Faster R-CNN and EfficientNet for Accurate Insect Identification in a Relabeled Yellow Sticky Traps Dataset

Citation for published version (APA):

Document status and date:
Published: 01/01/2021

DOI:
10.1109/metroagrifor52389.2021.9628708

Document Version:
Publisher's PDF, also known as Version of record

Document license:
Taverne

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.umlib.nl/taverne-license

Take down policy
If you believe that this document breaches copyright please contact us at:
repository@maastrichtuniversity.nl
providing details and we will investigate your claim.

Download date: 16 Sep. 2023
Faster R-CNN and EfficientNet for Accurate Insect Identification in a Relabeled Yellow Sticky Traps Dataset

1st Maurice Deserno
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
m.deserno@student.maastrichtuniversity.nl

2nd Alexia Briassouli
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
alexia.briassouli@maastrichtuniversity.nl

Abstract—Precision farming applications such as biological pest control use yellow sticky traps to catch insects and then classify and count them. A benchmark dataset of yellow sticky traps with insects has been recently made public for developing methods to identify and count them. However, this dataset contains missing or erroneous annotations, so we have corrected them and made them available \(^1\) for reliable benchmarking. We use these corrected annotations to propose and compare approaches based on State-of-the-Art (SoA) object detection/recognition, namely Faster R-CNN and EfficientNet, with appropriate pre-processing and data augmentation. Pre-processing and augmentation is crucial for achieving accurate results with this dataset, which faces challenges caused by the small size of the objects, occlusions, similarity in appearance and other factors. Our experiments demonstrate that our architectures can lead to very accurate recognition of challenging insect classes, with accuracies from 81.27\% to 99.1\%, while future work is proposed for reducing false alarms and improving performance.

Index Terms—Insect recognition, precision farming, EfficientNet, Faster R-CNN

I. INTRODUCTION

Yellow sticky traps are an important tool to monitor pests (insects), so they can be classified and counted. Usually farmers have to identify and count the insects by hand. We demonstrate how State-of-the-Art (SoA) Deep Learning (DL) architectures can automate this procedure with high accuracy on a recent benchmarking dataset. Some annotations in it were erroneous, so we corrected them to ensure the development of reliable and accurate classification and counting approaches.

The classification of insects on sticky traps poses several challenges. Insects of the same class may vary in their appearance, while different classes can be hard to distinguish, as some only vary slightly in size, or have a small difference in appearance. Additionally, insects can stick in different orientations, get torn, or even overlap.

We develop methods based on SoA deep learning classification architectures Faster R-CNN and EfficientNet, applied to the dataset after pre-processing and data augmentation, to demonstrate that they can achieve very high accuracy. The contributions of this work are twofold: (i) We make the correctly relabeled dataset available for benchmarking comparisons. (ii) We deploy SoA object classification methods on the relabeled dataset, with appropriate preprocessing tailored to the needs of our problem, demonstrating that they can achieve very high accuracy.

This paper is structured as follows: In Sec. II related work is presented. Sec. III describes the Yellow Sticky Traps dataset used and how it is corrected. The methods used are presented in Sec. IV and experimental results in Sec. V. Sec. VI summarizes the results and discusses future work.

II. RELATED WORK

The detection and recognition of pests is central in agriculture, and can significantly benefit from automated SoA object detection and recognition methods which are quickly gaining ground in precision farming.

Faster R-CNN, R-FCN and SSD with ResNet-50 [1] and VGG-16 [2] are tested in [3] to recognize disease and pest on tomato plants like leaf mold, whiteflies and nutritional excess. Faster R-CNN with VGG-16 did not have the best total mAP, but was selected in [3] because of its accuracy on the more challenging classes. Pest detection in oilseed rape in [4] uses Faster R-CNN, R-FCN and SSD. Their dataset has 12 different annotated species in 3022 different images with a total of 5016 labeled objects. Faster R-CNN with ResNet101 reached the highest mAP with 0.6890 followed by SSD with Inception (mAP of 0.6744). This work ultimately chose to use SSD, as it achieves comparable accuracy at a lower cost, so it can be deployed on a mobile device. YOLO for object detection combined with SVM for classification are tested in [5] using a dataset including six classes of flying insects: bee, fly, mosquito, moth, chafer and fruit fly. They reached an average counting accuracy of 92.50\%, average classifying accuracy of 90.18\% and classification recall of 92.52\%.

Commercially, BASF Digital Farming GmbH provides the “SCOUTING” [6] mobile application, for single and multiweed identification, disease identification and yellow trap anal-

\(^1\)https://github.com/md-121/yellow-sticky-traps-dataset

Authorized licensed use limited to: Cisco. Downloaded on April 30,2022 at 04:44:46 UTC from IEEE Xplore. Restrictions apply.
ysis. Using the yellow trap analysis function, three different classes of insects can be classified: weevils, pollen beetle and the cabbage stem flea beetle.

Recently, [7] introduced a dataset of images of yellow sticky traps with three different labeled insect classes. Faster R-CNN with Inception-ResNet-v2 [8] applied to it achieved a weighted averaged accuracy of 57.4%. It should be noted that this dataset contains some annotation errors, such as misplaced bounding boxes, or missing labels (see Sec. III).

In this work, we examine the same yellow stick traps dataset of [7], but we first correct the erroneous annotations (Sec III). We then deploy a method based on Faster R-CNN with Inception-ResNet-v2, as in [8], on the relabeled dataset, for detection and classification of insects. Faster R-CNN is chosen, as it is a SoA classification method, which is known to achieve high accuracy. However, it is computationally demanding, so more efficient architectures are also examined. In 2016 SqueezeNet [9] was introduced to address this issue, and in 2017 MobileNet [10], both of which achieved high accuracy at a lower computational cost. In 2020 EfficientNet [11] was introduced, which outperformed both SqueezeNet and MobileNet, and is currently one of the top performing computationally effective models. For this reason, we also deploy EfficientNet on the dataset, after its annotations have been corrected. Our experiments show that using these approaches with appropriate pre-processing and data augmentation, we can reach excellent classification accuracy, without high computational cost. To our knowledge, ours is the only work, besides [8], which examines the problem of pest identification in the Yellow Sticky Traps dataset of [7].

III. DATASET

The Yellow Sticky Traps dataset [7] consists of 284 images recorded in a static setup using Scoutboxes [12]. Each Scoutbox is equipped with a frame to hold the traps and a SLR camera recording images at a resolution of 5184 × 3456 pixels. The original dataset contains 7413 labeled insects of three classes (Table I, second column). There are also some labels for thrips insects as a fourth class, but only a few instances have been labeled, so this class is not used. Beside the labeled insects caught by the traps, there are also some other insects like bumblebees and flies, as well as plant parts like leaves (Fig. 2) present in the images, increasing its difficulty.

The insects included in this dataset are Macrolophus pygmaeus (MR), Nesidiocoris tenuis (NC), and Trialeurodes vaporariorum (TV) (see Fig. 2). The 3 mm to 4 mm large Macrolophus pygmaeus is a flying insect used to fight whiteflies. Macrolophus pygmaeus is mostly green with clear wings and a dark antenna scape (base). It prefers to eat the eggs and larvae of whiteflies but can also drink plant sap if other food sources are missing. Since it also uses plants as food source if there are not enough whiteflies to eat, it is important to calculate the exact amount of Macrolophus pygmaeus needed to keep a predator/prey balance [13]. The Nesidiocoris tenuis belongs to the same family as Macrolophus pygmaeus, but is stronger in general. With its size of around 5 mm to 6 mm, it is larger than the Macrolophus pygmaeus, it needs more food and reproduces faster. Compared to the smaller Macrolophus pygmaeus, it settles more in the upper plant area and eats younger plant parts, which will often cause more damage to the plant. The Nesidiocoris tenuis is also mostly green, but has black accents on the wings, black knees and a black ring on the neck, as well as black stripes on its antennas [14]. The Trialeurodes vaporariorum, also called Whitefly, is a 2 mm small flying insect which infests e.g. tomato plants. It drinks the plant sap and damages the plant this way. The Whitefly has white, powdery wings and a light yellow body. Because of its very small size it can be hard to recognize on images [15] even visually.

**TABLE I: Objects per class in original and corrected datasets**

<table>
<thead>
<tr>
<th>Class</th>
<th>Original</th>
<th>Corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macrolophus pygmaeus (MR)</td>
<td>1312</td>
<td>1619</td>
</tr>
<tr>
<td>Nesidiocoris tenuis (NC)</td>
<td>310</td>
<td>688</td>
</tr>
<tr>
<td>Trialeurodes vaporariorum (TV)</td>
<td>3591</td>
<td>5807</td>
</tr>
</tbody>
</table>

**Fig. 1: Classes of the dataset (MR [16], NC [17], TV [18])**

**Fig. 2: Example images from the yellow sticky traps dataset.**

**Dataset annotations issues and corrections:** We observed several issues in the original labeled dataset, which we corrected. Firstly, some of the images are oriented in landscape, while others are in portrait orientation. While this alone is not a problem, the bounding boxes cannot match the objects in the images, since they are oriented in landscape. Thus, we changed the Exif rotation information of the affected images from portrait to landscape, which fixed the problem.

Additionally, many ground-truth bounding boxes were badly positioned on the objects, or objects were not labeled at all. This can result in objects being incorrectly rejected because of low IoU scores with the misplaced bounding boxes, or because they are unlabeled. Indeed, during our first experiments we found several images in the test set with unlabeled objects (Fig. 3) which were correctly detected by the network.

After fixing the annotations, the corrected dataset contains in total 8114 (+701) insects (Table I, last column) and has been uploaded to https://github.com/md-121/...
yellow-sticky-traps-dataset for benchmarking purposes [19]. We split the dataset into training, testing and validation sets with a ratio of $0.7/0.2/0.1$. It should also be noted that the classes in this dataset are imbalanced, as 71.57% of all objects comprise of Trialeurodes vaporariorum, while Nesidiocoris tenuis only contributes with 8.48%. Because of this, it is important to also calculate other metrics like precision and recall, which we present in this work.

IV. METHOD

This section presents the approaches deployed here for object classification, based on Faster R-CNN with Inception-ResNet-v2, and EfficientNet as explained in Section II.

**Faster R-CNN Inception-ResNet-v2** is used in [8] for the Yellow Sticky Traps dataset, as it has been proven to achieve high accuracy for small objects. Faster R-CNN is able to perform detection and classification at the same time. The networks and datasets can be switched and trained in a unified way, to fine-tune and benchmark different networks. The networks used were pre-trained on COCO 2017 [20], as it contains more than 200K labeled images and 80 object categories. They are fine-tuned on our smaller dataset by freezing the weights of the top layers, which are trained on very small details to detect and classify small objects (pests). An overview of the workflow for applying Faster R-CNN can be seen in Fig 4. The images are split into subsets following two different approaches detailed in Sec. V, and then used to train and test the Faster R-CNN.

**EfficientNet** is a promising, relatively new network family, which is more shallow than the SoA, but still achieves high accuracy by mathematical scaling of the models’ parameters [11]. The base model is EfficientNet-B0, with upscaled baseline versions ranging from EfficientNet-B1 to EfficientNet-B7 [21]. We trained different EfficientNet versions as image classifiers. Our approach consists of two parts: (a) segmenting foreground objects using a threshold based mask, (b) classifying the foreground objects using EfficientNet. The goal of our approach is to carry out object detection through foreground extraction, and focus on improving the classification step, thus speeding up the complete process and increasing overall accuracy. An overview of the workflow for applying EfficientNet can be seen in Fig 5:

V. EXPERIMENTS

Our experiments involve preprocessing to improve speed and accuracy, and splits to help balance the dataset. We provide detailed descriptions of our experimental setup, to help in benchmarking efforts. Our results are only indirectly comparable to [8], as they use the original dataset, do not provide all details of their dataset split, and process differently resized images. We provide accuracy metrics best suited for each approach (weighted average accuracy for Faster R-CNN, as in [8], and precision/recall, accuracy for EfficientNet), and analyze the experiments in detail to encourage comparisons by future works with our relabeled dataset [19].

**Faster R-CNN Inception-ResNet-v2 experiments**

**Dataset preprocessing:** Faster R-CNN Inception-ResNet-v2 uses images with a minimum of 800 and a maximum of 1333 pixels per axis. We split the images in preprocessing into $800 \times 1320$ slices and 252 pixels overlap. The overlap was selected so that each insect class fits in the overlapping area. We annotate insects which are not truncated on image slices. Two different dataset splits (Fig. 6) are used to test the results: (a) images split into $800 \times 1320$ slices, (b) random splitting of the slices in training, testing and validation subsets. The latter aims at a better distribution of classes per subset, since some images have high class imbalance. The downside of this approach is, due to the overlap, the same object can appear in two subsets, so the second dataset split was created by switching the steps of the first split, to address this.

**EfficientNet-B7** [21]. We trained different EfficientNet versions as image classifiers. Our approach consists of two parts: (a) segmenting foreground objects using a threshold based mask, (b) classifying the foreground objects using EfficientNet. The goal of our approach is to carry out object detection through foreground extraction, and focus on improving the classification step, thus speeding up the complete process and increasing overall accuracy. An overview of the workflow for applying EfficientNet can be seen in Fig 5:

![Workflow for applying Faster R-CNN](image)

**EfficientNet** is a promising, relatively new network family, which is more shallow than the SoA, but still achieves high accuracy by mathematical scaling of the models’ parameters [11]. The base model is EfficientNet-B0, with upscaled baseline versions ranging from EfficientNet-B1 to EfficientNet-B7 [21]. We trained different EfficientNet versions as image classifiers. Our approach consists of two parts: (a) segmenting foreground objects using a threshold based mask, (b) classifying the foreground objects using EfficientNet. The goal of our approach is to carry out object detection through foreground extraction, and focus on improving the classification step, thus speeding up the complete process and increasing overall accuracy. An overview of the workflow for applying EfficientNet can be seen in Fig 5:

![Workflow for applying EfficientNet](image)

**V. EXPERIMENTS**

Our experiments involve preprocessing to improve speed and accuracy, and splits to help balance the dataset. We provide detailed descriptions of our experimental setup, to help in benchmarking efforts. Our results are only indirectly comparable to [8], as they use the original dataset, do not provide all details of their dataset split, and process differently resized images. We provide accuracy metrics best suited for each approach (weighted average accuracy for Faster R-CNN, as in [8], and precision/recall, accuracy for EfficientNet), and analyze the experiments in detail to encourage comparisons by future works with our relabeled dataset [19].

**Faster R-CNN Inception-ResNet-v2 experiments**

**Dataset preprocessing:** Faster R-CNN Inception-ResNet-v2 uses images with a minimum of 800 and a maximum of 1333 pixels per axis. We split the images in preprocessing into $800 \times 1320$ slices and 252 pixels overlap. The overlap was selected so that each insect class fits in the overlapping area. We annotate insects which are not truncated on image slices. Two different dataset splits (Fig. 6) are used to test the results: (a) images split into $800 \times 1320$ slices, (b) random splitting of the slices in training, testing and validation subsets. The latter aims at a better distribution of classes per subset, since some images have high class imbalance. The downside of this approach is, due to the overlap, the same object can appear in two subsets, so the second dataset split was created by switching the steps of the first split, to address this.

![Workflow for applying Faster R-CNN](image)

**EfficientNet** is a promising, relatively new network family, which is more shallow than the SoA, but still achieves high accuracy by mathematical scaling of the models’ parameters [11]. The base model is EfficientNet-B0, with upscaled baseline versions ranging from EfficientNet-B1 to EfficientNet-B7 [21]. We trained different EfficientNet versions as image classifiers. Our approach consists of two parts: (a) segmenting foreground objects using a threshold based mask, (b) classifying the foreground objects using EfficientNet. The goal of our approach is to carry out object detection through foreground extraction, and focus on improving the classification step, thus speeding up the complete process and increasing overall accuracy. An overview of the workflow for applying EfficientNet can be seen in Fig 5:

![Workflow for applying EfficientNet](image)

**V. EXPERIMENTS**

Our experiments involve preprocessing to improve speed and accuracy, and splits to help balance the dataset. We provide detailed descriptions of our experimental setup, to help in benchmarking efforts. Our results are only indirectly comparable to [8], as they use the original dataset, do not provide all details of their dataset split, and process differently resized images. We provide accuracy metrics best suited for each approach (weighted average accuracy for Faster R-CNN, as in [8], and precision/recall, accuracy for EfficientNet), and analyze the experiments in detail to encourage comparisons by future works with our relabeled dataset [19].

**Faster R-CNN Inception-ResNet-v2 experiments**

**Dataset preprocessing:** Faster R-CNN Inception-ResNet-v2 uses images with a minimum of 800 and a maximum of 1333 pixels per axis. We split the images in preprocessing into $800 \times 1320$ slices and 252 pixels overlap. The overlap was selected so that each insect class fits in the overlapping area. We annotate insects which are not truncated on image slices. Two different dataset splits (Fig. 6) are used to test the results: (a) images split into $800 \times 1320$ slices, (b) random splitting of the slices in training, testing and validation subsets. The latter aims at a better distribution of classes per subset, since some images have high class imbalance. The downside of this approach is, due to the overlap, the same object can appear in two subsets, so the second dataset split was created by switching the steps of the first split, to address this.

![Workflow for applying Faster R-CNN](image)

**EfficientNet** is a promising, relatively new network family, which is more shallow than the SoA, but still achieves high accuracy by mathematical scaling of the models’ parameters [11]. The base model is EfficientNet-B0, with upscaled baseline versions ranging from EfficientNet-B1 to EfficientNet-B7 [21]. We trained different EfficientNet versions as image classifiers. Our approach consists of two parts: (a) segmenting foreground objects using a threshold based mask, (b) classifying the foreground objects using EfficientNet. The goal of our approach is to carry out object detection through foreground extraction, and focus on improving the classification step, thus speeding up the complete process and increasing overall accuracy. An overview of the workflow for applying EfficientNet can be seen in Fig 5:

![Workflow for applying EfficientNet](image)

**V. EXPERIMENTS**

Our experiments involve preprocessing to improve speed and accuracy, and splits to help balance the dataset. We provide detailed descriptions of our experimental setup, to help in benchmarking efforts. Our results are only indirectly comparable to [8], as they use the original dataset, do not provide all details of their dataset split, and process differently resized images. We provide accuracy metrics best suited for each approach (weighted average accuracy for Faster R-CNN, as in [8], and precision/recall, accuracy for EfficientNet), and analyze the experiments in detail to encourage comparisons by future works with our relabeled dataset [19].

**Faster R-CNN Inception-ResNet-v2 experiments**

**Dataset preprocessing:** Faster R-CNN Inception-ResNet-v2 uses images with a minimum of 800 and a maximum of 1333 pixels per axis. We split the images in preprocessing into $800 \times 1320$ slices and 252 pixels overlap. The overlap was selected so that each insect class fits in the overlapping area. We annotate insects which are not truncated on image slices. Two different dataset splits (Fig. 6) are used to test the results: (a) images split into $800 \times 1320$ slices, (b) random splitting of the slices in training, testing and validation subsets. The latter aims at a better distribution of classes per subset, since some images have high class imbalance. The downside of this approach is, due to the overlap, the same object can appear in two subsets, so the second dataset split was created by switching the steps of the first split, to address this.
TABLE II: Precision (P), Recall (R) for the two dataset splits used in the Faster R-CNN Inception-ResNet-v2 experiments.

<table>
<thead>
<tr>
<th></th>
<th>Split 1</th>
<th></th>
<th>Split 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>TV</td>
<td>0.889</td>
<td>0.906</td>
<td>0.790</td>
<td>0.774</td>
</tr>
<tr>
<td>MR</td>
<td>0.744</td>
<td>0.968</td>
<td>0.728</td>
<td>0.927</td>
</tr>
<tr>
<td>NC</td>
<td>0.785</td>
<td>0.995</td>
<td>0.742</td>
<td>0.919</td>
</tr>
</tbody>
</table>

Training and Results: Image augmentation helps the network generalize, especially since the data was recorded under fixed conditions. We used momentum optimizer [22] with a value of 0.9, cosine decay learning rate with a base rate of 0.008 and 5,000 warm up steps. The training was setup for 200,000 steps and with a batch size of 2. We trained the network using the same configuration on our two different dataset splits. It should be noted that, because we annotated the slices only with entire insects, the total prediction count and total ground truth count can differ. This is because our network often detects insects which are truncated and only visible in small parts. However, they are not annotated, so they do not appear in the confusion matrix, but in the total prediction count. Using the first splitting approach, we achieved a weighted averaged accuracy of around 92.51%, where the weighted averaged accuracy is $W\text{AA} = \frac{\sum TP}{\sum Objects}$. On the second dataset split, the $W\text{AA}$ reduced to around 81.27%. The approach by [8] finds a $W\text{AA}$ of 87.4%. However, it is not directly comparable to our $W\text{AA}$, as it splits the images into another shape (6 times $1720 \times 1220$ pixels), but does not provide details about overlaps or how annotations were split.

Faster R-CNN with Inception-ResNet-v2 achieves high accuracy, but is a very resource demanding network. Since our computation time is limited the training was setup for 200,000 steps and with a batch size of 2. Using this setup the training took between 4 to 5 days in total, i.e. 96-120 hours. In the experiments that follow with EfficientNet, it is shown that comparable results can be achieved at a significantly lower computational cost (Table III).

EfficientNet experiments

Dataset preprocessing: Before we apply EfficientNet, we need to detect single objects in the training images. We investigate two approaches for this. We first use the ground truth data, i.e. one bounding box and label per object. We then apply EfficientNet models on these objects to obtain the results in Table V. The second approach detects foreground objects by removing the background using threshold based approaches. To this end, we converted the images of the dataset from RGB to HSV, as the saturation channel (S) is best suited to calculate the background and foreground objects are better separated from each other in the HSV colorspace than in RGB.

To find an optimal threshold, we used the image processing software Fiji [23], which tries out multiple threshold methods and outputs the resulting image, as well as the found threshold value. A montage of thresholding results on the saturation channel is shown in Fig. 7, which are examined to determine the optimal threshold value. The threshold value has to be selected in a way that all objects of the three dataset classes will be segmented from the background. If this value is not chosen carefully the segmentation lets too many background pixels pass or cuts off too much of the real foreground objects. When applying the threshold the insects legs get often segmented as individual objects which is undesirable. To avoid this, we applied gaussian blurring with $\sigma = 2$ to all images before thresholding [24]. A static threshold of 241, found by using [25], [26] and [27], was chosen, as it yields the best results after visual inspection. We keep objects between 435 and 40,000 pixels, to include the class smallest in size, i.e. trialeurodes vaporariorum (TV), as well as NC, the largest in size insect class.

Network choice: We tested EfficientNet versions B0 and B3, which each require different input image sizes and have different computational costs. The majority of our detected objects are smaller than 200 pixels per axis and therefore need to be upscaled. EfficientNet-B0 has the smallest input shape, so it is used to keep the required upscaling low. EfficientNet-B3 is chosen because it achieves higher accuracy while having similar computational cost and input shape with B0. For all networks we use the NoisyStudent + RandAugment ImageNet checkpoints [28], which achieve the highest accuracy compared to the baseline preprocessing.

Training: For training the two versions of EfficientNet we used the same setup to compare the results. The input images are augmented using random rotation, translation, flip and contrast changes. To improve robustness, we set a dropout rate of 0.2. A learning rate of 0.0001 was used with an Adam optimizer [29], and Categorical Cross-Entropy loss. Our network was trained for 500 epochs to avoid overfitting. All
versions were trained on an Nvidia Tesla K40m GPU with 12 GB memory. EfficientNet-B0 gave the best time/accuracy results as it reached the same accuracy as EfficientNet-B3 much faster, requiring 77 epochs instead of 450 (Table III).

It should be noted here that EfficientNet B0 took 3.2 hours to train, and EfficientNet B3 took 10.5 hours, both of which are significantly lower than Faster R-CNN, which required 4-5 days, i.e. 96-120 hours, with comparable accuracy.

<table>
<thead>
<tr>
<th>Network</th>
<th>Training</th>
<th>Avg. Loss</th>
<th>Validation Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>3.2 hours</td>
<td>0.036</td>
<td>0.99018 (77 epochs)</td>
</tr>
<tr>
<td>B3</td>
<td>10.5 hours</td>
<td>0.031</td>
<td>0.99018 (450 epochs)</td>
</tr>
</tbody>
</table>

**TABLE III:** Time to train different EfficientNet versions.

**Testing:** First, we evaluated the trained networks on the test set we created using the method described in section IV. With the bounding box based preprocessing method, the networks achieved an accuracy of over 98% and precision/recall values over 95% (Tables IV and V). Finally, we tested our classification approach combined with the threshold-based object segmentation method (Sec. IV). The segmentation method can also contain other insects and objects, like labels sticking to the traps. Experiments with the trained EfficientNet-B3 showed that the insect classes occurring in the Yellow Sticky Traps dataset are well recognized, in the case of TV sometimes even with 100% confidence. As our network was only trained on the three classes of interest, it falsely classified other objects, like flies, with high confidence as NC and empty images as TV. This can be explained by their similarity to the known classes. These false alarms could be reduced with an additional class for unknown objects, which is an area of future work.

**TABLE IV:** EfficientNet: Accuracy and Loss on test set.

<table>
<thead>
<tr>
<th>Network</th>
<th>Loss</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>0.0374</td>
<td>0.9864</td>
</tr>
<tr>
<td>B3</td>
<td>0.0326</td>
<td>0.991</td>
</tr>
</tbody>
</table>

**TABLE V:** Precision/Recall for EfficientNet B0, B3.

<table>
<thead>
<tr>
<th>Network</th>
<th>TV</th>
<th>MR</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>0.9909</td>
<td>0.9997</td>
<td>0.9751</td>
</tr>
<tr>
<td>B3</td>
<td>0.9886</td>
<td>0.9986</td>
<td>0.9791</td>
</tr>
</tbody>
</table>

**VI. CONCLUSIONS**

In this work we addressed the detection and classification of insects caught by yellow sticky traps in the dataset by [7]. We first corrected errors in the ground truth by improving the placing of bounding boxes, correcting class labels and labeling unlabeled objects. We made the relabeled dataset available on GitHub for future use [19]. We applied Faster R-CNN with Inception-ResNet-v2 on dataset splits that meet the required image shape and prevent truncation of objects. This resulted in many more objects per class than [8]. Depending on the dataset split, we achieved weighted averaged accuracies of 92.51% or 81.27%. The second approach used bounding boxes and foreground segmentation for object localization and EfficientNet for classification. EfficientNet-B3 reached the highest accuracy with 99.1%, as well as the best precision and recall, with good detection results even on torn apart insects. Future work aims at reducing false positives (misclassification of new classes) when combining foreground segmentation with EfficientNet, and further improving detection/classification accuracy.

**REFERENCES**


[9] Forrest Iandola, Song Han, Matthew Moskewicz, Khalid Ashraf, William Dally, and Kurt Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5mb model size,” 02 2016.


