Online]. Available: https://www.uni-muenster.de/Geoinformatics.cvmls/ (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