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*
515 622

*Timo Lietmeyer*
459 169

*Hendrik Lüning*
514 910


Münster, April 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 The YOLO algorithm

As the human gaze enables us to intuitively recognise, categorise and perceive objects, our brain, in conjunction with our eyes, is able to see quickly and accurately. These abilities allow us to perform complex tasks, such as cycling, which require the simultaneous use of several senses, with little conscious thought. [5].
The computer can be taught to do this with fast and accurate object recognition algorithms. Current systems use classifiers to recognise objects. This is applied at various points in variable scales in the test image to enable an object to be classified [5].

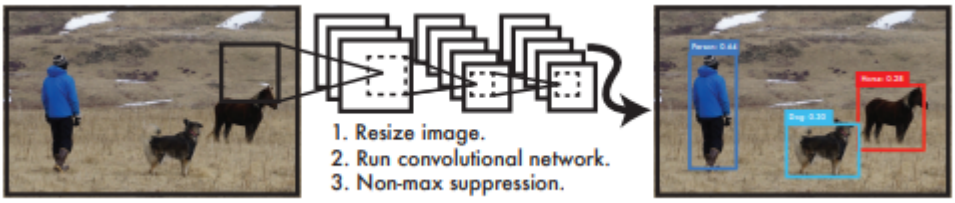


Figure 3.1: The YOLO object recognition system [5]

„You Only Look Once“ (YOLO) considers object recognition as a single regression problem by calculating directly from image pixel to bounding box coordinates and class probabilities. This algorithm analyses an image only once and directly predicts which objects are present where. As a result, the complexity of the structure of YOLO is very low, as shown in Fig. 3.1 [5].
The performance for object detection is increased by training YOLO with complete images. This standardised model offers several advantages over traditional object detection systems [5].
The first advantage of YOLO is the increased performance. This is made possible by the fact that object recognition on images is regarded as a regression problem and therefore no complex pipeline slows down the processing of an image. [5]. Secondly, YOLO analyses an image globally with

predictions for object detection. This allows YOLO to reduce the error of confusing background and foreground objects by half compared to Fast R-CNN. This is mainly due to the greater context that YOLO gains from the overall image analysis [5].
The third advantage is that YOLO was trained with generalised representations of objects to increase the error tolerance when applying it to new domains and unexpected inputs, due to the possibility of high generalisation [5].
One disadvantage of YOLO is its accuracy. The algorithm has difficulties in localising some objects, especially small ones [5].

YOLOv8 from Ultralytics is used in this project. This offers advantages in terms of performance and accuracy of object detection and is explained in more detail in the next chapter.

### 3.1.2 YOLOv8 von Ultralytics

In January 2023, Ultralytics released YOLOv8, which is based on YOLOv5. This version contains 5 different models (YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), YOLOv8x (extra large)), which were trained with data sets of different sizes [6].

The architecture of this implementation can be divided into 3 parts. These are the backbone, the neck and the head [6].
The detection of useful features from the input image takes place in the backbone, which is usually implemented as a CNN [6].
The neck is used between the backbone and head to aggregate and refine the features that the backbone outputs [6].
The last component is the head, which makes predictions based on the features provided by the backbone and neck. A post-processing step, such as non-maximum suppression (NMS), filters out overlapping predictions so that only the most reliable detections are used [6].
To improve performance, especially for object recognition of smaller objects, YOLOv8 uses CloU [7] and DFL [8] loss functions for bounding boxloss and binary cross entropy for the classificationloss [6].

This implementation is used in this project as it meets the requirements for precision and performance.

## 3.2 Computer Vision 2

Computer Vision 2 (CV2) is part of the „Open Source Computer Vision“ (OpenCV) library [9]. The current version 4.9.0 was released on 28/12/2023 [10].
This library contains algorithms for image and video processing.

## 3.3 PALMA

The High Performance Cluster of the University of Münster, PALMA („Paralleles Linux System für Münsteraner Anwender“) is a computer cluster with more than 3000 processor cores whose computing power is used for highly computing-intensive science and research [11]. This cluster was put into operation in 2010 and is operated with CentOS 7 [11], [12].

## 3.4 The camera trap

A camera trap is a device for taking unobtrusive pictures of animals without disturbing them in their natural habitat [13]. Due to the fast movement and relatively small size of insects, detecting insects on RGB images is challenging [14]. However, a „Dynamic Vision Sensor“ (DVS) can only record the movement of the insects, which results in very low energy consumption as only the movement in the image is recorded [14].
The videos used in this project were recorded with a DVS- and a RGB-camera. To accomplish that, a beam splitter was used to record the same scene with both cameras. The DVS-camera records the movement of the insects, while the RGB-camera records the scene in colour.

# 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 5.

### 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.

Figure 4.1: The output of the subtraction preprocessing

### 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.

Figure 4.2: The output of the background subtraction preprocessing

### 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.

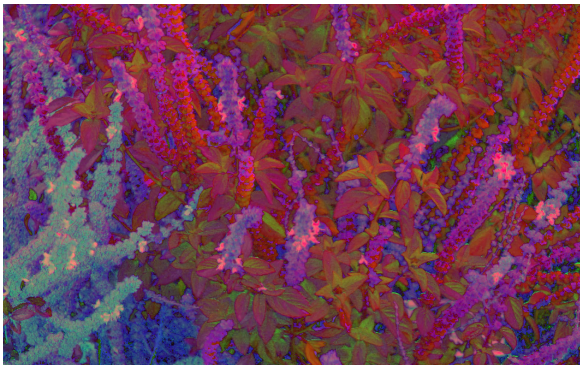

Figure 4.3: The output of the RGB to HSV preprocessing

### 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.

Figure 4.4: The output of the time offset (by 10) preprocessing

## 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

The training of the models was done using YOLOv8, and was run on the HPC of the University of Münster called PALMA. It was conducted using the preprocessed data from the previous steps and the standard YOLOv8 model.
To create comparable results, the training was done using the same parameters for all models. The training was done for 100 epochs with a patience of 25 and an image size of 640. The amount of devices used for the training was 2 to 4.
The method behind the training was to create models from multiple different preprocessing workflows and then compare the results. This comparison was done using the mAP (mean average precision) score. The mAP score is a common metric used to evaluate object detection models. It is calculated by taking the average of the precision-recall curve.

# 5 Results

The results of the project are presented in this chapter. For that, the results of the different approaches to preprocess the data before they were used for training are presented and compared. This chapter will take a look at the results one by one.

First af all the results of the default RGB training are presented. The default RGB training was done using the raw RGB data without any preprocessing. The results of this training are shown in Table 5.1.

The important values to look at are the precision and recall value of the epochs of the model. The precision and recall values should be as high as possible. The precision value shows the accuracy of detected objects and the recall value shows how many of the actual values were predicted correctly. Normally the two values are getting better with more epochs processed [15]. But in Figure 5.1 the precision is getting lower with each new epoch besides one spike and the recall value stays quite low with one low point. The box_loss is also an interesting value to look at. It shows the loss of the bounding boxes. The lower the value the better the bounding boxes are predicted. The box_loss is getting lower with each epoch which is at first sight a good sign. But in the box loss of the validation it varies a lot.

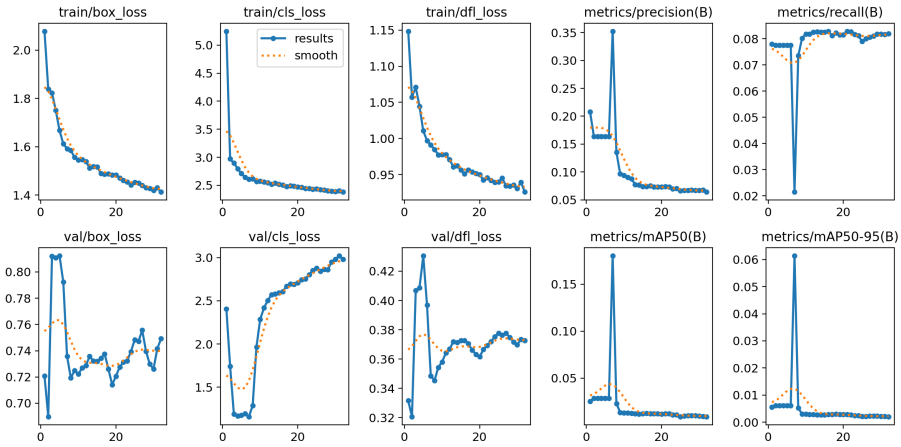


Figure 5.1: Results of the model training with RGB data

The background subtraction method was then used on the rgb data. The results of this training are shown in Table 5.2.

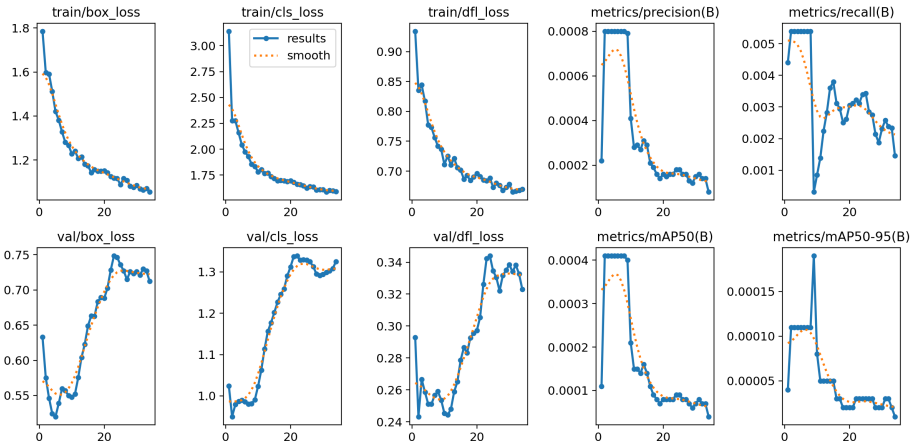


Figure 5.2: Results of the model training with background subtraction

The background subtraction model has much lower precision and recall values than the RGB model. At the final epochs it even goes down to lower than 0.0002 but for that the box_loss looks normal. All in all the the results of the background subtraction model not good.

The background subtraction method was then combined with the time offset method with an offset of 1. The results of this training are shown in Table 5.3.

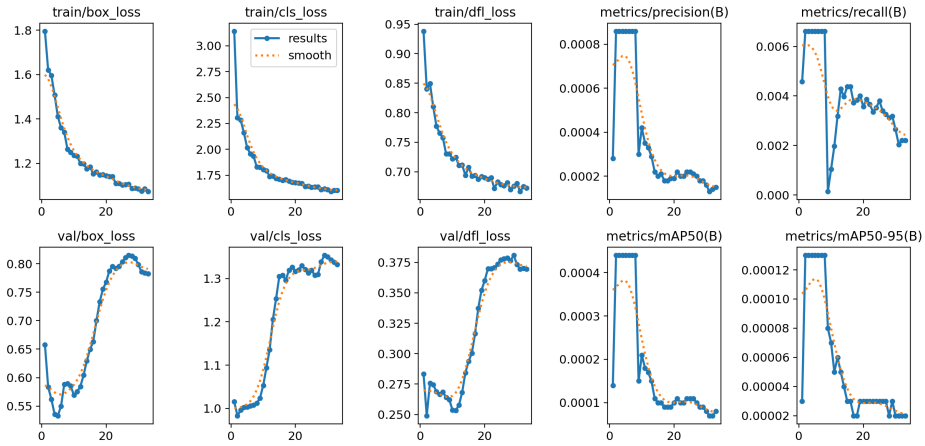


Figure 5.3: Results of the model training with background subtraction and time offset 1

Because the frames in the video data was just combined with one extra frame to visualize movement in pictures the results of the background subtraction + time offset 1 model are similar to the background subtraction model with almost the same results. So the time offset had no effect on the model.

The next results are from the training with the HSV data. The HSV data was created by converting the RGB data to the HSV color space. The results of this training are shown in Table 5.4.

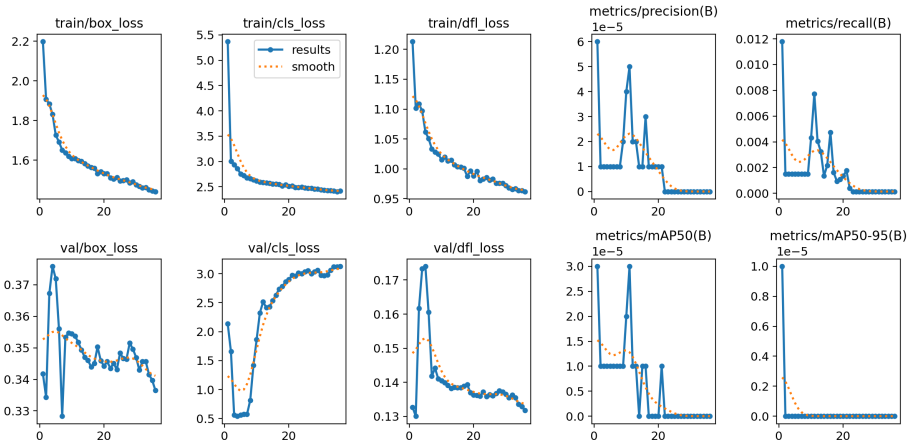


Figure 5.4: Results of the model training with HSV data

The precision and recall values of the HSV training are very low. They are not going down steadily as they should and even go near zero in the later epochs. The box loss is indeed going down steadily which is a good sign. These results show that the model is not able to detect the insects in the HSV data.

The HSV data was then combined with the background subtraction method. The results of this training are shown in Table 5.5.

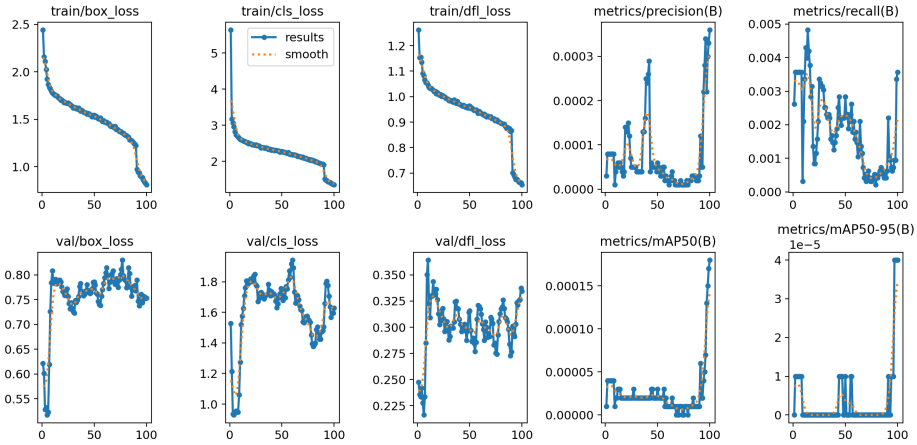


Figure 5.5: Results of the model training with HSV data and background subtraction

The results of the model trained with the HSV data and background subtraction is based on the same data as the HSV model. With the background subtraction added the model should perform better than the original HSV model. Firstly the precision and recall values are higher than the original

HSV model but still towards null. Because the precision and recall values are looking not normal the model is not able to detect the insects in the HSV data with the background subtraction.

# 6 Discussion

As seen in chapter 5, the results of the project are extremly bad, so much that there is no point in using the resulting models for any kind of further research. But the results are so bad, that for one it is unlikly that these results are representative, which means that some problems must have accured during the workflow, as well as that it is worth discussing why the results are so bad, what may have happend and what could be done to improve them.
Because these circumstaces this chapter will not discuss the results further, but will deal with the problems that may have caused these results.

The first point to examine would be the preprocessing of the data. There are multiple problems that could arise in this step. The most prominent would be, that the prepocessing of the videos would lead to unusable or corrupt videos files. But as shown in chapter 4 the outputs are correct. Some preprocessing steps provide visually better outputs to classify insect, like the background subtraction compared to the HSV, but all results servicable to yield some results.
Another problem could be, that the preprocessing of the images lead to a shift in the time steps of the videos.

This leads to the bounding boxes, as a shift in the time steps could lead to issues with mapping the bounding boxes correctly.

Figure 6.1: Combination of the preprocessed video using background subtraction and the corresponding bounding boxes

Figure 6.2: Combination of the preprocessed video using RGB to HSV and the corresponding bounding boxes

But as shown in the pictures above, the bounding boxes are correctly mapped to the videos.
As visible in the pictures the bounding boxes are not perfect and could be improved, but they are good enought to be used for training a model. The improvement of the bounding boxes could lead to minimal better results if you want to improve a already good model, but in this case the bounding boxes are not the problem.

As these points are not the problem, the next point to examine would be the training of the model. As shown in chapter 4 the training of the model was done using the YOLOv8 model. A possibility here could be that the default YOLOv8 model is not suitable to identify insects in the RGB or our preprocessed videos. But considering the results of the study shown in chapter 2, which used YOLO to identify and classify street signs in RGB videos with results up to 83.2% in the mAP@0.5 score, it is safe to assume that the YOLOv8 model should be suitable for this task.

All the above mentioned points were problems and approaches we checked and made sure that these are not the root of the problems. The next paragraphs will discuss some other points that could have impacted the results of the models.

A possibilty might be, that the RGB data and possible preprocessing steps do not provide resulting videos that are good enough, for the algorithm to learn sensible connections out of them. As seen in chapter 5 the default RGB training results in a model with an mAP@0.5 value of abou 18% which is definetly are very low value. However it is concerning that the results of the preprocessing steps are even worse, which should not be the case, because when you visually compare the used training videos for example the background subtraction should lead to way better results.
Continuing on the topic of trainingdata, the amout of training data can impact the results of the model. The training data used in this project contains 34 videos. This is a very low amount of data, which could lead to overfitting or bad results in generall.
A possible way to conteract the issue of the low amount of training data would be to incorporate cross validation into the training process.

The last point to improve would have been the hyperparameter of the YOLO training. These were primarily the defaults which could be improved by using a grid search to find the best hyperparameters. But because the results were so bad, it is unlikely that the hyperparameters are the problem and the attempt was made to find the bigger issue that the exact best hyperparameters.

# 7 Contributions by group members

## 7.1 Maximilian Elfers

The main task was the creation of the preprocessing code and workflow of the raw videos. During the course of the semester another big task was the continous training of models on PALMA, which turned out to be more difficult than hoped, because of technical problem on PALMA with YOLO.
In the written assignment the contribution was the related work and discussion chapter and shared tasks for the methods and results chapter.

## 7.2 Timo Lietmeyer

The preprocessing code and the conversion of bounding boxes have been extended. In addition, minor bugs in the code and during execution were fixed and problems with the use of YOLO on PALMA were solved.
In the written assignment, the Fundamentals part was added and the Methods, Results and Methods part was created in cooperation.

## 7.3 Hendrik Lüning

A major part of the work was the conversion of the bounding boxes described in 4, which involved setting up the build-dataset script and adapting it to the RGB data. As there were many problems with this script, a lot of debugging was done. Once the build-dataset script was complete, the focus shifted to training the models.
In the final draft, the introduction and, in collaboration, the methods-, results- and discussion-section were added.

# 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/ (visited on 03/15/2024) (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, 5).

[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 p. 5).

[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. 5).

[8] X. Li, W. Wang, L. Wu, *et al.*, "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. 5).

[9] *About - OpenCV*. [Online]. Available: https://opencv.org/about/ (visited on 03/28/2024) (cit. on p. 6).

[10] *Releases - OpenCV*. [Online]. Available: https://opencv.org/releases/ (visited on 03/28/2024) (cit. on p. 6).

[11] n. A., *Hpc-system palma*. [Online]. Available: https://www.uni-muenster.de/CoCoS/Systeme/PALMA.html (visited on 03/28/2024) (cit. on p. 6).

[12] Potthoff, Sebastian, *High performance computing*, 2024. [Online]. Available: https://confluence.uni-muenster.de/display/HPC/High+Performance+Computing (visited on 03/28/2024) (cit. on p. 6).

[13] A. F. O'Connell, J. D. Nichols, and K. U. Karanth, "Camera traps in animal ecology: Methods and analyses," *Camera Traps in Animal Ecology: Methods and Analyses*, pp. 1–271, 2011. DOI: 10.1007/978-4-431-99495-4 (cit. on p. 6).

[14] E. Gebauer, S. Thiele, P. Ouvrard, A. Sicard, and B. Risse, "Towards a dynamic vision sensor-based insect camera trap," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2024, pp. 7157–7166 (cit. on p. 6).

[15] G. Jocher, M. Rizwan Munawar, and A. Vina, *YOLO Performance Metrics — docs.ultralytics.com*, https://docs.ultralytics.com/guides/yolo-performance-metrics/, [Accessed 04-04-2024], 2023 (cit. on p. 12).

# 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, April 5, 2024

---

Timo Lietmeyer, Münster, April 5, 2024

---

Hendrik Lüning, Münster, April 5, 2024