Received: d month yyyy | Revised: d month yyyy | Accepted: d month yyyy | Published online: d month yyyy

RESEARCH ARTICLE

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

Journal of Computational and Cognitive Engineering
yyyy, Vol. XX(XX) 1–5
DOI: 10.47852/bonviewJCCEXXXXXXXXX

An Dinh Le [1], Duy Anh Pham [2], Dong Thanh Pham [3], Hien Bich Vo [4]*
1 Electrical and Computer Engineering Department, University of California San Diego, USA, d0le@ucsd.edu.
2 Osnabrück University, Joint Lab Artificial Intelligence and Data Science, Osnabrück, Germany, apham@uni-osnabrueck.de.
3 Graduate School of Bioresource and Bioenvironmental Sciences, Kyushu University, Japan, phamtdong0406@gmail.com.
4 Electrical and Computer Engineering Department, Vietnamese-German University, Vietnam, hien.vb@vgu.edu.vn.

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

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 throughput 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. The Raspberry Pi system successfully gathered environmental data and pest counts, sending them via email over 4G. However, running the full YOLO version in real time on Raspberry Pi isn't feasible, indicating a need for a lighter object detection algorithm for future research. Among model candidates, YOLOv4-tiny generally performs best, with SSD-MobileNetV2 also comparable and sometimes better, especially in scenarios with synthetic disturbances. SSD models excel in processing time, enabling real-time, high-accuracy detection. TFLITE versions of SSD models also process faster than their inference graph on TPU hardware, suggesting real-time implementation on edge devices like the Google Coral Dev Board. The results demonstrate the feasibility of real-time implementation of the fruit fly detection models on edge devices with high performance. In addition, YOLOv4-tiny is shown to be the most probable candidate, because YOLOv4-tiny demonstrates a robust testing performance towards citrus fruit fly detection. Nevertheless, SSD-MobileNetV2 will be the better model, considering the inference time.

Keywords: fruit fly, environmental data, smart IoT, edge computing, TPU

1. Introduction

Agriculture plays an important role in economic growth, and improving crop yield is of great concern [1] in Vietnam. On the one hand, insect pesticides can affect the metabolic processes of crops to degrade crop yield and quality [2]. 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 that 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 remain unidentified. The species which are harmful to fruits are of the common fruit fly species, namely B. cucurbitae and B. tau [3]. To optimize crop yields, agricultural workers tend to use a pesticide scheduler rather than consider the likelihood of pests' presence in the crop [4]. Thus, this not only causes many pesticide residues in agricultural commodities but also brings great pressure to the ecological environment [5]. The overuse of pesticides is partly because 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 [6, 7].

Traditionally, the information about the environment and pest species is acquired mainly through handcrafted feature engineering [6] 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 [8]. 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 implement and evaluate the object detection models by applying the models with testsets 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 TFLITE 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.

(2) We evaluate three different compact and fast object detection deep learning models, namely SSD-MobileNetV1, SSD-MobileNetV2, and the Yolov4-tiny. Nevertheless, we 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.

2. Literature Review

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 [9-20] are fully autonomous, whereas the semi-automatic ones [21-24] involve human interaction in the loop. For example, in [24], the images of insect body parts are classified to aid humans 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 optoacoustic 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 [9-12] and shallow learning [12], which both are sub-domains in the machine learning field, and image processing [13-20] techniques.

Shallow learning-wise, Kaya et al. [12] 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 [14-17]. 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 must take place prior to the classification. Doitsidis and colleagues [13] 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%. In [14], it was reported that a Wireless Sensor Network (WSN) was created 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. In [15], it was suggested that insect image processing, segmentation, and sorting algorithms could be used as 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 many of them. Using McPhail traps, a WSN was developed to detect the olive fruit fly and medfly in the field [16]. WSNs are sensor networks that gather data and may be built to process 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.

Near Infrared Spectroscopy (NIR) was used to identify infested olives in harvested crops [17]. The Genetic

Algorithm (GA) extracts the features from the collected full spectral data. The retrieved features serve as the input for the classifier. Hyperspectral imaging was deployed to identify contaminated mangos [18]. 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. To detect contaminated cherries, Xing et al. used reflectance and transmittance spectra [19]. 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. [15] used optoacoustic spectrum analysis to construct an olive fruit fly detection system. The optoacoustic 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 optoacoustic 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. [25] 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. Regarding 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. [9] 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 [26] is used for detection and coarse counting. To increase the number of training images required by the YOLO deep learning model, the scientists considered the six species of flying insects as a single class. The authors augment the images with translation, rotation, flipping, scaling, noise addition, and contrast adjustment to extend the data set size. They also employed a pre-trained YOLO to fine-tune its parameters on an insect dataset. Support Vector Machine (SVM) is used for classification and fine counting, with global features. The technique was run on Raspberry PI, with detection and counting performed locally in each trap. The system attained a 92.5 percent average counting accuracy and a 90.18 percent average categorization accuracy. The Dacus Image Recognition Toolkit (DIRT) was created by Kalamatianos et al. [10]. The toolkit includes MATLAB code samples for fast experimentation, as well as a collection of annotated olive fruit fly photos acquired by McPhail traps. On 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 [8]. 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 pre-processed 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 [11]. 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 proving its notable performance in the work [27] in pest detection. Especially, the reported results of YOLO v5 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 [28] and later fine-tuned on a training dataset of 4480 sub-images made from 280 images of yellow sticky pheromone traps. However, YOLO v5 is considered slower than YOLOv4. For the AI implementation on edge devices, works in [29, 30] demonstrate the AI applications on edge devices pest monitoring as well. In [30], Lynfield-inspired trap was used with naled-and fipronil-intoxicated methyl eugenol [31] 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 [27], the application Single Shot Multibox Detector with variant backbones and YOLOv4-tiny show significant speed performance to YOLO v5, while taking the raw images as input instead of segmented sub-images. Nevertheless, the work also showed limitation of applying detection models on edge device due to the slow

processing speed, which will be further addressed in this article.

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 [32 - 35], 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 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 [31]. The objects can be further reassured by the object detection system before getting counted.

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 re-port 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 an 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, 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 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 electro-chemical energy storage with a capacity of 5 Ah and a voltage of 12 V. The Arduino in the operating system will check voltage of the battery with a voltage sensor to make sure the battery voltage is above a certain level

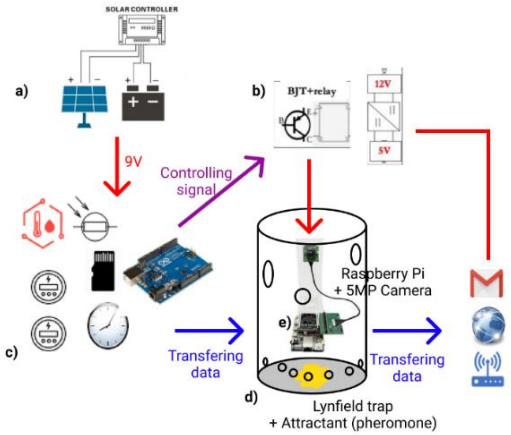

Figure 1. Overview of the trap system consisting of **a)** The solar panel system, **b)** The operation system, **c)** The sensor system, **d)** The modified Lynfield trap and **e)** The object detection system [30].

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 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. The

information from the sensors along with their corresponding time are stored in an 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

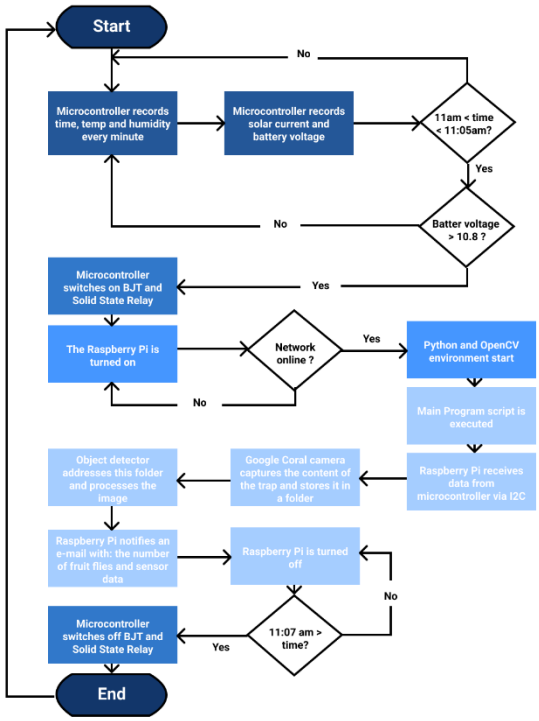

Figure 2. Flow chart of the trap system.

detection or not. The object detection system, shown in Figure 1e, 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 [31], which later helps to simplify the detection and classification problem. The Raspberry Pi module will receive data from the sensor system and send all data to the notification system to notify or alert farmers about the environmental 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. Moreover, the three models were selected because of their comparable parameter size and their feasibility for real-time implementation on edge devices. The pretrained models of these architectures were finetuned with our proposed insect dataset so that they can be used for the yellow fly detection application, as finetuning is also one of the common solutions for data scarcity problem in object detection. Because the models had been trained with COCO dataset [28], 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, finetuning 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 [36] by Wei Liu et al. and described as a one-stage object detection method that completely omits the region proposal and pixel/feature resampling 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 to produce a fixed-size collection of bounding boxes with corresponding shape offsets and confidence scores [36]. In addition, the early layers of the network are based on a standard image classification without classification layers, which is called the base network [36]. In this work, MobileNetV1 and MobileNetV2 are used as base networks for the SSD detection models. The elimination of region proposal and pixel/feature resampling 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 [37] 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 pointwise 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 hyperparameter to tune, and choose the size of the model.

3.3.3. MobileNetV2

The MobileNetV2 approach was first presented in [38] by M. Sandler et al. The approach is built based on MobileNetV1; therefore, it also makes use of the depth wise separable convolution architecture which consists of depth wise convolution layer and 1x1 pointwise convolution layer. In addition, the approach also utilizes linear bottleneck layers in convolutional blocks to optimize the neural architecture [38]. 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 [38].
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 [39]. 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 [40]. The YOLOv4 approach was created by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao [41]. 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 [42] 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. We also make the dataset available in [43].

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. An example of an original image and its synthetic versions are shown in Figure 3.

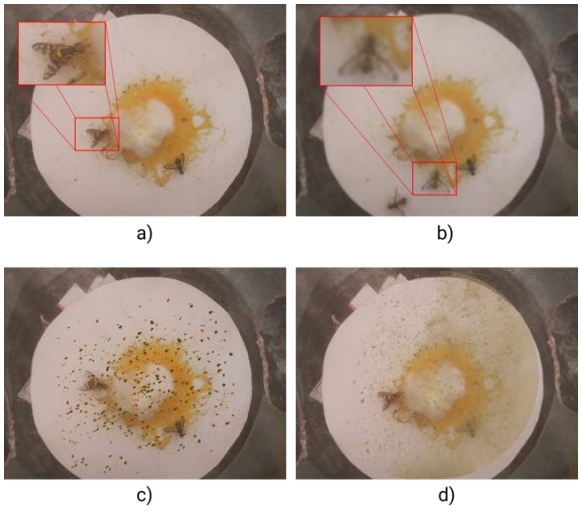

Figure 3. An example of an original image and its synthetic versions with **a)** original image, **b)** blurry filter **c)** adding salt-pepper disturbance, and **d)** adding dust disturbance.

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:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{All\ Detections} \quad (1)$$

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:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{All\ Ground\ Truths} \quad (2)$$

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:

$$F1 = 2\frac{Precision * Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (3)$$

4.3.4. Average Precision (AP)

By adjusting the confidence score threshold, the precision and recall will also be changed accordingly. 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 over-lapping 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.

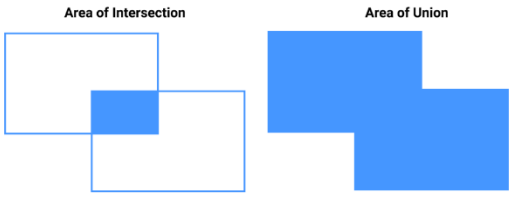

Figure 4. Area of Intersection (left) and Area of Union (right) examples.

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:

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \quad (4)$$

4.3.5. 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

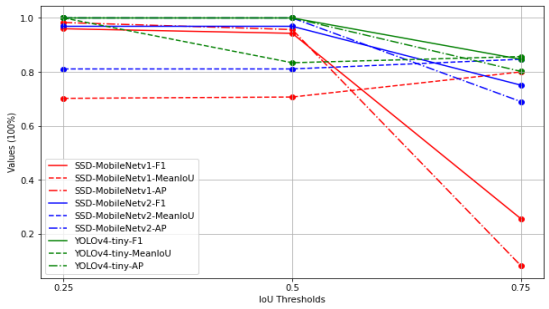

Figure 5. Evaluation results on Normal test set.

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.

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 a 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.

b) Blurry Testset

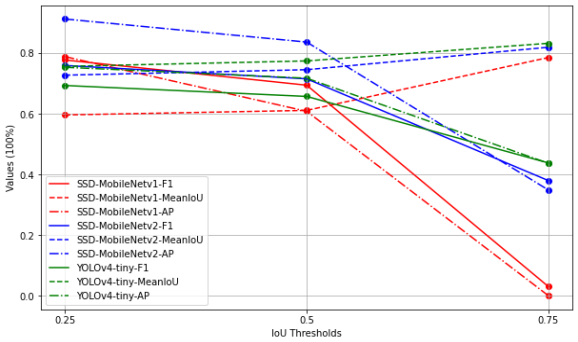

Figure 7. Evaluation results on Salt-pepper test set.

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

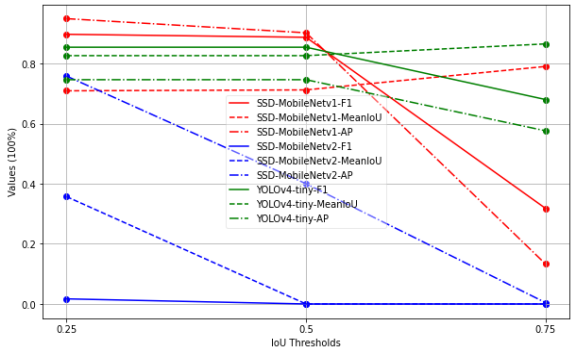

Figure 6. Evaluation results on Blurry test set.

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

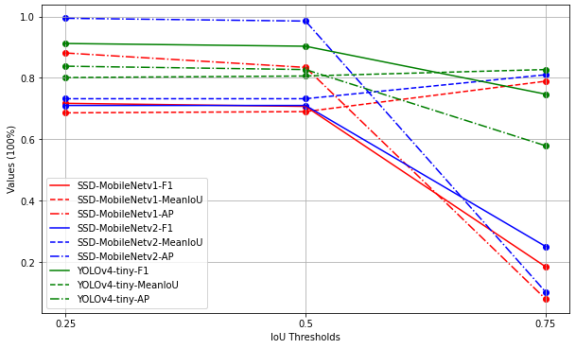

Figure 8. Evaluation results on Dust testset.

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 inference graphs run on the same device. In addition, we show 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, 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.

Table 1. Comparisons of the processing time of the trap object detection system between different types of processing units

Models	**CPU** Intel(R) Xeon(R) CPU @ 2.30GHz	**GPU** Tesla V100-SXM2-16GB	**TPU** Google TPU v2
SSD-MobileNetV1	3.414 FPS	29.140 FPS	1.025 FPS
SSD-MobileNetV2	3.485 FPS	21.000 FPS	1.024 FPS
SSD-MobileNetV1 TFLITE	_	_	8.743 FPS
SSD-MobileNetV2 TFLITE	_	_	10.058 FPS
YOLOv4-tiny	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 for 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.

Table 2. Overall assessment of the models, SSD-MobileNetV2 TFLITE and YOLOv4-tiny based on F1-Score and inference time.

Models	**F1-Score** (Normal Testset with 0.75 IoU threshold)	**Inference Time (**TPU Google TPU v2)
SSD-MobileNetV2 TFLITE	0.751	10.058 FPS
YOLOv4-tiny	0.847	0.545 FPS

Regarding the correctness, YOLOv4-tiny clearly outperforms the two SSD models over all 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 fly 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. 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 events in which the flies appear, our second choice, which is the SSD-MobileNetV2 model, should be considered. 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. The overall assessment table for YOLOv4-tiny and SSD-MobileNetV2 (TFLITE) is shown in Table 2 in terms of F1-Score and inference time.

4. Conclusion

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 in 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, with 0.847 F1-Score for IoU threshold 0.75, has the best performance among the three model candidates. Nevertheless, SSD-MobileNetV2 also has a comparable performance, 0.751 F1-Score for IoU threshold 0.75, to the YOLOv4-tiny model. Moreover, the SSD-MobileNetV2 model also outperforms the YOLOv4-tiny model in some test scenarios with synthetic disturbances. In addition, SSD models, especially SSD-MobileNetV2 model with 10.058 FPS on TPU, have a clear advantage over YOLOv4-tiny, with 0.545 FPS on TPU, 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.

Recommendations

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. From the test result with the synthetic test sets, the SSD family models were susceptible to disturbance and noise compared to the YOLOv4-tiny model. Our next attempt is also 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. Moreover, the performance of the detection models can also be enhanced with the implementation of wavelet analysis due to the preservation of detailed features. This has been proposed and tested in [44]. 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.

Funding Support

This research received no external funding.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest in this work.

Data Availability Statement

The data that support the findings of this study are openly available in AlertTrap-Dataset at https://github.com/a11to1n3/AlertTrap-Dataset?tab=readme-ov-file.

References

[1] Song, N. V., Phuong, N. T., Cuong, H. N., Diep, N. X., Diep, D. T., Huyen, V. N., Huyen, V. T., Tiep, N. C., & Trang, T. T. (2020). Vietnamese agriculture before and after opening economy. Modern Economy, 11(04), 894–907. https://doi.org/10.4236/me.2020.114067.

[2] Martinez, D. A., Loening, U. E., Graham, M. C., & Gathorne-Hardy, A. (2021). When the medicine feeds the problem; do nitrogen fertilisers and pesticides enhance the nutritional quality of crops for their pests and pathogens? Frontiers in Sustainable Food Systems, 5. doi:10.3389/fsufs.2021.701310.

[3] Liu, H., Wang, X., Chen, Z., & Lu, Y. (2022). Characterization of cold and heat tolerance of Bactrocera Tau (walker). Insects, 13(4), 329. doi:10.3390/insects13040329.

[4] Deguine, J.-P., Aubertot, J.-N., Flor, R. J., Lescourret, F., Wyckhuys, K. A. G., & Ratnadass, A. (2021). Integrated Pest Management: Good intentions, hard realities. A Review. Agronomy for Sustainable Development, 41(3). doi:10.1007/s13593-021-00689-w.

[5] Jacquet, F., Jeuffroy, M.-H., Jouan, J., Le Cadre, E., Litrico, I., Malausa, T., Reboud, X., & Huyghe, C. (2022). Pesticide-free agriculture as a new paradigm for Research. Agronomy for Sustainable Development, 42(1). https://doi.org/10.1007/s13593-021-00742-8.

[6] Dara, S. K. (2019). The new Integrated Pest Management Paradigm for the modern age. Journal of Integrated Pest Management, 10(1). doi:10.1093/jipm/pmz010.

[7] Tudi, M., Daniel Ruan, H., Wang, L., Lyu, J., Sadler, R., Connell, D., Chu, C., & Phung, D. T. (2021). Agriculture Development, pesticide application and its impact on the environment. International Journal of Environmental Research and Public Health, 18(3), 1112. https://doi.org/10.3390/ijerph18031112.

[8] Mamai, W., Maiga, H., Gárdos, M., Bán, P., Bimbilé Somda, N. S., Konczal, A., Wallner, T., Parker, A., Balestrino, F., Yamada, H., Gilles, J. R., & Bouyer, J. (2019). The efficiency of a new automated mosquito larval counter and its impact on larval survival. Scientific Reports, 9(1). https://doi.org/10.1038/s41598-019-43333-0.

[9] Zhong, Y., Gao, J., Lei, Q., & Zhou, Y. (2018). A vision-based counting and recognition system for flying insects in intelligent agriculture. Sensors, 18(5), 1489.

[10] Kalamatianos, R., Karydis, I., Doukakis, D., & Avlonitis, M. (2018). DIRT: The Dacus Image Recognition Toolkit. Journal of Imaging, 4(11), 129.

[11] Xia, D., Chen, P., Wang, B., Zhang, J., & Xie, C. (2018). Insect Detection and Classification Based on an Improved Convolutional Neural Network. Sensors, 18, 4169.

[12] Yasmin, R., Das, A., Rozario, L. J., & Islam, Md. E. (2023). Butterfly Detection and classification techniques: A Review. Intelligent Systems with Applications, 18, 200214. https://doi.org/10.1016/j.iswa.2023.200214.

[13] Doitsidis, L., Fouskitakis, G., Varikou, K., Rigakis, I., Chatzichristofis, S., Papafilippaki, A., & Birouraki, A. (2017). Remote monitoring of the Bactrocera oleae (Gmelin) (Diptera: Tephritidae) population using an automated McPhail trap. Computers and Electronics in Agriculture, 137, 69-78.

[14] Kumar, N., Nagarathna, & Flammini, F. (2023). Yolo-based light-weight deep learning models for insect detection system with field adaptation. Agriculture, 13(3), 741. https://doi.org/10.3390/agriculture13030741.

[15] Batz, P., Will, T., Thiel, S., Ziesche, T. M., & Joachim, C. (2023). From identification to forecasting: The potential of image recognition and artificial intelligence for aphid pest monitoring. Frontiers in Plant Science, 14. https://doi.org/10.3389/fpls.2023.1150748.

[16] Mamdouh, N., & Khattab, A. (2021). Yolo-based deep learning framework for olive fruit fly detection and counting. IEEE Access, 9, 84252–84262. https://doi.org/10.1109/access.2021.3088075.

[17] Casson, A., Beghi, R., Giovenzana, V., Fiorindo, I., Tugnolo, A., & Guidetti, R. (2020). Environmental advantages of visible and near infrared spectroscopy for the prediction of intact olive ripeness. Biosystems Engineering, 189, 1–10. https://doi.org/10.1016/j.biosystemseng.2019.11.003.

[18] Velásquez, C., Aleixos, N., Gomez-Sanchis, J., Cubero, S., Prieto, F., & Blasco, J. (2024). Enhancing anthracnose detection in Mango at early stages using hyperspectral imaging and machine learning. *Postharvest Biology and Technology*, *209*, 112732. https://doi.org/10.1016/j.postharvbio.2023.112732.

[19] Adedeji, A. A., Ekramirad, N., Rady, A., Hamidisepehr, A., Donohue, K. D., Villanueva, R. T., Parrish, C. A., & Li, M. (2020). Non-destructive technologies for detecting insect infestation in fruits and vegetables under postharvest conditions: A critical review. Foods, 9(7), 927. https://doi.org/10.3390/foods9070927.

[20] Diller, Y., Shamsian, A., Shaked, B., Altman, Y., Danziger, B.-C., Manrakhan, A., Serfontein, L., Bali, E., Wernicke, M., Egartner, A., Colacci, M., Sciarretta, A., Chechik, G., Alchanatis, V., Papadopoulos, N. T., & Nestel, D. (2022). A real-time remote surveillance system for fruit flies of economic importance: Sensitivity and image analysis. Journal of Pest Science, 96(2), 611–622. https://doi.org/10.1007/s10340-022-01528-x.

[21] Sciarretta, A., Tabilio, M. R., Amore, A., Colacci, M., Miranda, M., Nestel, D., Papadopoulos, N. T., & Trematerra, P. (2019). 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. https://doi.org/10.3390/agronomy9100608.

[22] Miranda, M. Á., Barceló, C., Valdés, F., Feliu, J. F., Nestel, D., Papadopoulos, N., Sciarretta, A., Ruiz, M., & Alorda, B. (2019). 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. https://doi.org/10.3390/agronomy9100620.

[23] Lello, F., Dida, M., Mkiramweni, M., Matiko, J., Akol, R., Nsabagwa, M., & Katumba, A. (2023). Fruit fly automatic detection and monitoring techniques: A Review. Smart Agricultural Technology, 5, 100294. https://doi.org/10.1016/j.atech.2023.100294.

[24] Wang, J., Chen, X., Hou, X., Zhou, L., Zhu, C., & Ji, L. (2016). 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, 1511-1528.

[25] Böckmann, E., et al. (2021). Rapid and low-cost insect detection for analyzing species trapped on yellow sticky traps. Scientific Reports, 11(1), 1-13.

[26] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[27] Yun, W., et al. (2022). Deep learning-based system development for black pine bast scale detection. Scientific Reports, 12(1), 1-10.

[28] Lin, T.-Y., et al. (2014). Microsoft COCO: Common Objects in Context. European Conference on Computer Vision. Springer, Cham.

[29] Nguyen, Q. M., Pham, D. A., Pham, D. T., Le, A. D., H. Vo, N. Q., & Vo, H. B. (2023). SmartTrap: An on-field insect monitoring system empowered by edge computing capabilities. *2023 RIVF International Conference on Computing and Communication Technologies (RIVF)*. https://doi.org/10.1109/rivf60135.2023.10471810.

[30] Pham, D., Le, A., Pham, D., & Vo, H. (2021). AlertTrap: On Designing an Edge-Computing Remote Insect Monitoring System. 2021 8th NAFOSTED Conference on Information and Computer Science (NICS) (NICS'21).

[31] Chen, P.-H., Wu, W.-J., & Hsu, J.-C. (2019). Detection of male oriental fruit fly (Diptera: Tephritidae) susceptibility to naled-and fipronil-intoxicated methyl eugenol. Journal of Economic Entomology, 112(1), 316-323.

[32] Wu, F., & Li, Y. (2023). Lightweight field insect recognition and classification model based on improved deep learning under complex background. *Security and Communication Networks*, *2023*, 1–9. https://doi.org/10.1155/2023/6560747.

[33] Xue, A., Li, F., & Xiong, Y. (2019). Automatic identification of butterfly species based on gray-level co-occurrence matrix features of image block. Journal of Shanghai Jiaotong University (Science), 24(2), 220–225. https://doi.org/10.1007/s12204-018-2013-y.

[34] Rajeena P. P., F., Orban, R., Vadivel, K. S., Subramanian, M., Muthusamy, S., Elminaam, D. S., Nabil, A., Abulaigh, L., Ahmadi, M., & Ali, M. A. (2022). A novel method for the classification of butterfly species using pre-trained CNN Models. Electronics, 11(13), 2016. https://doi.org/10.3390/electronics11132016.

[35] Almryad, A. S., & Kutucu, H. (2020). Automatic identification for field butterflies by Convolutional Neural Networks. Engineering Science and Technology, an International Journal, 23(1), 189–195. https://doi.org/10.1016/j.jestch.2020.01.006.

[36] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. Computer Vision – ECCV 2016, 21-37.

[37] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv, abs/1704.04861.

[38] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[39] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[40] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. ArXiv, abs/1804.02767.

[41] Bochkovskiy, A., Wang, C., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. ArXiv, abs/2004.10934.

[42] Jiang, Z., Zhao, L., Li, S., & Jia, Y. (2020). Real-time object detection method based on improved YOLOv4-tiny. ArXiv, abs/2011.04244.

[43] A11TO1N3/Alerttrap-Dataset: A dataset of fruitflies images captured by Alerttrap. (n.d.). Retrieved from https://github.com/a11to1n3/AlertTrap-Dataset?tab=readme-ov-file.

[44] Le, A. D., Jin, S., Bae, Y. S., & Nguyen, T. (2023). A novel learnable orthogonal wavelet unit neural network with perfection reconstruction constraint relaxation for image classification. 2023 IEEE International Conference on Visual Communications and Image Processing (VCIP). doi:10.1109/vcip59821.2023.10402772.