Benchmarking of Computer Vision Methods for Energy-Efficient High-Accuracy Olive Fly Detection on Edge Devices

José L. Mira[1*], Jesús Barba[1†], Francisco P. Romero[1†], M. Soledad Escolar[1†], Julián Caba[1†] and Juan C. López[1†]

[1*]Technologies and Information Systems Department, School of Computer Science, Paseo de la Universidad, 4, Ciudad Real, 13071, Castilla La Mancha, Spain.

*Corresponding author(s). E-mail(s): JoseLuis.Mira@uclm.es;
Contributing authors: Jesus.Barba@uclm.es; Franciscop.Romero@uclm.es; Soledad.Escolar@uclm.es; Julian.Caba@uclm.es; JuanCarlos.Lopez@uclm.es;
†These authors contributed equally to this work.

Abstract

The automation of insect pest control activities implies the use of classifiers to monitor the temporal and spatial evolution of the population using computer vision algorithms. In this regard, the popularisation of supervised learning methods represents a breakthrough in this field. However, their claimed effectiveness is reduced regarding working in real-life conditions. In addition, the efficiency of the proposed models is usually measured in terms of their accuracy, without considering the actual context of the sensing platforms deployed at the edge, where image processing must occur. Hence, energy consumption is a key factor in embedded devices powered by renewable energy sources such as solar panels, particularly in *energy harvesting* platforms, which are increasingly popular in smart farming applications. In this work, we perform a two-fold performance analysis (accuracy and energy efficiency) of three commonly used methods in computer vision (e.g., HOG+SVM, LeNet-5 CNN, and PCA+Random Forest) for object classification, targeting the detection of the olive fly in chromatic traps. The training and testing of the models were carried out using pictures captured in various realistic conditions to obtain more reliable results. We conducted an exhaustive exploration of the solution space for each evaluated method, assessing the impact of the input dataset and configuration parameters on the learning process outcomes. To determine their suitability for deployment on edge embedded systems, we implemented a prototype on a Raspberry Pi 4 and measured the processing time, memory usage, and power consumption. The results show that the PCA-Random Forest method achieves the highest accuracy of 99%, with significantly lower processing time (approximately 6 and 48 times faster) and power consumption (approximately 10 and 44 times lower) compared with its competitors (LeNet-5-based CNN and HOG+SVM).

Keywords: Insect Pest Monitoring, Olive fly, Edge Computing, Energy Harvesting, Computer Vision, Artificial Intelligence

1 Introduction

Currently, insect pests pose one of the greatest risks to crops on farms. To effectively control these pests, it is essential to accurately estimate the population of harmful insects in the surrounding environment. The insect population was estimated by counting the number of individuals captured in a series of chromatic traps that use pheromones to attract a specific type of specimen. Typically, this counting process is performed manually by a pest prevention technician who visually identifies the individuals. Counting insect populations is a crucial step in assessing the risk that insects pose to farms, enabling the implementation of preventive measures to mitigate potential damage.

The manual collection of data and counting of insects have certain limitations, including constraints on the quantity and frequency of these tasks. Additionally, human error can occur. Therefore, there is significant interest in automating this process. To enhance the efficiency and scalability of pest control systems, it is crucial to develop methods capable of remotely analysing visual information acquired daily. Thus, the response time can be significantly reduced, allowing for timely mitigation of the negative effects of insect pests with minimal resource usage. Moreover, these methods will have a positive impact on the ecosystem. However, individual insect identification presents challenges because of the wide range of orientations in which they may be attached to the trap [1] [2].

Computer vision techniques enable the automation of insect population counting in chromatic traps. By leveraging advanced algorithms and image processing, computer vision systems can accurately analyse the visual information captured by cameras. These systems can recognise and identify individual insects, allowing efficient and precise population estimation. The use of computer vision eliminates the need for manual counting by pest control technicians and offers a faster and more reliable approach to insect population assessment.

This paper proposes a comprehensive approach for the application of a series of computer vision algorithms coupled with machine learning techniques to accurately identify the presence of olive fruit flies in chromatic traps, while considering the energy dimension. On the one hand, the developed prototype is capable of providing a count of the detected individuals using a robust method, tested with actual pictures of chromatic traps. Working in a noncontrolled environment introduces several challenges that have been properly addressed in this study. This is in contrast to the majority of related literature reviewed, which often relies on ideal or controlled pictures as a starting point. On the other hand, the precision and accuracy results of the ML models examined in this study were compared with the computational resources required and their efficiency to select the optimal solution for image processing on edge devices powered by an energy harvesting infrastructure.

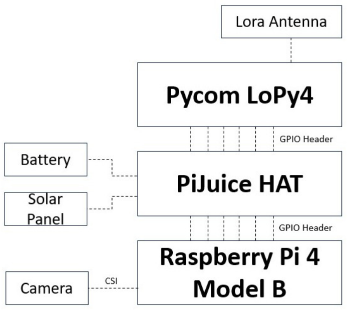
The selection of the three computer vision methods was driven by several factors, including: (1) their maturity and extensive practical experience in similar computer vision applications; (2) their wide adoption and familiarity within the research community; and (3) their demonstrated high accuracy and performance in classification problems. Thus, the final three algorithms considered are HOG+SVM, LeNet-5 CNN, and PCA+Random Forest.

In the next sections, the performance achieved by the three selected methods, as well as a thorough analysis of the impact of several factors (e.g. input format or model parameters) during the training phase, will be disclosed, providing a valuable tool for developers to better understand how the models develop in actual working conditions.

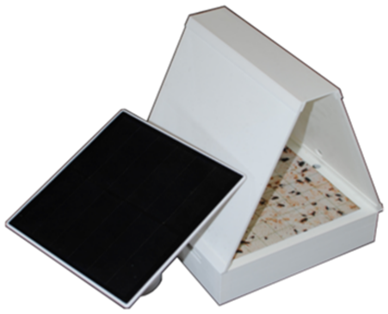
1.1 Project Contextual Background

This work is carried out as part of a project aiming to develop a revolutionary electronic chromatic trap called BIoTrap 4.0. This trap integrates three fundamental technologies: IoT (*Internet of Things*), computer vision algorithms, and the edge computing paradigm. BIoTrap 4.0 is built on a Raspberry Pi 4 Single Board Computer (SBC), to which several essential modules are connected, including a camera module, communication module, and power management module. Figure 1 visually depicts the comprehensive composition of this system.

The camera module is a vital component of the trap and is responsible for capturing images of insects. These images are analysed locally on the device, eliminating the need for continuous



(a) BIoTrap Schematic



(b) BIoTrap Hardware

Fig. 1 BIoTrap Prototype

internet connectivity and reducing data transmission requirements. This functionality proves highly valuable, especially in agricultural environments where network availability and bandwidth limitations are significant concerns.

To achieve efficient long-distance connectivity, the BIoTrap 4.0 prototype incorporates LoRa (Long Range), a wireless communication technology. LoRa uses a low-power radio protocol with widespread coverage, enabling long-range communication while minimising energy consumption. Furthermore, the use of unlicensed frequencies simplifies deployment and reduces costs. This technology is particularly well-suited for applications requiring broad coverage and efficient data transmission, such as smart agriculture.

Energy independence is another notable feature of BIoTrap 4.0. To achieve this, a lithium-polymer (LiPo) battery is integrated and connected to a 12W solar panel through a charge controller. This setup allows the trap to harness ambient conditions at the deployment location for energy generation. By leveraging solar energy, BIoTrap 4.0 becomes self-sufficient, eliminating the need for external power sources. In addition, the system can be programmed to enter a sleep mode and resume operation according to a predefined schedule, optimising energy use. The goal is to find an optimal solution that balances performance, accuracy in insect identification and energy demand.

Construction of the olive fruit fly image dataset is another key aspect of the project. To achieve this, a labelling application is used in which experts select image fragments that correspond to a positive match with this pest as shown in figure 2. These methods become valuable tools for expert decision making, while providing feedback for selecting parameters of these methods. In this way, a cycle of continuous improvement is established where the labelled data allows for the refinement and optimisation of the image analysis algorithms used in the BIoTrap 4.0 electronic trap.

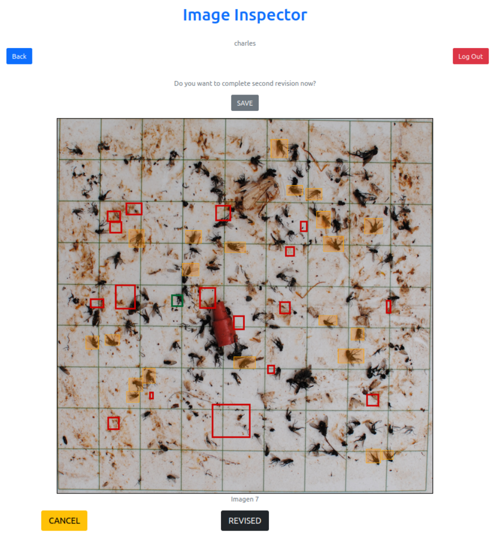


Fig. 2 Advisor app

Given that the system follows the energy harvesting paradigm, the methods used must be aware of the energy resources and minimize energy consumption while maximizing accuracy obtained.

This multi-view analysis approach represents a significant contribution, providing valuable insights to developers of edge artificial vision applications regarding the implications of different factors in the decision-making process. Consequently, it enhances the design and performance of their applications.

2 State of the art

Until recently, one of the most popular methods in several entomological classification studies was Support Vector Machines (SVM) combined with a feature extraction algorithm such as Histogram of Oriented Gradients (HOG) [3] [4] [5]. However, the feature extraction stage is highly sensitive to the environment in which pictures of individuals are taken to build the dataset. Consequently, many of the previous papers on insect detection and classification were developed in controlled environments or even based on entomological images that differ from the trapping conditions in a real environment [6] [7] [8]. Weather or the presence of other objects in the image often pose significant obstacles in establishing uniform processing parameters for all captured images. Variations in lighting conditions and the potential presence of dust, plant fragments, or other insects adhering to the chromatic trap contribute to these challenges. Additionally, feature extraction is typically a costly and unreliable process, as mentioned earlier. One of the primary challenges in designing an image processing pipeline is to obtain individual cut-outs of the regions of interest within the chromatic trap. This step is crucial before proceeding to the classification process using machine learning algorithms.

For all these reasons, the use of Convolutional Neural Networks (CNNs) has recently become the dominant approach for solving problems related to object classification or detection in images across various fields, including the automatic detection of insects [9] [10] [6] [8]. This is primarily due to two factors: (1) the robustness of the predictions even under varying conditions of the input dataset images, provided that the network is properly trained with a diverse dataset, and (2) the elimination of the need for a preprocessing stage. However, the Achilles heel of CNNs lies in the time-consuming training process, which is performed only once and requires expert input to avoid undesirable network behaviour. In brief, the use of CNN-based systems has captured the attention of the research community and has partly overshadowed other methods that could be worth considering when energy cost is a factor in the decision-making process.

In this article, in addition to SVM+HOG and CNN, the Principal Component Analysis (PCA) algorithm has been investigated, drawing inspiration from its success in specific works in hyperspectral and multispectral image analysis [11] [12], and its application in studying the influence of pathogenic diseases on crops using E-noses [13]. The combination of PCA and Random Forest can be beneficial in various scenarios. First, PCA helps to reduce the dimensionality of the dataset, which can be especially useful when working with highly correlated features or when reducing noise and redundancy in the data. This can improve computational efficiency and prevent model overfitting. Moreover, PCA can help identify the most important and representative features of the dataset, which can enhance the performance of Random Forest by reducing noise and focussing on the most informative characteristics.

In recent years, Spike Neural Networks (SNNs) have emerged as a captivating paradigm in the field of artificial intelligence, drawing inspiration from the intricate communication patterns of neurones in the human brain. The significance of SNNs lies in their unique ability to capture temporal dynamics, making them well-suited for tasks involving time-series data and dynamic information processing . Unlike conventional neural networks, SNNs process information through discrete events or 'spikes,' enabling event-driven computation and promising energy-efficient solutions, particularly in edge-based image classification applications.

However, the transition from Convolutional Neural Networks (CNNs) to SNNs in the realm of image classification introduces several noteworthy challenges. First, the conversion process itself, involving the transformation of continuous activations from CNNs into discrete, event-driven spikes characteristic of SNNs, demands careful consideration and exploration of suitable encoding mechanisms. For instance, in the work by [14], a set of 5x5 overlapping receptive fields, weighted according to Manhattan distance, is used to encode 16x16 regions of input pixels. The neural response is the membrane potential map, which is further converted into spike trains, with the spiking frequency proportional to the potential.

The need for novel training algorithms [15][16] and techniques tailored to the unique characteristics of SNNs has become obvious. In [17], the HESFOL model is proposed, utilising entropy theory to establish a gradient-based few-shot learning scheme in a recurrent SNN architecture. This model effectively improves the accuracy and robustness of the learning process, as demonstrated in image classification using the Omniglot dataset.

Furthermore, challenges related to hardware constraints and efficient implementation add complexity, given that the energy-efficient promise of SNNs necessitates specialised hardware architectures. Lakymchuck et al. [14] introduce a simplified and computationally efficient model of the spike response model (SRM) neuron with spike-time-dependent plasticity (STDP). The proposed solution combines spike encoding, optimal network topology, and an innovative neuron membrane model, achieving successful learning and classification of black and white images (Semeion dataset) with up to a x20 computing speed-up compared to other classic neuron models.

Beyond handwritten digit recognition, SNNs have been successfully validated in fields such as the classification problem of hyperspectral images in the edge computing environment [18], gesture recognition [19] or sonar image target classification [20].

While SNNs demonstrate promising features for embedded computing applications, the emphasis on static image analysis in this study, where temporal dynamics may not be fully utilized, has led to the decision to postpone the incorporation of SNNs to future versions of the BioTrap 4.0 system. The necessity to meticulously explore various neural models and alternative approaches for encoding input stimuli poses challenges that require a custom and careful consideration.

3 Materials and methods

3.1 Dataset creation

For this work, a dataset was created from actual images of chromatic traps taken on four different farms under variable lighting conditions. The original images are in RGB format and have a resolution of 3840x2160 pixels.

A total of 3043 clips of regions of interest (ROI) were extracted for their subsequent classification following an approach that will be briefly introduced for the shake of completeness.

The extraction of ROIs (see Figure 3) is influenced by the functional and not functional requirements of the platform where the algorithm will be deployed (that is, a RaspBerry PI 4 device powered by solar panels feeding a battery) and the image acquisition conditions, which are a real-life environment with variable lighting conditions, wind and dust. Thus, efficiency in the use of computing resources and variability management are two key driving factors for image preprocessing stage. Reducing the resolution of the picture and converting it to greyscale saves memory and computing time, while equalisation makes the process more resilient to changes in ambient conditions during image capturing. Once the picture has been cleaned (thresholding) and potential ROIs enhanced (morphological operations), the connected components the mask parameters (coordinates, area, etc.) to extract the final clips.

Subsequently, and the objective of the work presented in this article, it follows a classification process of these clips according to whether they correspond to regions where the olive fruit fly appears or not, in order to obtain the population indices. To be able to perform the training, these images were previously labelled by an expert in the field who has the knowledge to discern between what is and what is not olive fruit fly.

Thus, from the dataset of 3043 images, we obtain 358 elements of the positive class and 2685 of the negative class. It is worth recalling that such cut-outs were obtained directly from the output of the actual segmentation algorithm fed with real images. Thus, reproducing the working conditions provides input ROIs unideal (e.g. overlapping individuals, noisy background, etc.) but ensures a fair ground for all the machine methods to be tested, so the achieved performance levels are more reliable.

Figure 4 shows four examples of ROIs extracted after the segmentation phase with (Figure 4a) and without (Figures 4b, 4c and 4d) olive fly. As it can be seen, additional elements may appear because the size of the region is fixed (170x170 pixels). These elements could influence the learning process for the different

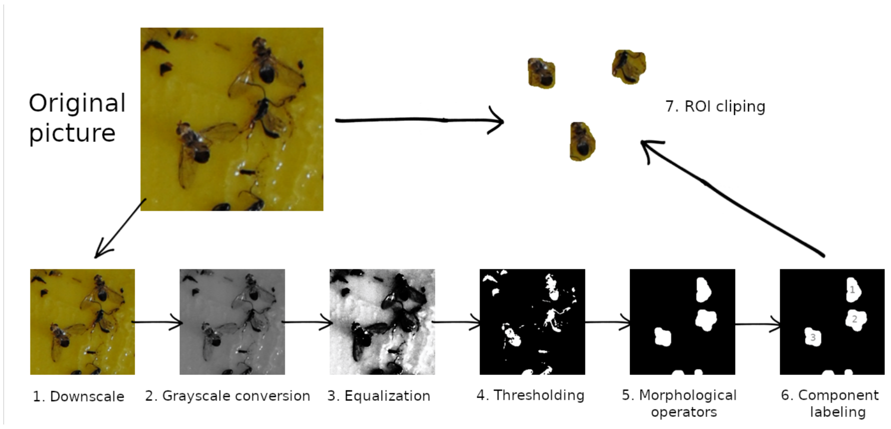


Fig. 3 Process of segmentation and extraction of ROIs in chromatic trap images

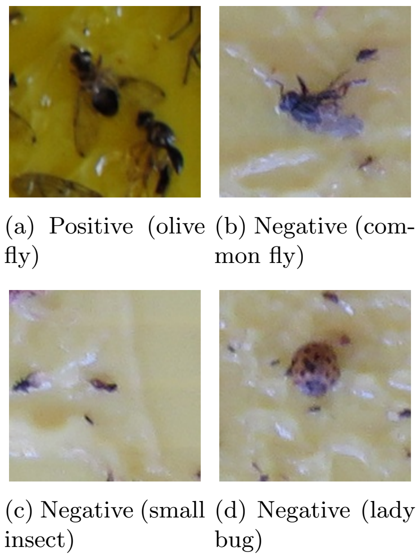


Fig. 4 Example of dataset images, extracted by the segmentation algorithm

machine learning methods to be studied, by introducing noise. This will also be evaluated in this work. Therefore, a background extraction process is additionally applied to this initial set of image clips, leaving only the region that was identified as an object or component (see Figure 5b). This process is simple; after stage 6 of the segmentation process (Figure 3) a binary mask is obtained from the connected components algorithm. This binary mask is then applied to the image clips, leaving the foreground element clean.

A little more than 3000 images is not a huge set to perform a good training of a classifier. Therefore, it is necessary to perform a "data augmentation" process. For each image of the original set, six new images are generated as the result of a clockwise rotation operation (90º, 180º, 270º and 360º) and mirroring on the horizontal and vertical axes. After this process 2872 images for the positive class and 21480 for the negative class made up the dataset.

3.2 Training procedure and metrics

The input images to the system have a resolution of 170 pixels wide by 170 pixels height. These images were normalised (i.e. the values of all its pixels were recalculated to lie in a floating point range between 0 and 1) before training the CNN and PCA+Random models. Normalization is a key operation to avoid problems in the internal calibration of these algorithms during training, which could be altered by outliers. On the other hand, it is not necessary to normalise the image set for the SVM+HOG use case.

When evaluating the performance of the models, two scenarios were analysed, each with a different dataset: regions with and without background extraction.

In terms of training, various situations have been explored using unbalanced and balanced datasets. Training with an unbalanced dataset respects the proportion of examples found in the real samples. This translates into using 18000 examples of the negative class and 2800 of the positive class. In the case of a balanced dataset, training was performed using 2800 instances of each class. The objective is to determine whether the prediction quality improves by balancing the

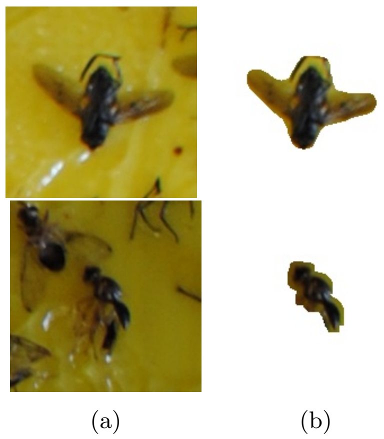


Fig. 5 Example of dataset images for training, with (a) and without (b) background subtraction

dataset despite having fewer samples to train the models.

Regardless of the dataset used and its characteristic (balanced versus unbalanced), it is split in a ratio of 80/20, intended for the training and test sets respectively. The training set, as its name suggests, will be used to train the model in order to subsequently perform the prediction on the test set and evaluate the results.

The training was performed using the cross-validation method, in which the training set itself was divided into 10 parts. One of the ten parts is selected as the test, and the rest are used for training. In cross-validation this, process is performed as many times as divisions are made to the set, so that the test is performed on a different subset each time.

In this work, Python was used for the development of the training algorithm and the machine learning models, relaying on libraries and software belonging to the Tensorflow and Scikit Learn environments.

In the following sections, the three selected models for evaluation are introduced: (1) Histogram of Oriented Gradients for feature extraction combined with a Support Vector Machine for classification; (2) Convolutional Neural Networks and; (3) Principal Components Analysis combined with Random Forest.

3.3 Evaluated models

3.3.1 HOG+SVM

HOG (*Histogram of Oriented Gradients*) is a method for obtaining descriptors or features that is widely used in computer vision and image processing applications. This algorithm assumes that the information in an image can be described and summarised by the distribution of gradient vectors calculated from groups of image pixels. This type of information is relevant when we want to keep the information that makes sense to distinguish shapes, as in our case, discarding non-necessary information such as color, etc.

To do this, the image is divided into small square subregions called cells and, a histogram of gradient vectors is generated for all pixels contained within each cell. The final result, or resulting feature vector, arises from the concatenation of each of these individual histograms.

To infer knowledge from the histogram information, one of the methods used is the SVM (*Support Vector Machines*) classifier. Roughly speaking, an SVM is a model that represents the sample points in space, trying to separate the points in classes by means of a hyperplane.

The function that computes the gradient histogram has several parameters that can be configured. First, a series of tests was performed to experimentally determine the best value for the cell size parameter. Taking into account that the size of the input image is 170×170 pixels, tests were performed with 16 different cell sizes, with values of pixel numbers in rows and columns within the set (85,42,21,10) for both RGB and greyscale datasets, with an unbalanced combination of positive and negative examples.

Because of the tests (see subsection 4.1), a 10x10 pixels cell size was selected due to its better robustness to noise. Thus, for each image cell (a total of 17×17 cells) 100 gradient vectors are obtained, which are summarised in a 9-set histogram. Each set represents a range of angles 0, 20, 40, 60 ... 160 (an unsigned gradient approximation has been used, as it is done in pedestrian detection application) and each gradient can contribute proportionally to two sets (or ranges of angles) depending on the direction value (e.g., if the direction value is 10, it will contribute equally to set 0 and set 20).

In summary, from an input image of 170×170 pixels (28,900 values in greyscale and 86,700 in RGB), we do obtain 17x17x9 (2,601) values or features representing such an image. These values or feature vectors are the input to the classifier.

For the classification part, the SVC model of the Scikit-Learn library was used, selecting a *Grid-Search* type search and evaluating the following parameters:

- *gamma*: scale and auto.
- *kernel*: linear, rbf, poly and sigmoid.

3.3.2 CNNs

Convolutional neural networks (CNNs) are a type of artificial neural networks intended to emulate the behaviour of neurons in the primary visual cortex of the human brain.

In this work, LetNet-5, one of the most popular CNN architectures, was chosen as the baseline to build a proposal customised for olive fly detection given the input dataset. The main reason for the popularity of this model is its simple and straightforward structure. This simplicity, applied to image recognition, makes it attractive for a restrictive computing environment such as the one planned for the deployment of our automatic olive fly identification and counting system.

Lenet-5 model was proposed by Yann LeCun and other authors in 1998 [21], to recognise handwritten and typewritten characters. In this study, the original structure has been maintained (Figure 6) but several architectural parameters need to be updated.

All convolution layers use a kernel size of 3×3 and relu-type activation. In addition, the Max-Pool stage uses a 2×2 reduction. The hyperparameters of the model were configured as follows:

- optimizer : adam
- loss : binary_crossentropy
- metrics : accuracy
- epochs : 6

It is worth mentioning that tests have been performed on more complex neural networks architectures. However, it has been observed that, in these cases, overfitting has occurred, resulting in worse results.

3.3.3 PCA + Random Forest

PCA (*Principal Components Analysis*) is another method widely used in machine learning that aims to reduce the complexity (dimensional and volume) of the datasets used to train the different models. This technique converts a set of correlated variables into a set of values lacking linear correlation, which we call principal components. This is done by constructing a new coordinate system for the dataset in which the largest variance is captured on the first axis and so on, assigning an extra dimension to each of the variances downwards. This transformation, which converts old coordinates to new ones, is precisely the linear transformation needed to reduce the dimensionality of the data.

The dataset treated with PCA was classified using a *RandomForestClassifier* method. The implementation used is that of *Scikit-Learn* in which the parameters chosen for the model have been selected based on a *GridSearch* type search also from the *Scikit-Learn* library. The parameters with which the model has been tested are listed below:

- criterion: gini, entropy.
- max_depth: None, 10, 100, 1000.
- min_samples_split: 2, 3, 5, 10.
- min_samples_leaf: 1, 2, 3, 5, 10.
- max_features: None, auto, sqrt, log2.
- n_estimators: 100, 500, 1000, 5000.

3.4 Energy consumption measurement

To accurately measure the energy consumed by each algorithm on the Raspberry Pi 4, a setup based on the OWON XDM2041 digital multimeter was designed. Two configurations were tested. Initially, the probes were connected to the Test Points (TP1 and TP2) on the Raspberry Pi dedicated to verifying the 5V power supply. However, this configuration proved to be unstable due to the inability to securely attach the probes to the board, which directly affected the reliability of the measurements. Therefore, a second configuration (shown in Figure 7) was used, where the power supply cable was tapped to insert the multimeter into the circuit. This setup allowed for unattended operation and improved stability. Furthermore, the inevitable modification of

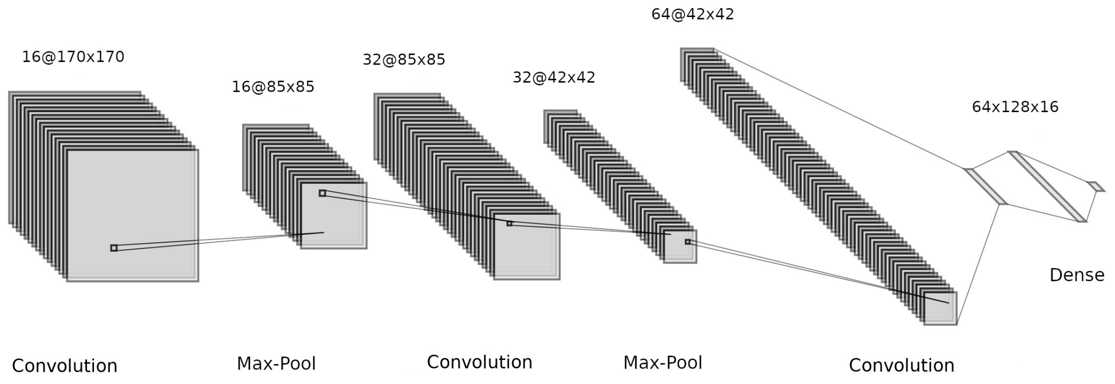


Fig. 6 LeNet-5 inspired architecture of the CNN model under evaluation for inference on an edge computing context

the system's working conditions under observation (due to the internal resistance of the multimeter used for current measurement) was minimised, as the Raspberry Pi 4 is capable of compensating and stabilising the input current.

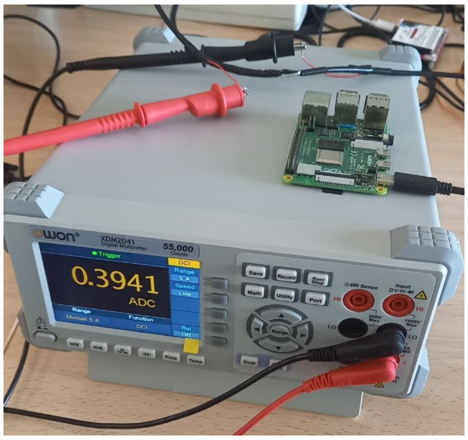


Fig. 7 Setup built for energy consumption measurement

The OWON XDM2041 digital multimeter can capture samples at a configurable rate, with an estimated error of 0.025% and 0.15% in voltage and amperage readings, respectively. To accommodate the latency requirements of the tests, the sampling process was customised with a 15ms period for CNN and PCA+Random, and a 111ms period for HOG+SVM. Each test involved the classification of 116 regions of interest (ROIs), and ten executions were performed for each test and algorithm being evaluated.

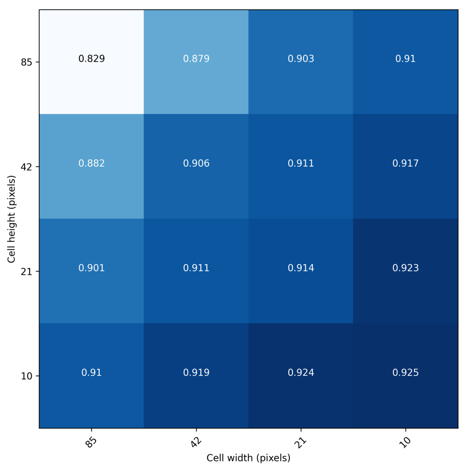
4 Results

In this section, we provide detailed information on the results obtained by the different models evaluated for all combinations of parameters and training data sets considered in this study. For each of the models, the best scenario will be selected and justified by analyzing the reasons for such behaviour and interpreting the metrics used for its assessment (i.e. Accuracy, Precision, Recall and F1). In all cases, tests were performed for the eight possible combinations, according to the characteristics of the training set described in subsection 3.1: with/without background extraction, RGB/greyscale and balanced/unbalanced class examples.

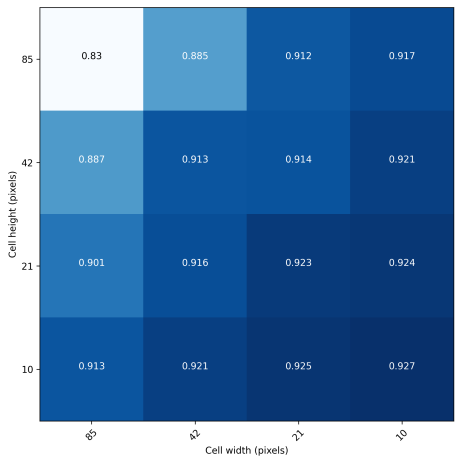
4.1 HOG+SVM

As it was introduced in subsubsection 3.3.1, sixteen different types of cells were firstly evaluated so that conclusions could be drawn about the impact of both the size and the geometry of the subdivisions for our application. As an example, in Figure 8 can be seen, in a graphical way, the variation of the accuracy metric for the RGB and greyscale datasets, using an unbalanced combination of positive and negative examples.

From the analysis of the figures, it can be concluded that, on the one hand, the larger the cell size, the less gradient vectors will be used to represent the image. On the other hand, the smaller the cell size, the less sensitive to noise is the information obtained. The use of colour or



(a) Grayscale dataset without background



(b) RGB dataset without background

Fig. 8 Impact of the cell size and geometry in the accuracy achieved by the HOG+SVM method for an unbalanced dataset

	RGB		*Grayscale*	
Background	*YES*	*NO*	*YES*	*NO*
Accuracy	0.70	0.82	0.66	0.82
Precision	0.84	0.90	0.79	0.90
Recall	0.46	0.73	0.43	0.75
F1	0.59	0.81	0.56	0.82

Table 1 Performance metrics of the HOG+SVM method for class-balanced input image sets.

	RGB		*Grayscale*	
Background	*YES*	*NO*	*YES*	*NO*
Accuracy	0.86	0.90	0.86	0.91
Precision	0.83	0.80	0.71	0.83
Recall	0.07	0.43	0.09	0.44
F1	0.13	0.56	0.16	0.58

Table 2 Performance metrics of the HOG+SVM method for class-unbalanced input image sets.

	RGB		*Grayscale*	
Background	*YES*	*NO*	*YES*	*NO*
Accuracy	0.85	0.87	0.86	0.90
Precision	0.83	0.86	0.82	0.88
Recall	0.87	0.89	0.92	0.93
F1	0.85	0.87	0.87	0.91

Table 3 Performance metrics of LeNet-5 CNN for class-balanced input image sets.

greyscale images does not make a significant difference in terms of the maximum accuracy obtained in both cases (92.5% versus 92.7% in favour of RGB images). However, the cell size does affect the results, with the smallest cell size (10×10 pixels) obtaining the best results due to its better robustness to noise.

In addition, the best combinations of parameters for the SVM classifier were: C $\rightarrow$ 1.0, class_weight $\rightarrow$ balanced, gamma $\rightarrow$ scale, kernel $\rightarrow$ poly.

Tables 1 and 2 show the results obtained for all metrics and datasets with the final configuration. As in all cases, these values are the average measured results after performing the rankings with all combinations resulting from the cross-validation method used. It should be noted that this method is quite sensitive to imbalances in the dataset. While accuracy improves when using a dataset with unbalanced classes, all other parameters are negatively affected.

4.2 CNN

Tables 3 and 4 show that the CNN model is sensitive to the training set, having a special impact on the Precision and Recall metrics. When the CNN is trained with a set of images where the number of negative class examples (non-fly) is predominant with respect to the positive class examples (fly), a better Accuracy value is obtained because of the higher probability that the class of the image to be classified is "non-fly". However, if we look at the rest of the metrics, they are clearly

	RGB		*Grayscale*	
Background	*YES*	*NO*	*YES*	*NO*
Accuracy	0.92	0.92	0.91	0.92
Precision	0.83	0.71	0.75	0.72
Recall	0.61	0.67	0.66	0.67
F1	0.70	0.69	0.70	0.69

Table 4 Performance metrics of LeNet-5 CNN for class-unbalanced input image sets.

favourable for the scenario of a training with a balanced set of images. Impact of the input image format (i.e. Grayscale versus RGB) is not significant, although a slight inferiority in prediction can be seen for black and white images compared with colour images. This difference is somewhat greater, but not much, in terms of the extraction or not of the background that is not part of the ROI, as indicated by the segmentation algorithm in the preprocessing phase.

Figure 9 shows the evolution of the accuracy and loss of the model as CNN training progresses. The analysis of the graph allows us to determine the instant in which the best prediction is obtained with a minimum mean error. The most interesting thing is to see how both parameters evolve for the test set (orange line) and not the training set, since in that case the CNN is being fed with class examples that did not participate in the first stage. The goal of this analysis is to determine the optimal point in this model fitting iterative process because there is a risk for our CNN to overfit and not being able to generalize. Thus, we came to the conclusion that, for both input images with and without background, 6 epochs give the best results. From that point on, although the accuracy of the model increases (slightly, due to the imbalance in the number of classes), the loss increases.

4.3 PCA+Random Forest

For this model, tests were performed by varying the % covariance maintained by the number of dimensions, as well as the number of remaining dimensions, and how these variations affect the metrics obtained by the prediction model.

These tests reflect the differences in the classification quality of the samples for the training dataset. Different reductions in the conserved variance have been progressively applied to this dataset as shown Tables 5 and 6. To evaluate the performance of the Random Forest algorithm in the classification of each of these datasets, the Accuracy, Precision, Recall and F1 metrics have been used. As in the previous cases, an 80/20 rule is followed for creating the training and test sets.

The combination of configuration parameters that gave the best results was as follows: criterion $\rightarrow$ entropy;max_depth $\rightarrow$ None;max_features $\rightarrow$ auto; min_samples_leaf $\rightarrow$ 1; min_samples_split $\rightarrow$ 3; and n_estimators $\rightarrow$ 100.

Tables 5 to 8 collect all the data generated for all parameter combinations. In summary, it can be concluded that the model behaves similarly for both balanced and unbalanced datasets. This is a feature that differentiates the PCA+Random Forest model from previous methods, where the influence of the percentage of positive class examples worked against the F1 metric but favoured the Accuracy metric. In this sense, it is more stable and homogeneous for all metrics obtained.

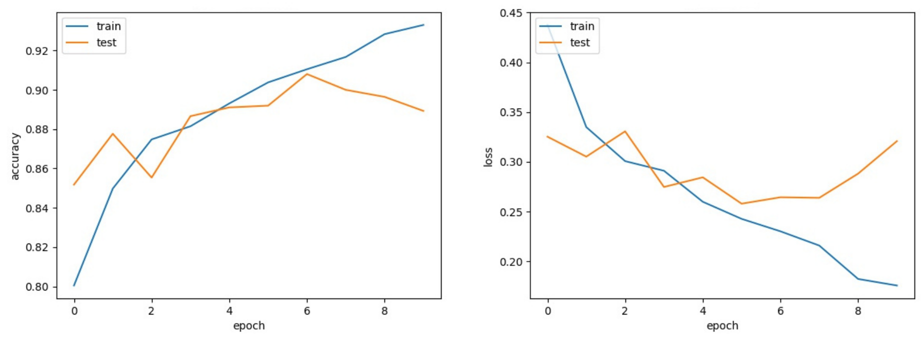
We can observe how, for the set of images with background, the reduction in the percentage of conserved variance causes the remaining dimensions to decrease more rapidly than that for the set of images in which the background has been extracted.

Taking the F1 metric as a reference, we can observe how the results remain comparable for the four combinations, although the RGB/No Background combination seems to stand out on average. However, the best values are obtained for the model using the set of images with background (these values are highlighted in bold in Tables 5 and 6). However, as stated, the difference is very small.

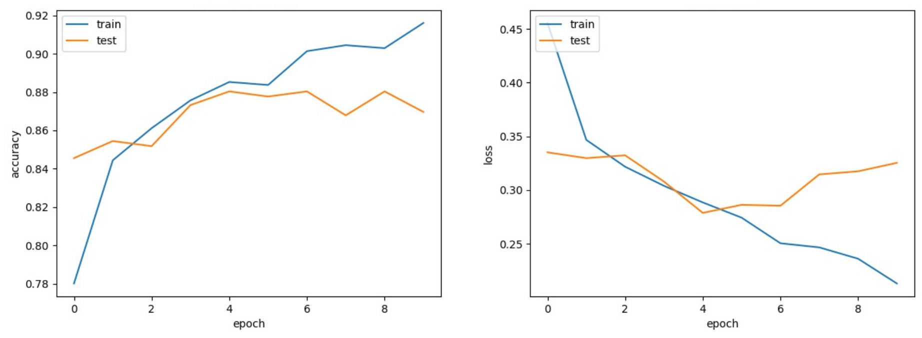
Another observation is that the maxima in terms of prediction quality are obtained at PCA reduction values ranging between 0.75 and 0.5 (depending on the dataset and method). This behaviour is interesting because it implies, as can be seen in Tables 7 and 8, a significant dimensional reduction that greatly simplifies the generated tree and therefore, the inference time and resources required.

4.4 Noise robustness in classification

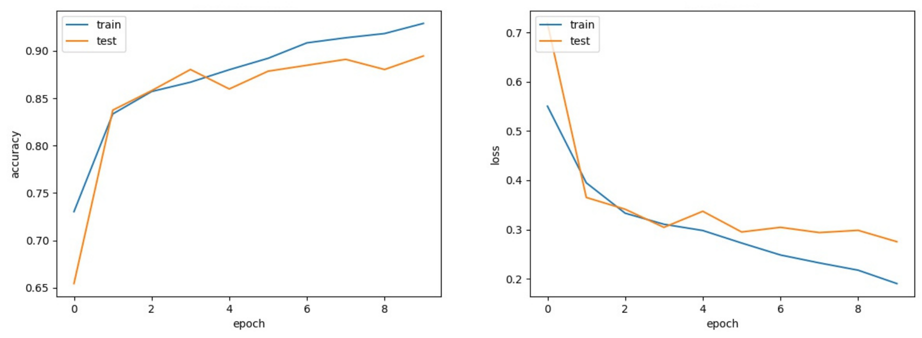
Adding noise to images is a commonly used technique for evaluating the robustness and generalisation of classification methods in the field



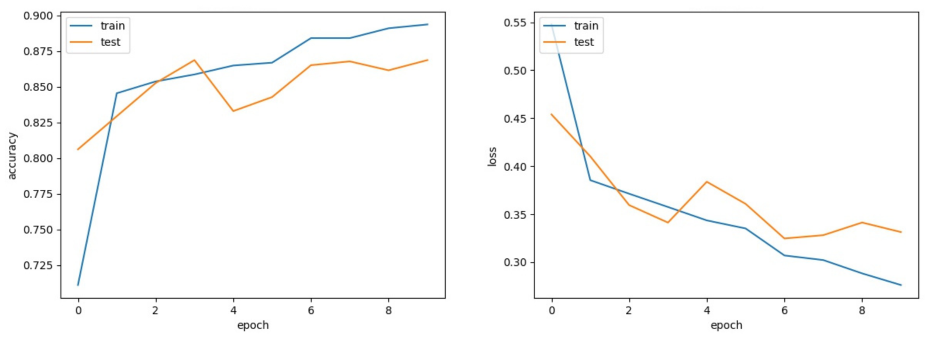
(a) RGB dataset without background



(b) Grayscale dataset without background



(c) RGB dataset with background



(d) Grayscale dataset with background

Fig. 9 Evolution of accuracy and loss of LeNet-5 based model during neural network training

	RGB								Grayscale							
	Background				No Background				Background				No Background			
Reduction	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
None	0.83	0.82	0.84	0.83	0.85	0.83	0.87	0.85	0.78	0.77	0.80	0.79	0.85	0.85	0.85	0.85
0.99	0.83	0.84	0.82	0.83	0.85	0.81	0.89	0.85	0.82	0.83	0.81	0.82	0.84	0.85	0.83	0.84
0.95	0.84	0.84	0.85	0.84	0.84	0.81	0.87	0.84	0.85	0.87	0.81	0.84	0.85	0.87	0.82	0.85
0.9	0.86	0.86	0.86	0.86	0.84	0.81	0.86	0.84	0.87	0.86	0.88	0.87	0.88	0.89	0.85	0.87
0.75	**0.94**	**0.91**	**0.99**	**0.95**	0.86	0.84	0.89	0.86	**0.92**	**0.88**	**0.96**	**0.92**	0.89	0.90	0.88	0.89
0.5	0.83	0.82	0.84	0.83	**0.94**	**0.91**	**0.98**	**0.94**	0.78	0.77	0.80	0.79	**0.95**	**0.92**	**0.98**	**0.95**
0.25	0.83	0.82	0.84	0.83	0.85	0.83	0.87	0.85	0.78	0.77	0.80	0.79	0.85	0.85	0.85	0.85
0.1	0.83	0.82	0.84	0.83	0.85	0.83	0.87	0.85	0.78	0.77	0.80	0.79	0.85	0.85	0.85	0.85

Table 5 Variation of prediction model quality metrics as a function of variability conservation. Balanced data set.

	RGB								Grayscale							
	Background				No Background				Background				No Background			
Reduction	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
None	0.93	0.75	0.73	0.74	0.90	0.66	0.62	0.64	0.92	0.69	0.71	0.70	0.90	0.62	0.61	0.62
0.99	0.90	0.63	0.69	0.66	0.90	0.64	0.65	0.65	0.92	0.71	0.73	0.73	0.90	0.63	0.64	0.63
0.95	0.91	0.68	0.71	0.70	0.92	0.70	0.71	0.70	0.94	0.76	0.78	0.77	0.92	0.68	0.71	0.70
0.9	0.92	0.75	0.72	0.74	0.92	0.73	0.71	0.72	0.95	0.83	0.81	0.82	0.93	0.71	0.74	0.72
0.75	**0.99**	**0.97**	**0.95**	**0.96**	0.94	0.79	0.79	0.79	**0.98**	**0.98**	**0.94**	**0.96**	0.95	0.81	0.85	0.83
0.5	0.93	0.75	0.73	0.74	**0.98**	**0.94**	**0.95**	**0.95**	0.92	0.69	0.71	0.70	**0.97**	**0.92**	**0.91**	**0.91**
0.25	0.93	0.75	0.73	0.74	0.90	0.66	0.62	0.64	0.92	0.69	0.71	0.70	0.90	0.62	0.61	0.62
0.1	0.93	0.75	0.73	0.74	0.90	0.66	0.62	0.64	0.92	0.69	0.71	0.70	0.90	0.62	0.61	0.62

Table 6 Variation of prediction model quality metrics as a function of variability conservation. Unbalanced data set.

	RGB		*Grayscale*	
Reduction	*BKG*	*No BKG*	*BKG*	*No BKG*
None	21675		7255	
0.99	874	685	581	593
0.95	177	196	90	156
0.9	47	83	26	67
0.75	3	20	2	17
0.5	1	4	1	3
0.25	1	1	1	1
0.1	1	1	1	1

Table 7 Number of components retained as a function of variability reduction. Balanced data set. With and without background (BK).

	RGB		*Grayscale*	
Reduction	*BKG*	*No BKG*	*BKG*	*No BKG*
None	21675		7255	
0.99	1163	951	741	789
0.95	188	211	99	164
0.9	46	82	27	66
0.75	2	18	3	15
0.5	1	3	1	2
0.25	1	1	1	1
0.1	1	1	1	1

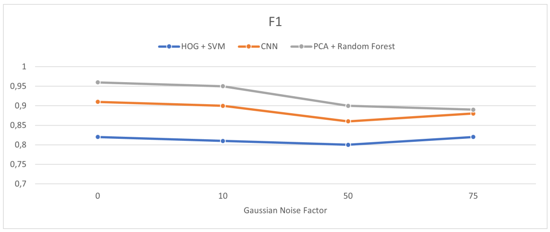
Table 8 Number of components retained as a function of variability reduction. Unbalanced data set. With and without background (BKG).

of image processing and machine learning. In real-world environments, images can be captured under less than ideal conditions, such as low lighting, lens distortions, or electromagnetic interference. Adding noise allows the simulation of these adverse conditions by introducing random variation in the pixel intensity of the image. In this particular case, samples from a normal distribution with different magnification factors were used to add noise to each pixel of the image. Subsequently, the results obtained with these noisy images are compared with the results of the original images to assess the robustness of the classification model. This analysis provides insights into the model's ability to generalise and maintain adequate performance in more challenging scenarios, which is crucial for effective real-world applications. By evaluating the quality of the results against the original images, we can determine whether the model is capable of handling and adapting to the inherent variability and complexity of real-world image conditions.

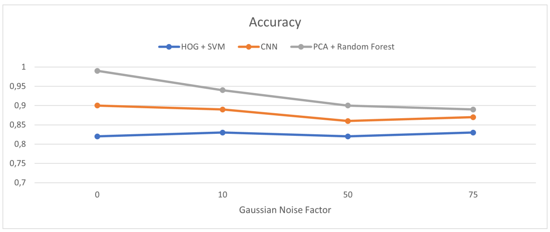
In Figure 10, the impact of using different Gaussian noise factors on the overall performance of the classifiers can be observed.

5 Discussion

Next, we present the best results achieved by the studied models and conduct a comparative analysis, considering the energy perspective. We draw conclusions regarding the suitability of each proposed method for the specific problem addressed in this study, which involves the identification of olive fruit fly specimens in real-life chromatic



(a) Gaussian noise effect on F1 metric



(b) Gaussian noise effect on Accuracy metric

Fig. 10 Effects of gaussian noise on methods performance

trap images. Additionally, we consider in our conclusions the fact that the final solution will be deployed on energy-harvesting computing platforms.

Tables 9 and 10 summarize the best results obtained from the three evaluated methods for each combination of input data. Among the eight testing scenarios, only one is selected (highlighted in bold type font) for later in depth analysis.

As can be seen, the PCA + Random Forest method obtains the best results. Above all, its reliability and stability stand out after training with an unbalanced dataset. For example, HOG+SVM and CNN can achieve good accuracy values in this case, but performs much worse in metrics such as Precision (which impacts F1 negatively). When training with a balanced dataset, CNN has an excellent performance, very close to that offered by PCA+Random Forest (although always below), but for a very specific case, which is the use of greyscale images and no background. Regarding the dataset to be used for the training and deployment of the models, it has been concluded that working with greyscale pictures results in a general improvement for all three methods. HOG + SVM and CNN achieve their best performance with images without background, whereas the opposite is true for PCA + Random Forest. Finally, the worst model in terms of detection performance and sensitivity to variations in the training data set is the HOG+SVM method.

Figure 11 details the behaviour of the three best cases selected for each method, showing the corresponding confusion matrixes and the behaviour of the ROC (Receiver Operating Characteristic) curve. As it can be seen, although the three models perform well, PCA+Random option stands out from the rest with an Area Under the Curve (AUC) of 1.

5.1 Performance analysis of the winner models

In this section, we delve into the details of the three top-performing models, each representing a different evaluated method, that have exhibited exceptional classification performance. By closely examining these models, our objective is to provide the reader with valuable insights and lessons learned that can be potentially applied to similar applications in the field of object detection, both in a general sense and specifically for insect detection (including other species).

SVM+HOG (Accuracy: 0.82 Precision: 0.90 Recall: 0.75 F1: 0.82)

The best results for this model are obtained by performing the training with the balanced dataset (2800 elements of each positive and negative classes) and by extracting the background from the images. With regard to the use of RGB or greyscale images, the results are very similar (although for B&W inputs a few extra decimals of accuracy are obtained). The tests carried out show that this model is particularly sensitive to training with images with the original background, which seems to indicate that it is particularly affected by noise, represented by the surface of the chromatic trap and other elements that may appear. In addition, the results suggest that the use of an unbalanced dataset also causes strong overfitting of the model.

CNN (Accuracy: 0.90 Precision: 0.88 Recall: 0.93 F1: 0.91)

The best results for this model were obtained by training with a balanced dataset (2800 elements of each positive and negative classes), with background extraction and greyscale images.

	RGB						Grayscale					
	Background			No Background			Background			No Background		
	SVM	CNN	PCA	SVM	CNN	PCA	SVM	CNN	PCA	SVM	CNN	PCA
Acc	0.70	0.85	0.94	0.82	0.87	0.94	0.66	0.86	0.92	**0.82**	**0.90**	0.95
Pre	0.84	0.83	0.91	0.90	0.86	0.91	0.79	0.82	0.88	**0.90**	**0.88**	0.92
Rec	0.46	0.87	0.99	0.73	0.89	0.98	0.43	0.92	0.96	**0.75**	**0.93**	0.98
F1	0.59	0.85	0.95	0.81	0.87	0.94	0.56	0.87	0.92	**0.82**	**0.91**	0.95

Table 9 Best results obtained by the three evaluated methods using a balanced data set.

	RGB						Grayscale					
	Background			No Background			Background			No Background		
	SVM	CNN	PCA	SVM	CNN	PCA	SVM	CNN	PCA	SVM	CNN	PCA
Acc	0.86	0.92	**0.99**	0.90	0.92	0.98	0.86	0.91	0.98	0.91	0.92	0.97
Pre	0.83	0.83	**0.97**	0.80	0.71	0.94	0.71	0.75	0.98	0.83	0.72	0.92
Rec	0.07	0.61	**0.95**	0.43	0.67	0.95	0.09	0.66	0.94	0.44	0.67	0.91
F1	0.13	0.70	**0.96**	0.56	0.69	0.95	0.16	0.70	0.96	0.58	0.69	0.91

Table 10 Best results obtained by the three evaluated methods using an unbalanced data set.

Tests with the unbalanced dataset show a slight overfitting of the network, resulting in a bias in the classification, which translates into worse results when running the model over the test dataset. Tests on the balanced dataset show better model performance with the set of images from which the background has been extracted.

PCA+Random (Accuracy: 0.99 Precision: 0.97 Recall: 0.95 F1: 0.96)

The best results for this model occur in the combinations of the unbalanced (2800 examples of positive class versus 18000 examples of negative class) and background-preserving datasets, with a small, almost negligible difference between the use of RGB and greyscale images. In general, the F1 metric of this model ranges between 0.91 and 0.96 for all configurations, thus outperforming the results of the previous two models. Contrary to SVM+HOG and CNN, this model seems to increase its performance as the number of samples used to train it increases, regardless of whether the dataset is balanced or not.

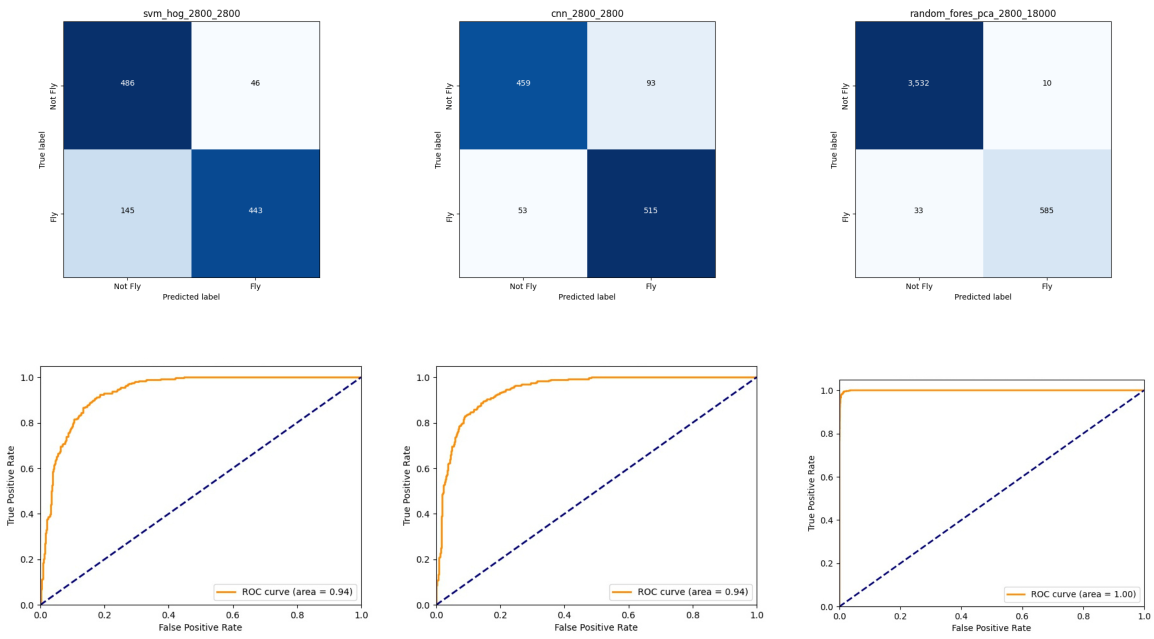
5.2 Energy efficiency analysis

In addition to the exceptional classification performance, computational efficiency plays a crucial role in the decision-making process for edge artificial intelligence solutions. This aspect becomes even more significant when considering the integration of classification models into low-cost, resource-constrained devices that rely on unstable energy sources such as solar panels. This work is conducted within the context of addressing these challenges and exploring the feasibility of deploying efficient and accurate classification models in such environments.

Model	Time (msecs)	Memory (MiB)	Power (A)	Energy (J)
SVM+HOG	883.54	127	0.631	2.681
CNN	46.71	101	0.638	0.143
PCA+Random	9.05	147	0.587	0.0256

Table 11 Classification time, peak memory usage, average intensity measured and energy of the three evaluated models. Average results per ROI classified, after 10 executions. Raspberry PI 4 average voltage measured during experiments 4.81V.

Table 11 presents a comprehensive overview of the execution times and memory usage for the three top-performing classification models investigated in this study. Despite the CNN model exhibiting significantly lower memory requirements than its counterparts, the trade-off between



(a) SVM+HOG (b) CNN (c) PCA+Random Forest

Fig. 11 Confusion matrix and ROC curve of the best cases for each method

execution time and memory favours the PCA-Random model. Notably, the PCA-Random model achieves an approximately 6-fold faster execution time compared to the LeNet-5 based neural network. Furthermore, when considering the investment of engineering time, the PCA model demonstrates a nearly 1% improvement in accuracy detection, further reinforcing its appeal. However, it is worth noting that memory usage should not be the sole determinant in selecting the most suitable method.

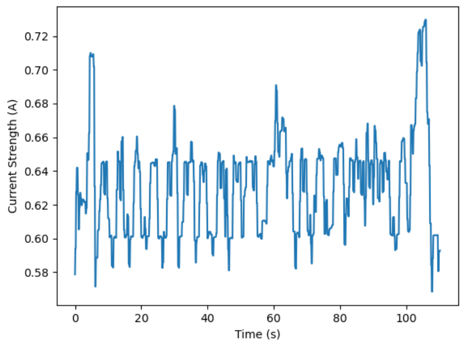
Power consumption in the Raspberry Pi 4 was also measured as it is a crucial consideration in our work for making informed decisions. The average power intake was observed to be approximately 5% lower in the case of PCA-Random, which can be attributed to the lower computational intensity of its operations. To minimise potential interference from operating system tasks or other sources of varying workload, we disabled unnecessary peripherals, the graphical subsystem, and network functionalities.

Figure 12 shows the variation in amperage during computation for the three scenarios and the details for a single ROI classification in the case of SVM+HOG (Figure 12b) for clarity.

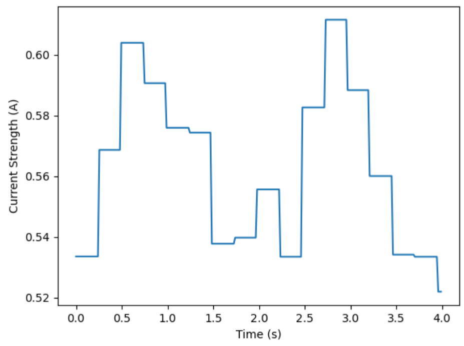
The characterisation of the models would allow the engineer to build different application profiles and select one depending on the execution context. For example, if memory is a bottleneck in our system, we could prioritise such parameters over classification performance and processing time (e.g. CNN model will be selected). However, if energy is a scarce resource (i.e. edge device uses solar panels), the PCA-Random model will be more suitable.

6 Conclusions

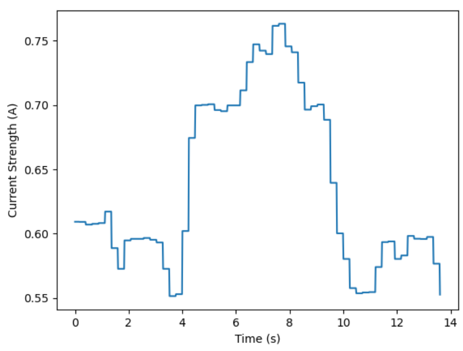
The performance of an artificial intelligence model for a specific application depends on multiple factors. In this study, we aim to evaluate the main supervised learning methods for the classification of the olive fly, analyzing their behavior under different working conditions (input image format, background consideration, light variation, etc.). The objective is to determine the best strategy for deploying a solution to control insect pest populations on edge devices, where resources are limited, without sacrificing accuracy and ensuring proper field operation. Within these systems, image capture conditions are a crucial aspect influenced by



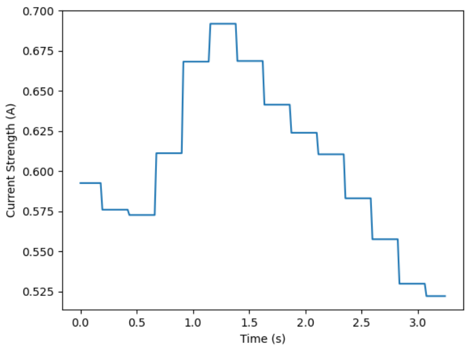
(a) SVM+HOG



(b) SVM+HOG (single ROI profile)



(c) CNN



(d) PCA+Random

Fig. 12 Power consumption profile for one test execution (116 ROIs classified)

environmental characteristics. Additionally, it is necessary to address the various parameter combinations used to select and configure the employed methods. Moreover, as the main objective of the study is to implement these techniques on energy-harvesting devices, another significant challenge is to achieve an optimal balance between model energy consumption and the accuracy they provide. Through a detailed study of the impact of parameter variation in each model, the best option was selected based on the F1 score, rather than relying solely on classification accuracy metric. The combination of PCA+Random Forest not only achieves the highest classification rates (99%), but also requires minimal processing time, resulting in lower energy demand. Therefore, this method was ultimately chosen to build a functional prototype on a Raspberry PI 4 computer board powered by solar panels, intended to be fully autonomous for extended periods.

6.1 Future Research Directions

Considering the limitations identified in this study and the potential extensions of our research, several avenues for future work in this field are open.

First, the exploration of automatic and systematic configuration parameter tuning for these algorithms is a crucial step towards obtaining solutions that aim to strike a balance between result accuracy and energy consumption.

Similarly, exploring a broader range of Artificial Neural Networks (ANN) models, especially those inspired by the brain, is crucial. As highlighted in Section 2, SNNs emerge as an appealing technology, particularly in applications with power constraints like the BioTrap 4.0 device. However, additional efforts are necessary to effectively encode static pictures into a time-based event input, as required by SNNs. Additionally, the exploration of suitable training and learning methods is pivotal for the success of this type of ANN and must be customized for each specific application. Furthermore, exploring optimized versions of the YOLO (*You Only Look Once*) object detection algorithm is worthwhile. Variants like Tiny YOLO or YOLO Nano are specifically designed for systems with limited memory or computing power. Initially, the CNN architecture complexity of these YOLO versions remains higher than that of the proposed LeNet-5. However, by assessing improvements in accuracy and analyzing overall processing time (given YOLO's ability to bypass the need for a pre-processing and segmentation stage), conclusive insights can be drawn in the opposite direction.

Lastly, the expansion of the study to a larger number of devices would require adapting the methods used to the available resources. By addressing these aspects, further advancements can be made in the development of efficient and accurate solutions for pest population control using edge devices with limited resources.

Supplementary information.

7 Funding

This work was partially funded by the Spanish Ministry of Economy and Competitiveness (MINECO) under project TALENT-BELIEF (Grant number PID2020-116417RB-C44) and by the European Commission under Horizon 2020 project SHAPES (Grant number 857159).

8 Statements and declarations

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

9 Consent for publication

Not applicable.

10 Ethics approval and consent to participate

Not applicable.

11 Availability of data and materials

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Acknowledgments. The authors are grateful to the staff of the Environmental Research Center "El Chaparrillo" of the Regional Government of Castilla-La Mancha for supporting the field visits and performing the expert classification and supervision of the datasets used in this work.

References

[1] Kang, S.-H., Cho, J.-H., Lee, S.-H.: Identification of butterfly based on their shapes when viewed from different angles using an artificial neural network. Journal of Asia-Pacific Entomology **17**(2), 143–149 (2014). https://doi.org/10.1016/j.aspen.2013.12.004

[2] Wen, C., Wu, D., Hu, H., Pan, W.: Pose estimation-dependent identification method for field moth images using deep learning architecture. Biosystems Engineering **136**, 117–128 (2015). https://doi.org/10.1016/j.biosystemseng.2015.06.002

[3] Gondal, D., Khan, Y.N.: Early pest detection from crop using image processing and computational intelligence. FAST-NU Research Journal ISSN: 2313-7045 **1** (2015)

[4] Venugoban, K., Ramanan, A.: Image classification of paddy field insect pests using gradient-based features. International Journal of Machine Learning and Computing, 1–5 (2014). https://doi.org/10.7763/ijmlc.2014.v4.376

[5] Zhang, L., Zhang, Z., Wu, C., Sun, L.: Segmentation algorithm for overlap recognition of seedling lettuce and weeds based on svm and image blocking. Computers and Electronics in Agriculture **201**, 107284 (2022). https://doi.org/10.1016/j.compag.2022.107284

[6] Kuzuhara, H., Takimoto, H., Sato, Y., Kanagawa, A.: Insect pest detection and identification method based on deep learning for realizing a pest control system. In: 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp. 709–714 (2020). https://doi.org/10.23919/SICE48898.2020.9240458

[7] Türkoğlu, M., Hanbay, D.: Plant disease and pest detection using deep learning-based features. Turkish Journal of Electrical Engineering and Computer Sciences **27**, 1636–1651 (2019). https://doi.org/10.3906/elk-1809-181

[8] Hassan, S.N.A., Rahman, S.A., Zaw, Z., Shoon, H., Win, L.: Automatic classification of insects using color-based and shape-based descriptors. International Journal of Applied Control, Electrical and Electronics Engineering (IJACEEE) **2** (2014)

[9] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[10] Rustia, D.J., Chao, J.-J., Chiu, L.-Y., Wu, Y.-F., Chung, J.-Y., Hsu, J.-C., Lin, T.-T.: Automatic greenhouse insect pest detection and recognition based on a cascaded deep learning classification method. Journal of Applied Entomology, 1–17 (2020). https://doi.org/10.1111/jen.12834

[11] Xia, J., Du, P., He, X., Chanussot, J.: Hyperspectral remote sensing image classification based on rotation forest. IEEE Geoscience and Remote Sensing Letters **11**, 239–243 (2014). https://doi.org/10.1109/LGRS.2013.2254108

[12] Rodarmel, C., Shan, J.: Principal component analysis for hyperspectral image classification. Surv Land inf Syst **62** (2002)

[13] Makarichian, A., Chayjan, R.A., Ahmadi, E., Zafari, D.: Early detection and classification of fungal infection in garlic (a. sativum) using electronic nose. Computers and Electronics in Agriculture **192**, 106575 (2022). https://doi.org/10.1016/j.compag.2021.106575

[14] Iakymchuk, T., Rosado, A., Guerrero-Martínez, J., Bataller-Mompeán, M., Frances-Villora, J.V.: Simplified spiking neural network architecture and stdp learning algorithm applied to image classification. EURASIP Journal on Image and Video Processing **2015** (2015). https://doi.org/10.1186/s13640-015-0059-4

[15] Lee, J.H., Delbruck, T., Pfeiffer, M.: Training deep spiking neural networks using backpropagation. Frontiers in Neuroscience **10** (2016). https://doi.org/10.3389/fnins.2016.00508

[16] Yang, S., Pang, Y., Wang, H., Lei, T., Pan, J., Wang, J., Jin, Y.: Spike-driven multi-scale learning with hybrid mechanisms of spiking dendrites. Neurocomputing **542**, 126240 (2023). https://doi.org/10.1016/j.neucom.2023.126240

[17] Yang, S., Linares-Barranco, B., Chen, B.: Heterogeneous ensemble-based spike-driven few-shot online learning. Frontiers in Neuroscience **16**, 850932 (2022). https://doi.org/10.3389/fnins.2022.850932

[18] Liu, Y., Cao, K., Wang, R., Tian, M., Xie, Y.: Hyperspectral image classification of brain-inspired spiking neural network based on attention mechanism. IEEE Geoscience and Remote Sensing Letters **19**, 1–5 (2022). https://doi.org/10.1109/LGRS.2022.3172410

[19] Safa, A., Bourdoux, A., Ocket, I., Catthoor, F., Gielen, G.G.E.: On the use of spiking neural networks for ultralow-power radar gesture recognition. IEEE Microwave and Wireless Components Letters **32**(3), 222–225 (2022). https://doi.org/10.1109/LMWC.2021.3125959

[20] Liu, Y., Tian, M., Liu, R., Cao, K., Wang, R., Wang, Y., Zhao, W., Zhou, Y.: Spike-based approximate backpropagation algorithm of brain-inspired deep snn for sonar target classification. Computational Intelligence and Neuroscience **2022**, 1–11 (2022). https://doi.org/10.1155/2022/1633946

[21] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791