NHL STENDEN
computer vision & data science

Mid-air aphid detection: A comparison between state of the art object detectors

NHL Stenden Lectoraat in Computer Vision & Data Science

Rick Scheper, Maurits Peereboom

Supervisors: Klaas Dijkstra, Lucas Ramos

Abstract—The potato virus Y can infect potatoes and heavily impact the strong position Dutch potato farmers have on the market. Because this virus gets carried by aphids, the usage of pesticides is crucial in order to prevent the food waste of hundreds of potatoes. Overuse of pesticides can strongly impact the environment. Which is why they can only be used when it is proven that aphids are present. Determining this will prove difficult: the insects are relatively small and can be mistaken for fruit flies. Computer vision may offer a solution here. Within this research, two detector models, YOLOv8 and RetinaNet are trained on a dataset containing winged aphids, wingless aphids and 2 kinds of fruit flies in order to determine which one is the best in detecting aphids in an early state as possible. Not only will these detectors be trained to be able to detect aphids from other insects, but also to determine exactly what the currently viewed insect is. The latter proves to be more difficult for the models with YOLOv8 giving a best F1 score of 0.84 and RetinaNet 0.53. As opposed to just determining an aphid from a non-aphid with F1 scores of 0.90 and 0.76. With this data, it can be determined that YOLOv8 is the better model for detecting aphids.

Index Terms—YOLOv8, RetinaNet, Aphids, Agriculture, Environment, Tiling, F1 score

1 Introduction

The Dutch export 800.000 tons of seed potatos yearly to more than 70 countries and have a marketshare of roughly 60 percent with a value worth of almost half a billion euro's. This strong export position is due to a high quality and health of the potato seeds. However, in the last five years there has been a rise in the total affected crops infected with Potato virus Y (PVY)[11]. This reduces the value of the product and results in a lower profit. To maintain the strong economic position the dutch have on these potato seeds a new method is required to stop the spread of PVY.

When a potato plant gets PVY, it will develop dark spots on its leafs which will eventually fall of, leaving a bare stem with muttled leaves at the top [9]. This greatly affects the yield of potato's which in turn threatens the strong economic position of the Dutch export in seed potato's.

PVY is spread by insects, particularly aphids, from infected to healthy plants. This assumption is supported by the direct relationship between the number of aphids and the percentage of the infected crops[5]. The detection of these aphids is really time consuming and needs to be made more efficient. As of now, the aphids get detected by putting a sticky yellow plate in the field. The aphids are attracted by the color and will get trapped once they land on the plate. They are then counted manually which is used to determine the amount of pesticides needed for the entire field.

This process is very slow and prone to human error; therefore, computer vision may be used to automate it. In a previous study, computer vision was implemented to automate the detection of aphids [12]. The object detector used in that study was trained to identify insects and aphids on a yellow plate. However, for this research, a different dataset containing aphids in flight will be utilized. This dataset better simulates real-world scenarios, where aphids are not stuck on a yellow sticky plate but flying in the field. Hence, it is crucial to investigate whether an object detector is capable of detecting these aphids in flight.

To detect these aphids in flight, an object detector will be used. Because there are a lot of different detectors, each with their own advantages and disadvantages, it is required to pick a selection of those. Part of this research will be determining which of these systems within the selection is the most desirable within the current stage of the aphid-detection project.

In short, to achieve minimal damage to the environment, while increasing the export of the potato harvest, confirming presence of aphids in an early stage is required. Which is why an object-detector, more advanced than the one used in the previous research [12] will be used. Because only one of all the available models will be used, finding the advantages and disadvantages of them is also necessary. The main research question is:

How accurate is the detection of aphids in flight, using computer vision?

In order to answer this question, research for the following sub-questions is necessary:

- *Which object detection model is the most fitting?*
- *What is the difference in performance when more classes are added?*

2 State of the art

The aim of this study is to determine the accuracy of computer vision-based detection of aphids in flight. The state of the art related to this question will be reviewed in previous research. Section 2.1.1 will provide a detailed analysis of the YOLOV8 object detector and assess its performance. Section 2.1.2 will focus on the RetinaNet object detector, explaining how it works and summarizing previous

Rick Scheper is an Applied mathematics student at the NHL Stenden University of Applied Sciences, E-mail: rick.scheper@student.nhlstenden.nl.

Maurits Peereboom is a Software Engineer student at the NHL Stenden University of Applied Sciences, E-mail: maurits.peereboom@student.nhlstenden.com.

Klaas Dijkstra is a Professor of Applied sciences at the NHL Stenden Lectoraat in Computer Vision & Data Science, E-mail: klaas.dijkstra@nhlstenden.com.

Lucas Ramos is a researcher at the NHL Stenden Lectoraat in Computer Vision & Data Science, E-mail: lucas.ramos@nhlstenden.com.

studies conducted using the detector. 2.2 looks at the previous research done on this subject and section 2.3 will provide extra information of object detection on insects in the field.

2.1 Object detection model candidates

After investigating different candidates, the selection of object detection models has come down to RetinaNet [**RetinaNet**] and YOLOv8 [4]. These are the state of the art object detectors and will therefore be used. The latter being the advanced version of the object detector used within the previous stage of the aphid-detection project: YOLOv5. it is also interesting to see whether YOLOv8 performs indeed better than it is predecessor. Therefore YOLOv5 will also be used in the experiments.

2.1.1 Object detection with YOLOV8

YOLOV8 is the latest object detector based on the YOLO framework. It is based on the DarkNet architecture [7]. Although this is a new version the frameworks of all the YOLO versions are the same. The main difference is that the YOLOv8 does not use anchors anymore. In previous versions of YOLO anchors were used and these were hard to compute [13]. YOLOv8 however computes the direct center of an object instead of the offset from a known bounding box. This step leads in a smaller computing time. The object detector is a single stage method that performs the detection and the classification of an object at the same time. There is a lot of research done with the YOLO framework and all have provided a good performance. One study for example trained YOLOv5 on a dataset consisting of more than 7000 images of insects [10]. Here YOLOv5 achieved a precision and recall of more than 90 for various Yolo models.

2.1.2 Object detection with RetinaNet

When it comes to object detection models, there is a difference between a one-stage and a two stage model: whilst one stage models can be faster and more simple, two stage models give better results, due to their higher accuracy. In order to achieve this, RetinaNet utilizes a focal loss function. The idea behind this function is that it reshapes the standard cross entropy loss. By doing this operation, the source lacking accuracy of one stage models is mitigated: the extreme foreground-background class imbalance, which occurs while training said models [15]. Even though this sounds promising, it has to be stated that the YOLOv8 model is more recent than the RetinaNet model. Previous research on the detection of different pills, concluded with RetinaNet having good accuracy, but slow detection speed, in comparison with YOLOv3. (The latter having less accuracy, but higher detection speed.) [8] This is important to remember at experimenting, because the model needs to detect flying aphids.

2.2 Previous research

In [12], yellow sticky plates of 250 by 250 mm have been used to trap aphids, among other insects, in order to train a detection model. This detection model was YOLOv5, a previous version of YOLOv8. In order to achieve desirable results, [12] has used tiling on the training, validating and testing batch of images. This means slicing the high resolution input image into a bunch of smaller cutouts. This way, the model can look at the high detail data, without the cost of computing power that comes with processing high resolution images. For the current stage of this project, high resolution input is used as well and tiling needs to be taken into consideration.

2.3 Previous research with insects in the field

Within this research, computervision used to detect insects within a lab environment. Because it is crucial to detect the aphids as early as possible, future research may use a dataset which consists not only of aphids on potato plants. But, more importantly, also of aphids in their flying state outside on the farm fields. Which is why it is required for the model to detect aphids mid-flight. Previous research has been performed on the matter. [3] [2] [14] And there are two notable factors which need to be taken into account for this current experiment. First of all, it seems that computer vision on detection of insects is very efficient. Especially in the case of convulational neural networks, where the model produced a 90% accuracy on the provided insect dataset [14]. However, all of the used datasets within the aforementioned research does not completely match with the goal of this research. This is because these dataset contain images of plants with bugs on top. The model which needs to be trained has aphids in flight in front of a yellow plate. With the right camera and image tiling, a process in which one big picture gets divided in smaller pictures, allowing for tiny details to become visible, the model should, in theory, achieve the desired result of detecting aphids in mid-air.

3 Materials and Methods

This section includes the presentation of the study requirements and the specific techniques that have been utilized. The acquisition of the dataset 'mixWinglessGreenpeachFruit' is discussed in 3.1. In section 3.2, the definition of the ground truth and the method of data division are elaborated. Section 3.3 covers the discussion of data pre-processing, while section 3.4 focuses on the utilization of object detection models. The evaluation metrics used in this research are shown in section 3.5.

3.1 Dataset acquisition

it is important in this research to detect aphids in their natural state rather than on yellow sticky plates [12]. Therefore, a new dataset comprising aphids and other insects in their natural state was collected. these aphids were dropped into a large wooden box and would pass a motion sensor. When a aphid was noticed by the sensor a camera would take a picture of the aphid. Many pictures were taken without any aphids present. The camera setup is shown in Figure 1.

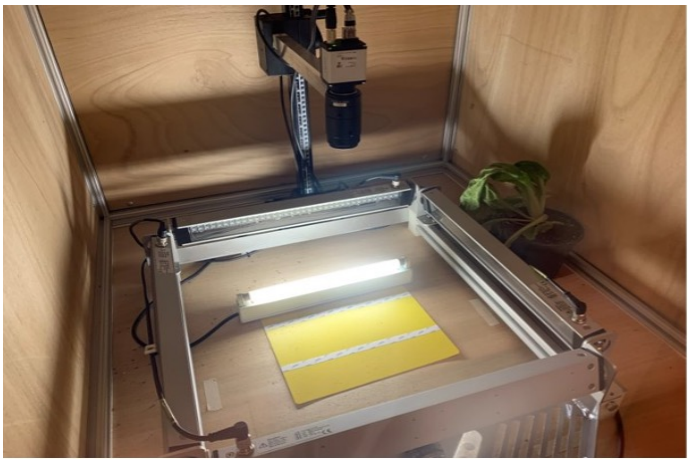

Fig. 1. Camera setup for data acquisition

With this setup, it is possible to capture these green peach aphids while they were spreading their wings. These images where taken in a controlled environment using a U3-3990SE Rev.1.2 camera. the specifications of this camera are shown in table 1.

Component	Specifications
Camera	Sony IMX541
Camera lens	KOWA LM16FC
Camera resolution	4512 x 4512
Sensor	1.1" CMOS

Table 1. Camera specifications.

3.2 Ground truth, annotations, and dataset division

3.2.1 Ground truth

The ground truth was collected by annotating the images using annotation software. This dataset contains aphids, virillis, and melanogasters. The location of these aphids is then annotated by surrounding bounding boxes around their bodies. After the annotations are complete, the combination of the photos, the class names, and the bounding boxes is considered the ground truth. This ground truth is then used to train the object detectors. Figure two shows a winged aphid with its bounding boxes.

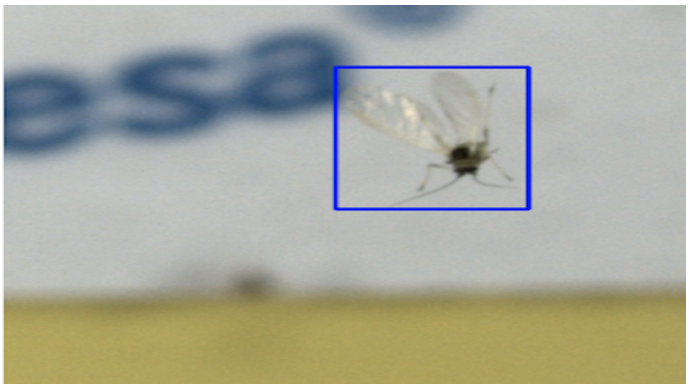

Fig. 2. A winged aphid with its corresponding bounding boxes

3.2.2 difference in annotation

This research we used two different types of annotations depending on the experiments. As mentioned, the dataset consists of aphids and non-aphids. In the first set of experiments, the images were annotated accordingly. To introduce additional complexity to the model, the following experiments have utilized more elaborate annotations. Instead of having two classes, five different classes have been created with the following labels: 'Aphid-winged,' 'Aphid_wingless,' 'Aphid_dropped,' 'Virillis,' and 'Melanogaster.' The subsequent experiments assessed whether this increased complexity affects the model's performance.

3.2.3 Dataset division

The combination of pictures and corresponding annotations resulted in the dataset: mix_wingless_greenpeach_fruit, which consists of 727 images. By dividing this dataset into train, test, and validation sets, the network produces a model that has not seen all the data. If the validation and test sets are also used for training, the model may become overtrained, resulting in worse performance on new data. This increases the changes the model will generalize to new, unseen data.
The amount of images for the train, test and validation sets can be found in table 2

Train	Validation	Test
556	62	109

Table 2. dataset splits

3.3 Tiling

To ensure that the details of the aphids were captured by the camera, the images were taken at a high resolution. The images in the dataset have a width and height of 4512 pixels. However, if these images were directly fed to the model, it would require a significant amount of computing power and time for processing during training. The testing process will also require a significant amount of computing power. Therefore, it is necessary to tile these images to only include the regions with aphids present. Tiling involves cutting smaller portions of the image without compromising its resolution, ensuring that all the details of the aphids remain visible. In this study, the tile size is set at 416x416 pixels.

3.3.1 Tiling approach

In this experiment, tiling was not implemented inside the object detector. This means the images were tiled before the training process and saved on the device. The object detector can then access the tiled images and train on them. It is important to ensure that the training, validation, and testing sets contain the right amount of information. To achieve this, two different approaches were used.
For the training set, images were tiled using a random positive tiler. This means that the tiler would only extract the parts of an image that contain aphids. This way, the model can effectively learn whether an aphid was present in the image.
The validation set was tiled in the same manner as the training set. However, it also needed to include some background images since the testing set consists of 90% background images. Therefore, a portion of those background images was transferred to the validation set
For the test set, it is crucial to include images without aphids. Otherwise, it will not be clear how the model performs when presented with negative data. To address this, a different tiling approach was necessary. A fixed tiler was used for the test set, which completely tiled all the images into 121 416x416 chunks.

3.4 Object detectors

3.4.1 YOLOv8

There are various models that can be used within YOLOv8, namely: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x6. The differences between these models is mainly the size of the backbone and the number of parameters. A model with a larger backbone may potentially lead to a better result, but it will take longer to achieve this result. It is therefore important to consider the tradeoff between model complexity and computing time. Because the dataset is relatively small the models this research will be using are: YOLOv8n, YOLOv8s and YOLOv8m. the other models are too complex and could likely start overfitting the data. This means the model could memorize the training set instead of learning what an aphid looks like.

3.4.2 RetinaNet

RetinaNet utilizes a Resnet backbone [15]. ResNet depths of 18, 34, 50, 101 and 152 being currently available. Note that in [15], a 101 backbone is used. During the experiments, different models with different backbones will be tested. It is important to mention that deeper models take more memory and time to train. The used optimizer is the Adam model. It will use a learning rate of 0.0001. And the employed scheduler will have a patience of 20.

3.4.3 YOLOv5

Besides the results within the project that YOLOv8 will produce, it is also interesting to see whether these results are better than its predecessor. Therefore the same experiments that are conducted with YOLOv8 will also be conducted with YOLOv5. In this comparison it will become clear if YOLOv8 is indeed faster and more accurate. The experiments will be done using a adjusted YOLOv5 model. This model was adjusted in such a way that it is compatible with YOLOv8. This information is obtained from a github page of YOLOv8 [6].

3.5 Train, test, and validation

3.5.1 Training and validation on the data

Before the training process, the high-resolution input is randomly tiled to create input for the training process. This random tiling does make sure that each tile has an insect. For validation, fixed tiling is applied, to check if the model can also detect tiles with only background. Following this, the model will iterate over this input. It

needs to do this a set number of times. Every epoch the model will validate over the validation input. At the end of validation, it is also decided if the model gets saved or not. This is done if the model shows improved results during validation. The model might actually give less accurate results on validating, due to a process called overtraining (see 3.2.2). All trained models have pre-trained weights trained on the coco-dataset [1].

3.5.2 Testing on the data

For the testing process, fixed tiling is applied. Furthermore, the testing input needs to have the same measurements, tile size, background and class types as the training and validation input. The output of the testing process should consist of the reconstructed image, with the predictions made by the model, as well as the ground truth.
One of the challenges during fixed tilling was that sometimes the same object of interest (insect) would be split between two different tiles. To understand this problem, firstly consider that each annotation and prediction is a box. This box makes sure that the object is completely inside it. It is the model's goal to get a box as closely as the annotation box. Because for fixed tiling, the objects are not taken into account, images will appear in which the object is sliced in half. The annotation is also sliced in half, but it will still take the other half into account. The model does not have that information, so it will not. This is why an intersection over union threshold of 0.1 is used, when calculating the statistics.

4 Experiments & Results

This section will cover the different experiments that were conducted. In section 4.1 the results of YOLOv8 and YOLOv5 aphid vs other insects are shown. Section 4.2 will show the results of YOLOv8 and YOLOv5 with classes: 'aphid_winged', 'aphid_dropped', 'aphid_wingless', 'virilis' , 'melanogaster'. Section 4.3 highlights how different ResNet backbones are trained on the dataset with 2 classes and what their performance is. In section 4.4, the same kind of experiments will be done but on the aformentioned 5 classes dataset.

4.1 Experiment 1: YOLOv8 with two classes.

This experiment involves training several YOLO models with different backbones. The models used in this experiment are the nano (n), small (s), and medium (m) variations of YOLOV8. The results are shown in Table 3. These experiments are performed on a dataset were two classes are used, as explained in section 3.2.2. It is interesting to see that YOLOv8 clearly outperforms YOLOv5 when the recall is considered. YOLOv8 detects a lot more than its predecessor. It is also remarkable that the nano variant of YOLOv8 has the best F1-score. The more complex and larger models do not improve on this metric. The only thing these larger models do better is in there average precision. Figure three shows the confusion matrix of the YOLOv8n model. This confusion matrix shows that 82 % of all the insects in the test set are detected and are assigned the correct label. 18 % of the insects in these background are not detected and were treated as if they were background.

4.2 Experiment 2: different YOLO models on 5 classes

In the second part of the experiments YOLOv8 and YOLOv5 were trained on a dataset with five classes explained in section 3.2.2. The results of these experiments can be found in table 3. It is interesting to see that when the YOLOv5 models are compared to their respective YOLOv8 counterpart, YOLOv5 outperforms YOLOv8, except the medium variant of the model provides better results. It is also worth noting that the precision of all the models (except for YOLOv8n) is very high. Whenever the model detects something, it is almost always correct. The problem with these models lies in their recall. The models find it difficult to detect these aphids, which results in many undetected insects. This can be seen in figure four which shows the confusion matrix of the YOLOv8s model. for each class the model detects the insects in roughly 70 % of the cases. The other 30 % are unnoticed and mostly gets treated as if it were background.

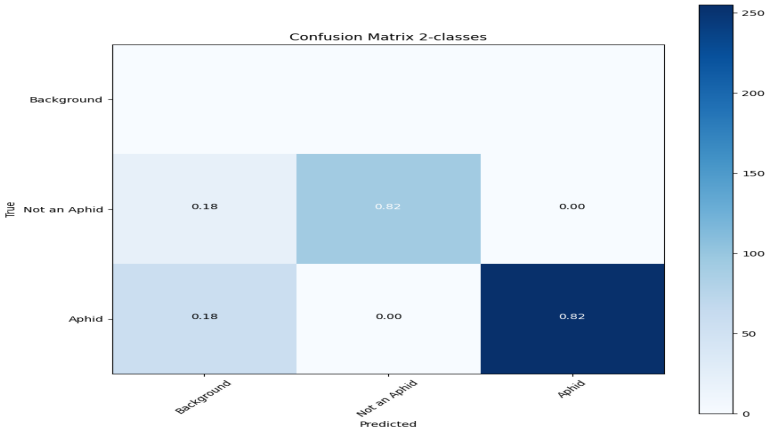

Fig. 3. Confusion matrix for YOLOv8s with two classes

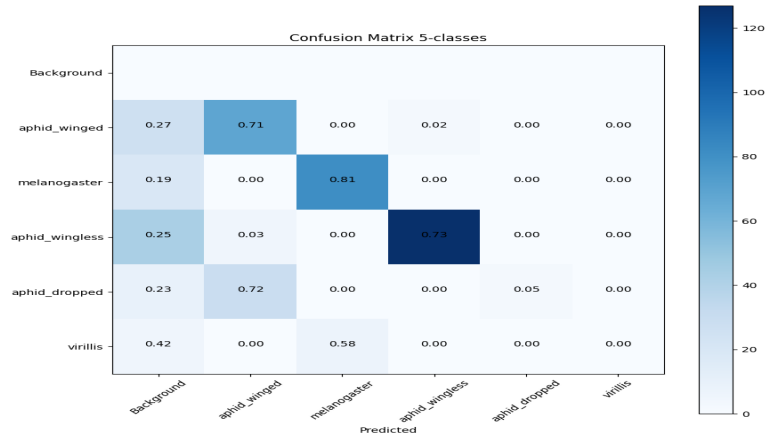

Fig. 4. Confusion matrix for YOLOv8s with five classes

4.3 Experiment 3: RetinaNet with two classes

As aforementioned, ResNet backbones of 18, 34, 50, 101 and 152 are available. Within this experiment, for each different backbone, a RetinaNet has been trained on the same dataset that was used in experiment 1, to determine which backbone delivers the most desired results. They have been trained on a maximum of 15 epochs, because earlier experiments have shown that the validation loss reaches its lowest point around there. The learning rate is 0.0001 and the batch size is 1. All models have been pretrained beforehand. For the intersection over union threshold within the predict function of the model, it has been decided that 0.5 is going to be used as well as 0.05. A threshold of 0.5 gives a nice balance of the precision and recall, while 0.05 gives more recall. More recall is desired for the end-result, because it is important for the farmers to detect aphids. A false positive is therefore more desirable than a false negative. The results of these models are shown in table four. Note that the results shown is the mean of all the statistics obtained for each class. Where the f1 score = $2 * (precision * recall)/(precision + recall)$

A RetinaNet with a ResNet backbone of 18 performs the best on this dataset. It has a precision of 0.65, a MAP with the same value and a F1 score of 0.76. Only its precission of 0.91 is outclassed by the

Table 3. Performance Metrics of the YOLO Models

Model	Precision	Recall	MAP-95%	F1-score
two classes				
YOLOv5n	0.90	0.76	0.75	0.81
YOLOv8n	**0.98**	**0.84**	**0.80**	**0.90**
YOLOv5s	0.93	0.71	0.79	0.81
YOLOv8s	0.98	0.82	0.83	0.89
YOLOv5m	0.96	0.72	0.82	0.83
YOLOv8m	**0.98**	**0.82**	**0.89**	**0.89**
five classes				
YOLOv5n	0.92	0.76	0.79	0.83
YOLOv8n	0.42	0.73	0.87	0.54
YOLOv5s	**0.98**	**0.75**	**0.83**	**0.85**
YOLOv8s	0.87	0.73	0.86	0.79
YOLOv5m	0.98	0.69	0.83	0.81
YOLOv8m	**0.99**	**0.73**	**0.89**	**0.84**

Table 4. Results for RetinaNet with different ResNet backbones, on a dataset with 2 classes, for two different IoU thresholds

Model	Precision	Recall	MAP	F1 score
IoU threshold = 0.5				
ResNet-18	0.91	**0.65**	**0.65**	**0.76**
ResNet-34	**0.94**	0.59	0.59	0.73
ResNet-50	0.82	0.62	0.61	0.71
ResNet-101	0.85	0.64	0.63	0.73
ResNet-152	0.82	0.63	0.62	0.71
IoU threshold = 0.05				
ResNet-18	**0.70**	**0.74**	**0.71**	**0.72**
ResNet-34	0.63	0.70	0.67	0.66
ResNet-50	0.61	0.73	0.69	0.66
ResNet-101	0.59	0.73	0.70	0.65
ResNet-152	0.63	0.71	0.67	0.67

ResNet-34 model, which has a result of 0.94. On the IoU threshold of 0.05, the ResNet-18 model gives the best results on all criteria. More shallow backbones outperforming deep backbones, is most likely the case due to the fact that this is a dataset with only two classes. Deeper backbones fit better with more complex datasets. (Like the COCO dataset which has 80 different classes.)

4.4 Experiment 4: RetinaNet with five classes

For this experiment, every ResNet model has been trained on the same dataset as the one used in experiment 2. Like experiment 3, a maximum of 15 epochs was used and the models have been pretrained. See table five for the results.

It seems like that especially the more shallow models seem to perform well on this dataset. Though the ResNet101 model with a threshold of 0.5 seems to have the biggest F1 score, due to an increase in precision. However, if a model with the highest recall is more desirable, the ResNet34 model, with a threshold of 0.05 will be the model of choice.

5 Discussion, Conclusion & Future Work

5.1 Discussion

The results of the performed experiments suggest that it is indeed possible to detect aphids in flight. The precision of these model lies around 90 % or higher depending on the model. Because two separate experiments were conducted per object detector, it also became clear that there are some drawbacks when the object detector needs to differentiate between aphids with wings and those without.

Table 5. Results for RetinaNet with different ResNet backbones, on the dataset with 5 classes, for two different IoU thresholds

Model	Precision	Recall	MAP	F1 score
IoU threshold = 0.5				
ResNet-18	**0.78**	0.39	0.38	0.52
ResNet-34	0.70	**0.43**	**0.39**	0.53
ResNet-50	0.58	0.37	0.34	0.45
ResNet-101	0.74	0.42	0.38	**0.53**
ResNet-152	0.57	0.35	0.30	0.44
IoU threshold = 0.05				
ResNet-18	**0.33**	0.49	0.43	0.40
ResNet-34	0.31	**0.57**	**0.46**	**0.40**
ResNet-50	0.25	0.54	0.39	0.34
ResNet-101	0.29	0.56	0.45	0.39
ResNet-152	0.31	0.48	0.36	0.38

The models that came out of these experiments can tell winged aphids apart from their younger counterparts; however, this comes with a cost in their recall. When the two confusion matrices in figure three and figure four are compared the big difference lies in the background class. For the models that contain five classes the number of insects that are not detected is higher than the models with two classes. This results in a lower recall for the models with five classes.

There is a complexity with the dataset with 5 classes, though. A difference is made between the classes 0 and 1, those being aphid_dropped and aphid_winged. The actual difference between these classes is small: both are winged aphids, but aphid_dropped are winged aphids that refuse to fly. In contrast to aphid_winged which are winged aphids which are flying. In practice, aphid_dropped is rarely encountered and due to their similarity to aphid_winged, the model confuses the two a lot. If these classes are counted as one class, the 5 classes dataset (then the 4 classes dataset) will surely give better results, closer to the 2-classes dataset.

This does show that with a relatively small dataset, the object detectors do a very good job of detecting aphids. The object detector that works the best in this case is YOLOv8. In both experiments, it is clear to see that YOLOv8 outperforms RetinaNet, and it also produces a model much faster.

Although these experiments produce good results with a relatively small dataset, it remains one of the shortcomings of this research. The main issue in our dataset lies in the differentiation between the classes 'aphid_dropped' and 'aphid_winged.' Among the total of 212 mature aphids that have grown wings, 65 of them are incorrectly labeled as 'aphid_dropped.' These aphids are both mature and have sprouted wings, but the dropped aphids do not display their wings.

5.2 Conclusion

This research has conducted various experiments on two different object detectors. Interpreting the results of all four experiments leads to the result that YOLOv8 is the better object detector for detecting aphids. It achieves the highest scores on all metrics, with an F1 score increase of 0.15 on the 2-classes dataset and an increase of 0.3 on the 5-classes dataset, and also requires considerably less time for training.

Furthermore, the experiments provide insights into the differences in classes. When comparing the metrics of the models with two classes to the model with five classes, it is clear to see that the model with 2 classes has much better recall. It detects more aphids, which leads to more reliable results when these models are used in the field.

5.3 Future Work

All used datasets have been acquired within a lab environment. The next step in this research requires a dataset taken of aphids within their natural habitat. This may prove a challenge, because getting pictures of aphids outside is more time consuming than in a lab. Not only that, this dataset can be only acquired within a very small time frame per year. An obvious solution is to use a training and validation dataset of lab photo's, while testing is done on natural photo's. The biggest difference of this dataset is the background colour. More research to a solution for problems this may cause might be required. An example is: more augmentations, that manipulate the color of the image, or an extra step in the detector that converts each image to gray-scale. This way, a dataset usable for training on natural data can be acquired more easily.

ACKNOWLEDGEMENTS

This project is financially supported by ELFPO and performed within the POP3+ Fryslân project Innovatie luizendetectie.

provinsje fryslân
provincie fryslân

NHL STENDEN
university of applied sciences

Europees Landbouwfonds voor Plattelandsontwikkeling: Europa investeert in zijn platteland

GEORGE PARS GRAANHANDEL B.V.

GeJo Grading Services B.V.

Fokkema Farming

Elyt. CONSULTANCY

REFERENCES

[1] URL: https://cocodataset.org/#home.

[2] Xie Chengjun et al. *Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning*. Research paper.

[3] Xia Denan et al. *Insect Detection and Classification Based on an Improved Convolutional Neural Network*. URL: https://www.mdpi.com/1424-8220/18/12/4169.

[4] Ayush Chaurasia Glenn Jocher. URL: https://docs.ultralytics.com.

[5] ing. dr. ing. and Verbeek Martin. *Virus- en vectorbeheersing in pootaardappelen*. 2019. URL: https://www.wur.nl/nl/onderzoek-resultaten/onderzoeksprojecten-lnv/soorten-onderzoek/kennisonline/virus-en-vectorbeheersing-in-pootaardappelen.html.

[6] Glenn Jocher. URL: hhttps://github.com/ultralytics/ultralytics/issues/1114.

[7] Redmon Joseph et al. "You Only Look Once: Unified, Real-Time Object Detection". In: (2016).

[8] Tan Lu et al. *Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identifcation*. URL: https://link.springer.com/article/10.1186/s12911-021-01691-8.

[9] Evans-Goldner Lynn. *potato virus y strains*. 2020. URL: https://www.aphis.usda.gov/aphis/ourfocus/planthealth/plant-pest-and-disease-programs/pests-and-diseases/nematode/potato/pvy.

[10] Shuyi Ma et al. "Deep Learning Based Detector YOLOv5 for Identifying Insect Pests". In: *Applied Sciences* (2021).

[11] *Projectplan POP3 profincie fryslân maatregel*.

[12] Klut Sander. *Detection of Aphids on Sticky Plates using YOLOv5 with Image Tiling*. Research paper.

[13] Jacob Solawetz. *What is YOLOv8? The Ultimate Guide.* URL: https://blog.roboflow.com/whats-new-in-yolov8/.

[14] Kasinathan Thenmozhi, Singaraju Dakshayani, and Uyyala Srinivasulu Reddy. *Insect classification and detection in field crops using modern machine learning techniques*. Research paper.

[15] Lin Tsung-Yi et al. *Focal Loss for Dense Object Detection*. URL: https://arxiv.org/pdf/1708.02002v2.pdf.