Detection of Aphids on Sticky Plates using YOLOv5 with Image Tiling

NHL Stenden Lectoraat in Computer Vision & Data Science

Sander Klut

Supervisors: Henry Maathuis, Maya Aghaei Gavari

Abstract—The health of seed potato crops is being threatened by viruses like the Potato Virus Y (PVY). In the last five years, more than 15% of harvested seed potatoes and sugar beans were infected with the Potato Virus Y [1]. There is an empirical relationship between the number of aphids and the number of infected plants [1]. Early detection of aphids creates the opportunity to instantly use pesticides. Currently, manually counting aphids is very time-consuming and the use of object detection models, to detect aphids and accordingly decide the necessity of applying pesticides, could offer a solution. This study considers a single-stage object detection model for the real-time detection of aphids on yellow sticky plates with YOLOv5. A dataset has been acquired with images of yellow sticky plates with aphids and other insects. High-resolution images are being utilized to capture the small size of aphids. To minimize computing power, a tiling technique is being explored. This involves dividing the images into smaller sections called tiles. The network is competent in selecting tiles that always contain either an aphid or an insect, named positive tiling. The YOLOv5l-model with positive tiling is the best-performing model with an F1-score of 0.525. Possible extensions of this study are being discussed, together with suggestions for future research.

Index Terms—Deep Learning, YoloV5, Insect Detection, Aphids, Agriculture, Yello Sticky Plates, Image Tiling

1 Introduction

The health of seed potato crops is being threatened by viruses [2]. Viruses like the Potato Virus Y (PVY) cause economic harm and jeopardize the strong export position of Dutch seed potatoes and sugar beans.

To maintain a strong export position, the health and quality of the product are paramount. In the last five years, more than 15 percent of harvested seed potatoes and sugar beans were infected with the PVY virus [1]. This diminishes the value of the product and thus results in less profit. It is believed that viruses such as PVY are spread by insects, with aphids in particular. This assumption is supported by the direct relationship between the number of aphids and the percentage of infected crops in the field [1].

Early detection of aphids creates the opportunity to use pesticides locally when needed, resulting in better protection of the crops and less use of pesticides. Here the speed of the detection process is crucial, since the control of the aphids needs to happen before further spreading of deceases.

Currently, the process of detecting aphids is time-consuming and not as efficient as it could be. With yellow sticky plates, insects are attracted and caught, after which the aphids are manually identified and counted. The results of these counts determine the number of pesticides used for the entire field, while aphids could be only present in certain regions. Here the use of object detection models, to detect aphids and accordingly decide the necessity of applying pesticides, could be a solution.

In object detection, there are two main methods: single-stage and two-stage. Single-stage performs detection and classification simultaneously, while two-stage has separate models for detection and classification.

- *Sander Klut is a Mechanical Engineering student, doing a minor in Computer Vision and Data Science at the NHL Stenden University of Applied Sciences, E-mail: Sander.klut@student.nhlstenden.com.*
- *Henry Maathuis is a researcher at the NHL Stenden Lectoraat in Computer Vision & Data Science, Email: henry.maathuis@nhlstenden.com.*
- *Maya Aghaei Gavari is a researcher at the NHL Stenden Lectoraat in Computer Vision & Data Science, E-mail: maya.aghaei.gavari@nhlstenden.com.*

This study uses the single-stage method. Single-stage object detection models have faster processing time from fewer operations, a simpler pipeline, and reduced memory and computational needs. In addition, they can have potentially higher accuracy for certain tasks due to end-to-end optimization.

The aim of this study is to design a single-stage object detection model able to localize and identify insects and aphids on yellow sticky plates.
Hence, the main research question is the following:

How can a single-stage object detection model be used to robustly find aphids on yellow sticky plates?

A dataset[1] has been collected containing images of aphids and insects. The images are high-resolution and require substantial computing power to process. To address this challenge, image tiling is applied to crop, and process smaller segments of the images. However, the dataset also has an unequal distribution of aphids and insects, which will be examined for its impact.
Therefore, the sub-questions are the following:

- *Does image tiling benefit the detection of tiny insects?*
- *Would an imbalanced dataset influence the detection results of tiny insects per class?*

[1]The dataset was collected by the Computer Vision and Data Science professorship.

2 STATE OF THE ART

The following paragraphs describe recent advancements in object detection techniques. 2.1 focuses on the single-stage object detection model YOLOv5, which has been shown to provide good performance in real-time object detection. 2.2 explains the use of RCNN-models for real-time monitoring of insect populations in greenhouses. 2.3 highlights the use of image tiling to balance accuracy and computational efficiency in small object detection, specifically for identifying pedestrians and vehicles onboard a micro aerial vehicle (MAV). Image tiling can result in faster performance, despite requiring extra processing compared to using raw images.

2.1 Object detection with YOLOv5

In [3], the study on the single-stage object detection model, YOLO is presented. Previous models were dependent on a two-stage approach in which, first, regions of interest are found after which a classifier processes these regions. You Only Look Once(YOLO) is a single model able to localize as well as classify objects.

YOLOv5 (You Only Look Once version 5) is an object detection model for real-time object detection released in 2020. The study is discussed in [4]. It's part of the YOLO (You Only Look Once) family of object detection models, which are known for their speed and accuracy. YOLOv5 uses a single neural network to detect objects in an image, allowing it to process frames in real-time. It has been developed and improved upon by the GitHub community, and has been designed for both efficiency and accuracy, making it a popular choice for real-time object detection tasks [5] [6] [7].

2.2 Field detection of tiny insects

In [8], a network using RCNN-models was constructed to detect thrips and whiteflies on yellow sticky plates in a greenhouse. The network was designed for real-time monitoring of the insect population in the greenhouse. R-CNN (Regions with Convolutional Neural Networks) is a computer vision algorithm for object detection and image classification. It uses a two-stage process. First, it generates candidate object regions using selective search, the so-called Region of Interest proposals. Then it uses a Convolutional Neural Network (CNN) to classify the regions and refine the bounding box locations.

RoI (Region of Interest) proposals refer to a technique used in computer vision and deep learning to identify and extract regions of an image that are most relevant or interesting to the task at hand. RoI proposals are generated using various algorithms, such as a sliding window. The goal is to reduce the computational cost and increase the accuracy of these models by focusing on regions that are likely to contain objects, rather than processing the entire image.

The best-performing model of the study was the Tpest-RCNN-model, consisting of pest feature learning with VGG16, Region of Interest proposals, and classification.

The TPest-RCNN model outperformed Faster R-CNN and other techniques, detecting multiple species in yellow sticky trap images with handcrafted features like color, shape, and texture. The model achieved a mean F1 score of 0.944 and average precision of 0.952.

2.3 Tiling for small objects

Researchers used images from a micro aerial vehicle to detect pedestrians and traffic in real-time, in [9]. To work on mobile GPUs, the challenge is finding a balance between accuracy and computational power.

PeleeNet is a deep neural network architecture for object detection and classification tasks. It is designed for efficient computation on mobile and embedded devices, with a focus on reducing computation and memory usage. PeleeNet is a light-weight network that uses a novel architecture to improve accuracy while still having a small number of parameters and low computational complexity.

In this study, PeleeNet in combination with image tiling is considered the most efficient for detecting pedestrians and vehicles with mobile GPUs. The suggested method restricts detail reduction in object detection by supplying the network with a constant input size. Training a network with high-resolution images using bigger feature maps leads to high computational and memory demands. The tiling approach proposed linearly increases computation time, while keeping memory constant, by processing tiles sequentially.

In trying to establish a well-performing and accurate model that trains fast, related studies have considered image tiling [8] [9] [10]. Image tiling requires extra processing power compared to raw data because it involves breaking down a large image into smaller overlapping tiles or segments, which must then be processed separately and later combined to form the final image. This additional step increases the computational complexity and requires more processing resources. However, it helps in processing large images that do not fit in memory by dividing the image into smaller tiles. In addition, this allows for efficient parallel processing and reduced memory usage, leading to improved processing speed and performance.

3 MATERIALS AND METHODS

The requirements for the study are presented in this section, together with the specific techniques that have been applied.

A dataset of yellow sticky plate images was collected from seed potato farms in Friesland, Netherlands, in 3.1. The images were taken with a SONY a7 MARK II camera in a controlled environment.

In 3.2 the images were cropped to a size of 4000x4000 pixels and annotated with bounding boxes and class names for 'aphid' and 'non-aphid'. The resulting dataset was divided into three groups for training, validation, and testing. To address misclassifications in the annotations, two additional groups were created using better-annotated images.

The images were pre-processed by selecting tiles from them to ease the processing and ensure the presence of insects in the tiles in 3.3. The object detection models are covered in 3.4, the hardware in 3.5, and how the results are presented in 3.6.

3.1 Dataset Acquisition

In preparation for this study, experiments[1] were carried out at seed potato farms across Friesland, the Netherlands. Since aphids and other insects appear to be attracted to bright yellow colors, yellow sticky plates were placed at the farms to attract these insects.

A dataset of images of yellow sticky plates has been collected, which contains both aphids and other insects. For four months, every two weeks these yellow sticky traps were collected and placed at ten Frisian farms, resulting in 80 yellow sticky plates. The collected traps were used for acquiring images.

In a controlled environment, images of the yellow sticky traps were taken with a SONY a7 MARK II SLR-camera. An image of each quarter of the yellow sticky plates was captured with the camera creating images with a resolution of 6000 x 4000 pixels, creating four individual images per yellow sticky plate. Resulting in a total of 320 images. The yellow sticky plates that have been used are 250x250 mm. Other specifications regarding the camera are shown in table 1.

Table 1: Camera Specifications

Component:	Specifications:
Camera	Sony A7II
Camera lens	Tamron F053S
Camera Resolution	6000x4000 pixels
Sensor Size	35.8x23.9 mm
Color depth/Bitrate	14-bit uncompressed Raw
Illumination	Low angle illumination

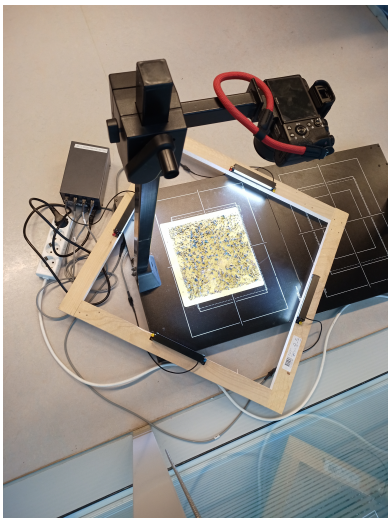


Fig. 1: Camera Setup for image acquisition

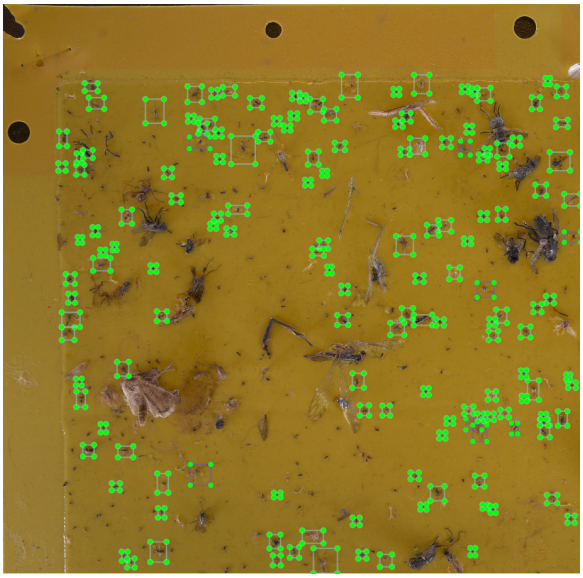


Fig. 2: Image with annotations made with LabelImg annotation software.

3.2 Ground Truth and Dataset division

3.2.1 Ground Truth

The images captured by the camera were cropped to create square images. The 6000x4000 pixel images were trimmed to a size of 4000x4000 pixels to accurately capture a quarter of the yellow sticky plates.

The data was annotated to generate a ground truth after cropping. The annotation software was utilized to outline the insects by constructing a bounding box around them [11]. Subsequently, the insects were manually classified as either 'aphid' or 'non-aphid.' The combination of bounding boxes, class names, and images constitutes the ground truth, which is used by object detection methods to train a model.

3.2.2 Train, Validation and Test Dataset

Dataset division in object detection plays a crucial role in the learning and evaluation process of the model. It helps the model to learn the features of objects effectively, avoid overfitting and evaluate its performance, and also facilitates comparison of different models.

From the images combined with their annotations, the 'Aphids and Other Insects Dataset' has been established. The dataset is divided into three parts: training, validation, and test sets. The model is trained on the training set, evaluated on the validation set, and finally tested on the test set to get a measure of its generalization performance.

For the purpose of finding the effect of the annotations on the results, another two groups are introduced in which the objects are again annotated, only now following other annotating rules. In the qualitative results of the general dataset, misclassifications in the annotations were noted, as well as annotations bigger than the objects. Insects with parts missing were not annotated. In the newly annotated data, all misclassifications have been adjusted and all insects and parts of insects were annotated.

The training and validation data remain unchanged. The grid search process in experiments has been split into two sections. The top hyperparameters from the first section were utilized in the second section, thus reducing the number of possible combinations in the grid search. The improved end-validation data was used to evaluate the performances of models through experiment 2, the first part of the grid search. Starting from experiment 3, the second part of the experiments, the improved testing data was used to prevent biased results.

The number of images per group and amount of aphids and non-aphids is shown in Table 2.

- Four tiles from the validation split are again annotated for End-Validation
- Four tiles from the test split are again annotated for the Improved Testing group.

Table 2: Dataset division

Model	Images	Non-Aphids	Aphids
Training	102	3302	402
Validation	47	1427	181
Testing	22	1200	136
End-Validation	4	369	50
Improved Testing	4	369	50

3.3 Pre-processing

3.3.1 Tiling

To improve the performance of an object detection model for detecting insects, especially aphids and non-aphids, high-resolution images are used. The images are taken from a close distance to capture more details of the insects. To reduce the computational cost, tiles, which are smaller crops from the images, are processed by the model instead of the entire images.

The tile size is set to 640x640 pixels, except for the experiment with tile sizes where a tile size of 320x320 is considered as well. Larger tile sizes would in this case allow for larger areas to be captured and would also increase computational cost and memory usage. Smaller tile sizes would reduce computational cost and memory usage but would limit the amount of detail captured, potentially leading to decreased accuracy. 640x640 was chosen as a balance between these factors.

To ensure that the model is trained to recognize both aphids and non-aphids, the coordinates of bounding boxes from the ground truth are picked and tiles around those boxes are created. This way, the tiles always contain either an aphid or a non-aphid. The possibility to balance the number of aphids versus non-aphids seen by the model during training is also introduced. The dataset has ten times more insects than aphids, causing an imbalance, so a probability of 50% is set to counteract this imbalance. The probability decides how often a tile with an aphid or a tile with a non-aphid will be created. If an

[1]The experiments were done by the professorship Computer Vision and Data Science.

image contains no aphids, the model will always create a tile with an insect. Despite the balanced tiling probability, the model still may see an imbalanced number of aphids and insects as each tile can contain multiple insects or aphids. Therefore, a probability of 100% aphid is also considered.

In testing, the images are partitioned into a fixed number of equal-sized tiles using the tiling algorithm. The object detection model is applied to each tile, and the outputs are combined to generate the final detection result for the entire image. This tiling approach is used to reduce computational costs in processing large images.

In the process of image tiling, margins are added to each tile to ensure that objects near the borders of the tile are not cut off. These margins, also known as padding or buffer zones, are added to ensure that the model can detect objects that are partially visible in the tile. The size of the margins can have a significant impact on the performance of the object detection model, as a larger margin size may increase the chances of detecting objects but also increase the processing time.

The size of the margins is a trade-off between accuracy and processing time. A margin size of the tile size divided by four is considered, based on the size of the objects and the receptive field of the model.

3.3.2 Augmentations

Data augmentation is used to increase the size and diversity of the training dataset. This can improve the detection ability of a machine learning model by reducing overfitting to the training data. Overfitting occurs when a model learns patterns that are specific to the training data, but the model does not predict well on new, unseen data.

Data augmentation techniques, such as flipping, rotating, cropping, and adding noise to images, are considered to expose the model to a wider range of variations in the input data, which can help it learn more robust and general representations of the input. This can lead to better performance on the validation and test sets, which consist of unseen data.

This study considers horizontal and vertical flipping, next to a color jitter, and applying noise multipliers.

Color jittering is a data augmentation technique used to increase the diversity of the training data by adding random variations to the color channels of the images. Color jitter is implemented by randomly altering the brightness, contrast, saturation, and hue of an image. The amount of color jitter can be controlled by specifying the magnitude of the random variations. Table 3 shows the hyperparameters of the color jitter.

Noise multipliers are used to adjust the amount of random noise added to a model during training. The purpose of adding noise is to make the model more robust to unseen data by encouraging it to learn more generalized representations of the input data. The noise multiplier determines the scale of the added noise, with a larger multiplier leading to more noise and a smaller multiplier leading to less noise. Table 4 shows those of the multiplicative noise.

Table 3: Hyperparameters for color jitter

Color Jitter:	Brightness	Contrast	Saturation	Hue
Value:	0.02	0.02	0.02	0.02

Table 4: Hyperparameters for multiplicative noise: min_mult = minimum multiplier, max_Mult = maximum multiplier.

Multiplicative Noise	min_Mult	max_Mult	Random sample
Value:	0.98	1.02	True

3.3.3 Normalising Images

The images are normalized to values between 0 and 1. For 16-bit images, this is done by subtracting mean = 0 and dividing by standard deviation, $2^{16} - 1$, as shown in equation 1.

$$normalized\_image = \frac{((x) - mean(x))}{(standard\_deviasion(x))} \quad (1)$$

3.4 Object Detection Model

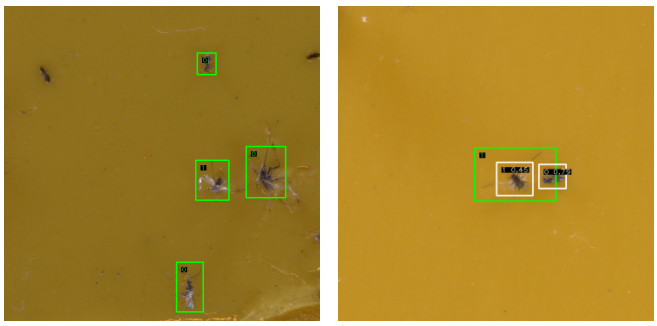
3.4.1 YOLOv5

YOLOv5 (You Only Look Once version 5) is an object detection model for real-time object detection. It's part of the YOLO (You Only Look Once) family of object detection models, which are known for their speed and accuracy. YOLOv5 uses a single neural network to detect objects in an image, allowing it to process frames in real-time. It has been developed and improved upon by the GitHub community, and has been designed for both efficiency and accuracy, making it a popular choice for real-time object detection tasks.

Several YOLOv5 models are available. In this study, the models YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x are considered. The YOLOv5 models are pretrained on the Common Objects in Context [12] dataset.

3.4.2 Training and Validation

Random tiles are created during training and validation. The tile size is one of the hyperparameters, standard set at 640x640 pixels. Figure 3a shows a tile created in training. Figure 3b shows a tile created in validation with the made predictions in white. Every two epochs of training, the model will validate whether learning has improved. The other hyperparameters are the number of tiles per image, batch size, learning rate, and the patience of the scheduler. For the baseline, the following values are used: number of tiles of 2 tiles per image, a batch size of 2, a learning rate of 0.001, and no patience.



(a) **Cropped tile with annotations in green during training.** (b) **Cropped tile with ground truth in green + predictions in white.**

Fig. 3: Cutout tiles in training and validation

3.4.3 Testing

Images are divided into fixed tiles and are run through testing. The tile size used for testing is 640x640 pixels. The tile size should always be the same in training and testing. The image with Bounding Boxes of the Ground Truth and the Predictions is reconstructed and saved. Figure 4 shows a testing image with ground truth in green and predictions in white. The yellow lines show the borders of the individually analyzed tiles.

3.5 Hardware

The machine used to carry out experiments consists of an Intel® Core(TM) i9-7960X CPU @ 2.80GHz (8 cores) CPU and an NVIDIA® GeForce RTX 2070(8.000 Gigabyte Memory) GPU.

3.6 Evaluation Metrics

The performance of the models is evaluated with intersection over union (IoU). With IoU, the ground truth bounding box is compared to the predicted bounding box. A threshold of 0.5 is used, meaning an intersection of 50 percent between the ground truth- and the predicted

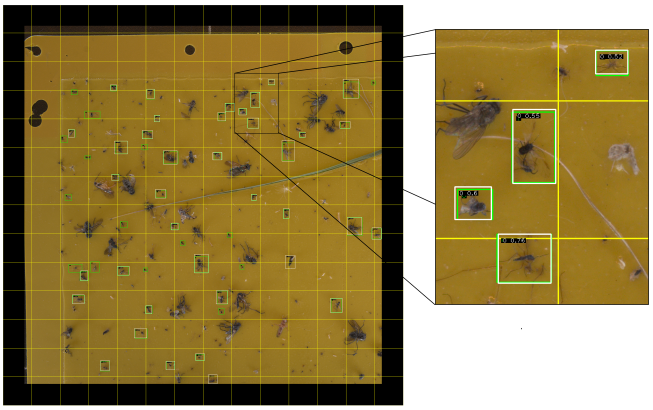


Fig. 4: Object detection and classification during Testing. Green = Ground truth, White = Predictions

bounding box is needed to label the prediction as a correct one.

To compare the performance of the models, the precision, recall, and F1-score, on the aphids and non-aphids, are considered. As well as the precision, recall, and F1-score on aphids only. A confusion matrix has been made for each class to identify where the models are struggling the most. With the data results of the performance, choices can be made between models.

During the experiments, the decision has been made that above all the precision, recall, and F1-score regarding only aphids should be considered. The results regarding only aphids are considered to be more useful since they will show the feasibility of detecting aphids with the models. The models are compared using F1-score on aphids, showing the overall strength of the models in this class.

Precision evaluates the accuracy of the model's predictions by calculating the ratio of true positive detections (TP) to the total number of predictions made, as shown in equation 2. True positives in object detection refer to instances when the model correctly detects the presence of an object and classifies it as the correct class. False positives, on the other hand, refer to cases where the model detects an object but misclassifies it or labels it incorrectly.

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

In equation 3 the equation for recall is shown. Recall is the score for the ratio between the number of correct predictions and the total number of the ground truth. TP being the True Positives, FN being False Negatives.

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

The F1-score arises from the average between the precision and the recall. This is shown in equation 4

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

The confusion matrices show the four values for each model for True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). For each class, there is a matrix. Table 5 shows the matrix for aphids. Table 6 shows the matrix for other insects.

4 Experiments and Results

Experiments have been conducted using different hyper-parameters and experimental settings, of the end-to-end tiled YOLOv5 tiled object detection model, on the 'Aphids and Other Insects Dataset'.

Table 5: Confusion Matrix: Aphids

	True Aphid	**No Aphid**
Predicted Aphid	TP	FP
Predicted No Aphid	FN	TN

Table 6: Confusion Matrix: Insects

	True Insect	**No Insect**
Predicted Insect	TP	FP
Predicted No Insect	FN	TN

The first experiment compares the different YOLO-models. YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x are compared. In the second experiment, different parameters for training and validation are considered, being the batch size, learning rate, and patience. In the third experiment, the effect of batch size, tile size, tiling, and use of augmentations is shown. The last experiment considers using different probabilities for positive tiling. The number of tiles per image of 2 is used for all experiments.

4.1 Experiment 1: Different YOLO-models

Four different YOLOv5-models, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, are compared. All hyper-parameters are equal for the four models, having a batch size of 2, the number of tiles per image is 2, and a learning rate of 0.0001. The experiments have been running for 100 epochs. No scheduler was used in this experiment. A summary of the experiment is shown in Table 7. The experiment is utilized in the conclusion to demonstrate a distinction between the models.

Table 7: Experiment 1: considering different YOLO-models, Batch = Batch size, Learning rt = Learning rate

	Model	Batch	No. Tiles	Learning rt.	Epochs
YOLOv5	s/m/l/x	2	2	0.0001	100

4.1.1 Results Experiment 1

Table 8 shows the results of experiment 1. From the F1-score in the table can be concluded that larger YOLOv5-models are outperforming the smaller models on the 'Aphids and Other Insects Dataset'. This is as expected since the larger YOLOv5-models are in fact larger algorithms requiring also more computing power.

Although this would suggest the use of the largest model, the size of the model is, due to available computational power, also inversely proportional to the batch size. This is further discussed in experiment 2. All models will still be considered in experiments 2 and 3.

Table 8: Results Table 1: Aphids and Non-aphids

Model	Precision	Recall	F1-Score
YOLOv5s	0.76	0.67	0.71
YOLOv5m	0.75	0.68	0.72
YOLOv5l	0.78	0.67	0.72
YOLOv5x	0.76	0.70	0.73

4.2 Experiment 2: Basic hyperparameters

Since the different YOLOv5-models are expected to deliver different results, depending on different parameters, all YOLO-models have been compared in combination with batch size, learning rate, and

patience. THe hyperparameter of patience introduces a scheduler that reduces the learning rate when the model is not improving for a given number of epochs, with patience being the number of epochs. The batch size in combination with tile size is limited by the amount of memory of the GPU. In this study, the maximum of (batch size * the number of tiles) for each model is:

- YOLOv5s-model<= 16
- YOLOv5m-model<= 8
- YOLOv5l-model<= 4
- YOLOv5x-model<= 4

For all experiments, a number of tiles of 2 per image has been used.

The best batch size, learning rate, and patience will be determined by the experiment. The learning rate and patience will be used in experiment 3. All batch sizes are still considered in experiment 3, due to find if there is a relation between batch size and tile size.

YOLO Model	Batch Size	Learn Rate	Patience
s/m/l/x	2/4/8	0.01/0.001/0.0001	0/20/40

4.2.1 Results Experiment 2

The best results of experiment 2 are shown in table 9. All best results have a learning rate of 0.001 and a patience of 20 epochs. In table 9 can be seen that larger batch sizes result in higher F1-scores.

Table 9: Results experiment 3. Pre_A = Precision Aphids, Rec_A = Recall Aphids, F1_A = F1-score Aphids.

Model	Batch	Epochs	Pre_A	Rec_A	F1_A
YOLOv5s	2	400	0.382	0.559	0.454
	4	400	0.449	0.419	0.433
	8	400	0.430	0.418	0.424
YOLOv5m	4	400	0.431	0.520	0.471
YOLOv5l	2	400	0.455	0.549	0.485

4.3 Experiment 3: Tile size, Tiling and Augmentations

The best YOLO-models, learning rate, and patience have been used as the baseline for experiment 3. This means that the batch size is still to be chosen. 24 experiments have been carried out differing in batch size, tile size, tiling, and augmentations.

Table 10: Experiment 3

Batch size	Tile size	Tiling	Augmentation
2/4/8	320/640	Tiling/Positive Tiling	No/Yes

4.3.1 Results experiment 3

The results are shown in table 11. YOLO-models YOLOv5m and YOLOv5l are used in combination with a learning rate of 0.001 and patience of 20 epochs.

As can be concluded from table 11, tiles with a size of 640x640 pixels seem to be preferable above tiles with a size of 320x320 pixels. The YOLOv5l-model with a batch size of 2 seems to outperform all other experiments with an F1-score on aphids of 0.525. Furthermore, no advantages have been found in the use of augmentations.

Table 11: Results experiment 3. Pre_A = Precision Aphids, Rec_A = Recall Aphids, F1_A = F1-score Aphids.

Model	Batch	Tile	Aug	Prec_A	Rec_A	F1_A
m	2	640	No	0.488	0.463	0.475
		640	Yes	0.576	0.390	0.465
	4	640	No	0.492	0.456	0.473
		640	Yes	0.389	0.478	0.429
		320	No	0.416	0.507	0.457
l	2	640	No	0.473	0.588	0.525
	2	640	Yes	0.593	0.397	0.476

4.4 Experiment 4: Probability

Experiment 4 considers the change in tiling probability. Random Positive Tiling involves randomly cropping and scaling the positive examples from the training images. Tiling probability is one of the hyperparameters of the tiling algorithm. The tiling probability decides how often a tile is created around an aphid and how often around a non-aphid. A 50% probability for each is compared with a 100% probability of creating a tile with an aphid. The 100% probability counters the imbalance between the number of aphids and non-aphids seen by the network.

Table 12: Experiment 4

YOLO-model	Batch size	Probability
s/m/l/x	2/4/8	50/50 , 100/0

4.4.1 Results experiment 4

While it was expected that removing the imbalance by changing the probability would result in better performances of the network, table 13 shows that there is no difference between the results of the different probabilities. With a probability of 50% over 400 epochs, the F1-score is 0.424, while the probability of 100% over 400 epochs results in an F1-score of 0.421. However, during these experiments, another interesting point came forward. When looking at the F1-scores on aphids, the probability of 100% aphids, seems to train much faster regarding the detection of aphids.

Table 13: Results experiment 4. Pre_A = Precision Aphids, Rec_A = Recall Aphids, F1_A = F1-score Aphids.

Model	Epochs	Probability	Prec_A	Rec_A	F1_A
YOLOv5s	100	50/50	0.356	0.451	0.398
	100	100/0	0.358	0.495	0.415
	400	50/50	0.430	0.418	0.424
	400	100/0	0.333	0.574	0.421

5 Discussion, Conclusion and Future Work

This section discusses the main findings of the study and states the conclusions that have been drawn from these findings. In addition, several options for future research are given.

5.1 Discussion

The study suggests that the use of larger YOLO-models results in better performances on the 'Aphid Dataset'. Overall, a larger batch size results in better performances and faster training of the network, as shown in table 8 and 9. Where the YOLOv5s-model achieved an F1-score of 0.71 while the YOLOv5x-model achieved an F1-score of 0.73. The batch size in combination with the number of tiles per image is limited by the amount of memory of the GPU.

It is believed that even higher batch sizes in combination with the YOLOv5x-model would result in even better performances. To

accomplish these results, more computing power is needed, as well as a larger dataset. From table 11 can be concluded that larger batch sizes result in faster training. This in combination with the occurrence of better performances on non-aphids with larger batch sizes at the end of the training, results in the assumption that the network needs more aphids to train on.

The study has also shown that positive tiling, where only tiles are selected containing either an aphid or a non-aphid, outperforms regular tiling.

The use of augmentations has not proven to benefit the results on the 'Aphid dataset', as can be concluded from table 11. Further experiments must show if there are other augmentations than the ones discussed in the paper, that do benefit the results. More test results on this part would especially be useful since augmentations can counter the disadvantages of a small dataset.

Concerning the main hyperparameters, a learning rate of 0.001 and a scheduler with patience of 20 epochs have been shown to give the best results in this setting (table 9). A tile size of 640x640 pixels is preferred over a tile size of 320x320 pixels (table 11).

The experiments done regarding the class probabilities in positive tiling, called tiling probability, have shown that a balance between the number of objects per class results in faster training (table 13). At 100 epochs, the 100/0% tiling probability model performs with an F1-score of 0.415, being superior to the 50/50% tiling probability model with an F1-score of 0.398. The F1-scores are equal at 400 epochs. Because in the remaining part of the tile, around the chosen aphid or non-aphid, there could be more aphids or non-aphids, it is difficult to perfectly balance the number of objects seen per class.

A solution could be considered which would perfectly balance the classes. Taken into account has to be that the process of tiling would still have to be random. A more obvious solution could be to create a model with only one class for aphids. A single-class model is a suitable choice when the objective is to detect just one type of object, as it simplifies the model and lessens the need for large amounts of training data.

The model seems to do some false predictions on parts of insects that are stuck to the yellow sticky plates. Furthermore, regarding the 'Aphids and Other Insects Dataset' there is still room for improvement. Attempts have been done in improving the annotations for the testing data, resulting in the End-Validation split and the Improved Testing split. Due to time restrictions, no further improvements to the dataset have been done. Especially in the data for training, aphids are annotated as non-aphids and the other way around.

At a certain point in training and validation, the network has seen most of the aphids in the dataset at least once. A possible explanation for the results could therefore be that the network is beginning to overtrain. Overtraining is the principle where the network is trained too specifically on the data at hand during training. Because of this, the network would perform worse on data it has never seen before.

5.2 Conclusion

This section answers the research questions. The conclusion states the main highlights of the results and relates them to the goals of the study.

This study proposes a single-stage object detection model as a solution to find aphids on yellow sticky plates. Primary, the sub-questions are answered, after which the answer to the main research question is stated.

5.2.1 Does image tiling benefit the detection of tiny insects?

For the detection of tiny objects images with a high resolution have been used. Processing high-resolution images takes a lot of computing power. Hardware limitations make it impossible to process full images. For this reason, the model crops tiles from the images which are then processed. The use of high-resolution images with image tiling has given promising results. More complex algorithms seem to outperform less complex algorithms.

5.2.2 Would an imbalanced dataset influence the detection results per class?

Around farms, in general, there are a lot more insects than aphids. The collected yellow sticky plates have approximately ten times as many insects on them as aphids. This creates a natural imbalance in the dataset.

The probability of the tiling algorithm creating a tile with either an aphid or a non-aphid affects the ratio between classes. An imbalanced dataset is now autonomously balanced towards a desired ratio. A higher probability of a specific class has not been shown to increase results for this class. The higher probability of a class does result in faster training for the specific class.

5.2.3 How can a single-stage object detection and qualification model be used to robustly find aphids on yellow sticky plates?

An end-to-end method to find aphids on yellow sticky plates has been established for use by farmers. A YOLOv5 model in combination with image tiling has been shown to deliver optimistic results but not yet on a level that can be put to use in the field. However, the feasibility of detecting aphids with the proposed methods has been shown.

The overall best result achieved in the study is an F1-score on aphids of 0.525. The model has a precision on aphids of 0.473, and a recall on aphids of 0.588. The indication of the number of aphids in a field of crops can help farmers in their decision-making around the use of pesticides.

5.3 Future Work

This segment discusses the possibilities for future research after this study. The next steps are discussed in direct relation to this study. After which, potential future studies are considered.

5.3.1 Improve Dataset

For future work, improving the data going into the model should be considered. A bigger training dataset is desirable. The dataset could be extended with more images or configurations for augmentations could be examined to give the idea of a larger dataset.

The annotations of the 'Aphids and Other Insects Dataset' should be considered enhancing. Instead of doing this manually, the network could also return images of objects the network is not certain about.

Due to time restrictions, no further improvements to the dataset have been done. Especially in the data for training, there is room for improvement. Some aphids are now still annotated as non-aphids and the other way around.

5.3.2 Potential future studies

Future research could be done on single-stage object detection. Something that can be attempted is using a model with only a class for aphids. When only one type of object needs to be detected, using a single-class model could reduce the complexity of the model and reduces the amount of training data required.

In this study, different kinds of aphids are all considered to fit into one class. Some kinds of aphids differ from each other in color, shape, and size, potential research could be done in categorizing the aphids per kind.

Parallel to this study, a system is being developed, that attempts to identify and analyze the aphid population with an in-flight setup at a seed potato farm. A test station has to be designed that is able to collect necessary data, such as images or videos, and send it to a remote location. Here, the software designed specifically to regulate the use of pesticides per region can access and process data.

ACKNOWLEDGEMENTS

The study is covered in the consortium Innovation in Aphid Detection, a partnership between GeJo Grading, Fokkema Farming, George Pars Graanhandel B.V., and the professorship Computer Vision and Data Science of NHL Stenden. The author wishes to thank the consortium for their contributions to the study and expertise. As well as, the professorship of Computer Vision and Data Science of NHL Stenden University, for all the resources used for the study.

REFERENCES

[1] Paula Westerman. Virus- en vectorbeheersing in pootaardappelen, 01 2022.

[2] PROJECTPLAN POP3+. Provincie fryslân maatregel 7: Samenwerking voor innovaties 2021, 2021.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[4] Glenn Jocher. ultralytics/yolov5, 08 2020.

[5] Xiaohong Han, Jun Chang, and Kaiyuan Wang. Real-time object detection based on yolo-v2 for tiny vehicle object. *Procedia Computer Science*, 183:61–72, 2021.

[6] Yonghui Lu, Langwen Zhang, and Wei Xie. Yolo-compact: An efficient yolo network for single category real-time object detection, 08 2020.

[7] Achyut Morbekar, Ashi Parihar, and Rashmi Jadhav. Crop disease detection using yolo. *2020 International Conference for Emerging Technology (INCET)*, 06 2020.

[8] Wenyong Li, Dujin Wang, Ming Li, Yulin Gao, Jianwei Wu, and Xinting Yang. Field detection of tiny pests from sticky trap images using deep learning in agricultural greenhouse. *Computers and Electronics in Agriculture*, 183:106048, 04 2021.

[9] F. Ozge Unel, Burak O. Ozkalayci, and Cevahir Cigla. The power of tiling for small object detection, 2019.

[10] Delia M. Pinto-Zevallos and Irene Vänninen. Yellow sticky traps for decision-making in whitefly management: What has been achieved? *Crop Protection*, 47:74–84, 05 2013.

[11] Darren Tzutalin. Labelimg, 08 2022.

[12] Coco - common objects in context.