Machine Learning meets Insect Monitoring

Jakob Danel*
jakob.danel@uni-muenster.de
Institute for Geoinformatics
Münster, Germany

Frederick Bruch*
f_bruc03@uni-muenster.de
Institute for Geoinformatics
Münster, Germany

1 INTRODUCTION

The decline of insect populations poses a significant threat to global ecosystems and human livelihoods. As these diminutive animals play pivotal roles in pollination, nutrient cycling, and pest control, their dwindling numbers could trigger cascading effects throughout food webs, leading to ecosystem instability and agricultural challenges. Amidst growing concerns over insect declines, there is an urgent need for innovative methodologies to monitor and understand insect populations in diverse environments.

Our study project, *"Machine Learning meets Insect Monitoring: Detecting Tiny Objects in Cluttered Natural Outdoor Environments"*, addresses this pressing issue by leveraging advanced technologies at the intersection of machine learning and insect ecology. We aim to develop novel approaches for quantifying insect presence in complex outdoor settings.

1.1 Context and Motivation

The motivation for our project stems from the alarming decline in insect populations and its potential catastrophic consequences for ecosystems and human society. Quantitative data paints a dire picture, indicating that 41% of global insect species have declined over the past century alone [21]. This decline is not limited to terrestrial insects but also affects non-terrestrial species. Furthermore, while vertebrate populations have garnered significant attention, it's crucial to recognize that more than 90% of all animal species are invertebrates, with insects making a big part of it. Insects are indispensable for ecosystem health, with 90% of flowering plants benefiting from their pollination efforts [15]. Their contribution extends to global agriculture, where insect pollinators promote the production of 75% of major crops, valued at approximately €150 billion [6, 9]. However, monitoring and quantifying invertebrates pose significant challenges due to their vast diversity and the complexity of their habitats. Thus, our project aims to address these challenges by developing innovative technological solutions to monitor and quantify insect populations in natural outdoor environments effectively.

Traditional methods for monitoring insect populations, such as manual sampling or trap-based surveys, are labor-intensive, time-consuming, and often provide limited spatial and temporal coverage. Moreover, these methods may not adequately capture the dynamic nature of insect behavior in natural environments. To address these challenges, our project seeks to harness the power of emerging technologies, particularly in the realms of computer vision, machine learning, and sensor technology, to revolutionize insect monitoring efforts.

1.2 Objectives

The primary objective of our study project is to develop a state-of-the-art visual insect monitoring system capable of detecting and tracking tiny objects in cluttered outdoor environments. By integrating machine learning algorithms with advanced imaging modalities, such as Dynamic Vision Sensors (DVS), we aim to achieve real-time, high-precision insect detection and tracking.

Our study project delves into the critical issue of declining insect populations and their profound implications for ecosystems and human welfare. Recognizing the limitations of conventional insect monitoring methods, we emphasize the pressing need for technological advancements to address this challenge. Central to our approach is the development of a Smart Insect Camera Trap, a novel concept that integrates spatial and temporal cues for enhanced insect detection in outdoor environments. We highlight the significance of this innovation in overcoming the complexities of monitoring fast-moving insects amidst cluttered backgrounds. Moreover, we underscore the potential of Dynamic Vision Sensors in capturing rapid insect motion with precision and efficiency. Our project's organizational structure is delineated through distinct phases, including data annotation, algorithm development, and evaluation.

2 RELATED WORK

To address the problem of significant quantity of photos that lack useful information (namely, photographs without insects), a substantial amount of energy consumption, and real-time image processing on limited hardware, many methods have been tried. Naqvi et al. showed that scheduled frame capturing outperformed motion-activated imaging [14]. Ratnayake et al. used kNN background subtraction and YOLOv2 to process recordings to detect honeybees [19] and later added flower recognition using YOLOv4 [18]. However, challenges remain in accurately capturing and saving insect imagery due to factors such as small object size, fast insect motion, cluttered scenes, and dynamic backgrounds. Previous work by Thiele et al. has demonstrated progress towards visual insect camera traps.

They proposed a methodology that incorporates temporal cues, specifically using the HSV* color space, to improve insect detection accuracy. Their quantitative evaluation of detection networks, including YoloV3, Faster RCNN, and MobileNet, demonstrated promising results in terms of average precision, precision, recall, and computational efficiency [24].

To address the limitations mentioned above, researchers have explored alternative technologies such as DVS. DVS offers several advantages over conventional RGB cameras, as it is specifically designed to detect motion events with high sensitivity, ultra-fast response times, and low energy consumption [4]. The utilization of DVS technology holds promise for enhancing insect tracking

*Both authors contributed equally to this research.

capabilities within the Smart Insect Camera Trap. Building upon this foundation, Gebauer et al. extend the capabilities of insect camera traps by introducing DVS-based sensing in combination with a real-time small object detection algorithm, demonstrating superior detection performance and lower processing times compared to state-of-the-art deep learning models [10].

3 FUNDAMENTALS

3.1 Hardware and Camera Setup

The insect monitoring system (see Figure 1) relies on a carefully designed hardware setup comprising two key components: a conventional full-frame RGB camera and a DVS.

3.1.1 RGB Camera. The RGB camera, a Basler a2A1920-160ucBAS with a resolution of 2.3 MP, is equipped with a Kowa LM16JC1MS 16mm/F1.4 C-Mount lens. This high-resolution RGB camera captures visual data at a rapid frame rate of 160 fps, providing detailed imagery of the monitored environment.

3.1.2 Dynamic Vision Sensor. The DVS operates on neuromorphic vision principles, detecting events triggered by local brightness changes above a threshold. It offers high-temporal-resolution event data suitable for tracking fast-moving insects in cluttered and dynamic environments. The DVS used is a Prophesee EVK4 connected with a Trigger Connector.

3.1.3 Beamsplitter. The RGB camera and the DVS are positioned in front of the same beamsplitter, with 70% of the light reserved for the RGB camera and 30% for the DVS. Both cameras are housed in a 3D-printed enclosure to ensure they have a similar view of the captured scene. Both cameras use a Kowa LM16JC1MS 16mm/F1.4 C-Mount lens, and the focus is set to a plant about one meter away from the camera.

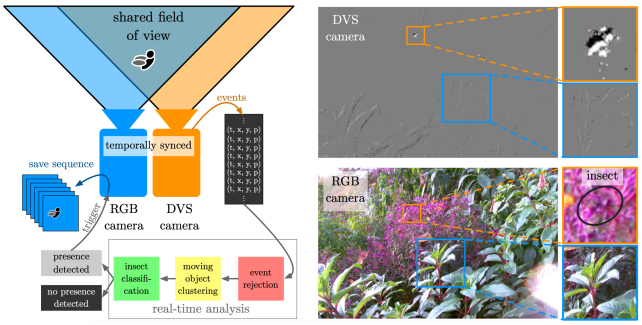


Figure 1: RGB-DVS Camera System [10].

3.2 CNN's and YOLO

The first precursor of a convolutional neural network (CNN), named Neocognitron, was published in 1990. It established the ideas of feature extraction, pooling layers, and convolution in a neural network [8]. The term "convolutional neural network" was coined to describe the architecture of the LeNet[13], which was developed by Yann LeCun and his team from 1989 to 1998 for the handwritten digit recognition task. With AlexNet[12], the first ever CNN won the ImageNet[1] classification challenge in 2012, reducing the error rate from 26.2% to 15.3%. Since then, every winner of the 'ImageNet large scale visual recognition challenge (ILSVRC)' was based on a CNN.

In 2016, a new convolutional neural network (CNN) named YOLO (You Only Look Once) was released by Joseph Redmon and his team [20]. This revolutionary model (see Figure 2) was one of the first models utilizing single-stage method for object detection, surpassing previous detectors in both speed and accuracy. Its remarkable performance quickly established YOLO as one of the most favored architectures for object detection tasks across academic and non-academic circles [22]. Since then, numerous researchers and groups have iteratively refined YOLO, with the most recent major version being YOLOv9[26].

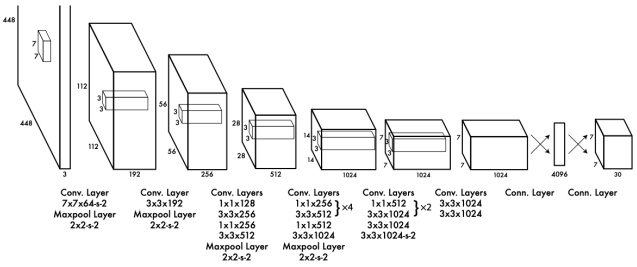


Figure 2: YOLO Architecture.

3.3 Preprocessing

Since we will be manipulating the original video streams obtained from the RGB and DVS camera, we will now discuss some fundamental preprocessing techniques.

3.3.1 Background Subtraction. Background subtraction is aimed at isolating foreground objects from the background in an image or video stream. The process involves extracting the regions of interest by identifying and removing the stationary elements or the background from the input data. In this study, we utilized the MOG2 (Mixture of Gaussians) background subtraction algorithm from the OpenCV library[2]. MOG2 is a robust and widely-used method for background modeling, capable of adaptively updating the background model over time to accommodate gradual illumination changes and dynamic backgrounds.

3.3.2 Temporal Filter. Temporal filtering is used in signal and image processing to reduce noise and smooth out fluctuations over time. The basic principle behind temporal filtering involves analyzing the variations in intensity or pixel values across consecutive frames in a video sequence. By considering the temporal relationship between adjacent frames, temporal filters aim to suppress random noise while preserving the underlying structure and motion in the data. One common approach to temporal filtering is the use of moving average or low-pass filters, which compute the average or weighted average of pixel values over a sliding window of frames. In the following experiments, the exponential moving

[1] https://image-net.org/index.php
[2] https://pypi.org/project/opencv-python/

average has been used. This effectively attenuates high-frequency noise components, resulting in a smoother output signal or image sequence. Temporal filtering is particularly useful in scenarios where the input data exhibit temporal instability or jitter, such as in surveillance footage, video streams from mobile cameras, or recordings in noisy environments.

3.3.3 RGB to HSV. For transformation from RGB to HSV a classical approach is used, to transform the RGB input stream into the HSV color scheme [3]. The HSV values are then written to the RGB-channels of the output. The transformation begins with defining the maximum (Max = max(R, G, B)) and minimum (Min = min(R, G, B)) for an input pixel R, G, B. Then HSV is defined as following:

$$\text{Hue (H)} = \begin{cases} 0, & \text{if Max} = \text{Min} \\ 60° \times \left(\frac{G-B}{\text{Max}-\text{Min}} \mod 6\right), & \text{if Max} = R \\ 60° \times \left(\frac{B-R}{\text{Max}-\text{Min}} + 2\right), & \text{if Max} = G \\ 60° \times \left(\frac{R-G}{\text{Max}-\text{Min}} + 4\right), & \text{if Max} = B \end{cases}$$

$$\text{Saturation (S)} = \begin{cases} 0, & \text{if Max} = 0 \\ 1 - \frac{\text{Min}}{\text{Max}}, & \text{otherwise} \end{cases}$$

$$\text{Value (V)} = \text{Max}$$

3.4 Object Detection Measurements

Confusion matrices (Table 1) are used in object detection tasks to assess the effectiveness of a model. This matrix facilitates the evaluation of our model's performance in identifying true positives (correct detections), false positives (incorrect detection), true negatives (correct rejections), and false negatives (missed detections).

		Predicted	
Actual		**Positive**	**Negative**
Positive		TP	FP
Negative		FN	TN

Table 1: Confusion matrix explaining true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN).

From this matrix, we derive important metrics like *Precision* ($\frac{TP}{TP+FP}$) and *Recall* ($\frac{TP}{TP+FN}$) [1], valuing all detections with confidence > 0.25 and $IoU > 0.45$ as TP, where confidence represents the model's confidence in the detection and *IoU* is defined for calculated bounding box B_p and expected bounding box B_e as: $\frac{B_p \cup B_e}{B_p \cap B_e} = \frac{\text{area of overlap}}{\text{area of union}}$ [23].

Mean Average Precision (mAP) is calculated as the area under the Precision-Recall Curve [7]. The Precision-Recall Curve is calculated by examining all confidence thresholds and calculating Precision and Recall.

Additionally, we introduce the concept of fitness to assess the model's performance comprehensively. The fitness metric is defined as the best fitness over all epochs and is calculated as follows:

$$\text{fitness} = 0.1 \times \text{mAP@0.5} + 0.9 \times \text{mAP@0.5-0.95}$$

Here, mAP@0.5 represents the mean average precision at an IoU threshold of 0.5, while mAP@0.5-0.95 represents the mean average precision across IoU thresholds ranging from 0.5 to 0.95. This composite metric provides a balanced assessment of the model's performance across different IoU thresholds, with greater emphasis placed on higher IoU thresholds to ensure robustness in object detection.

4 METHODS

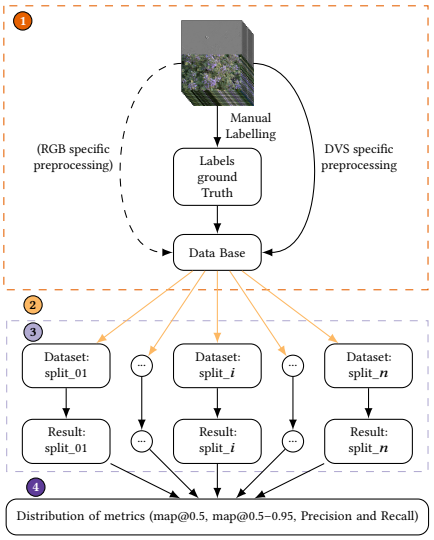


Figure 3: Workflow for an experiment with a unique preprocessing combination for DVS (and RGB). Split_i describing the workflow for one specific scene used for the cross validation (where $i \in \{i \mid i$ is an integer, $1 \leq i \leq n\}$ and n is the amount of scenes in the dataset).

This section delves into the methods employed throughout our research and provides a rationale for their selection. To clarify essential terms: a "scene" is the setup of the camera within the specific environment, while a "video" consists of smaller segments taken from scenes. Our experimental design encompasses two primary approaches: The DVS only experiment, wherein solely DVS images are used for the detection task, and the RGB + DVS experiment, which involves the utilization of a merged image of both input types for the detection task. To assess the robustness of our experiments, we implemented a cross-validation strategy following the leave-one-out principle. Each scene was designated as validation data for one split, resulting in a distribution of performance quality scores that facilitates meaningful statistical comparisons. A key part of our work involves exploring the impact of different computer vision preprocessings on raw data. Through these experiments, we sought to discern potential enhancements in the detection task performance. The overall workflow of our experiments is visualized in Figure 3, and the subsequent sections will expound upon the applied methods, following the structured framework illustrated in the diagram.

Initially, manual labeling of videos and individual preprocessing procedures are conducted for both RGB and DVS videos, by labeling

the DVS stream and mapping the labels to the RGB stream (depicted as ①). Then we construct a dataset that can be used within the YOLO environment for each validation split, with an additional step involving the combination of DVS and RGB data (②). The third step involves employing YOLO for training a model and generating validation results (③). Finally, in the fourth step (④), the study filters the best scores for each validation split and proceeds to compare the distributions of different preprocessing techniques.

4.1 Preparing the data

The used dataset was conducted at the Botanical Garden of Muñster. The filming took place from September 28, 2023, to October 13, 2023. The filming process utilized a dual camera setup, as detailed in Section 3.1. Ten distinct scenes were captured during this period, each scene generating videos comprising a maximum of 4000 frames. In sum, the dataset comprises 97,077 frames.

4.1.1 Labelling. The labeling process was facilitated using the Labelbox tool[3]. Bounding boxes were drawn around insects identified in each video. Setting bounding boxes for each frame is not required. It is possible to maintain the position of the bounding box unchanged across successive frames.

4.1.2 Preprocessing. Video processing was carried out in a stacked format (DVS + RGB), as illustrated in Figure 3 at the initial node. This approach provided the advantage of applying the same preprocessing steps to both DVS and RGB data if needed. Given that the DVS input comprises only one-channel information, distinct preprocessing steps were employed on different channels. Furthermore, a sequential preprocessing workflow was implemented. The implementation was done using Python and the framework OpenCV [2] which is capable of modifying images on pixel base.

In Section 3.3, each preprocessing concept is introduced, while here the methodology behind temporal filtering gets explained. The specific sequence of preprocessing steps employed for a given experiment will be elaborated upon in the results section.

Temporal Filtering. In our research, we employed a temporal filtering technique to process single-channel frame streams. This technique is governed by the following formula:

$$out_i = \left(1 - \frac{1}{fs}\right) \times out_{i-1} + \frac{1}{fs \times in_i}$$

Here, fs denotes the filter size, indicating the weighting of previous frames considered in the filtering process. Throughout our experiments, we set $fs = 5$. The input to the filtering process, denoted as in_i, represents the selected input channel k of the ith frame of the input video, structured as $M_{h \times w}$, where h and w correspond to the frame's height and width, respectively. The resulting output, out_i, also maintains the same dimensions, $M_{h \times w}$.
For initialization, we adopted the following configuration:

$$out_0 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{h \times w}$$

This approach involves incorporating both, the current and the previous frames and accumulating them. Temporal dependent preprocessings are a common approach to increase the accuracy of insect detection models [24]. The approach we employ diverges from a linear methodology; instead, it assigns significantly higher weights to the current and more recent frames compared to those further in the past. Specifically, the weighting scheme entails assigning a weight of $\frac{1}{fs}$ to the current frame, while the p-th previous frame (indexed as $i - p$) contributes to the output frame with a weight calculated as $\left(1 - \frac{1}{fs}\right)^p \times \frac{1}{fs}$. This weighting mechanism ensures that the influence of past frames diminishes exponentially as they recede into the past. Consequently, frames closer to the present exert a stronger influence on the output frame, aligning with the temporal dynamics of the video stream (see Figure 4). We opted for $fs = 5$ to have a balance between maintaining a significant influence from frames with higher p values and assigning considerably higher importance to frames nearer in time (see Figure 4). Such an approach aims to enrich the current frame with information from the recent past, enabling the capture of motion even when the subject remains static. This capability proves particularly valuable, for instance, in scenarios where insects exhibit minimal movement, ensuring that past motion remains accounted for.

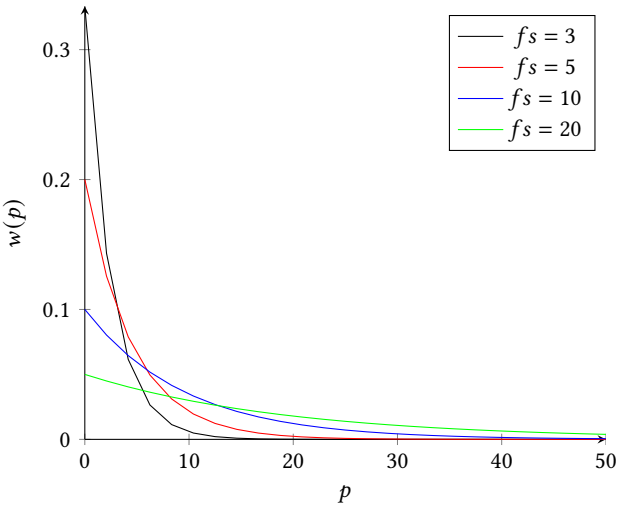


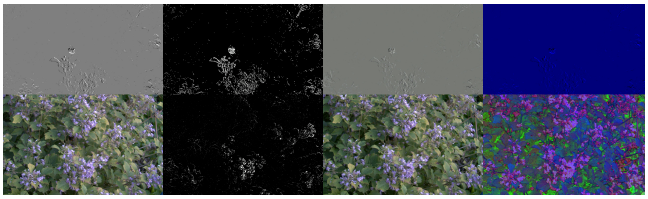
Figure 4: Weights used in temporal filtering for different fs: $w(p) = \left(1 - \frac{1}{fs}\right)^p \times \frac{1}{fs}$. Describing the decreasing impact of the previous frame to the output frame in temporal filtering approach.

4.2 Setup cross validation

To facilitate the cross-validation process, we established an infrastructure for each scene(totaling $n = 10$). An essential component of this infrastructure is a Bash script that iterates over the scenes, submitting the setup process to the Slurm[4] task management tool on the computational unit. To improve performance, we implemented

[3] https://labelbox.com/

[4] https://slurm.schedmd.com/documentation.html



(a) Original (b) BS (c) TF ($fs = 5$) (d) RGB to HSV

Figure 5: Visual depiction of various preprocessing steps applied to the original stacked data set.

dependencies to ensure that only one split preparation process runs at any given time. This script also prepares the dependencies and parameters for the subsequent training process (see Section 4.3). The setup of one cross-validation split encompasses several key tasks. Subsequently, bounding boxes are transformed from Labelbox format to the YOLO format [25]. Videos are then transformed into images of individual frames. For DVS, the upper part of the image is cropped. For DVS + RGB, each image gets undistorted using the respective camera calibration matrix [27]. The image is warped to map pixels of RGB frames to DSV frames [11]. Then, from DVS and RGB four channels will be selected (this is configurable for the user) and merged into an RGBA frame. Frame extraction is implemented in Python using OpenCV [2]. Lastly, the script extracts the number of available cores on the hardware and leverages the Python multiprocessing framework to maximize hardware utilization. This involves processing a set of videos simultaneously to enhance efficiency. By executing these tasks, we establish a robust infrastructure for each scene, ensuring the seamless execution of cross-validation splits with optimized resource utilization.

4.3 Train detection models using YOLO

In our experiments, we adhered to specific parameters and setup configurations. For a general workflow, we conducted training for 100 epochs. We implemented a patience of 25 epochs during training, whereby if there was no improvement in performance within this window, the training process was halted. Original image resolution was maintained to ensure comparability across experiment types. For the DVS experiment, we employed YOLOv8[5], utilizing the yolov8n model as the base for our training. An automatic batch size option determined a feasible batch size for the available hardware, while default hyperparameter configurations of YOLOv8 were applied. In the DVS + RGB experiment, YOLOv7 [25] was utilized with a modification to incorporate the Alpha Channel as input [22]. A batch size of 96 was manually determined as optimal for hardware utilization. Training commenced with a pretrained model from Scharf, chosen for its 4-channel architecture rather than the specific initialization weights. The default yolov7 configuration was applied for training parameters.

4.4 Compare the results

The validation of the data will be performed on the following criteria, each of which is measurable for each training of the data: We evaluate map@0.5 and map@0.5-0.95, as it is common in YOLO implementations. To assess the model's performance, we consider the best fitness over all epochs. For each validation split, we gather these metrics to obtain a distribution of Precision, Recall, map@0.5, and map@0.5-0.95 scores. To compare two preprocessings, we apply a paired t-Test to test if the mean of the differences μ_d is significantly different from 0. We apply an alternative hypothesis H_A that assumes $\mu_d > 0$ to test directly for the better performance of a specific model. As input, we use the best fitness score of each validation split, a known method for comparing cross-validation results [5]. The use of the t-Test comes with violations of the assumptions for t-Tests, as metric differences are not independent due to being calculated on the same dataset [17]. Despite drawbacks, especially the higher probability of false positives [5], other approaches are not feasible:
One-fold McNemar test [5, 17] was not applicable, because different settings in each scene necessitate a test that validates performance across all scenes. The 5×2 validation method by [5] was not feasible due to the 50%/50% split between test and validation data, which made the training set too small.

5 EXPERIMENTS

In our experiment naming scheme, we separate individual channels using underscores (_). For each node in the preprocessing chain, we list the input and various preprocessing steps applied, separated by hyphens (–). If a preprocessing is applied to all channels, we group them within brackets. This scheme provides a concise and systematic way to represent the transformations applied to different channels in our research. The detailed abbreviations used to describe the components are listed in Table 2. We will give a brief explanation of each experiment in this chapter.

Abbreviation	Type	Explanation
DVS	input	DVS(Event)–Stream
RGB		RGB(color)–Stream
BS	preprocessing	background subtraction
TF		temporal filtering
HSV		RGB to HSV
R	extraction	extract red color
G		extract green color
B		extract blue color

Table 2: Different abbreviations and their explanations used in the experiments

5.1 Three channels

Firstly we focused on using the DVS stream for insect detection. Therfore we used three channel RGB images as data base.

5.1.1 DVS_DVS_DVS (01). Here we tested the original videos, by using the DVS stream in all three image channels.

5.1.2 DVS_DVS–TF_DVS (02). Here, we utilized the temporal filtering preprocessing for testing the impact of temporal filtering on original data.

[5] https://docs.ultralytics.com/de

5.1.3 DVS_DVS–BS_DVS-TF (03). We combined temporal filtering and background subtraction by applying both preprocessing on different channels while still maintaining the original DVS stream.

5.1.4 DVS_DVS–BS_DVS–BS–TF (04). A variant of the previous attempt, where we applied the background subtraction to a temporal filtered DVS stream, rather than the original one. Figures 6 and 7 demonstrate that this approach of preprocessing makes it easier to identify insects. Especially in the RGB stream at the bottom.

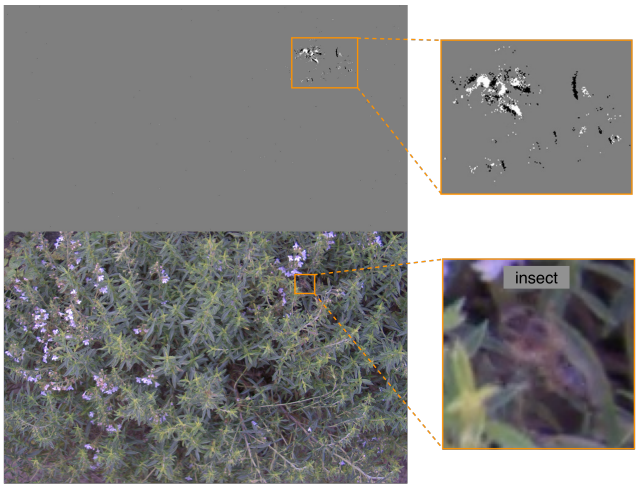


Figure 6: DVS and RGB frames stacked.

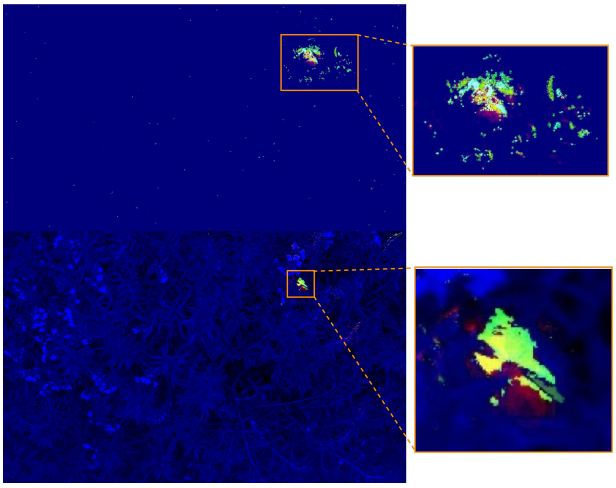


Figure 7: Manipulated DVS and RGB frames stacked.

5.2 HSV

For all previous mentioned experiments (see Section 5.1) we performed also an HSV transformation leading to the following four experiments:

- (DVS_DVS_DVS)–HSV (05)
- (DVS_DVS–TF_DVS)–HSV (06)
- (DVS_DVS–BS_DVS–TF)–HSV (07)
- (DVS_DVS–BS_DVS–BS–TF)–HSV (08)

5.3 Four channels

For combining RGB and DVS stream we used a four channel approach with an additional alpha channel.

5.3.1 RGB-R_RGB-G_RGB-G_DVS (09). Here, we used all four possible input streams as the input to test the performance of the original data.

5.3.2 DVS_DVS–BS_DVS–BS–TF_RGB–BS (10). We performed an experiment by combining the best DVS stream preprocessing approach with the best approach tested on the RGB data[6]

		ID	Name
3-Channels	RGB	01	DVS_DVS_DVS
		02	DVS_DVS–TF_DVS
		03	DVS_DVS–BS_DVS-TF
		04	DVS_DVS–BS_DVS–BS–TF
	HSV	05	(DVS_DVS_DVS)–HSV
		06	(DVS_DVS–TF_DVS)–HSV
		07	(DVS_DVS–BS_DVS–TF)–HSV
		08	(DVS_DVS–BS_DVS–BS–TF)–HSV
4-		09	RGB-R_RGB-G_RGB-B_DVS
		10	DVS_DVS–BS_DVS–BS–TF_RGB–BS

Table 3: Assignment of experiment name to IDs. Used in later sections for better readability.

6 RESULTS

6.1 Precision

In Experiment 08, the mean precision attained was 0.279, with a median value of 0.269 (see Figure 8). Conversely, Experiment 04, incorporating HSV preprocessing, demonstrated a slightly higher mean precision of 0.294, with a similar median value of 0.287 (see Figure 8). These findings suggest that the inclusion of HSV preprocessing did not substantially affect precision compared to the configuration without it. Similarly, Experiment 06 yielded a mean precision of approximately 0.293, while Experiment 02 achieved a slightly higher mean precision of 0.317. Both configurations exhibited comparable median precision values, indicating consistent performance despite the addition of HSV preprocessing. Notably, Experiment 10, employing RGB preprocessing, showcased a significantly higher mean precision of about 0.360 compared to Experiments 08 and 04, which achieved mean precisions of 0.297 and 0.306, respectively. This indicates that RGB preprocessing may have a more pronounced positive impact on precision within this experimental context. Furthermore, when comparing Experiment 01 and Experiment 05, marginal differences in mean precision were observed, with values of 0.321 and 0.301, respectively, suggesting that the inclusion of HSV preprocessing did not substantially alter precision outcomes. Lastly, Experiment 09 exhibited the highest mean

[6]This was performed and validated by another project group.

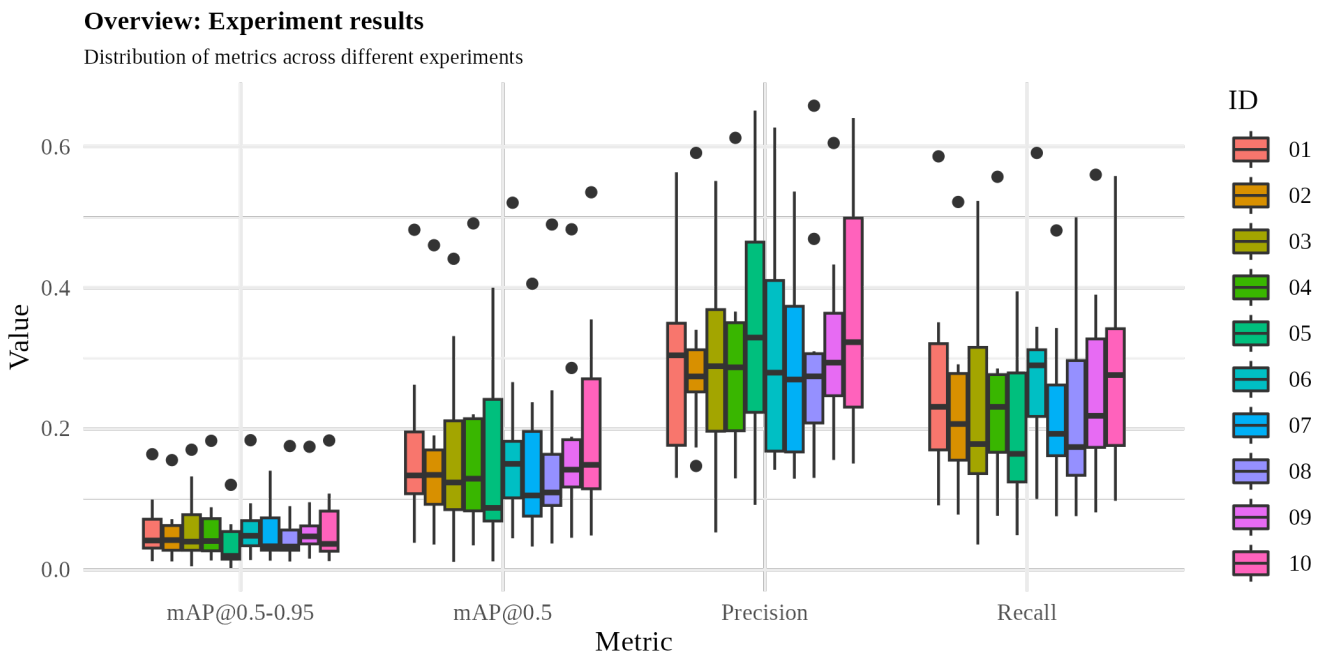


Figure 8: Boxplots describing the distribution of different metrics measured in the described experiments. Experiments are ordered and labelled as introduced in Section 5.

precision among all configurations, at 0.340, indicating the potential efficacy of RGB preprocessing in enhancing precision metrics compared to configurations solely within the DVS framework.

6.2 Recall

The findings from the experiments conducted demonstrate varying levels of recall across different configurations. Experiment 08, for instance, yielded a mean recall of 0.222, with values ranging from 0.076 to 0.481, and a median of 0.193. In comparison, Experiment 04 exhibited a slightly higher mean recall of 0.240, with values ranging from 0.076 to 0.557, and a median of 0.231. Similarly, Experiment 06 and Experiment 02 displayed mean recalls of 0.230 and 0.254 respectively, with corresponding maximum, minimum, and median values. Notably, Experiment 10, which employed RGB preprocessing, showed a significantly higher mean recall of approximately 0.281, compared to Experiments 08 and 04, which utilized HSV preprocessing, with mean recalls around 0.254 and 0.280 respectively. This discrepancy suggests a potentially more pronounced positive impact of RGB preprocessing on recall within this experimental framework. Furthermore, marginal differences in mean recall were observed between Experiment 01 and Experiment 05 configurations, with both hovering around 0.224. The disparities in maximum, minimum, and median values between these configurations were negligible. Lastly, Experiment 09 displayed the lowest mean recall among all configurations, approximately 0.196, indicating that RGB preprocessing may not have as significant a positive effect on recall when compared to configurations solely within the DVS framework.

The outcomes of various experimental configurations highlight the variability in mean mAP@0.5 values. Experiment 08, for instance, demonstrated a mean mAP@0.5 of 0.147, with values ranging from 0.033 to 0.406, and a median of 0.105. Conversely, Experiment 04 exhibited a slightly higher mean mAP@0.5 of 0.167, with values ranging from 0.034 to 0.491, and a median of 0.129. Similarly, Experiment 06 and Experiment 02 yielded mean mAP@0.5 values of approximately 0.156 and 0.179 respectively, with corresponding maximum, minimum, and median values. A significant difference was noted between Experiment 10 and Experiment 08/04 configurations. The former, integrating RGB preprocessing, showcased a mean mAP@0.5 of approximately 0.205, compared to the latter, which displayed mean mAP@0.5 values around 0.171/0.180. This suggests a potentially more pronounced positive impact of RGB preprocessing on mAP@0.5 within this experimental context. Furthermore, when comparing Experiment 01 and Experiment 05 configurations, marginal differences in mean mAP@0.5 were observed, along with negligible discrepancies in maximum, minimum, and median values. Lastly, the Experiment 09 configuration exhibited the lowest mean mAP@0.5 among all configurations, approximately 0.150, indicating that RGB preprocessing may not have as significant a positive effect on mAP@0.5 when compared to configurations solely within the DVS framework.

6.3 mAP@0.5–0.95

Experiment 08 resulted in a mean mAP@0.5–0.95 of 0.053, with values ranging from 0.013 to 0.140 and a median of 0.033. Conversely, Experiment 04 displayed a slightly higher mean mAP@0.5–0.95 of 0.059, with values ranging from 0.013 to 0.183 and a median of

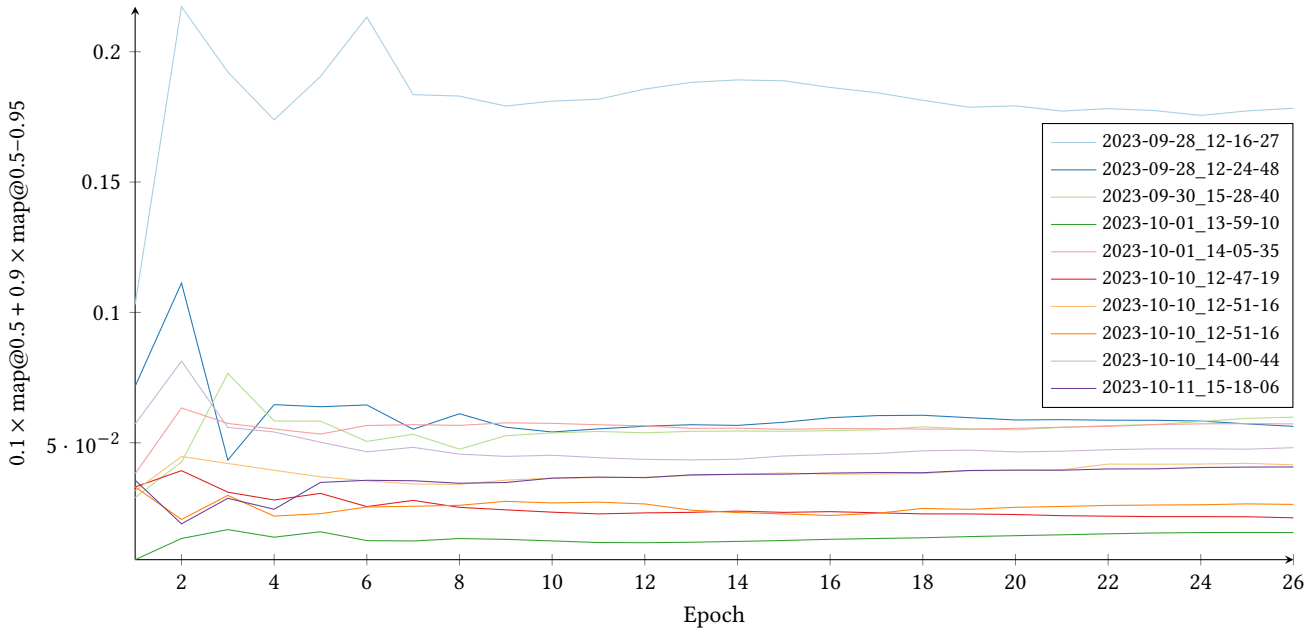


Figure 9: Evolution of fitness values along the epochs, in cross-validation of Experiment 04.

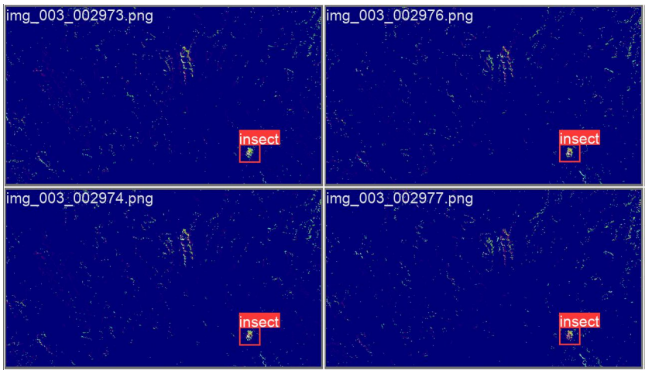
0.041. Similarly, Experiment 06 showed a mean mAP@0.5–0.95 of approximately 0.053, with values ranging from 0.012 to 0.155 and a median of 0.042, while Experiment 02 yielded a slightly higher mean mAP@0.5–0.95 of 0.060, with values ranging from 0.016 to 0.174 and a median of 0.047 (refer to Figure 8). A notable difference was observed between Experiment 10 and Experiment 08/04 configurations. The former, integrating RGB preprocessing, demonstrated a mean mAP@0.5–0.95 of approximately 0.059, with values ranging from 0.012 to 0.183 and a median of 0.037, compared to the latter, which exhibited mean mAP@0.5–0.95 values of around 0.058/0.059. Their respective values ranged from 0.164/0.184 to 0.012/0.014, with medians of 0.041/0.048 (refer to Figure 8). This suggests that RGB preprocessing may exert a more pronounced positive impact on mAP@0.5–0.95 compared to HSV preprocessing within this experimental context. Furthermore, when comparing Experiment 01 and Experiment 05 configurations, marginal differences in mean mAP@0.5–0.95 were observed, with values of approximately 0.052 and 0.056 respectively. The maximum, minimum, and median values also exhibited negligible discrepancies between these configurations. Lastly, Experiment 09 exhibited the lowest mean mAP@0.5–0.95 among all configurations, approximately 0.036, with values ranging from 0.002 to 0.120 and a median of 0.019. This suggests that RGB preprocessing may have a less pronounced positive effect on mAP@0.5–0.95 compared to configurations solely within the DVS framework.

	10	02	03	08	01	06	07	05	09
04	.491	.342	.117	.116	.048	.029	.041	.010	.008
10	NA	.447	.333	.358	.226	.136	.154	.176	.030
02	NA	NA	.296	.220	.027	.019	.051	.001	.001
03	NA	NA	NA	.472	.166	.057	.094	.063	.013
08	NA	NA	NA	NA	.116	.077	.043	.023	.003
01	NA	NA	NA	NA	NA	.158	.195	.109	.003
06	NA	NA	NA	NA	NA	NA	.462	.421	.014
07	NA	NA	NA	NA	NA	NA	NA	.462	.016
05	NA	NA	NA	NA	NA	NA	NA	NA	.021

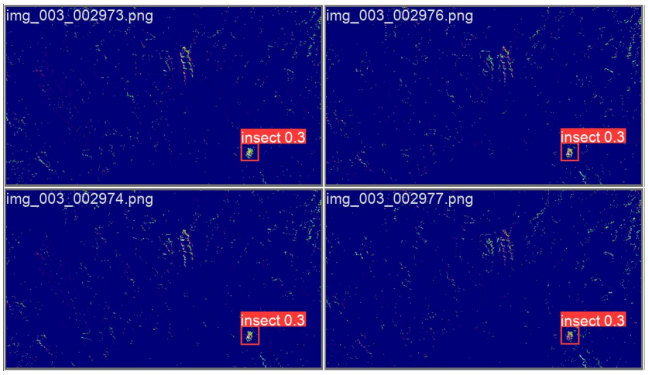
Table 4: Analyzing t-Test Results, by comparing the distribution of maximum fitness values across cross-validation elements. The rows and columns are arranged in descending order based on the mean of each experiment. The null hypothesis (H_0) posits that the true mean difference between the column value and the row value is not greater than zero, while the alternative hypothesis (H_A) suggests that the true mean difference is greater than zero. Cells are color-coded green if H_0 can be rejected at a significance level of $\alpha = .05$, indicating statistical significance; otherwise, they are colored red.

6.4 Comparison

Experiment 04 emerges as the top performer overall, boasting the highest mean fitness value among all experiments. Notably, it surpasses the original DVS stream (Experiment 01) with statistical significance. However, the introduction of RGB background subtraction in Experiment 04 does not yield a significant impact, as evidenced by the comparison with Experiment 10 in Table 4. Conversely, the amalgamation of the original RGB and DVS streams in Experiment 09 demonstrates inferior performance compared to all other experiments, as indicated in Table 4. Furthermore, the incorporation of temporal filtering to the DVS stream, as observed



(a) Labels (Ground Truth)



(b) Prediction

Figure 10: Comparison of labels vs. prediction

in Experiment 02, leads to a significant enhancement in results relative to Experiment 01. While overall HSV appears to underperform compared to corresponding RGB experiments, RGB does not exhibit significantly higher means, except in the case of Experiments 02 and 06.

7 DISCUSSION

In this study, we explored the feasibility of detecting tiny insects in cluttered environments using RGB and DVS-event stream images. The results we found not only show that this kind of detection is possible, but they also show how complex environments affect the complex link between preprocessing methods and detection success.

The use of temporal-based preprocessing approaches appeared as an important component in our research. By leveraging temporal filtering, we were able to isolate non-moving insects within DVS streams, thereby reducing the impact of background noise and enhancing the clarity of insect-related motion patterns. This finding underscores the importance of considering temporal dynamics in the preprocessing pipeline for insect detection tasks.

Our experiments utilizing background subtraction techniques produced promising results. By applying background subtraction, we successfully delineated moving insects from stationary background elements, such as plants. This preprocessing step not only facilitated the extraction of relevant motion features but also contributed to the mitigation of false positives, thus improving overall detection accuracy.

However, despite the efficacy of temporal-based and background subtraction preprocessings, our analysis revealed the susceptibility of detection outcomes to various environmental factors. Factors such as the size of insects, the presence of non-insect objects in the scene, and the speed of movement of these objects exerted significant influences on detection performance. Addressing these challenges necessitates the development of robust preprocessing strategies capable of adapting to diverse environmental conditions.

While temporal-based techniques exhibited substantial improvements in detection accuracy, the impact of the HSV transformation on model outcomes was relatively minimal. This observation suggests that color space transformations may have limited utility in certain insect detection scenarios, highlighting the need for careful consideration of preprocessing techniques based on the specific characteristics of the input data. However, the tested combinations of preprocessings were able to detect insects in DVS streams (see Figure 10).

Our investigation into the effects of incorporating RGB background subtraction as supplementary information yielded intriguing insights. Even though this preprocessing step didn't seem to improve the results of our tests, more research is needed to fully understand how it might have affected the results. Future research efforts should explore alternative approaches to leveraging RGB information effectively in insect detection tasks, considering its potential significance in other contexts.

In conclusion, our study contributes to the growing body of research on insect detection by showcasing the effectiveness of preprocessing techniques in enhancing detection accuracy in challenging environments.

8 LIMITATIONS

Despite the promising potential of our approach, several limitations must be acknowledged. The relatively small size of the dataset used for training and validation poses a primary constraint, potentially limiting the generalization capability of machine learning models and increasing the risk of overfitting, particularly in diverse natural environments where insect populations and behaviors may vary significantly. Moreover, while efforts were made to ensure accurate manual labeling of the dataset, the inherent complexity of insect behavior and the intricacies of outdoor environments may introduce uncertainties and biases into the annotations. Variations in annotator interpretation, labeling consistency, and the inherent subjectivity involved in identifying and categorizing insect species could impact the performance of trained models, potentially leading to suboptimal detection and tracking results. Moreover, the suitability of single-stage CNNs like YOLO for detecting small objects, such as insects, is a concern, as these architectures may struggle with accurately capturing the small size and rapid motion of insects in cluttered outdoor environments Thus, while our study provides valuable insights into the potential of machine learning for insect monitoring, further research and refinement in dataset collection, and model development are necessary to address these challenges effectively and enhance the robustness and applicability of insect monitoring technologies in real-world scenarios.

9 FURTHER WORKS

Utilizing the same YOLO version and configuration throughout experiments is imperative for ensuring consistency and comparability. We employed both YOLO versions 7 and 8 to facilitate this objective. While most experiments were conducted within a three-channel environment, further exploration into the impact of additional RGB input on the fourth channel is warranted. Detailed testing in this regard could provide deeper insights into the implications of utilizing the alpha channel as a YOLO input channel. Experiment 10, which combines DVS results with RGB results, allocated only one channel for the RGB stream. However, three-channel approaches exist within the RGB domain. One potential solution could involve employing leave-one-out cross-validation, thereby disregarding two of the six input bands from DVS and RGB for the experiment. Default hyperparameters were utilized in our experiments. Maximizing the hyperparameter space could potentially enhance results further. The choice of a qualitative metric for selecting a fs value for temporal filtering was employed in our experiments. Adopting a more quantitative setup, such as comparing detection metrics for different fs values in preprocessing, could validate the quality of this choice. The field of computer vision offers numerous other preprocessing possibilities beyond those explored in our experiments. Non-AI-based object detection processes, in particular, could be investigated for their efficacy in preprocessing. Low batch sizes have not been extensively studied in object detection problems [16]. Exploring the influence of different batch sizes on results could shed light on their impact and potential for improving generalization.

10 CONCLUSION

In this study, we have delved into the feasibility of detecting minuscule insects within cluttered environments using RGB and DVS-event stream images. The findings of our research not only affirm the viability of such detection but also shed light on the intricate interplay between preprocessing methodologies and detection efficacy in complex environments.

In summary, our study makes a valuable contribution to the expanding field of insect detection by demonstrating the efficacy of preprocessing techniques in improving detection accuracy, particularly in challenging environments. However, it is essential to acknowledge several limitations, including the relatively small dataset size, potential biases in manual labeling, and concerns about the suitability of single-stage CNN architectures for detecting small objects such as insects. To address these limitations and advance the robustness and applicability of insect monitoring technologies in real-world settings, further efforts are needed in refined dataset collection, model development, and exploration of alternative preprocessing methods. Future research endeavors should investigate additional preprocessing approaches, and conduct thorough evaluations of hyperparameters to drive progress towards more effective insect monitoring solutions.

ACKNOWLEDGMENTS

Calculations (or parts of them) for this publication were performed on the HPC cluster PALMA II of the University of Münster, subsidised by the DFG (INST 211/667-1).

REFERENCES

[1] Hamed Habibi Aghdam, Elnaz Jahani Heravi, et al. 2017. Guide to convolutional neural networks. *New York, NY: Springer* 10, 978-973 (2017), 51.

[2] Gary Bradski. 2000. The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer* 25, 11 (2000), 120–123.

[3] Vladimir Chernov, Jarmo Alander, and Vladimir Bochko. 2015. Integer-based accurate conversion between RGB and HSV color spaces. *Computers & Electrical Engineering* 46 (2015), 328–337. https://doi.org/10.1016/j.compeleceng.2015.08.005

[4] Tobi Delbruck. 2016. Neuromorphic vision sensing and processing. In *2016 46Th european solid-state device research conference (ESSDERC)*. IEEE, European Solid-State Device Research Conference, Lausanne, 7–14.

[5] Thomas G. Dietterich. 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation* 10, 7 (10 1998), 1895–1923. https://doi.org/10.1162/089976698300017197 arXiv:https://direct.mit.edu/neco/article-pdf/10/7/1895/814002/089976698300017197.pdf

[6] Elisabeth J Eilers, Claire Kremen, Sarah Smith Greenleaf, Andrea K Garber, and Alexandra-Maria Klein. 2011. Contribution of pollinator-mediated crops to nutrients in the human food supply. *PLoS one* 6, 6 (2011), e21363.

[7] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88 (2010), 303–338.

[8] Kunihiko Fukushima. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics* 36, 4 (1980), 193–202.

[9] Nicola Gallai, Jean-Michel Salles, Josef Settele, and Bernard E Vaissière. 2009. Economic valuation of the vulnerability of world agriculture confronted with pollinator decline. *Ecological economics* 68, 3 (2009), 810–821.

[10] Eike Gebauer, Sebastian Thiele, Pierre Ouvrard, Adrien Sicard, and Benjamin Risse. 2024. Towards a Dynamic Vision Sensor-Based Insect Camera Trap. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Winter Conference on Applications of Computer Vision, Waikoloa, 7157–7166.

[11] Chris A Glasbey and Kantilal Vardichand Mardia. 1998. A review of image-warping methods. *Journal of applied statistics* 25, 2 (1998), 155–171.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012), 9.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[14] Qaim Naqvi, Patrick J Wolff, Brenda Molano-Flores, and Jinelle H Sperry. 2022. Camera traps are an effective tool for monitoring insect–plant interactions. *Ecology and Evolution* 12, 6 (2022), e8962.

[15] Jeff Ollerton, Rachael Winfree, and Sam Tarrant. 2011. How many flowering plants are pollinated by animals? *Oikos* 120, 3 (2011), 321–326.

[16] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. 2018. MegDet: A Large Mini-Batch Object Detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Conference on Computer Vision and Pattern Recognition, Salt Lake City, 9.

[17] Sebastian Raschka. 2018. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *CoRR* abs/1811.12808 (2018), 49. arXiv:1811.12808 http://arxiv.org/abs/1811.12808

[18] Malika Nisal Ratnayake, Adrian G Dyer, and Alan Dorin. 2021. Towards computer vision and deep learning facilitated pollination monitoring for agriculture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. Conference on computer vision and pattern recognition, Nashville, 2921–2930.

[19] Malika Nisal Ratnayake, Adrian G. Dyer, and Alan Dorin. 2021. Tracking individual honeybees among wildflower clusters with computer vision-facilitated pollinator monitoring. *PLOS ONE* 16, 2 (02 2021), 1–20. https://doi.org/10.1371/journal.pone.0239504

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 10.

[21] Francisco Sánchez-Bayo and Kris AG Wyckhuys. 2019. Worldwide decline of the entomofauna: A review of its drivers. *Biological conservation* 232 (2019), 8–27.

[22] Paula Scharf. 2023. Monitoring of insects in the wild: Investigating the feasibility of machine learning based tiny object detection in time-lapse video recordings. (3 2023). "Master Thesis".

[23] Richard Szeliski. 2022. *Computer vision: algorithms and applications*. Springer Nature, London.

[24] Sebastian Thiele, Lars Haalck, Marvin Struffert, Christoph Scherber, and Benjamin Risse. 2021. Towards Visual Insect Camera Traps. (2021), 4 pages.

[25] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. 2023. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. Conference on computer vision and pattern recognition, Vancouver, 7464–7475.
[26] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. 2024. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. arXiv:2402.13616 [cs.CV]
[27] Zhengyou Zhang. 2000. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence* 22, 11 (2000), 1330–1334.

A INDIVIDUAL CONTRIBUTIONS

Disclaimer: The following responsibilities are outlined based on the collaborative nature of our project. Throughout all phases, close cooperation was maintained between team members. Tasks were distributed with one individual taking primary responsibility for specific topics, while the other provided support and assistance in their development. The subsequent list details the individual responsibilities of each team member.

A.1 Fred

During the research phase, I focused on several important aspects for the project's success. I delved into different evaluation metrics suitable for assessing insect detection performance, such as precision, recall, F1 score. I looked into version differences of YOLOv7 and YOLOv8 to understand each data structure and parameters. My involvement in implementation spanned various tasks. One contribution was the development of preprocessing pipelines aimed at facilitating the application of different preprocessing techniques to our image data. This involved implementing mechanisms to apply preprocessings uniformly across all video streams, ensuring consistency and reproducibility in our experiments. I also put a lot of effort in implementing various preprocessing techniques, such as optical flow using Farneback method, morphological transformations, contrast enhancements, temporal filtering and background subtraction. I've contributed to the transformation from RGB to DVS streams allowing for a precise bounding box location in the RGB streams. Within the preprocessing pipeline, I implemented workflows for processing of input channels, allowing for selective application of preprocessings to specific channels. In terms of project management, I defined the structure of our data and outlined clear steps for project progression. This included establishing robust data structuring practices to ensure consistency and ease of access throughout the project lifecycle. I also set up a project board to schedule tasks effectively to optimize workflow efficiency. Documentation was another area where I contributed. I took charge of documenting various steps of our research and implementation processes, ensuring that insights, methodologies, and results were comprehensively recorded for future reference and report writing.

A.2 Jakob

During the research phase, my tasks involve thoroughly exploring various methodologies relevant to the project objectives. This includes studying existing literature, exploring state-of-the-art techniques, and analyzing previous research findings in the field. I also conduct comprehensive experimentation and validation procedures, employing cross-validation techniques to rigorously test the robustness and generalization capabilities of different models or algorithms. In the implementation stage, my responsibilities extend to the practical execution of tasks necessary for achieving project goals. For instance, when converting labels from Labelbox to YOLO format, I carefully design and implement algorithms or scripts to extract frames from video data. This involves considering video formats, frame rates, and quality to ensure accurate frame extraction. Additionally, I develop algorithms to convert label annotations from JSON format to the specific text format required by the YOLO model. Partitioning the dataset into validation and training sets requires thoughtful consideration of data distribution, ensuring that both sets are representative of the overall dataset. This involves techniques such as stratified sampling or random sampling, depending on the nature of the data and the objectives of the project. Moreover, I implement strategies to handle imbalanced datasets, ensuring fair representation of all classes during training and evaluation. In terms of multithreading techniques for video preprocessing and dataset construction, my tasks include designing and implementing efficient parallel processing algorithms. This involves breaking down preprocessing tasks into smaller, independent units of work that can be executed concurrently across multiple threads or processes. By leveraging the computational resources available on the system, I aim to minimize processing time and improve overall efficiency. When initiating cross-validation, I plan and script each step of the validation pipeline meticulously. This includes tasks such as data preprocessing, model training, evaluation, and result analysis. Automation plays a crucial role in streamlining this process, ensuring consistent and reproducible results across different cross-validation folds or experiments. Throughout the project, I continuously monitor the execution of PALMA processes to ensure they are functioning as expected. This involves real-time monitoring of system performance metrics, identifying bottlenecks or issues, and implementing corrective measures as needed. Additionally, I optimize hardware usage to maximize resource utilization on the HPC cluster, enhancing throughput and efficiency while minimizing costs.

B RESULTS OF EXPERIMENTS

B.1 DVS_DVS_DVS

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.5513	0.523	0.44096	0.17012	0.76884	4.1461	0.821
2023-09-28_12-24-48	2	0.35731	0.3133	0.23516	0.08439	1.108	1.8614	1.2905
2023-09-30_15-28-40	3	0.38376	0.32128	0.20846	0.07582	1.8045	3.5738	2.0543
2023-10-01_13-59-10	4	0.15446	0.07509	0.03597	0.01184	3.0686	4.5362	3.8138
2023-10-01_14-05-35	4	0.35912	0.16769	0.1239	0.03882	2.0395	4.1501	2.1871
2023-10-10_12-47-19	2	0.14012	0.17131	0.06201	0.01965	1.9942	3.1462	1.8642
2023-10-10_12-51-16	27	0.25491	0.13634	0.09683	0.03086	2.83	3.7793	2.8406
2023-10-10_13-31-14	1	0.25245	0.18499	0.12336	0.04074	1.5973	4.0174	1.7338
2023-10-10_14-00-44	54	0.39074	0.22959	0.16609	0.05423	0.68406	2.6204	0.66271
2023-10-11_15-18-06	1	0.3639	0.11819	0.09284	0.03481	1.8836	4.6243	1.7537

Table 5: Cross-validation results of Experiment DVS_DVS_DVS. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.2 DVS_DVS–TF_DVS

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.60516	0.56015	0.48277	0.17429	0.76353	2.4551	0.83168
2023-09-28_12-24-48	2	0.4328	0.3526	0.28608	0.09562	1.1627	3.4866	1.2716
2023-09-30_15-28-40	24	0.36725	0.25127	0.18839	0.06449	2.0873	3.7211	2.2464
2023-10-01_13-59-10	20	0.16759	0.08126	0.0452	0.01561	3.1263	5.1574	3.8228
2023-10-01_14-05-35	5	0.31885	0.1728	0.13856	0.04383	2.1029	4.1312	2.3102
2023-10-10_12-47-19	2	0.15563	0.18949	0.08609	0.02874	1.98	2.8322	1.8228
2023-10-10_12-51-16	5	0.26842	0.17535	0.12694	0.03629	2.7279	3.554	2.5508
2023-10-10_13-31-14	3	0.24403	0.39004	0.17211	0.05048	1.5736	2.0685	1.7568
2023-10-10_14-00-44	4	0.25464	0.24692	0.14504	0.05464	0.66716	3.2493	0.68603
2023-10-11_15-18-06	57	0.35293	0.12294	0.11405	0.03711	1.9949	4.1047	1.8141

Table 6: Cross-validation results of Experiment DVS_DVS–TF_DVS. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.3 DVS_DVS–BS_DVS–TF

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.61245	0.55722	0.49113	0.18276	0.77857	2.8979	0.81968
2023-09-28_12-24-48	1	0.29828	0.27801	0.22029	0.0885	1.0762	2.1502	1.3277
2023-09-30_15-28-40	22	0.36597	0.27262	0.19919	0.06864	2.0715	3.9635	2.2423
2023-10-01_13-59-10	33	0.12947	0.0764	0.03446	0.01308	3.665	5.2655	4.369
2023-10-01_14-05-35	3	0.32541	0.18567	0.14072	0.04565	2.047	3.9494	2.3319
2023-10-10_12-47-19	1	0.1721	0.19532	0.07232	0.02423	2.0295	2.9002	2.084
2023-10-10_12-51-16	2	0.20218	0.16006	0.09369	0.03247	2.5918	3.4586	2.4855
2023-10-10_13-31-14	3	0.19528	0.28528	0.11731	0.03548	1.6305	2.5073	1.7367
2023-10-10_14-00-44	4	0.35856	0.26648	0.21906	0.07358	0.58948	2.9788	0.57537
2023-10-11_15-18-06	19	0.27569	0.12769	0.07995	0.02482	1.7149	4.4976	1.6175

Table 7: Cross-validation results of Experiment DVS_DVS–BS_DVS–TF. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.4 DVS_DVS–BS_DVS–BS–TF

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.62708	0.5911	0.52043	0.18359	0.77642	3.6354	0.82215
2023-09-28_12-24-48	2	0.44734	0.31459	0.26599	0.09407	1.1109	2.495	1.2586
2023-09-30_15-28-40	3	0.28888	0.34448	0.18508	0.06452	1.8421	3.6513	2.0832
2023-10-01_13-59-10	3	0.14516	0.10016	0.04451	0.0136	2.7758	4.4311	3.5375
2023-10-01_14-05-35	2	0.34214	0.21358	0.16429	0.05214	2.034	4.0387	2.3878
2023-10-10_12-47-19	2	0.15233	0.30372	0.09433	0.03319	1.9289	3.1916	1.7498
2023-10-10_12-51-16	2	0.21552	0.22859	0.12435	0.03589	2.5927	3.0114	2.4695
2023-10-10_13-31-14	1	0.14165	0.30287	0.09171	0.02655	1.5098	3.3903	1.6476
2023-10-10_14-00-44	2	0.26999	0.27671	0.17285	0.07117	0.63254	2.9519	0.60749
2023-10-11_15-18-06	47	0.43283	0.12009	0.13579	0.04395	2.1166	4.1803	1.9382

Table 8: Cross-validation results of Experiment DVS_DVS–BS_DVS–BS–TF. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.5 (DVS_DVS_DVS)–HSV

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.65805	0.49974	0.48961	0.17533	0.79112	2.9134	0.84445
2023-09-28_12-24-48	2	0.46916	0.3232	0.25435	0.09005	1.1342	2.4803	1.2976
2023-09-30_15-28-40	9	0.28913	0.29695	0.17647	0.06308	1.8912	3.9148	1.9978
2023-10-01_13-59-10	2	0.15557	0.07594	0.03709	0.01156	2.7436	4.4434	3.5269
2023-10-01_14-05-35	7	0.29653	0.14156	0.10964	0.03318	2.1196	4.2706	2.2126
2023-10-10_12-47-19	2	0.13019	0.18505	0.06778	0.02054	2.1607	3.1994	1.9219
2023-10-10_12-51-16	23	0.24366	0.13125	0.09995	0.03203	2.7338	3.6823	2.6282
2023-10-10_13-31-14	3	0.19624	0.29548	0.10928	0.03234	1.6457	2.3726	1.8238
2023-10-10_14-00-44	46	0.30964	0.16274	0.1249	0.03551	0.67285	3.5326	0.65249
2023-10-11_15-18-06	6	0.25923	0.12738	0.0881	0.02608	1.8705	3.7292	1.7368

Table 9: Cross-validation results of Experiment (DVS_DVS_DVS)–HSV. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.6 (DVS_DVS–TF_DVS)–HSV

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.59115	0.52154	0.46017	0.15536	0.78617	4.1007	0.85585
2023-09-28_12-24-48	1	0.30219	0.26381	0.17626	0.07158	1.0597	2.9597	1.2593
2023-09-30_15-28-40	21	0.31508	0.28301	0.19031	0.06281	2.2014	4.212	2.3446
2023-10-01_13-59-10	5	0.14708	0.07811	0.03555	0.01166	3.4744	4.4598	4.2447
2023-10-01_14-05-35	4	0.3402	0.16155	0.1262	0.04199	2.1239	4.1924	2.2692
2023-10-10_12-47-19	5	0.17337	0.20332	0.08548	0.02185	2.2287	3.1915	1.879
2023-10-10_12-51-16	22	0.27575	0.15321	0.11422	0.03647	2.7768	3.6379	2.7178
2023-10-10_13-31-14	6	0.26292	0.29123	0.14257	0.04179	1.8247	2.2985	1.8576
2023-10-10_14-00-44	4	0.27235	0.20986	0.15029	0.06196	0.65428	3.5422	0.64054
2023-10-11_15-18-06	19	0.24854	0.13086	0.08039	0.02452	1.8345	3.8239	1.7155

Table 10: Cross-validation results of Experiment (DVS_DVS–TF_DVS)–HSV. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.7 (DVS_DVS–BS_DVS–TF)–HSV

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.53623	0.48113	0.40552	0.14027	0.78307	2.3771	0.86457
2023-09-28_12-24-48	3	0.37944	0.34271	0.23745	0.0916	1.0523	2.2179	1.184
2023-09-30_15-28-40	24	0.37721	0.27212	0.20887	0.07397	2.0416	3.8827	2.1826
2023-10-01_13-59-10	23	0.12918	0.07581	0.03288	0.01272	3.6493	4.8803	4.4396
2023-10-01_14-05-35	3	0.28112	0.16735	0.11358	0.03529	1.987	4.0329	2.2879
2023-10-10_12-47-19	1	0.17092	0.18218	0.06537	0.02144	1.9579	3.0386	2.0934
2023-10-10_12-51-16	1	0.16561	0.15971	0.08255	0.03038	2.7603	3.2855	2.6035
2023-10-10_13-31-14	1	0.12906	0.2313	0.09688	0.031	1.6211	2.7079	1.7562
2023-10-10_14-00-44	2	0.36248	0.20313	0.15764	0.07176	0.60315	4.4363	0.57795
2023-10-11_15-18-06	1	0.25826	0.1033	0.07372	0.02681	2.1206	3.6766	1.8846

Table 11: Cross-validation results of Experiment (DVS_DVS–BS_DVS–TF)–HSV. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.8 (DVS_DVS–BS_DVS–BS–TF)–HSV

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	2	0.56359	0.58623	0.48209	0.16371	0.78566	2.5672	0.84185
2023-09-28_12-24-48	2	0.42213	0.35088	0.26231	0.09951	1.1045	2.2193	1.2706
2023-09-30_15-28-40	2	0.3547	0.34673	0.21086	0.07588	1.7042	3.4564	2.095
2023-10-01_13-59-10	4	0.13022	0.09124	0.03812	0.01195	2.9066	4.5044	3.6173
2023-10-01_14-05-35	6	0.32983	0.18038	0.1434	0.04659	2.1499	4.0052	2.3014
2023-10-10_12-47-19	2	0.14714	0.23178	0.0821	0.02779	1.9168	2.8835	1.7974
2023-10-10_12-51-16	27	0.27822	0.16639	0.12385	0.03606	2.6266	3.5838	2.5393
2023-10-10_13-31-14	1	0.14558	0.24195	0.10873	0.03016	1.5867	3.0195	1.7611
2023-10-10_14-00-44	4	0.2642	0.23014	0.14908	0.05836	0.63114	2.4242	0.62397
2023-10-11_15-18-06	47	0.33383	0.11925	0.10739	0.03192	1.9835	4.4006	1.8298

Table 12: Cross-validation results of Experiment (DVS_DVS–BS_DVS–BS–TF)–HSV. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.9 RGB-R_RGB-G_RGB-B_DVS

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	55	0.6511	0.3947	0.3999	0.1203	0.03937	0	0.004529
2023-09-28_12-24-48	6	0.4477	0.2941	0.2722	0.06106	0.0722	0	0.005339
2023-09-30_15-28-40	15	0.4702	0.2833	0.2586	0.06439	0.09284	0	0.01846
2023-10-01_13-59-10	33	0.1558	0.1274	0.04861	0.009769	0.1007	0	0.02414
2023-10-01_14-05-35	46	0.2265	0.1234	0.06769	0.01831	0.09055	0	0.02889
2023-10-10_12-47-19	3	0.09192	0.0489	0.01175	0.002066	0.07221	0	0.01162
2023-10-10_12-51-16	55	0.222	0.1811	0.09328	0.02065	0.05841	0	0.01553
2023-10-10_13-31-14	3	0.4296	0.266	0.1902	0.03299	0.04384	0	0.004939
2023-10-10_14-00-44	7	0.2288	0.1474	0.07283	0.01379	0.03431	0	0.00183
2023-10-11_15-18-06	47	0.4725	0.09537	0.0819	0.0181	0.05152	0	0.00893

Table 13: Cross-validation results of Experiment RGB-R_RGB-G_RGB-B_DVS. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.

B.10 DVS_DVS–BS_DVS–BS–TF_RGB–BS

		Metrics				Validation Loss		
Name	Epoch	Precision	Recall	mAP@0.5	mAP@0.5-0.95	Box	Cls	Dfl
2023-09-28_12-16-27	4	0.6406	0.5583	0.5353	0.1832	0.03171	0	0.00567
2023-09-28_12-24-48	0	0.2546	0.3269	0.1631	0.03879	0.05182	0	0.006787
2023-09-30_15-28-40	1	0.5225	0.3972	0.3549	0.09688	0.08129	0	0.02249
2023-10-01_13-59-10	6	0.1817	0.09767	0.04849	0.01215	0.07865	0	0.03429
2023-10-01_14-05-35	11	0.2783	0.1639	0.1138	0.03036	0.08178	0	0.04567
2023-10-10_12-47-19	1	0.1506	0.2304	0.07404	0.01807	0.06409	0	0.01793
2023-10-10_12-51-16	0	0.2226	0.2129	0.1175	0.02476	0.08592	0	0.02262
2023-10-10_13-31-14	2	0.4273	0.3467	0.2278	0.04142	0.06033	0	0.007622
2023-10-10_14-00-44	4	0.5519	0.3216	0.2849	0.1078	0.02651	0	0.003197
2023-10-11_15-18-06	9	0.367	0.1524	0.1342	0.03424	0.06514	0	0.0175

Table 14: Cross-validation results of Experiment DVS_DVS–BS_DVS–BS–TF_RGB–BS. For each validation split the epoch with best fitness and the respective metrics and validation loss is listed.