ALERTTrap: A study on object detection in remote insect trap monitoring system using on-the-edge deep learning platform

An D. Le
Electrical and Computer Engineering Department
University of California San Diego
San Diego, USA
d0le@ucsd.edu

Duy A. Pham
Computer Science Department
Bonn-Rhein-Sieg University of Applied Sciences
Bonn, Germany
duyanhpham@outlook.com

Dong T. Pham
Graduate School of Bioresource and Bioenvironmental Sciences
Kyushu University
Fukuoka, Japan
phamtdong0406@gmail.com

Hien B. Vo *
Electrical and Computer Engineering Department
Vietnamese-German University
Binh Duong, Vietnam
hien.vb@vgu.edu.vn

March 4, 2022

ABSTRACT

Fruit flies are one of the most harmful insect species to fruit yields. In AlertTrap, implementation of SSD architecture with different state-of-the-art backbone feature extractors such as MobileNetV1 and MobileNetV2 appear to be potential solutions for the real-time detection problem. SSD-MobileNetV1 and SSD-MobileNetV2 perform well and result in AP@0.5 of 0.957 and 1.0 respectively. YOLOv4-tiny outperforms the SSD family with 1.0 in AP@0.5; however, its through-put velocity is considerably slower, which shows SSD models better candidate for real-time implementation. We also tested the models with synthetic test sets simulating expected environmental disturbances. The YOLOv4-tiny had better tolerance to these disturbances than the SSD models.

Keywords fruit fly · environmental data · smart IoT · edge computing · YOLOv4-tiny · MobileNet · Raspberry Pi · TPU

1 Introduction

Agriculture plays an important role in economic growth, and improving crop yield is of great concern Unnevehr et al. [2007] in Vietnam. On the one hand, insect pesticides can affect the metabolic processes of crops to degrade crop yield and quality Muralidharan and Pasalu [2006]. On the other hand, fruit flies are known to cause 50 to 100% crop loss unless timely interventions are implemented. There are just a small number of fruit fly species have been discovered, namely Bactrocera dorsalis, B. correcta, B. cucurbitae, B. tau, B. latifrons, B. zonata, B. tuberculata, B. moroides and B. albistriga, while some species still remain unidentified. The species which are harmful to fruits are of the common fruit

*Correspondence author
fly species, namely B. cucurbitae and B. tau [Vargas et al., 2015]. In order to optimize crop yields, agricultural workers tend to use a pesticide scheduler rather than take into account the likelihood of pests’ presence in the crop [Hazarika et al., 2009]. Thus, this not only causes a large number of pesticide residues in agricultural commodities but also brings great pressure to the ecological environment [Miller and Spoolman, 2014]. The over-use of pesticides is partly because the information about pest species and densities cannot be provided in a timely and accurate way. In contrast, if the information is provided in a timely fashion, it could be possible to take proper prevention steps and adopt suitable pest management strategies including the rational use of pesticides [Cho et al., 2007] [Zhang and Swinton, 2009].

Traditionally, the information about the environment and pest species is acquired mainly through hand-crafted feature engineering [Cho et al., 2007] such that workers manually use sensors and compare a pest’s shape, color, texture, and other characteristics with justification from the domain experts. Likewise, counting is typically time-consuming, labor-intensive, and error-prone [Zhang and Swinton, 2009], and therefore, it is urgent and significant to establish an autonomous and accurate pest identification system. There is a growing tendency of utilizing machine vision technology to solve these problems with promising performance in the agriculture research field.

In this work, we focus on developing a solution to detect oriental yellow flies which usually harm citrus fruits such as oranges and grapefruits. We further evaluate the object detection models proposed for yellow fly detection problem in Pham et al. [2021] by testing the models with test set simulating potential disturbances occurring in real scenario. Additionally, the work presented in this paper will not only focus on the use of different types of object detection algorithms but also apply the TF-LITE format of the models compatible to edge device system such as TPU processors. This direction of study is to develop real-time detection application with the emerging edge computing technology to enhance the performance of the system in terms of detection accuracy, power efficiency, and latency reduction with the purpose of detecting the living fruit flies beside the stuck and dead ones on the trap. Moreover, the article will describe the hardware implementation so that the work can be reproduced and further developed. Our contributions are:

1. We constructed, developed, and provided a more in-depth discussion of the end-to-end camera-equipped trap, named AlertTrap with installation of a Lynfield-inspired sticky trap, to instantly detect fruit flies and the solar-energy powering system controlled by a separate Raspberry Pi. This work may also provide more detailed information complementing the hardware description of the system provided in Pham et al. [2021].

2. Like our previous work in Pham et al. [2021], we continue to evaluate three different compact and fast object detection deep learning models, namely SSD-MobileNetV1, SSD-MobileNetV2, and the Yolov4-tiny; nevertheless, we further introduce artificial disturbances imitating inference effects which may compromise the detection performance in real-time scenario. Moreover, we also evaluate the SSD-MobileNetV1 and SSD-MobileNetV2 models with their TFLITE format versions on a TPU device. With the results, we compare not only their ability to accurately detect and localize the fruit flies which we had trained them to predict, but also the increase in processing speed as well as the power saving factor.

3. Finally, we contributed a dataset of 200 images of fruit flies lying on the sticky trap. The dataset can be used to finetune a pretrained model so that it can be retrained with a short training time and used for the yellow fly detection problem. In addition, we also provide our synthetic data for evaluation with disturbances along with the original data which is disturbance-free. The link to the image source can be found in Data Availability Statement.

2 Related Works

Insect detection techniques can be classified into three system types, namely manual, automatic, or semi-automatic systems. Manual insect detection techniques are known as a process in which trained workers count the trapped flies on a daily basis. These turn out to be error-prone, time-consuming, and labor-intensive, while semi-automatic and automatic systems can address the disadvantages with the replacement of highly accurate and autonomous emerging technological software and hardware.

Specifically, the remaining two types of insect detection systems are often called e-traps as they are fueled by electronic components with extensive computer algorithms such as a center-controlled unit connecting with a camera and the trap actuators. Thus, they are also known as vision-based insect traps. As suggested in the names, the automatic insect detection systems [Zhong et al., 2018], Kalamatianos et al. [2018], Xia et al. [2018], Kaya and Kayci [2014], Doitsidis et al. [2017], Tirelli et al. [2011], Sun et al. [2016], Philimis et al. [2013], Moscetti et al. [2015], Haff et al. [2013], Xing and Guyer [2008], Potamitis et al. [2017] are fully autonomous, whereas the semi-automatic ones Sciarretta et al. [2019], Miranda et al. [2019], Shaked et al. [2018], Wang et al. [2017] involve human interaction in the loop. For example, in Wang et al. [2017], the images of insect body parts are classified in order to aid the human to better categorize the insects. Generally, the e-traps are equipped with a wide range of post-processing techniques to detect and classify trapped insects. These techniques are recognized by the sensor type that is used to capture the
existence of insects in the trap. Particularly, they are image-based, spectroscopy-based, and opto-acoustic techniques, which correspond respectively to the visible-light camera, the near-infrared (NIR) camera, and the ultrasound sensor.

The image-based techniques consist of three sub-domain techniques, namely deep learning [Zhong et al. 2018], Kalamatianos et al. [2018], Xia et al. [2018], Kaya and Kayci [2014] and shallow learning [Xia et al. 2018], which both are sub-domains in the machine learning field, and image processing [Dotsidis et al. 2017], Tirelli et al. [2011], Sun et al. [2016], Philimis et al. [2013], Moscetti et al. [2015], Haff et al. [2013], Xing and Guyer [2008], Potamitis et al. [2017] techniques.

Shallow learning-wise, Kaya et al. Xia et al. [2018] created a machine-learning-based classifier that can differentiate between 14 butterfly species. The texture and color characteristics are extracted by the writers. A three-layer neural network is used to process the extracted features. The categorization accuracy achieved is 92.85 percent.

The detection approach is based on image processing as described in Tirelli et al. [2011], Sun et al. [2016], Philimis et al. [2013], Moscetti et al. [2015]. While image-processing techniques are simpler than deep learning techniques, their accuracy is reasonable (70-80%) and the system is wired with the illumination environment. However, extensive feature engineering has to take place prior to the classification. Doitsidis and colleagues [Kaya and Kayci 2014] created an image processing method to detect olive fruit flies. By using auto-brightness adjustment, the algorithm first reduces the effect of changing lighting and weather conditions. Then, using a coordinate logic filter improves the edges by amplifying the difference between the dark bug and the bright background. Finally, the technique uses a circular Hough transform followed by a noise reduction filter to identify the trap’s limits. The achieved accuracy rate is 75%.

Tirelli et al. Dotsidis et al. [2017] created a Wireless Sensor Network (WSN) for detecting pests in greenhouses. The image processing technique first removes the effect of light changes from the photos, then denoises them, and finally recognizes the blobs. Sun et al. Tirelli et al. [2011] suggested an insect image processing, segmentation, and sorting algorithm for insect “soup” images. In insect “soup” photos, the insects float on the liquid surface. The method was evaluated on 19 soup images by the authors, and it worked well for the majority of them. Using McPhail traps, Philimis et al. created a WSN that detects the olive fruit fly and med-fly in the field Sun et al. [2016]. WSNs are sensor networks that gather data and may be built to process the information and transfer it to humans. WSNs may also have actuators that respond to specific events. The template comparison algorithm is the detection algorithm. The identification is based on the detection of specific anatomical, patterning, and color characteristics.

Moscetti et al. used Near Infrared Spectroscopy (NIR) to identify infested olives in harvested crops Philimis et al. [2013]. The Genetic Algorithm (GA) extracts the features from the collected full spectral data. The retrieved features serve as the input for the classifier. With 93.75 percent accuracy, Quadratic Discriminant Analysis (QDA) is the most accurate. Haff et al. used hyperspectral imaging to identify contaminated mangoes Moscetti et al. [2015]. The algorithm’s overall error proportion of high-infested samples ranges between 2% and 6%, whereas the algorithm’s overall error rate for low-infested samples is 12.3 percent. In order to detect contaminated cherries, Xing et al. used reflectance and transmittance spectra Haff et al. [2013]. According to the extent of damage, the cherries were separated into two categories: “acceptable” and “non-acceptable.” On transmittance spectra, Canonical Discriminant Analysis (CDA) achieved 85 percent classification accuracy.

Potamitis et al. Tirelli et al. [2011] used opto-acoustic spectrum analysis to construct an olive fruit fly detection system. The opto-acoustic spectrum analysis detects the species of insects based on wingbeat analysis. The authors examined the recorded signal’s temporal and frequency domains. The random forest classifier is fed the retrieved features from the time and frequency domains. The random forest classifier had a precision of 0.93, a recall of 0.93, and an F1-Score of 0.93. The opto-acoustic approach, on the other hand, cannot distinguish between different types of fruit flies, including peaches and figs. Furthermore, solar radiation affects sensor readings, and the trap is susceptible to sudden strikes or shocks that cause false alarms on windy days.

Böckmann et al. Oraby et al. [2021] utilizes Bag of Visual Words (BoVW) to encode clusters of key points extracted by scale-invariant feature transform (SIFT) into some meaningful local features in a so-called visual codebook. This kind of dictionary is then used to incorporate how frequent each feature appears in each patch of newly extracted key points as the input to train an SVM classifier for different classes of flies as well as one background class for a patch of nothing of interest. In contrast, the precision values decreased after 7 days of the insects remaining on the Yellow Sticky Paper by approximately 20% compared to the test results of the initialization measurement on day 0. With regard to class mean accuracy, the dictionary size had no obvious influence but on the recall in individual categories. Within the individual categories, the recall of the background class was the highest, as expected. A maximum value of 99.13% was achieved without differences in color space conversion or dictionary size. The best classification results were achieved with greyscale images and dictionary sizes of 200 and 500 words.

Regarding deep learning techniques, Zhong et al. Ding and Taylor [2016] created a deep-learning-based multi-class classifier that can classify and count six different types of flying insects. The You Only Look Once (YOLO) algorithm
Most of the time, insects are not stationary, so it is difficult to get a clear image of flying insects. In studies by Wang et al. [2012a], Wang et al. [2012b], Kang et al. [2014], Kang et al. [2012], the authors chose insect specimens that were well-preserved in an ideal laboratory environment to capture images of the insects at high resolution. However, since fewer environmental factors are considered in this method, it is limited in specific applications. In this study, we designed a unique automatic autonomous environment data reading and pest identification system to try to eliminate the above problems.

Being largely motivated by preventing the oriental fruit flies from destroying citrus fruits such as oranges and grapefruits, we come up with a trap which targets only that one type of the species, which is specifically named as B. Dorsalis. This can be achieved by replacing the yellow sticky paper with the naled-and fipronil-intoxicated methyl eugenol attractant Chen et al. [2019] in replacement of the yellow sticky paper trap combined with the DIRT dataset, the authors tested various forms of the pre-trained Faster Region Convolutional Neural Networks (Faster-RCNN) deep learning detection technique. Prior to classification, RCNNs are convolutional neural networks containing region proposals that suggest the regions of objects. Faster-RCNN had a mAP of 91.52 percent, where mAP is the average maximum precision for various recall levels. The authors demonstrated that image size has a substantial impact on the detection, but RGB and grayscale images have almost the same detection accuracy. Because Faster RCNN is computationally costly, each e-trap regularly uploads its collected image to a server for processing. Ding et al. created a technique for detecting moth flies Zhang and Swinton [2009]. Translation, rotation, and flipping are used to enhance the visuals. To balance the average intensities of the red, green, and blue channels, the photos are preprocessed with a color-correcting algorithm. The moths in the photos are then detected using a sliding window Convolutional Neural Network (CNN). CNNs are supervised learning algorithms that use learned weights to apply filters on picture pixels. Backpropagation is used to learn the weights. Finally, Non-Max Suppression is used to remove the overlapping bounding boxes (NMS). Using an end-to-end deep learning neural network, Xia et al. detect 24 kinds of insects in agriculture fields Kalamatianos et al. [2018]. A pre-trained VGG-19 network is utilized to retrieve the features. The insect’s position is then determined through the Region Proposal Network (RPN). The proposed model had a mAP of 89.22 percent.

Recently, YOLO is once again proving its notable performance in the work Yun et al. [2022] in pest detection. Especially, the reported results of YOLO v5l by the authors illustrate the mAP of 94.7 percent, where it has the highest recall score of 0.92 among all the other state-of-the-art methods, such as Fast RCNN, Faster RCNN and RetinaNet. The models have been pretrained on COCO dataset Lin et al. [2014] and later fine-tuned on a training dataset of 4480 sub-images made out of 280 images of yellow sticky pheromone traps. However, YOLO v5l is considered slower than YOLOv4. In our previous study, Pham et al. [2021], we used Lynfield-inspired trap with naled-and fipronil-intoxicated methyl eugenol Chen et al. [2019] in replacement of the yellow sticky paper trap combined with object detection system to detect only targeted oriental yellow flies. Unlike the yellow sticky paper, the substance is proved to only attract harmful fruit flies and the detection problem is thus reduced to one-class detection for detecting the existence of the fruit flies and verifying whether the detection is correct. The work showed primary work and provided foundation to further develop real-time system for yellow fly detection in on-field scenario. Compared to Yun et al. [2022], the application Single Shot Multibox Detector with variant backbones and YOLOv4-tiny show significant speed performance to YOLO v5l, while taking the raw images as input instead of segmented sub-images. Nevertheless, the work also showed limitation of ap-playing detection models on edge device due to the slow processing speed, which will be further addressed in this concurrent work.

3 Methodology

3.1 Overview of the Trap System

Most of the time, insects are not stationary, so it is difficult to get a clear image of flying insects. In studies by Wang et al. [2012a], Wang et al. [2012b], Kang et al. [2014], Kang et al. [2012], the authors chose insect specimens that were well-preserved in an ideal laboratory environment to capture images of the insects at high resolution. However, since fewer environmental factors are considered in this method, it is limited in specific applications. In this study, we designed a unique automatic autonomous environment data reading and pest identification system to try to eliminate the above problems.

Being largely motivated by preventing the oriental fruit flies from destroying citrus fruits such as oranges and grapefruits, we come up with a trap which targets only that one type of the species, which is specifically named as B. Dorsalis. This can be achieved by replacing the yellow sticky paper with the naled-and fipronil-intoxicated methyl eugenol attractant to assure only B. Dorsalis flies are lured into the trap. It eases the classification and counting process as no other insects will get attracted by the methyl eugenol attractant Chen et al. [2019]. The objects can be further reassured by the object detection system before getting counted.
Figure 1: Overview of the trap system consisting of a) The solar panel system, b) The operation system, c) The sensor system, d) The modified Lynfield trap and e) The object detection system [Pham et al. 2021].

The system involves a two-fold setting: a) an electronic system reads environment data with a sticky trap installed and a digital camera is set up to collect images of the flies, b) the object detection software to recognize fruit flies on the image before sending all information (environmental data and number of fruit flies) via email or SMS to alert farmers independently. The whole system is autonomous and powered by a solar system. This system is implemented on an Arduino Uno and Raspberry Pi system. The results provide precise prevention and treatment methods based on the combination of pest information and other environmental information. Based on this edge computing design, the computation pressure on the server is alleviated and the network burden is largely reduced.

The edge-computing traps are designed to work separately and individually report the count of fruit flies to the farmers. They are spread, based on the effectiveness of the attractant, such that each 2-3 devices can cover the area of 1000 square meters.

3.2 Hardware

Overall, the hardware part of the system consists of five interconnected subsystems with distinctive functions and behaviors, which are described in Figure 1, which is also shown in [Pham et al. 2021], namely the solar panel system, the control system, the sensor system, the trap, and the object detection and communication system.

The power system of the trap contains of a solar panel, a battery, and a solar charge controller (Figure 1a). The solar panel converts the solar energy to DC current with 830 mA to power the trap system. The converted energy is stored in an electrochemical energy storage with a capacity of 5 Ah and a voltage of 12 V. The Arduino in the operation system will check voltage of the battery with a voltage sensor to make sure the battery voltage above a certain level required for the system’s operation. If the condition is not met, the object detection module will not be operated. The Pulse Width Modulation (PWM) solar charge controller is used in order to control the device voltage, open the circuit, and halt the charging process if the battery voltage is above a certain level.

The operation system (Figure 1b) is controlled by an Arduino microcontroller board. As aforementioned, the Arduino module reads the battery voltage with a voltage sensor from the sensor system to decide whether to turn on or off the object detection system, which is controlled by the Raspberry module. The SSR10D is used to control activate and deactivate the object detection system. The SSR10D is a solid-state relay and uses lower power electrical signal to generate an optical semiconductor signal as an activate signal for the opto-transistor to allow high voltage going into and powering the device’s output device, which is the Raspberry device in this case. In addition, the lower electrical signal is the output from the 2N2222 bipolar junction transistor receiving control signal from the Arduino module. Hence, the Arduino can stop the Raspberry Pi 3b+ computer drawing current from the solar system after it is shut down. The sensor system (Figure 1c) takes responsibility for measuring the three important factors, temperature, humidity and light. Also, it records the current created by the solar system and the voltage battery. The humidity and temperature, which also affect the living environment of the yellow flies, are measured with the AM2315 I2C sensor. RGB and clear light is measured with the TCS34725 light sensor with IR filter and white LED. In addition to sensor system, INA219 is used to read the solar current and battery voltage information. Moreover, a DS1307, which is a battery-backed real time clock (RTC), is used to help the microcontroller keep track of time. These information from the sensors along with their...
corresponding time are stored in a SD card attached on the device. These two factors, the operation system and sensor system, help the microcontroller decide whether to turn on the object detection or not. The object detection system, shown in Figure 1, is operated by the Raspberry Pi 3b+ and collect images for its fruit fly detection algorithm with a waveshare Pi camera with 5 MP. The camera is placed at the top of a double-size Lynfield shape trap with several holes at the bottom, shown in Figure 1d. To attract and capture only the yellow flies, methyl eugenol is used as the attractant to the insects Chen et al. [2019], which later help to simplify the detection and classification problem. The Raspberry Pi module will receive data from the sensor system and sends all data to the notification system to notify or alert farmers about the environment data and the number of detected fruit flies through email or SMS. The behavior of the whole system is described in the flow chart shown in Figure 2.

3.3 Software - Object detection pipelines

The architectures used to train the yellow fly detection models are SSD with MobilenetV1 and MobilenetV2 backbones, and YOLOv4-tiny. The selected models are all single-stage detection models since, compared to their counterpart, the two-stage detection models, the single-stage detection models have been shown to have a faster processing speed with a competitive performance. The pre-trained models of these architectures were fine-tuned with our proposed insect dataset so that they can be used for the yellow fly detection application, as fine-tuning is also one of the common solutions for data scarcity problem in object detection. Because the models had been trained with COCO dataset Lin et al. [2014], which is a large dataset having over 200,000 labeled images with 1.5 million object instances for 80 object categories, and, hence, contains common features for object detection problem, fine-tuning the models with 200 yellow fly images helped the models perform the yellow fly detection task.

3.3.1 SSD

To solve the real-time object detection in the yellow fly detection problem, variants of Single-Shot Multibox Detector (SSD) are used. The SSD method was first proposed in Liu et al. [2016] by Wei Liu et al. and described as a one-stage object detection method that completely omits the region proposal and pixel/feature re-sampling stages used in region proposal-based techniques such as Faster-RCNN. The SSD network is based on a feed-forward network that uses default bounding boxes with different shapes, ratios, and scales in order to produce a fixed-size collection of

![Flow chart of the trap system.](image)
bounding boxes with corresponding shape offsets and confidence scores\cite{liu2016ssd}. In addition, the early layers of the network are based on a standard image classification without classification layers, which is called the base network \cite{liu2016ssd}. In this work, MobileNetV1 and MobileNetV2 are used as base networks for the SSD detection models. The elimination of region proposal and pixel/feature re-sampling stages helps to improve the processing speed of the model compared to two-stage techniques such as Faster-RCNN with a small trade-off in the model’s accuracy, which enables the implementation of real-time object detection with high accuracy on embedded system for yellow fly detection problem.

### 3.3.2 MobileNetV1

The approach was first proposed in \cite{howard2017mobilenets} by A. Howard et al. and was described as a lightweight deep neural network for mobile and embedded system applications with an efficient trade-off between latency and accuracy. The model is based on depth wise separable convolution including depth wise convolution layer which is used to apply a single filter per input channel, and point-wise convolution layer, which creates a linear combination of the output of the depth wise layer. In addition, to construct the model further less computationally expensive, width multiplier, which is used to thin the network uniformly at each layer, and resolution multiplier, which is applied to input images and the internal representation of each layer, were introduced as a hyper-parameter to tune and choose the size of the model.

### 3.3.3 MobileNetV2

The MobileNetV2 approach was first presented in \cite{sandler2018mobilenetv2} by M. Sandler et al. The approach is built based on the MobileNetV1; therefore, it also makes use of the depth wise separable convolution architecture which consists of depth wise convolution layer and 1x1 point-wise convolution layer. In addition, the approach also utilizes linear bottleneck layers in convolutional blocks to optimize the neural architecture \cite{sandler2018mobilenetv2}. Moreover, inverted residual design is also used in the model to implement shortcuts between bottlenecks with the purpose of improving the ability of gradient propagation across the multiplier layers. Nevertheless, the implementation of the inverted design also showed better performance and significantly more memory efficiency in the work \cite{sandler2018mobilenetv2}.

The training and evaluation of the SSD with MobileNetV1 and MobileNetV2 base networks is based on the pre-trained models provided in the Object Detection API in TensorFlow Model Garden \cite{huang2017speed}. The models were also trained on the Google Colab Pro environment to utilize the provided GPUs for the training purpose.

### 3.3.4 YOLOv4-tiny

The YOLOv4 model is a method for detecting objects that was developed from the YOLOv3 model proposed in \cite{redmon2018yolov3}. The YOLOv4 approach was created by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao \cite{bochkovskiy2020yolov4}. It is twice as fast as EfficientDet in terms of performance. Furthermore, when compared to YOLOv3, AP (Average Precision) and FPS (Frames Per Second) in YOLOv4 have increased by 10% and 12%, respectively. This is because a CSPDarknet53 backbone and a PANet path-aggregation neck along with the YOLOv3 head make up for the YOLOv4 architecture. YOLOv4-tiny \cite{jiang2020yolov4} is the compressed version of YOLOv4. Based on YOLOv4, it is suggested that the network topology should be simplified, and parameters should be reduced so that it may be implemented on mobile and embedded devices. The YOLOv4-tiny model can be trained in a shorter time than the YOLOv4.

### 4 Evaluation

The proposed models, YOLOv4-tiny, SSD-MobileNetV1, and SSD-MobileNetV2, were evaluated by using performance metrics such as precision, recall, F1 score, mean IoU, and average precision (or AP). In addition, the models’ processing time was also evaluated to check for real-time application feasibility. The processing time of the models was calculated while they were run on CPU, GPU, and TPU hardware. The hardware provided by Colab Pro service was Tesla V100-SXM2-16GB, TPU V2, and Intel(R) Xeon(R) CPU @ 2.30GHz, which were used to find the average processing time of the models on GPU, TPU, and CPU respectively. Moreover, besides the regular test set, the models were also checked with synthetic test images originating from the original test set. The augmentation effects were implemented to simulate disturbances that can be captured in practice, such as leaf, insect, dust, and flaring effect.
4.1 Training Dataset

The contributed 200 images of fruit flies in the trap are consolidated into two parts, namely training dataset and test dataset, with the proportion of 75% and 25%, respectively. Therefore, there are 150 images being used to train the models incorporated in this paper. The remaining 50 images are responsible for evaluating the performance of such models.

4.2 Test Dataset

For the evaluation of the proposed models, the original test set, which includes 50 images, was used along with other four synthetic datasets generated from the original test images. The synthetic datasets were used to simulate the common disturbances which could be captured in real-life scenarios such as blurry, dust, salt-pepper and flaring effect. Therefore, in general, the total test set has 250 images, in which 50 images come from the original dataset and the other 150 images are copies of them with augmentation effects. The example of an original image and its synthetic versions are shown in Figure 3.

The synthetic images with dust and salt-pepper augmentation are to simulate the disturbances coming from the weather and environmental conditions, while the blurring augmentation is used to simulate foggy or out-of-focus issues captured by the camera in the field and maybe disrupt the classification of the targeted object class.

4.3 Evaluation Metrics

The performance of the trained detectors was evaluated by being tested with a test dataset. The true positive (TP), false positive (FP), and false negative (FN) were counted from the detection results and used to find the precision, recall, and F1 score metrics. TP is the number of correctly detected objects, while FP shows the amount of the falsely detected object and FN informs the number of targeted objects which were missed during the detection process. The three metrics can then be used to evaluate further aspects of the models’ performance such as precision, recall, F1-Score, and average precision, which were also used for model evaluation in this work. In addition, the average processing time of the models during the detection process was also measured and used as another evaluation criterion, and Mean IoU was also used to evaluate the localization of the detection.

4.3.1 Precision

Precision is an evaluation metric calculated with true positives and false positives. The metric indicates how accurate the detector’s performance is by showing the percentage of correctly detected objects out of the total detected objects. The relationship between the precision, true positive and false positive numbers can be expressed as the following:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{AllDetections}$$ (1)
4.3.2 Recall

Recall is an evaluation metric calculated with counted true positives and false negatives. The recall value decreases when the number of false negatives is increased. The metric measures how many objects are missed by the evaluated detector by showing the percentage of correctly detected objects out of the total true objects. The relationship between the recall and true positive and false negative numbers can be expressed as the following:

\[
Recall = \frac{TP}{TP + FN} = \frac{TP}{AllGroundtruths} \tag{2}
\]

4.3.3 F1-Score

F1-Score is an evaluation metric calculated with both precision and recall metrics, or with all counted true positives, false positives, and false negatives. The F1-Score metric combines both precision and recall with equal weight to show the balance and relative relation between the precision and recall metrics. The mathematical expression of the F1-Score metric regarding other evaluation metrics is shown as:

\[
F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3}
\]

4.3.4 Average Precision

By adjusting the confidence score threshold, the precision and recall will also be changed accordingly. By calculating the mean value of the precision score corresponding to the confidence score threshold varied from 0 to 1, average precision, or AP for short, can be found.

4.3.5 Mean IoU

IoU stands for intersection over union. The intersection can be understood as the overlapping area between a ground-truth bounding box and its corresponding detected bounding box, while the union is the total area of the ground-truth bounding box and the corresponding detected bounding box with the omission of the overlapping area. The graphical representation of the mentioned definition is shown in Figure 4.

Then, the IoU is the ratio of the intersection area over the area of union of the two bounding boxes. The higher the value of the IoU, the more matched the two bounding boxes are. Therefore, the mean IoU, which is the mean value of the IoU values of all available pairs of detected bounding boxes and their corresponding ground-truth bounding boxes, shows the localization of the examined algorithm. The mathematical relation between the IoU, the area of intersection, and the area of union can be expressed with the following formula:

\[
IoU = \frac{Area of Intersection}{Area of Union} \tag{4}
\]
4.3.6 Processing Time

To examine the feasibility of real-time implementation of the trained models, the models’ processing time was also measured and used as an evaluation metric. The processing speed of a detector is found by calculating the mean processing speed of the detector during its detection operation over 250 test images.

4.4 Experimental Results

4.4.1 Precision, Recall, F1-score, mean IoU, and AP evaluation

In this work, the models were evaluated for F1-score, mean IoU, and AP evaluation metrics with different choices of IoU threshold value. The following shows the performance tables of the trained detectors through different testing scenarios with IoU threshold values of 0.25, 0.50, and 0.75, which may reflect the performance of the algorithms with different localization demands. Additionally, in this paper, the result performance of the models in normal testset or disturbance-free is further discussed from the discussion in Pham et al. [2021]. The discussion presented here will complement the evaluation shown in Pham et al. [2021].

a) Normal Testset

In the normal test case with no augmented and synthetic disturbances, it can be observed in all IoU thresholds that YOLOv4-tiny is the model having the best performance in all aspects among the three examined algorithms, YOLOv4-tiny, SSD-MobileNetV1, and SSD-MobileNetV2. Especially, for IoU threshold values 0.25 and 0.5, the YOLOv4-tiny model achieved perfect F1-Score, and AP metrics with the value of 1.0. Moreover, with high localization constraints, YOLOv4-tiny still has good F1-score which is 0.847. In addition, the detection from the YOLOv4-tiny model also has great localization based on the mean IoU metric, which is 0.834 for IoU threshold 0.25 and 0.50, and 0.857 for IoU threshold 0.75. Nevertheless, SSD-MobileNetV2 also has comparable performance to the performance of YOLOv4-tiny, since for IoU threshold 0.25 and 0.5, SSD-MobileNetV2 also achieved a great F1-Score, and AP metrics with the values of 0.969 and 1.0 respectively. In the extreme case of IoU threshold 0.75, SSD-MobileNetV2 can also have good performance with values of 0.751 for F1-score, and 0.69 for AP metric. In addition, SSD-MobileNetV2’s detection also has similar localization compared to YOLOv4-tiny, which are 0.811 for IoU threshold 0.25 and 0.50 and 0.847 for IoU threshold with the value of 0.75. The SSD-MobileNetV1 also has good performance for IoU threshold 0.25 and 0.50; however, with IoU threshold of 0.75, the model’s performance was heavily compromised. Moreover, in all three IoU threshold cases, the SSD-MobileNetV2 has better performance than SSD-MobileNetV1 in all aspects. The performance results on different localization constraints are shown in Figure 5. The results can also be found in Pham et al. [2021].

b) Blurry Testset

In Blurry testset, a Box filter of size 30x30 is convolved over the images in the Normal testset to create the blur effect which aims to replicate the foggy weather inside the trap or the out-of-focus issue of the camera. Generally, this testset changes the overall features of the fruit flies because of the averaging effect of the filter; therefore, the models
might not work as expected, which is clearly shown via the results of SSD-MobileNetV2. However, According to Figure 6, YOLOv4-tiny still performs stably on the testset. Specifically, YOLOv4-tiny can maintain its metrics values in the variation range of 0.2 throughout three IoU thresholds; whereas, SSD-MobileNetV2 is totally collapsed when the IoU threshold increases and SSD-MobileNetV1 significantly drops in F1-Score and AP by a value of 0.6 at the extreme case IoU threshold.

c) **Salt-pepper Testset**

The salt-pepper disturbances imply the appearance of unwanted tiny fraction of leaves or other insects that are accidentally flown into the trap by the wind. It is named “Salt-pepper” because the effect looks visually like salt and pepper on a dish, which is not related to the salt-and-pepper noise in image processing point of view. It can be observed from Figure 7 that YOLOv4-tiny, with extreme localization constraint, IoU threshold 0.75, the model outperformed other models, and it also has best mean IoU in all IoU threshold constraints. However, adding the Salt-pepper disturbance makes all three models’ performance worse than their performance on the remaining testsets. Specifically, SSD-MobileNetV1 suffers the most where in the extreme case, both its F1-Score and AP drop to 0.

d) **Dust Testset**

With test images with added dust disturbance, it can be observed in all IoU thresholds that YOLOv4-tiny has the best performance in all aspects. Nevertheless, compared to its performance with the normal dataset, it can be observed
that the disturbance effect did compromise the performance of the model based on the F1-score. The same conclusion can also be drawn for SSD-MobileNetV1 and SSD-MobileNetV2 models based on their performance on the testset. In addition, the dust disturbance also proves to have a great negative effect on the SSD-MobileNetV2 model when the extreme localization constraint is applied, since while in the Salt-pepper testset with IoU threshold of 0.75, the SSD-MobileNetV2 could still perform well with an F1-Score value of 0.38, while with dust disturbance, the model can only achieve 0.251 in F1-Score. Based on the mean IoU metric, the dust disturbance also decreases the localization of all three models’ detection results.

4.4.2 Processing time evaluation

In this work, the model candidates are examined with CPU, GPU, and TPU hardware. Moreover, the SSD-MobileNetV1 and SSD-MobileNetV2 models with TFLITE convert support were also converted into TFLITE format which is compatible with TPU and can exploit the advantage of the hardware. Table 1 shows the processing time of each model on CPU, GPU, and TPU hardware. It can be observed that SSD models have a faster processing time than the YOLOv4-tiny model on GPU, CPU, or TPU hardware. Moreover, the TFLITE version of SSD-MobileNetV1 and SSD-MobileNetV2 models run on TPU is much faster than SSD-MobileNetV1, SSD-MobileNetV2, and YOLOv4-tiny in-ference graphs run on the same device. In addition, compared to the work in [Pham et al. 2021], we update the processing speed in FPS for TFLITE models with SSD-MobileNetV1 and SSD-MobileNetV2 architecture on a TPU device. TPU processors can be found in edge device such as Google Coral Dev board, and by comparing the processing speed of the models in TFLITE format in TPU shown in this work and the processing speed of the models in Raspberry Pi shown in [Pham et al. 2021], it can be shown that with the same architectures, the TFLITE models’ processing speed on a TPU device is approximately 6 times faster than the inference models on Raspberry Pi. This shows that TFLITE model on TPU edge device would be a more feasible for real-time application implementation.

| Models              | CPU  | GPU  | TPU  |
|---------------------|------|------|------|
| SSD-MobileNetV1     | 3.414 FPS | 29.140 FPS | 1.025 FPS |
| SSD-MobileNetV2     | 3.485 FPS | 21.000 FPS | 1.024 FPS |
| YOLOv4-tiny TFLITE  | 1.282 FPS | 1.207 FPS | 0.545 FPS |

5 Discussion

The assessments in this research are dedicated to search for the most appropriate object detection method among the current state-of-the-art algorithms which have been implemented for insect and fly recognition under our hardware constraints and problem definition. As we only target one type of fruit flies that particularly causes harm to the citrus
fruits, we have replaced the yellow sticky paper with a white disc containing the special attractant as a hard refinement to pick up only the flies we are interested in. The object detection problem is then simplified to only one-class object detection, which eases the need of exhausting feature extraction. However, the general constraints, such as correctness and fastness, for an object detection task on an edge-device still hold since early detection and separation of the infected areas are extremely important to the fruit yield. Ultimately, SSD-MobileNetV1, SSD-MobileNetV2 and YOLOv4-tiny are the best candidates for these requirements because they utilize extracted features from a backbone classification model to automatically propose object-related regions instead of using a region-proposal module to pool the related regions before classifying them as many two-stage object detection models, such as Fast-RCNN and Faster-RCNN, normally do.

Regarding the correctness, YOLOv4-tiny clearly outperforms the two SSD models over all of the evaluations on four different types of testset with very high and stable results. This could make YOLOv4-tiny become the most probable candidate, because YOLOv4-tiny demonstrates a robust testing performance towards citrus fruit flies detection although it has been fine-tuned only on a training dataset without augmentation effects. SSD-MobileNetV2 shows appropriate robustness given its small number of trainable parameters by yielding good results in two over four testsets, while SSD-MobileNetV1 only works with the original testset. Nevertheless, SSD-MobileNetV2 fails dramatically with the Blurry testset, which simulates a very frequent event that could happen in a fruit field. By all means, YOLOv4-tiny is no doubt the chosen one among the three methods if we would not have taken other aspects into account.

Conventionally, highly accurate object detection methods trade their processing speed for its better performance due to the employment of more parameters in their architecture. YOLOv4-tiny is not an exception where its processing speed is far from real-time (1.2 FPS compared to 30 FPS). While missing a fraction of time could lead to undetectable event in which the flies appear, our second choice which is the SSD-MobileNetV2 model should be taken into account. In order to realize this choice after extensive performance analysis with four different testsets, SSD-MobileNetV2 must have been fine-tuned with more augmented versions of the original training dataset before going to production to leverage its robustness to the level of YOLOv4-tiny while retaining its processing speed. Moreover, TFLITE version of SSD-models are also tested on a cloud TPU Google engine, TPUv2, for the feasibility of edge-device deployment.

6 Conclusion and Outlook

Experimental results show that the Raspberry Pi system successfully gained environmental data and number of counted pests which were transferred to email addresses through the 4G network. The full YOLO version cannot run real time on Raspberry Pi which poses the need of a lighter Object detection algorithm for future research.

From the results, it can also be concluded that in general, YOLOv4-tiny has the best performance among the three model candidates. Nevertheless, SSD-MobileNetV2 also has a comparable performance to YOLOv4-tiny. Moreover, the SSD-MobileNetV2 model also outperforms the YOLOv4-tiny model in some test scenarios with synthetic disturbances. In addition, SSD models have a clear advantage over YOLOv4-tiny in processing time criterion, which makes real-time detection application with high accuracy feasible. Furthermore, the TFLITE versions of SSD models also process faster than the SSD models’ inference graph on TPU hardware, suggesting a feasibility of real-time implementation of the SSD models on edge devices with TPU processors such as Google Coral Dev Board with edge TPU. Therefore, in the future work, Google Coral Dev Board will be implemented on the system, which can be used to compare with the Raspberry Pi 3b+’s in accuracy and processing time aspects. In addition, in-field operation of the system will be tested to check the system’s practicability and for further improvement. In addition, from the test result with the synthetic test sets, it can be seen that the SSD family models were susceptible to disturbance and noise compared to the YOLOv4-tiny model. Therefore, our next attempt is to improve the SSD models’ performance training the detectors with augmented and synthetic data synthesized from the original dataset. Moreover, by building several trap devices, we will try to apply federated learning on the multiple on-field traps so that the detection algorithm can be trained, and improved while being applied on the field. Hence, the detection performance of the traps can be further boosted. Furthermore, we also would like to further develop our detection solution to other types of insects so that it may not only enhance the yellow fly detection performance but also make the solution applicable for other insect detection problems. To achieve the goal, we will need to expand our dataset so that it would contain other types of insects.

7 Data Availability Statement

The image dataset can be found at the following link: https://github.com/a11to1n3/AlertTrap-Dataset.
References

Laurian J Unnevehr et al. Causes of and constraints to agricultural and economic development: Discussion. *American Journal of Agricultural Economics*, 89(5):1168–1169, 2007.

K Muralidharan and IC Pasalu. Assessments of crop losses in rice ecosystems due to stem borer damage (lepidae: Pyralidae). *Crop protection*, 25(5):409–417, 2006.

Roger I Vargas, Jaime C Piñero, and Luc Leblanc. An overview of pest species of bactrocera fruit flies (diptera: Tephritidae) and the integration of biopesticides with other biological approaches for their management with a focus on the pacific region. *Insects*, 6(2):297–318, 2015.

Lakshmi K Hazarika, Mantu Bhuyan, and Budhindra N Hazarika. Insect pests of tea and their management. *Annual review of entomology*, 54:267–284, 2009.

G Tyler Miller and Scott Spoolman. *Sustaining the earth*. Cengage Learning, 2014.

Jongman Cho, Junghyeon Choi, Mu Qiao, CW Ji, HY Kim, KB Umh, and TS Chon. Automatic identification of whiteflies, aphids and thrips in greenhouse based on image analysis. *Red*, 346(246):244, 2007.

Wei Zhang and Scott M Swinton. Incorporating natural enemies in an economic threshold for dynamically optimal pest management. *Ecological Modelling*, 220(9-10):1315–1324, 2009.

Duy A Pham, An D Le, Dong T Pham, and Hien B Vo. Alerttrap: On designing an edge-computing remote insect monitoring system. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 323–328. IEEE, 2021.

Yuanhong Zhong, Junyuan Gao, Qilun Lei, and Yao Zhou. A vision-based counting and recognition system for flying insects in intelligent agriculture. *Sensors*, 18(5):1489, 2018.

Romanos Kalamatianos, Ioannis Karydis, Dimitris Doukakis, and Markos Avlonitis. Dirt: The dacus image recognition toolkit. *Journal of Imaging*, 4(11):129, 2018.

Denan Xia, Peng Chen, Bing Wang, Jun Zhang, and Chengjun Xie. Insect detection and classification based on an improved convolutional neural network. *Sensors*, 18(12):4169, 2018.

Yilmaz Kaya and Lokman Kayci. Application of artificial neural network for automatic detection of butterfly species using color and texture features. *The visual computer*, 30(1):71–79, 2014.

Lefteris Doitsidis, George N Fouskitakis, Kyriaki N Varikou, Iraklis I Rigakis, Savvas A Chatzichristofis, Androniki K Papafilippaki, and Athanasia E Birouraki. Remote monitoring of the bactrocera oleae (gmelin)(diptera: Tephritidae) population using an automated mcphail trap. *Computers and Electronics in Agriculture*, 137:69–78, 2017.

P Tirelli, NA Borghese, F Pedersini, G Galassi, and R Oberti. Automatic monitoring of pest insects traps by zigbee-based wireless networking of image sensors. In *2011 IEEE International Instrumentation and Measurement Technology Conference*, pages 1–5. IEEE, 2011.

Changming Sun, Paul Flemons, Yongsheng Gao, Dadong Wang, Nicole Fisher, and John La Salle. Automated image analysis on insect soups. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6. IEEE, 2016.

Panayiotis Philimis, Elias Psimolophitis, Stavros Hadjiyiannis, Alessandro Giusti, Josep Perelló, Albert Serrat, and Pedro Avila. A centralised remote data collection system using automated traps for managing and controlling the population of the mediterranean (ceratitis capitata) and olive (dacus oleae) fruit flies. In *First International Conference on Remote Sensing and Geoinformation of the Environment (RSCy2013)*, volume 8795, page 87950X. International Society for Optics and Photonics, 2013.

Roberto Moscetti, Ron P Haff, Elisabetta Stella, Marina Contini, Danilo Monarca, Massimo Cecchini, and Riccardo Massantini. Feasibility of nir spectroscopy to detect olive fruit infested by bactrocera oleae. *Postharvest Biology and Technology*, 99:58–62, 2015.

Ronald P Haff, Sirinnapa Saranwong, Warunee Thanapase, Athit Janhiran, Sumaporn Kasemsumran, and Sumio Kawano. Automatic image analysis and spot classification for detection of fruit fly infestation in hyperspectral images of mangoes. *Postharvest Biology and Technology*, 86:23–28, 2013.

Juan Xing and Daniel Guyer. Comparison of transmittance and reflectance to detect insect infestation in montmorency tart cherry. *Computers and Electronics in Agriculture*, 64(2):194–201, 2008.

Ilyas Potamitis, Iraklis Rigakis, and Nicolaos-Alexandros Tatlas. Automated surveillance of fruit flies. *Sensors*, 17(1):110, 2017.
Andrea Sciarretta, Maria Rosaria Tabilio, Armando Amore, Marco Colacci, Miguel Álvarez Miranda, David Nestel, Nikos T Papadopoulos, and Pasquale Trematerra. Defining and evaluating a decision support system (DSS) for the precise pest management of the Mediterranean fruit fly, Ceratitis capitata, at the farm level. *Agronomy*, 9(10):608, 2019.

Miguel Ángel Miranda, Carlos Barceló, Ferran Valdés, José Francisco Feliu, David Nestel, Nikolaos Papadopoulos, Andrea Sciarretta, Maurici Ruiz, and Bartomeu Alorda. Developing and implementation of decision support system (DSS) for the control of olive fruit fly, Bactrocera oleae, in Mediterranean olive orchards. *Agronomy*, 9(10):620, 2019.

B Shaked, A Amore, C Ioannou, F Valdés, B Alorda, S Papanastasiou, E Goldshtein, C Shenderey, M Leza, C Pontikakos, et al. Electronic traps for detection and population monitoring of adult fruit flies (Diptera: Tephritidae). *Journal of Applied Entomology*, 142(1-2):43–51, 2018.

Jiang-ning Wang, Xiao-lin Chen, Xin-wen Hou, Li-bing Zhou, Chao-Dong Zhu, and Li-qiang Ji. Construction, implementation and testing of an image identification system using computer vision methods for fruit flies with economic importance (Diptera: Tephritidae). *Pest management science*, 73(7):1511–1528, 2017.

Tamer Oraby, Michael G Tyshenko, Jose Campo Maldonado, Kristina Vatcheva, Susie Elsaadany, Walid Q Alali, Joseph C Longenecker, and Mustafa Al-Zoughool. Modeling the effect of lockdown timing as a Covid-19 control measure in countries with differing social contacts. *Scientific reports*, 11(1):1–13, 2021.

Wei-guang Ding and Graham Taylor. Automatic moth detection from trap images for pest management. *Computers and Electronics in Agriculture*, 123:17–28, 2016.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

Po-Hung Chen, Wen-Jer Wu, and Ju-Chun Hsu. Detection of male oriental fruit fly (Diptera: Tephritidae) susceptibility to naled and fipronil-intoxicated methyl eugenol. *Journal of Economic Entomology*, 112(1):316–323, 2019.

Seung-Ho Kang, Su-Hee Song, and Sang-Hee Lee. Identification of butterfly species with a single neural network system. *Journal of Asia-Pacific Entomology*, 15(3):431–435, 2012.

Seung-Ho Kang, Jung-Hee Cho, and Sang-Hee Lee. Identification of butterfly based on their shapes when viewed from different angles using an artificial neural network. *Journal of Asia-Pacific Entomology*, 17(2):143–149, 2014.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Seung-Ho Kang, Menglong Zhu, Bo Chen, Dmitri Kalenchchenko, Wei-jun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNet: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Serge Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

Joseph Redmon and Ali Farhadi. Yolov4: An incremental improvement. *arXiv preprint arXiv:1904.13211*, 2019.