Detecting *Halyomorpha halys* Using a Low-Power Edge-based Monitoring System

Amin Kargar[a,c], Dimitrios Zorbas[b], Salvatore Tedesco[a], Michael Gaffney[c], Brendan O'Flynn[a]

[a] *Tyndall National Institute, University College Cork, Cork, Ireland*
[b] *Nazarbayev University, School of Engineering & Digital Sciences, Astana, Kazakhstan*
[c] *Horticulture Development Department, Teagasc Ashtown Food Research Centre, Dublin, Ireland*

Abstract

Smart monitoring systems in orchards can automate agriculture monitoring processes and provide useful information about the presence of insects, such as the Brown Marmorated Stink Bug (BMSB), that threaten the production quantity and quality of fruit such as pears. Unlike other approaches in the literature, we propose a low-cost image monitoring system which exhibits a very low power consumption without compromising much of the accuracy that existing expensive systems incorporating significant computing and processing capability can achieve in such applications. The proposed system relies on a microcontroller unit and a camera which can take pictures of a double-sided sticky insect trap which, with the help of novel machine learning algorithms, can report on the presence of BMSB via a long-range communication link. The Internet of Things data capture and analysis system has recently been deployed in a real orchard in Italy which is subject to BMSB infestation and the first images have been analyzed. This paper presents how the system works, the image processing, detection and classification algorithms, as well as a demonstration of the memory and energy consumption associated with the processing algorithms. The system achieves an accuracy of over 90% with multiple times less memory and energy consumption compared to other similar approaches in the literature.

Keywords: *Halyomorpha halys*, Edge Computing, Machine Learning, Deep Learning

1. Introduction

Based on forecasts by the United Nations Food and Agriculture Organization (FAO), up to 40% of global crop production is estimated to be lost to pests annually. Over $220 billion in economic losses are attributed to plant diseases, and at least $70 billion of these losses are attributed to the impact of invasive insects every year[1]. As such, based on the FAO estimation, invasive insects have a remarkable impact on the economy and threaten food security for millions of people. Moreover, the situation is expected to worsen as invasive species spread into countries where certain pests were absent previously. Events such as climate change, globalization and global trade will contribute to this spread as environmental conditions change around the world [1].

The Brown Marmorated Stink Bug (BMSB) or *Halyomorpha halys* (see Figure 1) is one example of an invasive insect. This insect pest is native to East Asia and can be found in China, Japan, Korea, and Taiwan. It is a highly polyphagous pest. Fruit trees, vegetables, and ornamental plants are among the plants that BMSB feed on [2]. It caused a damage of €588 million to fruit production in 2019 in northern Italy [3]. Several factors make the control of BMSB difficult, such as their high reproduction rate, mobility, tendency to feed on multiple types of plants, and the inefficacy of general insecticides against them [4]. Due to its high invasion and high production loss that it causes, this insect was selected as a research focus within the EU-funded research project named, HALY.ID, which aims to develop innovative technologies to monitor insects, specifically BMSB, in orchards in Europe where it is now present.

Generally, pesticides are used by growers to control the insect pest in their orchards, but they have adverse effects on human health and the environment [5]. The implementation of an integrated Pest Management strategy (IPM) aims to address the impact of invasive insect pests on crops in a human- and eco-friendly and sustainable way. This strategy suggests using multiple techniques to control pests including cultural, mechanical/physical, biological, and chemical control [6]. By implementing IPM, pesticide applications can be reduced significantly without affecting yield quality or quantity [5]. Monitoring is an important part of IPM strategies that specify the timing of insecticide applications, identification of pest species, and information on the size of the insect population. The monitoring data helps growers to take an informed decision such as to the correct insect control strategy to use and choose

Email addresses: `amin.kargar@tyndall.ie` (Amin Kargar), `dimitrios.zorbas@nu.edu.kz` (Dimitrios Zorbas), `salvatore.tedesco@tyndall.ie` (Salvatore Tedesco), `michael.gaffney@teagasc.ie` (Michael Gaffney), `brendan.oflynn@tyndall.ie` (Brendan O'Flynn)

[1] `https://www.fao.org/news/story/en/item/1402920/icode/`

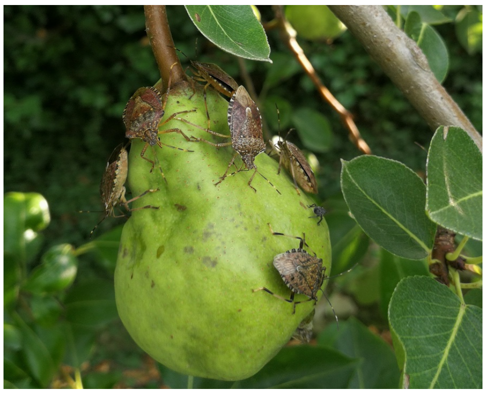


Figure 1: The Brown Marmorated Stink Bug or *Halyomorpha halys* on a pear fruit (Source: HALY.ID[3]).

the best possible controlling method accordingly [4]. In traditional insect pest monitoring, farmers must periodically visit their farms and visually analyse and count insect species manually. This can be problematic for large-scale orchards due to the amount of time, effort, and manpower cost it requires [7]. In addition, sometimes visually identifying insect species with the human eye can be difficult because of their high visual similarity [8]. These factors can increase the error in the establishment of the estimated insect population and lead to incorrect use of pesticides. Therefore, devoting significant time and resources to this procedure may not always result in accurate results. Automation of the identification and counting task would make this estimation more accurate and more efficient. The automation of insect monitoring has widely been studied in recent years, using image processing and machine vision [9]. Such systems employ camera sensors in orchards capable of taking images of crops or plants. The generated images are then analysed for insect detection and classification.

However, there are several parameters in the development of automation processes that should be considered during the design of such a system as listed below:

- Power consumption: Orchards generally have no power lines, hence the monitoring system should be low-power and last for extended periods without the need for battery maintenance.
- Accuracy: The system should be accurate enough so that farmers can trust its information and make appropriate decisions accordingly.
- Cost of purchase and maintenance: The cost of such a system should be as low as possible. If such a system has a high cost, farmers are not likely to use it since many of them should most likely be required to be deployed in orchards. Moreover, the maintenance cost should also be as low as possible by using commercially available consumables (e.g., traps).
- Ease of deployment and maintenance: The system must not require any prior knowledge of electronics or information technology processes for the farmers.
- Communication of the results: There may be a lack of reliable internet connectivity and communication infrastructure since orchards are outside of cities in remote areas.

To this extent, the study described in this publication tackles the limitations of pest monitoring systems and in particular energy consumption and cost. The main contributions of this study are summarized as follows:

- A low-power consumption and low-cost vision based smart insect monitoring system to detect and count insects in orchards, specifically the Brown Marmorated Stink Bug, is proposed.
- A light-weight and low-power deep-learning (DL) model for image classification to be implemented on resource-constrained microcontrollers (MCU) with a memory capacity in the range of kilobytes (KB) is proposed, evaluated, and compared to other solutions in the literature.
- A power characterization of the proposed system is performed to confirm its low-power operation and the power efficiency of the proposed ML approach.

2. Related Work

Camera-based insect monitoring systems use a camera to capture images and a processor for analysis to identify and count the relevant insect population. In recent years, several studies employ a local system (i.e., edge devices) or a system running on the cloud for image analysis. For example, Guo et al. [10] proposed an automatic system to monitor flying vegetable insects with an RGB camera and a YOLO-based detection algorithm [11] named YOLO-SIP. The system periodically captures images and sends them to a cloud server over a wireless network for insect identification and classification. Similarly, other studies [12, 13] used similar image acquisition and detection systems. Most cloud or remote server-based systems aimed to increase insect detection accuracy, and they were not concerned about computational power, memory usage and power consumption. In such systems, since the images need to be transferred to the cloud, the system must be equipped with a powerful, high-bandwidth, communication system, which directly affects the system cost and energy consumption which can cause issues in remote deployments.

Several other recent studies have been conducted using edge devices. In such systems, the objective is to perform the image analysis task on the monitoring device itself. However, edge devices typically have a considerably limited resource budget compared to cloud or server-based

[3]`https://www.haly-id.eu/`

systems. Li et al. [14] developed a few-shot recognition method to identify cotton pests with an accuracy of over 95% accuracy. They benefited from Convolutional Neural Network (CNN) and their system consisted of an FPGA for computation and an ARM to control the program.

An ultra-low power smart pest detection system was proposed in [15]. The proposed system was based on GAP8 [16]. Their device worked only with low-resolution grayscale images (244x324 pixels) and extracted some potential area of the captured image and utilised a Machine Learning (ML) model for classification. This system needed 3.5mJ of energy to perform the image processing and the classification and achieved an accuracy of 93%.

Saradopoulos et al. [17] proposed an edge- and camera-based system for insect counting in the context of smart cities. The performance of three different devices including a Raspberry Pi (RPi), an Espressif ESP32, and a Google Coral were compared. ESP32 was suggested as the best module for their application with an inference time of 51 seconds and power consumption of 6mA in deep sleep mode. Moreover, the quantization technique was used to reduce the model size to 0.5MB and fit it into the device memory. This study used regression to count the insects and achieved an accuracy of 95%, thus species classification was ignored.

Brunelli et al [18, 19] developed an automated pest detection edge device that used Deep Neural Network (DNN) to detect Codling Moths in a pheromone-based trap. Their system consisted of an RPi for image capturing and processing. The power supply consisted of a battery and a solar panel to recharge the battery. The system detection pipeline included regions of interest (RoI) extraction and RoIs classification. For the classification part, three different DNN models including LeNet, VGG16 and MobileNetV2 were trained, and gained an accuracy of over 95%. In terms of power consumption, several configurations were analysed, and a system based on RPi3 and LeNet consumed the lowest amount of energy at 123.2J.

Finally, Sütő [20] proposed an embedded insect monitoring system using an OpenMV Cam board and the target insect was the Codling Moth. The algorithm used a selective search method for object proposals, and the MobileNetV2 was used as a base model for the classification part. The author also proposed a system based on an RPi and a camera [21]. The author used the same algorithm for insect identification and counting as with the previous work and the system required approximately 400 seconds for one cycle operation with an average power consumption of over 2.1W.

In the mentioned works a variety of different types of hardware were used for insect monitoring applications. Table 1 compares the most common hardware used in such applications. Based on the table, it is clear that microcontroller units (MCUs) have a much lower cost and power consumption than other solutions, thus, they can be considered the best option for insect monitoring applications deployed in remote areas. However, it is important to establish whether their severely limited processing resources are sufficient to provide reliable results for the detection of BMSB.

This work describes an optimised algorithm for insect detection and identification designed to be used on a resource-constrained platform. For this aim, the image analysis algorithm was separated into two functions that run separately. By doing so, each part can independently use the whole available memory for its computations. This network model jointly improved hardware metrics by decreasing memory usage, model size, model complexity, and computation time without considerably compromising accuracy.

3. Proposed System

The proposed system works in repeated cycles, where in each cycle, a picture of a double-sided trap is taken, the image analysis carried out using DNNs and relevant results transmitted over a communication link. The device then enters a long deep sleep mode between each processing cycle. In this section, details regarding the proposed system processing implementation, the techniques used, as well as the deep learning model are described.

3.1. System Components & Processes

Figure 2a illustrates the image acquisition and data processing system first introduced, and described in detail, in our preliminary work [22]. This automated insect monitoring tool was developed and just recently deployed in an Italian orchard for a trial use (see Figure 2b). The design specifications of the system can be found in [22], while here, a brief overview of its key components and functionalities is provided.

The system consists of a double-sided trap[4], and a pheromone is applied to attract BMSBs. A white colour trap was employed because it provides the highest contrast with mottled brownish grey BMSBs and improves the detection accuracy. Furthermore, the pheromone is unique and attracts only BMSB, which reduces the probability of other insects being trapped.

Components within the system include an OpenMV[5] board which is an MCU based board with an integrated camera, an RGB LED to provide illumination and enable the system to capture images at night, and a servo motor to rotate the trap to capture images from both sides of the trap. In addition, as electrical supply cannot be assumed to be available in remote orchards, the system uses batteries, which can be recharged using an integrated solar panel. In this version of the device, the data is transferred via an LTE (4G) connection.

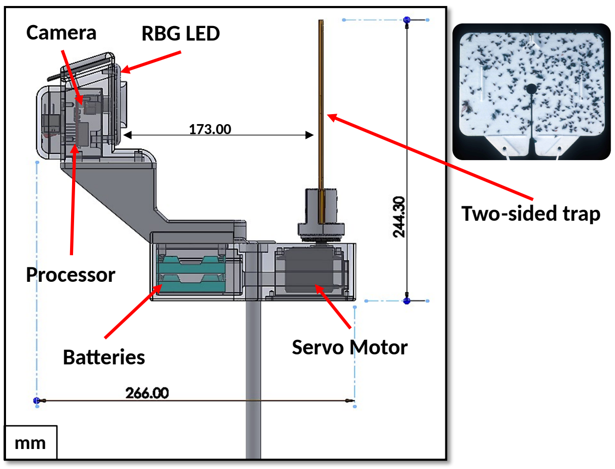
[4]`https://www.andermattuk.com/rebell-white-trap`

[5]`https://openmv.io/collections/products/products/openmv-cam-h7-plus`

Table 1: Examples of hardware used in smart insect monitoring systems in the literature.

Platform	**Example**	**Memory Capacity (MB)**	**Storage Capacity (GB)**	**Max Power Consumption*** **(W)**	**Price**** **($)**
Cloud/Server	GPU Nvidia V100	≥16384	≥1024	250	≥5K
FPGA	Artix-7	≥1024	≥8	≥1	≥500
Mobile Phone	iPhone 11	≥4096	≥64	8	≥500
Raspberry Pi	RPi 4B	≥1024	≥8	5	≥60
Microcontroller (MCUs)	STM32H7	≤1	≤0.002	0.3	30

*Power consumption is approximate as it depends on several factors.
**Prices are approximate due to the global chip shortage issue at the time of this report.



(a) System diagram

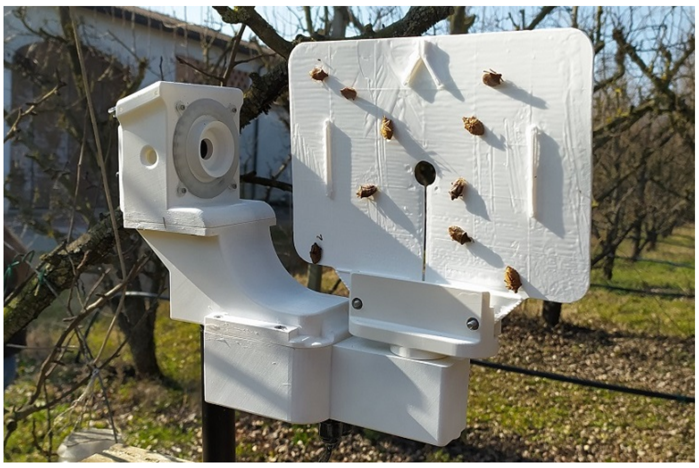

(b) Deployed device

Figure 2: System overview and components as described in [22].

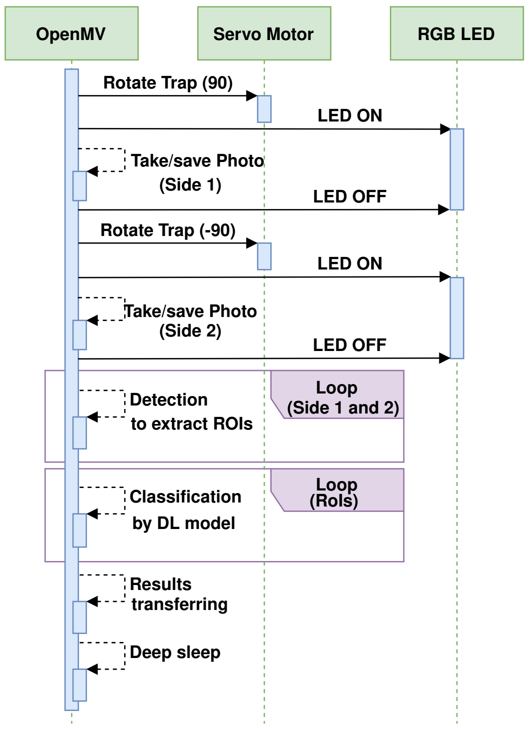


Figure 3: Diagram of steps followed per cycle.

Figure 3 shows the system timing diagram for an image capture and processing cycle. In order to keep track of the insect population trends during the growing season, the system goes into the active mode for the insect detection process twice in a period of 24 hours. This happens one hour after the sunset and one hour before the sunrise. After that, it goes into the deep sleep mode for the rest of the day to decrease energy consumption. Since sunset and sunrise vary during the year, they are calculated based on the location and the time zone of the deployment. The system operation is divided into five main stages in each operational cycle as follows:

Capture of images: The system takes images from both sides of the trap using the integrated RGB camera. As mentioned earlier in the text, images are captured during nighttime hours, one hour before sunrise and one hour after the subset using the LED light. By doing so, the lighting condition is controlled and remains consistent in all images which is important for the object detection and classification processes.

The next step is the image analysis phase which consists of two sub-steps including detection and classification, shown in Figure 4. These two sub-steps are explained in the following paragraphs:

Processing and detection: In this step, the system executes an image processing algorithm on the two captured images to detect and extract the potential Regions of Interest (RoIs); An RoI is an area where the target insect is likely to be present. To this purpose, as shown in Figure 4, the algorithm converts the RBG image to grayscale. Then, a smoothing filter is used to reduce some details, such as insect legs or antennas (step 1). In step 2, Otsu's method is used to find an optimal threshold from the image histogram to convert the image to a black-and-white image and separate the foreground (insects) and background (trap surface). The last stage of the process is the use of a blob detector to locate the potential areas where the target insect may be present (step 3). The blob detector detects blobs based on primary criteria, called size thresholds. Blobs are filtered out if their bounding box parameters are lower or higher than these thresholds. These thresholds are defined and adjusted by the minimum and maximum dimensions of the target insect size, which typically ranges from approximately 12 to 17 mm in length [23]. Since the distance between the camera and the trap is fixed, these thresholds are valid and reliable. The detected areas (RoIs) are cropped from the original image for further analysis and are sized 150x150 pixels (step 4). Details of this process are available in [22].

Deep learning classification: In step 5, the extracted RoIs are fed into a deep learning model for classification. Based on the model result, the cropped images are labelled as 'Yes' or 'No' indicating whether the image belongs to a BMSB or not (step 6). The critical point for this phase is that the model should be optimised, be small in size, and be sufficiently efficient to run and fit in resource-constrained MCUs.

Results transfer: In this phase, the results are transferred to the cloud for further decision-making. To reduce power consumption associated with communications, instead of sending images of the whole trap, the system only transfers small (in terms of data size) results as regards the presence of the relevant insect species. The JPEG compression method with a quality of 90% is used to compress the images which results in a typical image size of a few KB ($\leq$4 KB).

Sleep mode: Finally, the system goes to deep sleep mode to reduce power consumption until the next scheduled image acquisition round needs to take place.

3.2. Datasets

As mentioned in the previous section, the image analysis process is divided into two separate functions including detection and classification. In the detection phase, the system analyzes images sized at 1600x1200 pixels to extract RoIs, while in the classification phase, the system analyses 150x150 pixels images to recognize BMSBs. Thus, two separate datasets were created for these two phases, the first one was used to evaluate the performance of the detection phase and the second one was used for the classification phase. For this aim and because the system was deployed in the orchard only recently, dead BMSB specimens were used and glued on the surface of the trap to create the first dataset of insect images in the laboratory. In addition, some images of real BMSB captured by the deployed system were added to this dataset named Trap_Dataset. This dataset contains approximately 40

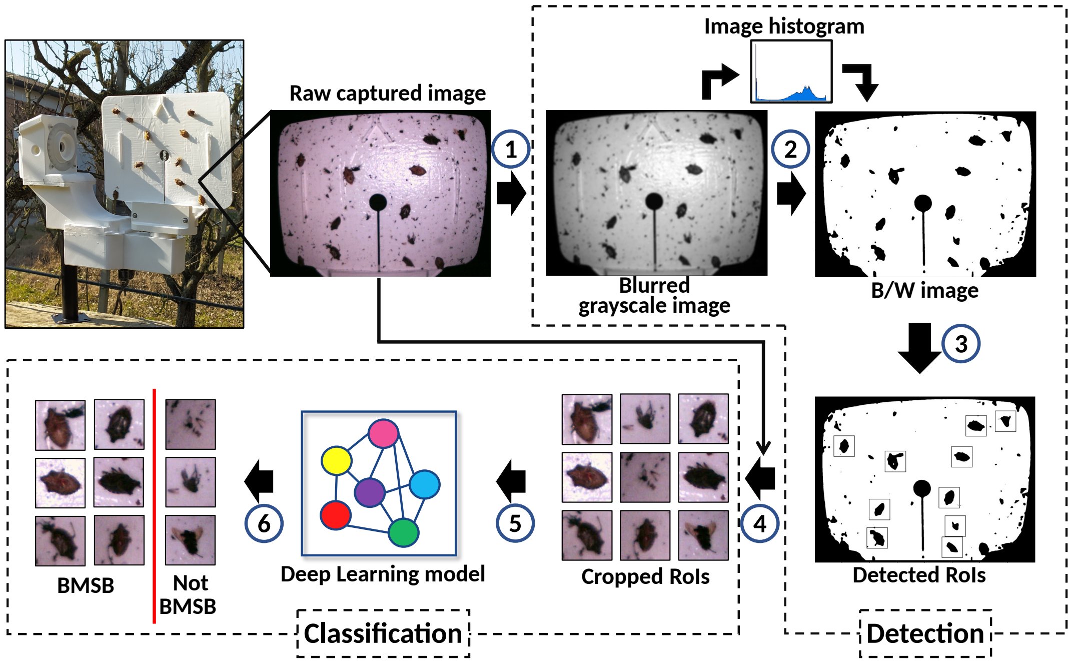


Figure 4: Image processing algorithm, from the raw captured image to classified RoIs.

laboratory-captured images and 70 images from the deployed system in the orchard, all captured at a resolution of 2MP (1600x1200) by our system.

The second dataset (BMSB_Dataset) was created by using images of insects gathered from the deployed system, available on the internet, and cropped BMSB images from Trap_Dataset. Augmentation methods were also employed to increase the number of images. Image augmentation applies some predefined transportation on original images to create new ones. In this study, rotation, horizontal flipping, and vertical flipping were used for augmentation. To this end, the dataset was first divided into train, validation, and test sets which 75%, 12.5%, and 12.5% of the total data is assigned to each set respectively. Then, augmentation was applied to the train set to decrease the model overfitting and increase the generalization of the model by increasing samples in the train set.

3.3. Classification Model

The goal of the classification model is to recognize whether the cropped image contains a BMSB or not. The proposed deep learning model is based on CNN which is specifically designed for image classification. The main part of the CNN model is the convolutional layers which are responsible for recognising patterns, edges, and textures as well as for extracting this information from the input image. These layers scan an input image using small kernels (also called filters) and create feature maps in which each kernel extracts specific features and generates a corresponding feature map. Then, an activation function is applied to the feature maps to introduce nonlinearity into the network, and thus, allows the approach to recognize complex patterns and relations in data. In

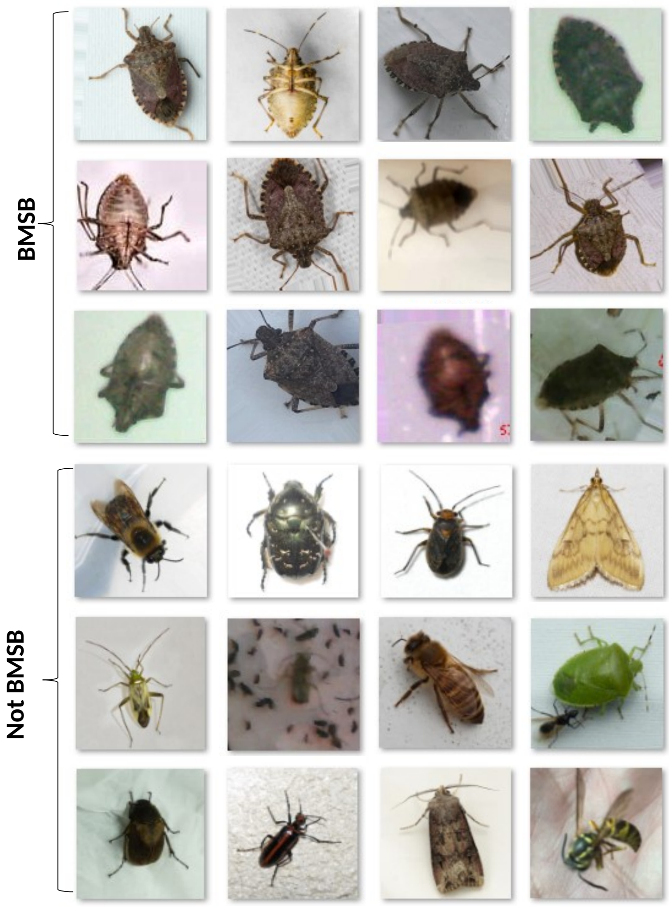


Figure 5: Dataset samples.

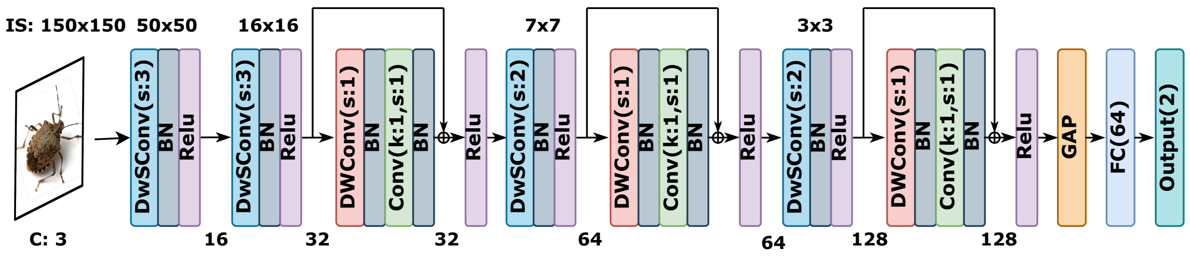


Figure 6: Proposed classification model; IS: Input Size, C: Number of Channels, DwSConv: Depthwise Separable Convolution, Conv: Convolution, BN: Batch Normalization, GAP: Global Average Pooling, FC: Fully Connected.

the final stage – which is known as the classifier part – the extracted feature maps are combined and analyzed to make a decision on the content of the input image.

In this section, we explain how the DNN model was created to meet criteria of being small in size and low in complexity so that it can run on resource-constrained MCUs. For this goal, Depthwise Separable Convolution (DwSConv) was employed instead of Standard Convolution (SConv) which is generally used to capture and extract important features and patterns from an input image. DwSConvan is an effective method for reducing computations and parameters while maintaining a high accuracy level [24]. DwSConv factorizes a Standard Convolution into two steps; a depthwise convolution followed by a pointwise convolution. This significantly decreases the amount of computation as pointed out in [24], for example; a DwSConv with a kernel size of 3 decreases the computational cost by around 9 times.

Moreover, skip connections were used to improve the proposed model performance by allowing data to flow easily from input to output and combining the features from different layers and, also, to help reduce the problem of vanishing gradients during network training [25].

The suggested architecture is depicted in Figure 6. As mentioned, DwSConv was used instead of SConv to extract features from an input image. In the suggested architecture, four DwSConv (with a kernel size of 3 and a stride of 3 or 2) were used to extract features and reduce the input dimension. A residual/skip block was used after each DwSConv, except for the first one. This block is a DwSConv that adds its input and output before the activation function (ReLU). Therefore, in this architecture, the dimension size decreases from 150x150 in input to 3x3 after the last convolution layer. The channel number correspondingly increases from 16 in the first DwSConv to 128 in the last one. For the classifier part, a Global Average Pooling (GAP) layer was used followed by a Fully Connected (FC) layer. This is a small and lightweight CNN-based model that has around 46,000 parameters.

In order to deploy the trained model on MCUs with extremely limited flash storage, it was quantized to 8-bit integers in the range of [-128, 127]. Quantization reduces the model size, however, it may cause a slight reduction in the accuracy compared to the float data type model which uses 32 bits of precision. Based on our experiments, the difference between the accuracy of quantized and unquantized option was approximately 0.2%, thus, the accuracy of the quantized model of our proposed network remained the same as with the unquantized model, while its model size decreased by over two times.

4. Evaluation & Discussion of the Results

This section describes how the experiments were carried out, presents the evaluation results, and compares them to other similar approaches in the literature. In order to create a comprehensive evaluation, several important aspects relevant to edge-based systems were investigated.

To evaluate the power consumption performance of the approaches, we considered the analysis of six BMSBs on each side of the trap and powered the device with Otii Arc Pro power analyser[6]. The power consumption of the system was measured at different phases of operation using the power analyser. Besides, several CNN-based architectures such as LeNet [26], MobileNetV1 [24], MobileNetV2 [27], SqueezeNet [28], and VGG16 [29] have been trained on our BMSB_Dataset and used to compare the performance of the different approaches. MobileNetV1, MobileNetV2 and SqueezeNet are lightweight network models that are widely used in mobile and edge-based systems. Besides these models, LeNet (as one of the simplest network models) and VGG16 (as a bigger and more complex model) were also trained to evaluate their performances and the possibility of using them on MCUs.

The proposed classification model was implemented in the TensorFlow[7] framework using Keras[8]. As there are two classes ('Yes' and 'No'), binary cross entropy was used as a loss function. To reach the final optimized model, hyperparameters were tuned by varying epochs, batch size, and learning rate. Therefore, the Adam algorithm was used as an optimizer with a learning rate of 0.005, and the epochs and batch size were set to 200 and 64, respectively. ReLU6 was used as an activation function, and the dataset was divided into 80% and 20% for training and test sets. In addition, dropout and spatial dropout techniques were used to prevent overfitting during model training.

[6]https://www.qoitech.com/otii-arc-pro/
[7]https://www.tensorflow.org/
[8]https://keras.io/

4.1. Evaluation Metrics

The accuracy of detection and classification results were evaluated separately and based on different metrics. For the detection phase, the accuracy was calculated by averaging the ratios of the correct number of detected RoIs to the actual number of RoIs on each image as follows:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{d_i}{n_i}, \qquad (1)$$

where N is the number of images, d_i is the number of correct detected RoIs of the i-th image and n_i is the number of actual RoIs of the i-th image.

For the classification phase, four different metrics including accuracy, precision, recall, and F1-score were used. The accuracy represents the correct predictions over the total number of predictions. Precision indicates the proportion of correct positive prediction and the total positive prediction, while recall indicates the proportion of correct positive prediction and actual positive. The F1-score shows the balance between precision and recall metrics for a model [30]. These are metrics that are generally used for binary classification models.

In addition to accuracy, the performance of the proposed classification model in terms of computation metrics including model complexity, size, and peak memory usage was also assessed. The model complexity was evaluated by the number of MACs (Multiply-Accumulate) [31]. MACs represent the number of multiplications and additions performed during one inference. Moreover, the total device energy consumption and run time of one cycle were measured and evaluated to have a comprehensive assessment.

4.2. Accuracy & Model size

The detection algorithm achieved an accuracy of 93.8% on the OpenMV platform. It was observed that most of the detection errors occurred in the areas located at the edges of the image or trap. The algorithm can detect the black objects that are surrounded by the white colour (background). These objects could be any object with the same size as our target insects. For example, it could be any insect species, such as a bee or a fly, or even a small leaf. When the trap image is converted to a black and white image (see the black and white image in Figure 4 step 2), insects located at the edges are mixed with the black area around the trap surface, thus the algorithm is not able to recognize them as a separate object on the trap surface since they are not surrounded by white colour. It should also be noted that in high-density insect cases (when the number of trapped insects is high), as the probability of insects laying next to or over each other increases, the error might increase as well. The reason is that overlapping can increase the size of the detected blob and it might get filtered out due to its size. Thus, it is recommended to replace the trap weekly to decrease the chance of overlapping.

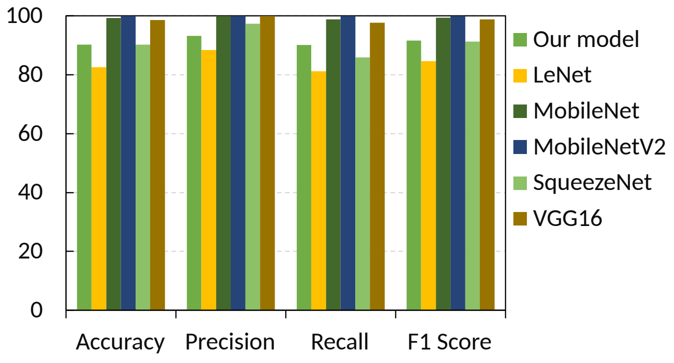


Figure 7: Comparison of the proposed system with different classification models.

Table 2: Performance of the proposed classification model and comparison with other models.

DL Models	Model size (MB)	Parameters (M)	Inference Time (s)
Our model (DwSConv)	0.08	0.046	0.07
Our model (SConv)	0.3	0.3	0.1
LeNet	2.3	2.4	0.25
MobileNet	3.4	3.2	1.41
MobileNetV2	2.6	2.3	1.17
SqueezeNet	0.8	0.8	1.88
VGG16	14.5	14.7	48.22

Figure 7 illustrates the performance of the classification models in terms of accuracy, precision, recall, and model size. The accuracy of our DwSConv model was 90.2% and a 95% confidence interval (CI) of [89.3,91.1], the precision was 93.2% (95% CI, 92.4-94.1), the recall was 90.2% (95% CI, 88.9-91.4), and the F1-score was 91.6%(95% CI, 90.8-92.4). In comparison with the SConv model, only provided here as a reference, the DwSConv-based model had better performance and raised the accuracy by 2%. Moreover, in comparison with LeNet, the proposed model achieved better accuracy, however, it was lower than the other more complex models including MobileNet, MobileNetV2, and VGG16. This was an expected result because the proposed model is designed to be as light-weight as possible.

However, as we can see in Table 2, our model had 0.046 million parameters and its size was only 80 KB which is much smaller than the other models size, over 10 times less than SqueezeNet and approximately 30 times less when compared to other models. This is important as the amount of flash storage in typical MCUs is less than 1MB. This means that our model can easily fit in memory-constrained MCUs without any significant loss of accuracy, which is critical for edge-based systems. Moreover, the proposed architecture achieved considerably low inference time compared to other models. The proposed DwSConv model needed 0.07 s for each prediction. This is 3.6 times faster than LeNet, 20 times faster than MobileNet, 16.7 times

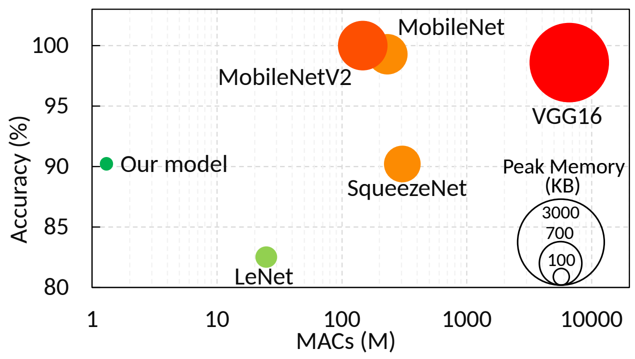


Figure 8: Comparison of accuracy, complexity (MACs) and peak memory usage; bubbles size represents the peak memory usage, and its colour indicates the possibility of running the corresponding model on typical MCUs.

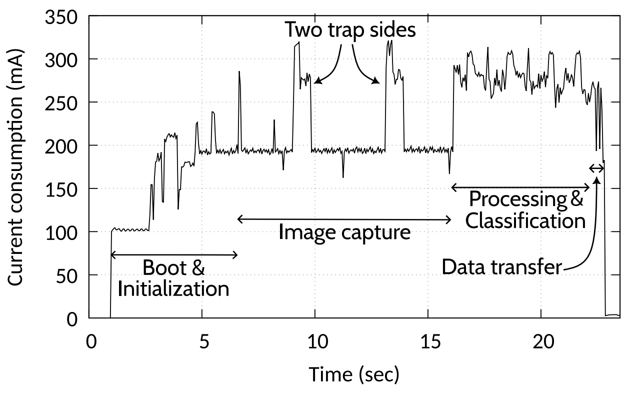


Figure 9: Proposed system's current consumption over different phases of a cycle.

faster than MobileNetV2, 27 times faster than SqueezeNet, and approximately 688 times faster than VGG16. However, the inference time is not a critical factor for insect monitoring applications, but it has a direct impact on the energy consumption. Since computational resources are used for a longer period of time, increasing inference time directly correlates with an increase in power consumption as it is further explained in Section 4.4.

4.3. Memory usage & Complexity

Memory usage is a key factor when a classification model is supposed to be implemented on a resource constrained MCU. The peak memory usage of different models is shown in Figure 8 with the size of the bubbles representing the memory usage in conjunction with levels of accuracy associated with the different models as well as their complexity. The peak memory usage of the proposed model was 75 KB which was 2.7, 9.7, 14.8, 8.4 and 38 times less than LeNet, MobileNet, MobileNetV2, SqueezeNet and VGG16, respectively. Therefore, the proposed model can easily run on general-purpose MCUs with less than 512 KB of RAM. This is not possible for the other models with the exception of LeNet.

The same figure can also reveal the relative complexity of the different approaches. The model complexity was evaluated based on the number of MACs needed for one inference. As Figure 8 illustrates, the proposed model had 1.3 million MACs in one inference while this factor for other models was much greater, being 24.8 M for LeNet, 231 M for MobileNet, 146.6 M for MobileNetV2, 305 M for SqueezeNet, and over 6 billion MACs for VGG16. The impact of this factor is reflected to the inference time and consequently to the energy consumption. The results clearly indicate that the proposed DwSconv architecture has a reasonable balance between accuracy, MACs, and peak memory usage when it is supposed to run on resource-constrained MCUs.

4.4. Power Consumption & Run time

The power consumption of the different approaches is another critical factor which affects both the deployment feasibility and the cost of such a system as the one described. Keeping the system in low power sleep mode for as long as possible between image acquisition cycles will significantly reduce the power consumption, but low power consumption algorithms are essential to enable scenarios that require more frequent sampling duty cycles.

The OpenMV current consumption over a single cycle is illustrated in Figure 9. The voltage was set to 5V. Since the camera should capture two images (from both sides of the trap), there are two peaks in the power profile associated with image capture. Also, there are two peaks associated with image analysis for each of the images, for the two sides respectively. In addition, the current consumption during the classification component reaches the highest value, thus reducing the classification time significantly reduces the power consumption. Moreover, the system current consumption was about 7mA in the deep sleep mode which is almost 30 times lower compared to the active mode (see Figure 9). The impact on the battery lifetime is significant.

A comparison of the energy consumption of the proposed system and another edge-based pest detection system [19] for each of the five operations, mentioned in Section 3.1, is depicted in Figure 10.In the work reported by Albanese et al., the authors utilized a RPi for image capture, processing and classification. Another difference between their system and ours is that this system only takes and analyses one image, while our system works on a two-sided trap, so it takes and analyses two images in one run time. Based on this fact, our system consumed more energy in capturing mode as the trap needs to rotate two times during this task, taking 9.75 seconds in total. Therefore, in capturing function, our system consumed 12.09J (of which 2.7J was consumed by the servomotor and LED) while Albanese et al's device consumed 1.67J. Overall, our system consumed less energy in one processing cycle, particularly in the boot and classification part. Our device

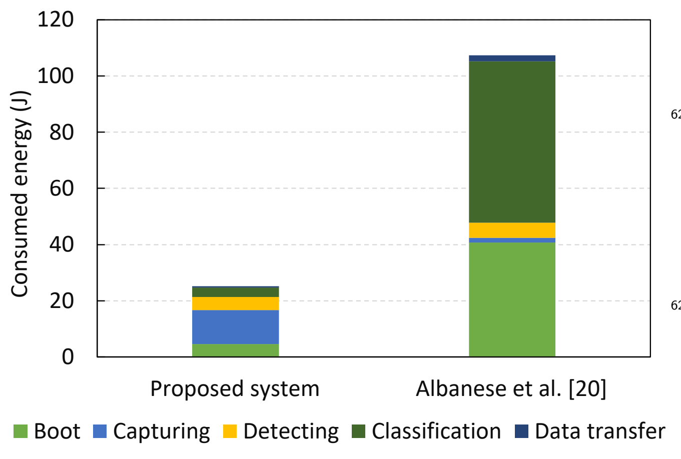


Figure 10: A breakdown energy consumption of the proposed system and [19].

Table 3: Energy consumption for different models run on OpenMV.

DL Models	Total runtime[†] (s)	Total energy (J)	CL time[*] (s)	CL energy[*] (J)
Our model (DwSConv)	16.37	17.9	2.52	3.24
LeNet	18.34	24	4.95	6.26
MobileNet	31.5	44.4	18.5	26.8
MobileNetV2	30.3	42.4	17.2	24.6
SqueezeNet	36.2	50.6	22.5	34.5
VGG16	618.54	932	605.4	914.07

[*]Classification (CL) time/energy consumption for 12 images
[†]The booting phase is not considered in this table

needed 4.63J to boot while Albanese et al's device needed 40.7J. In addition, in the classification function, our system consumed just 3.24J while Albanese et al's system needed 57.4J, meaning that around 18 times more energy was expended by their system. Finally, in data transfer mode, 0.6J and 2.15J was consumed by the proposed system (wired serial connection) and Albanese et al's device which used LoRa, respectively.

As previously mentioned, the classification component's energy consumption is dependent on the network model used. Noted that the classification time depends on the inference time meaning that a lower inference time leads to a lower classification time as well. In this regard, the classification time and energy consumption for different models on the OpenMV system are reported in Table 3. As can be observed, the system using the DwSConv model consumed only 17.9J. This is 1.3, 2.5, 2.4, 2.8 and 52 times less than LeNet, MobileNet, MobileNetV2, SqueezeNet and VGG16, respectively. As compared to the total energy consumption, the classification component corresponds to only 18.1% of it when DwSConv was used. Remarkably, by increasing the complexity and, consequently, the inference time (see Table 2), the contribution of the classification in the energy consumption increases significantly for the other models. For example, when MobileNet was used, more than 50% of the energy consumption was dedicated to classification. As a result, the classification model plays a key role in such a system's energy consumption.

4.5. System Lifetime

System lifetime is a critical factor in insect monitoring systems since they should last for at least a growing season when they are deployed in orchards. In this version of the device, two packs of four series-connected batteries (AA NiMH rechargeable battery, 2.45Ah, 1.2V) are connected in parallel so the battery bank voltage and capacity are 4.8V and 4.9Ah, respectively.

To calculate the battery lifetime, the energy consumption of the device in one day should be computed. As mentioned earlier, the system goes into active mode twice a day and it remains in sleep mode for the rest of the day. In each of the active mode periods, it is assumed that the system works for 30 seconds during which it consumes an average of 300mA (see Figure 9). Moreover, the system consumes 7mA in sleep mode. Hence, every 24 hours, the system consumes 300mA for 60 seconds and 7mA for the remainder of the day. According to these values, the total consumption for one day is about 0.83Wh, and as the battery capacity is 23.52Wh (4.9Ah · 4.8V), the system could continuously work for over 28 days without any charging activity.

In addition, the system benefits from a solar panel to charge the battery bank. The typical charging voltage of the solar panel is 5.5V, the current is 100mA, and the conversion rate is up to 17%. Thus, based on these values, the charging time needed to restore the battery to its full capacity is around 8.9 hours considering that 0.83Wh is consumed per day. However, this is a theoretical estimation and it is expected to be higher in cloudy weather conditions. But based on the fact that the day is much longer than the estimated charging time, the battery bank is capable of being fully charged within a sunny day meaning that the system can work on the battery for a long time. It is worth mentioning that by using bigger solar panels, the charging time can be decreased. For example, the charging time can be decreased to 4.5 hours by using two solar panels (the one used in this study) in parallel.

5. Conclusions & Future work

In this paper, a novel low-cost and low-power monitoring system was investigated for the detection of *Halyomorpha halys*, an invasive insect which causes a lot of damage to fruit. The system employs light but efficient image processing and deep learning mechanism which can run on very low-cost microcontrollers. A series of experiments were run on a blend of real and artificial datasets. The experimental results confirmed that it is feasible to run complex deep learning approaches on MCU-based systems on the edge. Compared to other approaches in the literature, the comparison showed significant gains in terms of energy consumption and model size which greatly affect the cost of the deployment. The compromise in terms of detection accuracy was only a few percentile units away from large in-size models.

To achieve a low memory usage, a separate detection and classification approach was suggested in this study. Thanks to this approach, the system could load images from the memory several times (i.e., load captured images from both sides and run a detection algorithm to detect, crop and save potential areas; then again load cropped images for classification). The process of loading images from the memory increases the image processing time as well as the power consumption. To tackle this issue, we are planning to work on a single-stage algorithm that does the detection and the classification simultaneously in order to decrease the image loading time, and thus, reduce the energy consumption.

In addition, we are also planning to apply video analysis and deep learning techniques for the detection of other insects, especially those of smaller size, which could be really useful in terms of studying insect behaviour. Moreover, we will extend the proposed platform to support other widely available commercial traps.

Data availability

The dataset is available at `https://doi.org/10.5281/zenodo.7887046`

Acknowledgments

This project is co-funded by the European Regional Development Fund (ERDF) under Ireland's European Structural and Investment Funds Programmes 2014–2020. This work was carried out as part of the Haly.ID project – 2020EN508 funded by Ireland's Department of Agriculture, Food and the Marine under Grant: 2020 Trans National ERA-NET. The first author is supported by a Walsh Scholarship funded by Teagasc, The Irish Food and Agriculture Authority. Aspects of this work have been supported by Science Foundation Ireland under Grant 12/RC/2289-P2-INSIGHT, 13/RC/2077-CONNECT, 16/RC/3835-VISTAMILK and 16/RC/3918-CONFIRM which are co-funded by the European Regional Development Fund (ERDF) under Ireland's European Structural and Investment Funds Programme 2014-2020. Moreover, we would like to express our gratitude to postdoc researcher, Dr. Mariusz Wilk, for his contribution to the design and development of the system as described in [22].

References

[1] D. R. Paini, A. W. Sheppard, D. C. Cook, P. J. De Barro, S. P. Worner, M. B. Thomas, Global threat to agriculture from invasive species, Proceedings of the National Academy of Sciences of the United States of America 113 (27) (2016) 7575–7579.

[2] I. Secretariat, Scientific review of the impact of climate change on plant pests, FAO on behalf of the IPPC Secretariat, 2021.

[3] G. Bulgarini, C. Castracani, A. Mori, D. A. Grasso, L. Maistrello, Searching for new predators of the invasive Halyomorpha halys: the role of the black garden ant Lasius niger, Entomologia Experimentalis et Applicata 169 (9) (2021) 799–806.

[4] V. Ferrari, R. Calvini, B. Boom, C. Menozzi, A. K. Rangarajan, L. Maistrello, P. Offermans, A. Ulrici, Evaluation of the potential of near infrared hyperspectral imaging for monitoring the invasive brown marmorated stink bug, Chemometrics and Intelligent Laboratory Systems 234 (2023) 104751.

[5] R. C. Stephenson, C. E. Coker, B. C. Posadas, G. R. Bachman, R. L. Harkess, J. J. Adamczyk, P. R. Knight, Economic Effect of Insect Pest Management Strategies on Small-scale Tomato Production in Mississippi, HortTechnology 30 (1) (2020) 64–75.

[6] Steven Frank, Lucy Bradley, Kathleen Moore, Integrated Pest Management, Chapter 8 (feb 2022).

[7] A. Bereciartua-Pérez, L. Gómez, A. Picón, R. Navarra-Mestre, C. Klukas, T. Eggers, Insect counting through deep learning-based density maps estimation, Computers and Electronics in Agriculture 197 (2022) 106933.

[8] W. Albattah, M. Masood, A. Javed, M. Nawaz, S. Albahli, Custom CornerNet: a drone-based improved deep learning technique for large-scale multiclass pest localization and classification, Complex and Intelligent Systems 9 (2) (2022) 1299–1316.

[9] M. Preti, F. Verheggen, S. Angeli, Insect pest monitoring with camera-equipped traps: strengths and limitations, Journal of Pest Science 94 (2) (2021) 203–217.

[10] Q. Guo, C. Wang, D. Xiao, Q. Huang, Automatic monitoring of flying vegetable insect pests using an RGB camera and YOLO-SIP detector, Precision Agriculture 24 (2) (2023) 436–457.

[11] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, arXiv preprint arXiv:2004.10934 (2020).

[12] W. Li, Z. Yang, J. Lv, T. Zheng, M. Li, C. Sun, Detection of Small-Sized Insects in Sticky Trapping Images Using Spectral Residual Model and Machine Learning, Frontiers in Plant Science 13 (2022) 2013.

[13] W. Li, D. Wang, M. Li, Y. Gao, J. Wu, X. Yang, Field detection of tiny pests from sticky trap images using deep learning in agricultural greenhouse, Computers and Electronics in Agriculture 183 (2021) 106048.

[14] Y. Li, J. Yang, Few-shot cotton pest recognition and terminal realization, Computers and Electronics in Agriculture 169 (2020) 105240.

[15] D. Brunelli, T. Polonelli, L. Benini, Ultra-low energy pest detection for smart agriculture, Proceedings of IEEE Sensors 2020-Octob (oct 2020).

[16] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, L. Benini, GAP-8: A RISC-V SoC for AI at the Edge of the IoT, Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors 2018-July (aug 2018).

[17] I. Saradopoulos, I. Potamitis, S. Ntalampiras, A. I. Konstantaras, E. N. Antonidakis, Edge Computing for Vision-Based, Urban-Insects Traps in the Context of Smart Cities, Sensors 2022, Vol. 22, Page 2006 22 (5) (2022) 2006.

[18] D. Brunelli, A. Albanese, D. D'Acunto, M. Nardello, Energy Neutral Machine Learning Based IoT Device for Pest Detection in Precision Agriculture, IEEE Internet of Things Magazine 2 (4) (2020) 10–13.

[19] A. Albanese, M. Nardello, D. Brunelli, Automated Pest Detection with DNN on the Edge for Precision Agriculture, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 11 (3) (2021) 458–467.

[20] J. Sütő, Embedded System-Based Sticky Paper Trap with Deep Learning-Based Insect-Counting Algorithm, Electronics 2021, Vol. 10, Page 1754 10 (15) (2021) 1754.

[21] J. Suto, A Novel Plug-in Board for Remote Insect Monitoring, Agriculture 2022, Vol. 12, Page 1897 12 (11) (2022) 1897.

[22] A. Kargar, M. P. Wilk, D. Zorbas, M. T. Gaffney, B. Q'Flynn, A Novel Resource-Constrained Insect Monitoring System based on Machine Vision with Edge AI, 5th IEEE International Image Processing, Applications and Systems Conference, IPAS 2022 (2022).

[23] A. L. Acebes, Host plant effects on the biology, behavior and ecology of brown marmorated stink bug, Halyomorpha halys (Stål) (Hemiptera: Pentatomidae) (2016).

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[25] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-Decem (2015) 770–778.

[26] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2323.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[28] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size, arXiv preprint arXiv:1602.07360 (2016).

[29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[30] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, Journal of Big Data 2021 8:1 8 (1) (2021) 1–74.

[31] S. S. Saha, S. S. Sandha, M. Srivastava, Machine learning for microcontroller-class hardware-a review, IEEE Sensors Journal (2022).