Computer Vision

Project: Counting bees

Sven Schnydrig
sven.schnydrig@student.unisg.ch

January 12, 2024

1 Introduction

The goal of this project was to count and detect bees in images. We started from simple methods like color tresholding, then moved to detection with machine learning models using hog features and a sliding window approach and last but not least using deep learning with state-of-the-art pretrained YOLOv8 models. The colors tresholding yielded poor results as the colors of the bees and background is very similar in certain places, the machine learning approach yielded better results and the deep learning approach yielded excellent results.

2 Color thresholding

Figure 1: Example of color thresholding applied to an image

In figure 1 we can see that the tresholding becomes especially difficult in the section below the middle and above the bottom where the color of the background and the bees is very similar.

3 Machine Learning Part

For the machine learning part I extracted all the bee patches of all the images in patch sizes of 150x150 with a center crop based on the coordinates in the label files. I extracted also 4 times as many random background patches as bee patches per image in order to be able to train a classifier to distinguish between bee and background patches. I then used those bee patches and background patches and put them together and computed hog features for every patch. Subsequently I trained various machine learning models like KNN, Logistic Regression, SVM, LightGBM and XGBoost to classify whether a patch contains a bee (1/True) or not (0/false) with the hog features as inputs and the labels (1/0 - bee/background) as targets. This yielded satisfactory results in terms of Precision, Recall and F1 score on the validation set of the patches:

Classifier	Class	Precision	Recall	F1-Score
KNN	background	0.84	1.00	0.91
	bee	0.99	0.22	0.36
Logistic Regression	background	0.98	0.98	0.98
	bee	0.91	0.93	0.92
Light GBM	background	0.98	0.99	0.98
	bee	0.97	0.90	0.93

Table 1: Classifier Performance Metrics on validation set

The chosen logistic regression model stood out due to its computational efficiency and its capability to facilitate the comparison of prediction confidence through probability scores. Therefore, it was selected for further analysis in this project. To generate predictions on actual images from the test set, a sliding window approach was employed. This approach utilized a window size of 150x150 pixels with a stride of 50 pixels. To mitigate the issue of double-counting bees we apply Non-Maximum Suppression (NMS). For each patch where a prediction was made, a ground truth label of "bee" was assigned if the intersection over union (IoU) with any of the labeled bee bounding boxes exceeded 0.5; otherwise, it was considered a background patch.

The accuracy of the bee count per image was evaluated by comparing the number of positive class predictions against the actual count of bees. This comparison was quantified using three key metrics: the Mean Absolute Percentage Error (MAPE), the Mean Absolute Error (MAE), and the Root Mean Squared Error (RMSE). The results of this analysis are presented in Table 2.

Metric	Value
MAPE	20.77%
MAE	3.55
RMSE	4.65

Table 2: Model Error Metrics in bee count over images - test set

These results indicate a moderate level of accuracy in bee counting with the logistic regression model. While the MAE and RMSE values suggest reasonable error margins, the MAPE of 20.77% reflects room for improvement in the model's precision, particularly in scenarios with high bee densities or complex backgrounds.

3.1 Qualitative Evaluation

Figure 2: Example of machine learning predictions with sliding window on a single image of test set

In Figure 2 we can see the predictions of our machine learning model with the sliding window approach described above. The good news is that many bees get recognized correctly, the bad news is that we also have multiple false positives due to darker shades in the wood and some bees that don't get recognized due to the strong similarity with the background. Furthermore the fit of the bounding boxes is also not always perfect as both the stride and size of the boxes may be too large.

4 Deep Learning Part

Using the ultralytics library, a pretrained YOLOv8l model was fine-tuned for bee detection. The process involved hyperparameter tuning over 100 iterations for 10 epochs to find suitable hyperparameters for the model. The final model was then trained for 50 epochs with the hyperparameters that offered the best performance on the validation set during tuning.

The fine-tuned YOLOv8l model demonstrated excellent performance metrics, as shown in Table 3. These results underscore its high precision and recall, along with impressive mean Average Precision (mAP) scores, affirming its effectiveness in the bee detection task.

Metric	Value
Precision (B)	0.9572
Recall (B)	0.9431
mAP50 (B)	0.9767
mAP50-95 (B)	0.7473
Fitness	0.7702

Table 3: Performance Metrics on test set

In terms of error metrics for the bee count over images, presented in Table 2, the YOLOv8l model achieved significantly lower metrics than the machine learning approach with a MAPE of 9.89%, along with a MAE of 1.69 and RMSE of 2.15, highlighting its accuracy and reliability in bee counting.

Metric	Value
MAPE	9.89%
MAE	1.69
RMSE	2.15

Table 4: Model Error Metrics in bee count over images - test set

4.1 Qualitative Evaluation

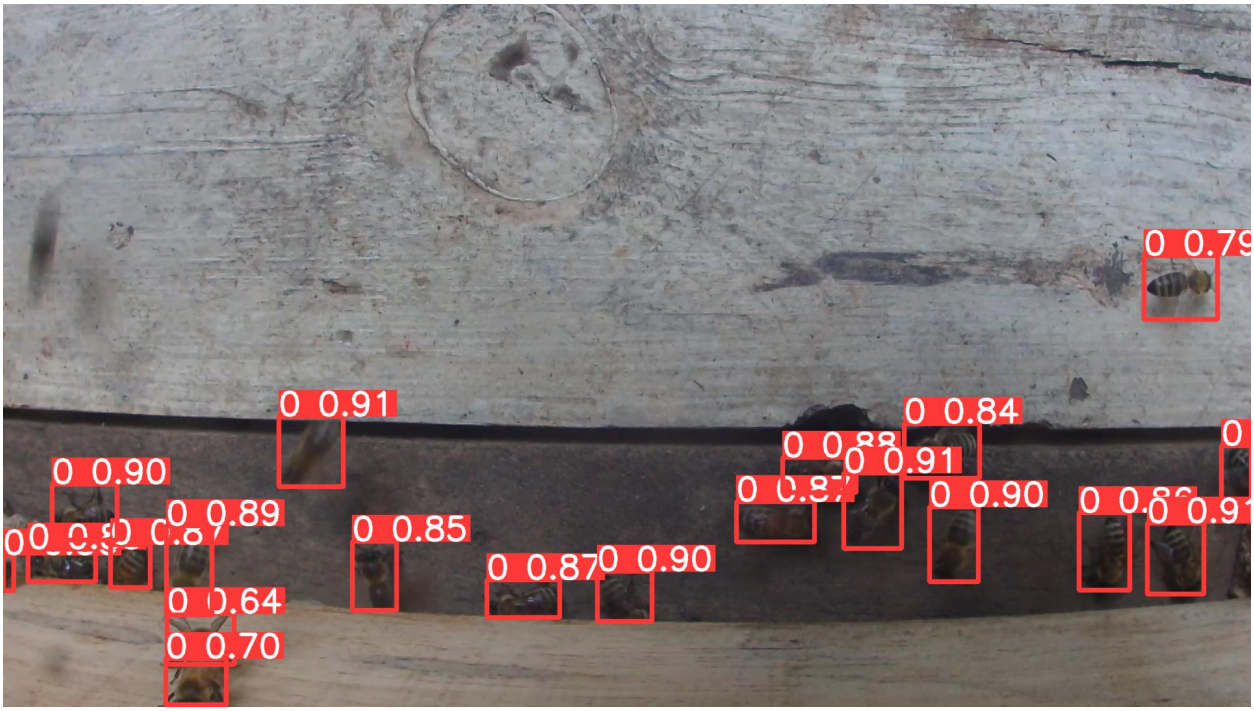

Figure 3: Example of YOLOv8l prediction on a single image of test set

When looking at the same picture example than before but with the YOLO detection in figure 3 we can see that the detection is significantly better. Even bees with strong overlap to other bees or bees on very dark background get recognized correctly. Moreover, the bounding boxes around the bees are now much more precise and snug.

5 Comparison

5.1 Quantitative Comparison

The ML approach had an MAE of 3.55, while the DL approach achieved a significantly lower MAE of 1.69. This suggests that the DL approach is more precise in counting the number of bees, likely due to its more powerful features extraction and adaptive bounding boxes which translates to better generalizability to detect bees in various poses and positions. Nevertheless the machine learning approach showed also impressive results based on simple hog features. The disadvantage of the machine learning approach was that a lot of manual experimenting and finetuning of variables like patch size, stride and features to use for training was necessary, as opposed to the deep learning approach where all of this happens automatically. Furthermore because of the fixed patch size and

the sliding window approach our patches may not always encapsulate the bees fully and the predicted boxes are less snug.

5.2 Qualitative Comparison

When we look at Figure 2 and Figure 3 for the qualitative comparison we can see that while both models perform well, the deep learning approach clearly has the upper hand. Even bees with very strong overlap or bees on very dark background get recognized with deep learning and are often missed with the machine learning approach. Furthermore the deep learning approach does not have any false positive in this particular example opposed to the machine learning models who predicts a bee in multiple areas where there is not any bee close by.

6 Conclusion

In this study, two distinct approaches were evaluated for the task of bee detection: the Machine Learning (ML) approach using HOG features combined with Logistic Regression and a sliding window approach with NMS, and the Deep Learning (DL) approach utilizing YOLOv8.

The ML approach, characterized by its simplicity and ease of interpretation, showed potential for faster training on less sophisticated hardware. However, it exhibited limitations in terms of accuracy in object counting. The approach's reliance on manual feature extraction and a computationally intensive sliding window preprocessing technique with Non-Maximum Suppression (NMS) further contributed to its reduced robustness, especially in handling variations in object size and background conditions.

Conversely, the DL approach, employing the YOLOv8 model, demonstrated significantly higher accuracy. It proved robust against variations in object size and challenging background conditions and was efficient in managing overlaps and different object densities. Despite these advantages, the YOLOv8 model necessitated much greater computational resources for training and presented a more complex model architecture, which could be challenging to interpret.

Overall, while the ML approach offers benefits in terms of simplicity and resource efficiency, the DL approach, particularly YOLOv8, stands out in terms of accuracy and adaptability to varying conditions, albeit at the cost of increased computational demand and complexity. Furthermore while the machine learning models are computationally cheaper and therefore faster to train than the YOLO models, they're slower at inference due to the sliding window mechanism with NMS. It's also worthwhile to note that the deep learning approach was significantly easier to code, although that is only the case thanks to the available pretrained YOLO models and the ultralytics library.