International Journal of Technology

http://ijtech.eng.ui.ac.id

Research Article

Fruit Fly Detection Based on Wingbeat Sound Using Embedded Artificial Intelligence

Van-Khanh Nguyen[1,2*], Vy-Khang Tran[1], Chi-Ngon Nguyen[3]

[1]Programmable Logic Controller Technology and IIoT Laboratory, College of Engineering, Can Tho University, Can Tho 94119, Vietnam

[2]College of Engineering, Can Tho University, Can Tho 94119, Vietnam

[3]Nam Can Tho University, Can Tho 94118, Vietnam

[*]Corresponding author: vankhanh@ctu.edu.vn

Abstract: The highly invasive oriental fruit fly has caused significant agricultural losses worldwide. Electronic traps have been widely studied for fruit fly detection and counting. However, research focusing on applying acoustic sensors to identify fruit flies based on their wingbeat sound is currently lacking. This study focused on identifying trapped oriental fruit flies based on wingbeat sound data. An acoustic sensor was integrated into the funnel trap to record the wingbeat sounds of trapped flies along with ambient environmental noise. The trap was deployed in an apple orchard for two months to collect data. A spectrogram transformation and Mel-filter bank were applied to process the captured audio, generating two distinct sets of spectrogram images. A deep learning model based on convolutional neural network architecture was then designed and deployed on an ESP32 microcontroller to classify the wingbeat sounds of fruit flies and other environmental sounds. The trained model's field experiment in the orchard showed that the model could classify the sound of fruit fly wingbeat in real-time audio streams with an accuracy of up to 96.86%. This demonstrates the practical applicability of the sound-sensor-based fruit-fly identification method. In addition, implementing the deep learning model on a microcontroller results in a compact, low-power, and cost-effective electronic trap. As a result, the compact design and low power consumption make this solution a promising approach for real-time monitoring and early pest detection in agricultural environments. However, its broader applicability requires further validation across more diverse datasets, longer deployment periods, and varying environmental conditions.

Keywords: Automation trap; Acoustic sensor; CNN architecture; Embedded system; Pest detection

1. Introduction

Tephritid fruit flies are major agricultural pests distributed worldwide, with species diversity that continues to expand each decade (Trombik et al., 2023). They exhibit endophytic behavior, causing quantity and quality yield losses. Therefore, they pose a serious threat to global fruit and vegetable production. (Opoku et al., 2025; Jaffar et al., 2023; Kibira et al., 2015). Bactrocera dorsalis (Hendel, 1912) (*Diptera: Tephritidae*), commonly known as the oriental fruit fly, is one of the most common and dangerous fruit flies. They appear in major quarantine regulations in many countries and have been reported in 75 countries worldwide (Zeng et al., 2019). They have been reported to attack over 481 host plant species (Liquido et al., 2017), including many economically valuable fruit trees, such as mango, banana, guava, and orange (Wijekoon et al., 2024; Mutamiswa et al., 2021). In Vietnam, oriental fruit flies are the primary fruit pest, infesting orchards across vast areas. They are prevalent in many regions, particularly impacting high-value export fruit crops in the Mekong Delta (Tran and Nguyen, 2023; Long et al., 2022; Oanh and Duc, 2020; Hien et al., 2019). These studies have demonstrated the destructive

https://doi.org/10.14716/ijtech.v17i3.8284
Received December 2025; Revised February 2026; Accepted April 2026

potential of *B. dorsalis* and emphasized the need for fruit fly control and prevention measures.

Many measures have been implemented to eliminate and prevent the impact of fruit flies. In particular, IPM has gained widespread adoption due to its notable benefits, such as reducing the amount of pesticides used, protecting natural enemies, significantly reducing losses caused by fruit flies, and increasing farmers' incomes (Muriithi et al., 2016). However, traditional IPM relies on human labor to control and kill fruit fly populations. Manual measures are tedious, labor-intensive, and prone to error. The monitoring process can be improved by integrating electronic traps equipped with multiple sensors to address these limitations, which may improve accuracy and efficiency (Lello et al., 2023). Electronic traps have been developed to improve traditional insect traps by integrating sensing circuits to automatically detect and count insects (Lello et al., 2023; Navarro-Llopis and Vacas, 2014). Although detection is relatively simple, accurate identification and counting remain challenging. Two primary approaches have been explored: wingbeat frequency analysis (Kalfas et al., 2022; Sandrini-Moraes et al., 2019; Potamitis et al., 2014) and image-based recognition (Hahn et al., 2023; Molina-Rotger et al., 2023; Huang et al., 2021; Le et al., 2021; Martins et al., 2019). However, both approaches have inherent limitations. Wingbeat-based methods are influenced by differences in age and sex and often fail when flies crawl into traps instead of flying. On the other hand, image-based traps typically rely on sticky surfaces to capture flies for later analysis. These sticky surfaces degrade and accumulate debris over time, making recognition increasingly unreliable. Moreover, image processing requires substantial computational resources and energy, which often require minicomputers or cloud servers, which are costly and unsuitable for large-scale, battery-powered field systems. These limitations highlight the need for a more efficient and autonomous solution that can directly operate on low-power microcontrollers.

Acoustic sensing is a promising alternative for real-world deployment. Fruit flies produce characteristic wingbeat sounds within specific frequency ranges (Nguyen et al., 2025; Mankin et al., 2006), which can serve as distinctive features for identification. Acoustic features have been applied in various applications, such as mosquito presence detection, insect classification (Khalid et al., 2025; Shetty and Kumar, 2025), and pest detection (Ali et al., 2025). Previous studies have explored acoustic-based insect detection using traditional signal processing techniques, such as discriminant analysis for termite detection (Nanda et al., 2018), demonstrating the feasibility of distinguishing insect-generated signals from environmental noise in practical sensing systems. However, most of these studies rely on existing datasets or data collected under controlled conditions and are not yet ready for use in real-world environments. Moreover, the majority of research focuses on evaluating the feasibility of PCs or minicomputers; therefore, deployment on resource-constrained devices, such as microcontroller-based platforms, would further enhance the practical applicability of such systems. The presence of diverse background noises poses additional challenges in orchard environments, emphasizing the importance of collecting and analyzing data under real conditions to develop robust identification algorithms. Acoustic-based monitoring could reduce maintenance demands and eliminate the need for sticky traps, thereby lowering operational costs.

Recent advances in deep learning have further enhanced this potential. Convolutional neural networks (CNNs), widely used in audio classification and speech recognition (Abdel-Hamid et al., 2013), have recently been applied to insect sound analysis (Varma et al., 2021). CNNs trained on wingbeat spectra have demonstrated promising results for fruit flies (Kalfas et al., 2022). Mel-spectrogram representations can better capture relevant frequency features, improving robustness under noisy outdoor conditions. Combining acoustic sensing with CNN-based models offers a practical path toward efficient, low-cost, and autonomous fruit fly identification in real agricultural environments.

Most edge devices are based on processors with limited memory and computational resources. Microcontrollers (MCUs) help reduce energy consumption and prolong device operating time. However, these devices generally have very limited memory, which poses challenges for integrating neural network models into practical applications. For example, the ESP32, a

widely used MCU in edge devices, provides only 520 kB of SRAM and typically 4 MB of onboard flash memory (Espressif Systems, 2021). Although the ESP32 has been used to deploy TinyML models, memory constraints remain a significant challenge in system design.

The objective of this study is to develop a CNN-based deep learning model capable of running on microcontrollers to identify fruit flies by their wingbeat sounds under real-world conditions. The integration of a microphone into an electronic trap, enabling continuous sound acquisition in orchard environments where both insect signals and background noise coexist, is a key feature of this work. This study emphasizes the practical deployment of the model on resource-constrained devices, providing a more realistic evaluation of its effectiveness compared to simulations. Ultimately, this work provides an initial step toward acoustic-based monitoring of fruit fly populations; however, the current experimental scope limits its broader applicability and requires further validation under more diverse and long-term field conditions.

2. Methods

2.1 Electronic Trap Design

We developed an electronic fruit fly trap based on the design proposed in our previous study (Nguyen et al., 2025). The trap structure followed a conventional model, with an additional middle compartment inserted between the lid and body to house the sensing modules (Figure 1). To detect insect entry and exit, two pairs of infrared transceivers were mounted at the entrance of a funnel-shaped tube, while a microphone placed near the attractant source captured wingbeat sounds. An ESP32 microcontroller continuously recorded audio signals at 16 kHz and infrared sensor data to a microSD card. The daily energy consumption of the trap is around 120 mAh. Powered by batteries, the trap operated autonomously in a 2000 m² apple orchard (Figure 1) from January 17 to February 17, 2025, and was active from 6:00 AM to 6:00 PM to align with fruit fly activity patterns. Approximately 100 audio files were generated daily and processed using Audacity (Audacity Team, 2017).

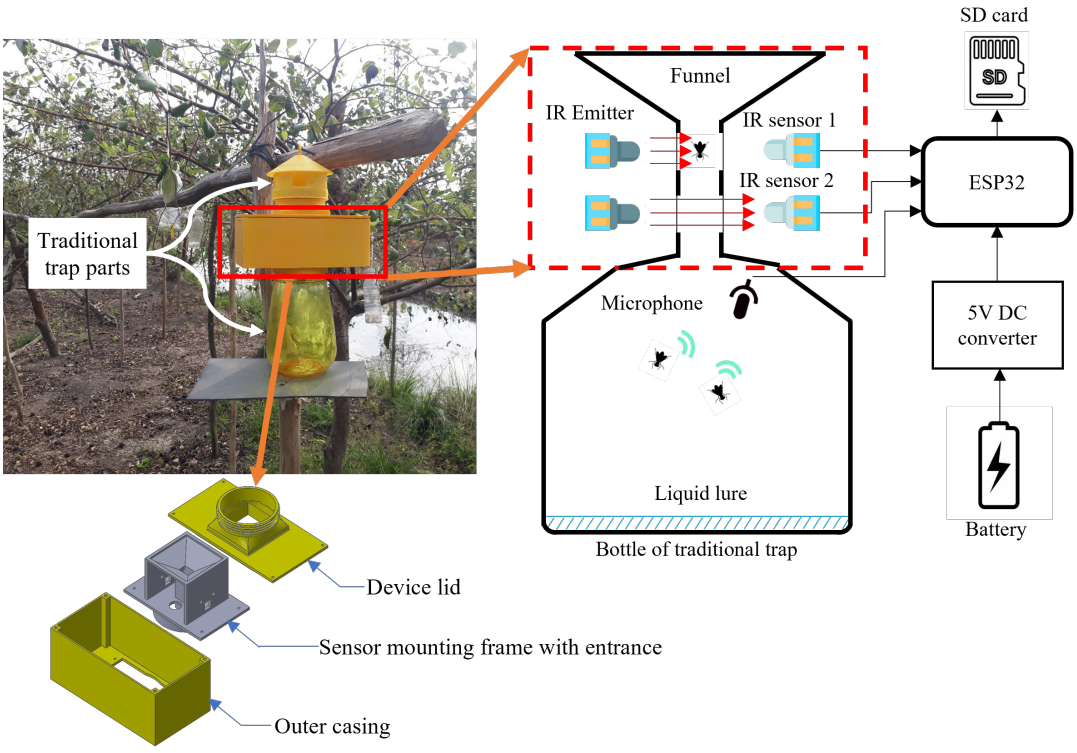

Figure 1 Experimental setup and electronic trap design

A detailed description of the wingbeat sound dataset and its spectral characteristics was presented in our previous work (Nguyen et al., 2025). In summary, the dataset included over

1,100 annotated wingbeat and non-wingbeat sound segments, each 400 ms in duration, stored in WAV format. The analysis revealed two primary frequency groups: a low-frequency band below 1.5 kHz, characterized by a fundamental frequency around 168 Hz and its harmonics, and a high-frequency band between 5 and 8 kHz. These acoustic patterns were found to be distinctive compared with other ambient sounds, such as bird calls, dog barking, and machinery noise. The current study expands on that dataset by incorporating additional field recordings and using these characteristic frequency groups as key features for machine learning–based classification.

2.2 Real-Time Spectrogram Imaging Algorithm

This study builds upon a previously collected acoustic dataset (Nguyen et al., 2025) to develop a recognition framework for fruit fly wingbeat sounds. The time-domain signals were converted into spectrogram images and used as input to a convolutional neural network (CNN). Two filtering approaches were applied during spectrogram generation: the Mel filter and a simple average filter. The Mel filter compresses the frequency information by grouping neighboring frequencies into a single pixel, emphasizing key low-frequency components characteristic of wingbeat sounds (Warden and Situnayake, 2019). Conversely, an IM filter can be applied if dominant features occur in higher frequency regions. The simple average filter further reduces spectrogram dimensionality by averaging adjacent frequency bins while preserving the spectral characteristics identified in our previous analysis (Nguyen et al., 2025).

Each spectrogram image consisted of 20 rows, corresponding to a signal of 400 ms in the time domain. Because fruit fly wingbeat events typically occur over short durations, a sliding window of 400 ms was adopted to capture each event, as shown in Figure 2. This window length also provided a practical balance between temporal resolution and memory usage for microcontroller-based implementation. Each window contained 6,400 samples at a sampling rate of 16 kHz, which can be efficiently processed in real time.

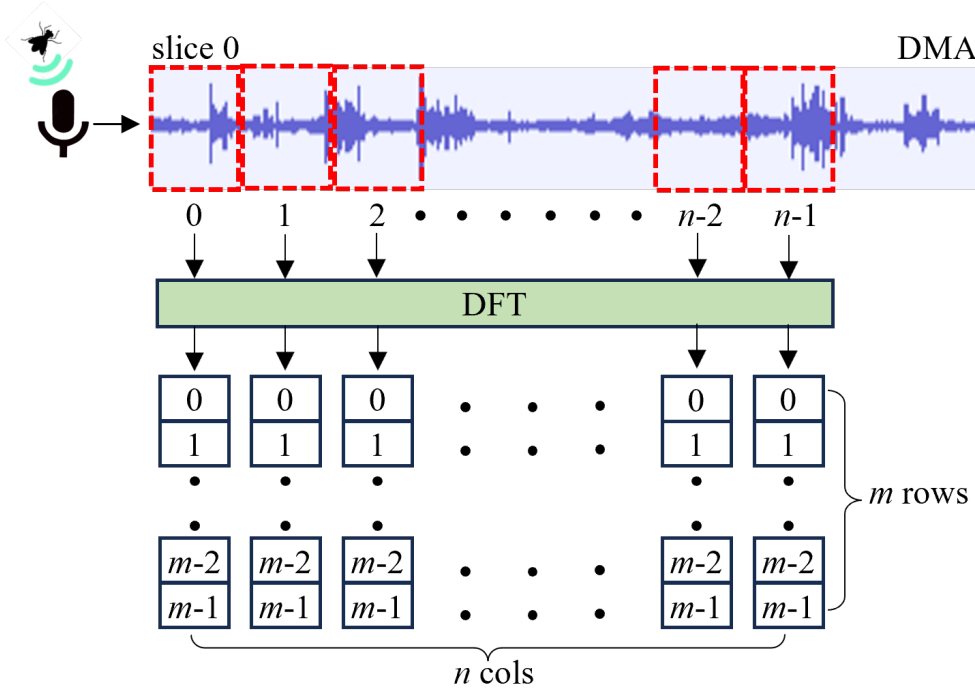

Figure 2 Spectrogram of the imaging mechanism

The number of spectrogram columns depended on the filter type: spectrograms generated using the Mel filter contained 100 columns, whereas those produced by the average filter contained 128 columns. Figure 2 shows the real-time spectrogram generation process. The audio signals captured by the INMP441 MEMS microphone were transferred to the ESP32 via the I2S interface and stored in the DMA buffer. The buffered audio signal was segmented into 20-

ms frames, each of which was transformed into a spectrogram column using a discrete Fourier transform (DFT). The resulting columns were updated in a rolling manner using 400 ms sliding windows to maintain continuous real-time visualization without memory overflow. First, the system applies the DFT to transform these data into 512 frequency components using the following formula (Oppenheim and Schafer, 2009):

$$X_k = \sum_{n=0}^{N-1} A(k)\, W^{jk}, \qquad j = 1, 2, 3, \ldots, N-1 \tag{1}$$

where $A(k)$ is a complex number, and W is calculated as follows:

$$W = e^{2\pi i/N} \tag{2}$$

The power of each frame is calculated using the following formula:

$$P_i(k) = \frac{1}{N}|X_k|^2 \tag{3}$$

Because the result is symmetric, only 256 frequency components are used. This study uses the Kiss FFT library (Bogerding, 2017) to perform the DFT transformation because this library is optimized for MCU and runs efficiently for non-power-of-2 data sample numbers. A filter is then applied to reduce the image column size. Two filters are applied. The first filter is the Mel filter, which is used to group frequency bands into n groups. The formula for converting the frequency scale from Hertz to the Mel scale is presented as follows (Abdul and Al-Talabani, 2022):

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right) \tag{4}$$

The inverse transformation is given by the formula:

$$f = 700\left(10^{\frac{m}{2595}} - 1\right) = 700\left(e^{\frac{m}{1127}} - 1\right) \tag{5}$$

The frequencies f_{high} and f_{low} are converted to the Mel scale to determine the upper M_{high} and lower M_{low} bounds of the filter bank, m linearly spaced values within the range $[M_{low} : M_{high}]$ are calculated and converted back to the frequency scale using the inverse transformation formula. Subsequently, the filters are generated using the following formula:

$$H_m(k) = \begin{cases} 0, k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \le k \le f(m+1) \\ 0, k > f(m+1) \end{cases} \tag{6}$$

where m is the desired number of filters and f() is the list of frequencies spaced by $m + 2$ Mel. In this study, M equals 100 or the filter has 100 frequency groups. For the averaging filter, two adjacent frequencies are averaged to reduce from 256 to 128.

The frequency spectrum generation algorithm was installed and executed directly on the MCU to ensure that the model was trained and achieved good results on the embedded system. The program converts the spectrum from audio clips stored on a memory card and saves it as a dataset. Furthermore, the program can be transformed in real time in parallel with the data acquisition process so that the system can perform recognition with real-time data streams.

2.3 Overview of the implementation method

This study was conducted in three main stages to identify fruit flies using embedded AI (Figure 3). In stage 1, traps were deployed in the fruiting apple orchard to record sounds for a month. Then, the recorded sounds, such as fruit fly wingbeats and bird sounds, glass trimmer sounds, and dog barking, were extracted manually. These sounds were then generated into spectrograms and divided into two groups to train the deep-learning models to classify them. In stage 2, two CNN-based deep learning models were proposed to classify fruit fly wingbeat and other sounds. At this stage, the models were created on a personal computer and trained using the training dataset extracted from the collected dataset. Then, the models were evaluated using the test dataset, and additional data were collected. If the accuracy of the model is not achieved at this stage, parameters such as the kernel size, number of features, number of convolutional layers, and drop-out ratio are used to improve the accuracy on the embedded system. In the final stage, the post-quantization model is embedded in the MCU on the trap to conduct real-time testing of the embedded system. There are two types of tests: static and field tests. The static test is a post-quantization model that is evaluated with a set of spectral images stored in the memory card, similar to the set used on the personal computer. The field test is a model inferred from the actual sound stream when the trap is placed in the garden. Both the classification results and the corresponding audio data are stored on the memory card for later evaluation and accuracy assessment. After these three stages, the machine learning model can be used to develop related applications.

2.4 Deep learning network model for fruit fly recognition based on wingbeat signals

This research aims to identify fruit flies based on their wingbeat sounds using embedded artificial intelligence. The goal is to directly integrate AI into an embedded system for real-time fruit fly detection. Currently, no flexible tool is available that allows both training and inference to be performed entirely on embedded hardware. As a result, this study uses TensorFlow Lite to deploy a post-quantized AI model onto the embedded system. A CNN model is proposed based on two collected datasets, one containing wingbeat sounds and the other consisting of non-wingbeat sounds, to classify and distinguish between the two categories. Suppose the two labels to be classified are denoted as $C = c_1, c_2$ with c1 and c2 representing wingbeat_sound and other_sounds, respectively. The training dataset after labeling is denoted as $X = x_i, Y_i | 1 \leq i \leq m$ with m is the number of training data samples, x_i and Y_i are the spectrogram and its corresponding labels or $Y_i \epsilon$ C. The CNN model to classify the above two labels with the parameter set W is mathematically expressed as follows:

$$\mathcal{F}(X|W) = f_n\left(\cdots f_2\left(f_1(X|W_1)|W_2\right)\cdots|W_n\right) \tag{7}$$

Where $f_j(X|W_i)$ is the j^{th} layer of the n-layer model. The structure of the two CNN network models proposed in this study is illustrated in Table 1. The two models have the same layer structure except for the input size. Models 1 and 2 have input sizes of 20 100 (Mel filter spectrogram) and 20 128 (average filter spectrogram), respectively. The number of filters and the output size of each layer is shown in the adjacent box. The model consists of two convolutions alternating with two max-pooling layers, a flatten layer followed by a drop-out layer with a ratio of 50%, and a fully connected (FC) layer. A convolutional layer can be mathematically represented as follows:

$$f_j(X_i|W_i) = h(\theta * X_i + b) \tag{8}$$

where θ are the kernels or filters, and b and $h(\cdot)$ are the bias and activation functions, respectively. The convolutional and Max-pooling layers use kernels of size 3×3 and 2×2, respectively.

The activation function h uses the rectified linear unit (ReLU) function with the following formula:

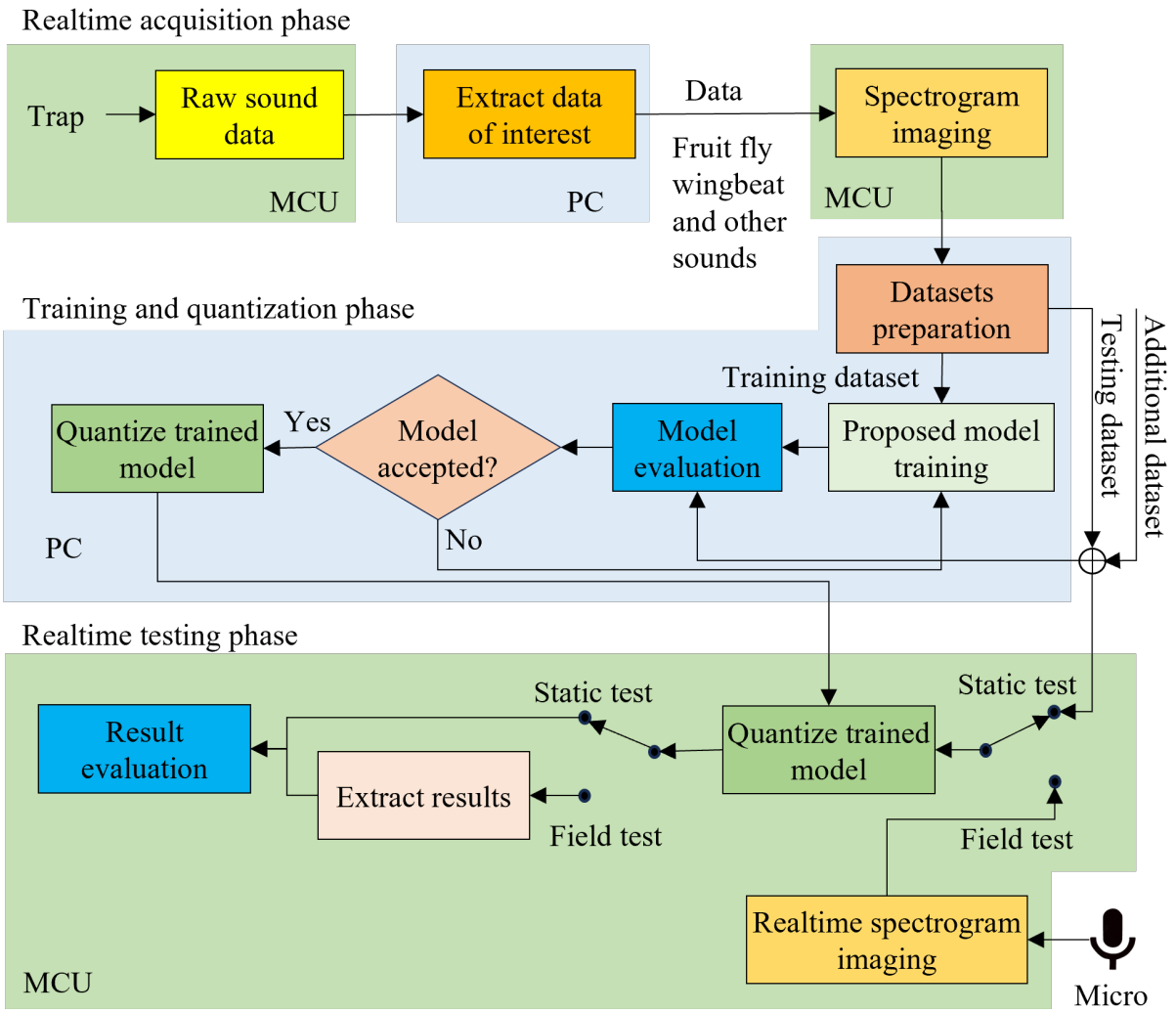

Figure 3 Overview of the proposed research methodology. The framework consists of three integrated phases: (1) Realtime acquisition phase for capturing raw sound data via MCU and generating spectrograms on PC; (2) Training and quantization phase where the model is trained, evaluated, and compressed (quantized) on a PC platform; and (3) Realtime testing phase involving the deployment of the quantized model onto the MCU for performance validation through static and field tests

Table 1 Proposed CNN models.

Layer	Activation function	Output shape	
		Model 1	Model 2
Input	-	(20, 100)	(20, 128)
Conv2D	ReLU	(16, 96, 8)	(16, 124, 8)
MaxPooling2D	-	(8, 48, 8)	(8, 62, 8)
Conv2D	ReLU	(6, 46, 16)	(6, 60, 16)
MaxPooling2D	-	(3, 23, 16)	(3, 30, 16)
Flatten	-	(1104)	(1140)
Dropout (0.5)	-	(1104)	(1104)
Dense (FC)	ReLU	(32)	(32)
Dense (FC)	Softmax	(2)	(2)

$$h(x) = \max(0, x) \tag{9}$$

For the FC layer, the first Dense layer has 32 outputs and also uses the ReLU function, whereas the second Dense layer uses the Softmax function with cross-entropy for the classification problem with output posterior probability, which is mathematically expressed as follows:

$$f_i(z) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{10}$$

Here, z are the CNN scores inferred for each label in C. The loss function is computed as follows:

$$Loss = -\log\left(\frac{e^{z_p}}{\sum_j e^{z_j}}\right) \tag{11}$$

where z_p is the model score of the positive class. Both models will be created, trained, and tested with the dataset in Table 2 before being quantized for real-time testing embedding in MCU.

Standard CNN architectures are typically designed for high-resource platforms and cannot be directly deployed on constrained microcontrollers such as the ESP32 due to limited SRAM and Flash capacity, which are insufficient to store full-precision weights and intermediate activations. To address this, model compression techniques, such as quantization or pruning, are required. In this work, the default Post-Training Quantization (PTQ) provided by TensorFlow is applied to convert model parameters from 32-bit floating-point to 8-bit integer representation, significantly reducing memory usage and enabling efficient inference (Google, 2024). The PTQ method is computed based on the following transformation:

$$q = round\left(\frac{x}{s}\right) + z \tag{12}$$

Here, x denotes the original floating-point value, q is the quantized integer value, s is the scaling factor, and z is the zero-point that maps zero in floating-point to an integer value. This transformation allows the model to operate using low-precision arithmetic while preserving numerical fidelity, making it suitable for deployment on memory-constrained embedded devices.

2.5 Evaluation indicators

This study uses a confusion matrix to evaluate the performance of the trained DL network model. Accuracy (ACC), precision *(P)*, recall *(R)*, and F1-score *(F1)* are calculated from the confusion matrix using formulas (13), (14), (15), and (16):

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$P = \frac{TP}{TP + FP} \tag{14}$$

$$R = \frac{TP}{TP + FN} \tag{15}$$

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{16}$$

Here, *TP, TN, FP,* and *FN* are the true positive, true negative, false positive, and false negative values, respectively. Specifically, *TP* refers to cases where the model correctly recognizes fruit fly wingbeat sounds, and *TN* represents cases where the model correctly identifies non-wingbeat sounds. In contrast, *FP* and *FN* correspond to incorrect classifications. *FP* occurs

when non-wingbeat sounds are mistakenly identified as wingbeats, whereas *FN* occurs when wingbeat sounds are not correctly recognized. In this study, *P, R,* and *F1* are reported as macro-averaged values, denoted as *MP, MR,* and *MF1,* respectively, and calculated according to the following equations:

$$MP = \frac{P_{wing\_beat} + P_{other\_sounds}}{2} \tag{17}$$

$$MR = \frac{R_{wing\_beat} + R_{other\_sounds}}{2} \tag{18}$$

$$MF1 = \frac{F1_{wing\_beat} + F1_{other\_sounds}}{2} \tag{19}$$

3. Results and Discussions

3.1 Spectrogram conversion results

Figure 4 illustrates two types of spectrograms of the fruit fly wingbeat sounds used in this study. Figure 4a shows the spectrogram generated using the Mel filter, which emphasizes low-frequency components and suppresses high-frequency ones according to the Mel-scale characteristics. Figure 4b shows the spectrogram generated using the average filter, which retains all the major frequency bands of the unfiltered signal (Figure 4c). Only Mel-filtered and average-filtered spectrograms were employed to train and evaluate the fruit fly recognition CNN model. The spectrograms were derived from the previously established annotated audio dataset. Each spectrogram was categorized into two classes: wingbeat and non-wingbeat sounds, with 1000 images per class. Non-wingbeat spectrograms are generated from bird call and machinery noise recordings, with 500 samples for each category. These sound sources were selected due to their prevalence in the study area's orchards. The dataset was randomly divided into 60% for training and 40% for internal testing.

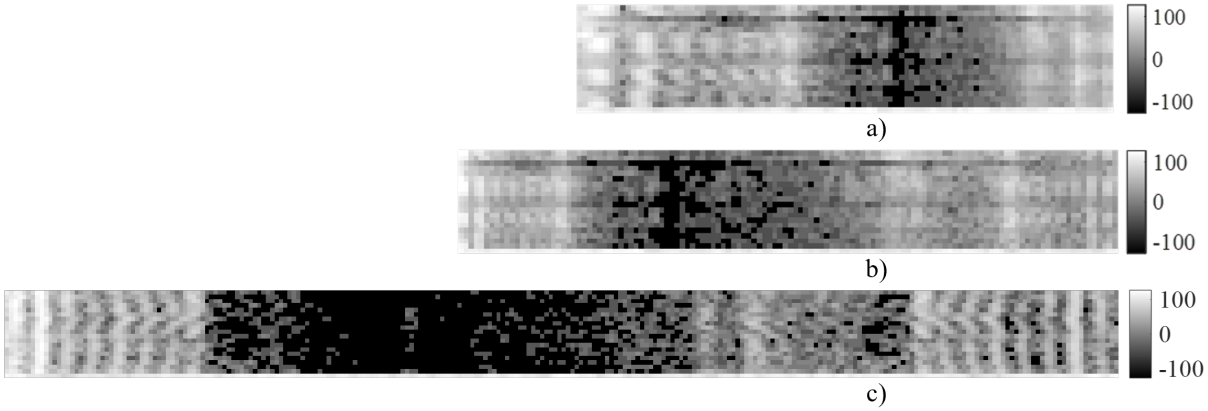

Figure 4 Spectrogram conversion results: (a) Mel-filter spectrogram, (b) Average-filter spectrogram, (c) Original spectrogram

An independent test dataset consisting of 1000 newly collected spectrograms for each class was prepared using additional audio recordings to further assess the generalization performance of the CNN. These new samples were not included in the training process, resulting in 2800 images used exclusively for the testing dataset (Table 2).

3.2 Model training and quantization results

All parameters of the layers in the CNN model were tuned using a trial-and-error approach to achieve optimal training and testing accuracy while minimizing overfitting. The two models

Table 2 Distribution of audio samples across the training, testing, and supplementary test datasets

Data label	Train set	Test set	Supplementary test dataset
wingbeat sound	600	400	1000
other sounds	600	400	1000

were developed and trained using the root mean squared propagation optimizer on Google Colab with a learning rate of 0.001. Figure 5 presents the training results and confusion matrices obtained by testing the models on 1400 samples. Figures 5a and 5b illustrate the training performance of the two models after 40 epochs, demonstrating good convergence without overfitting.

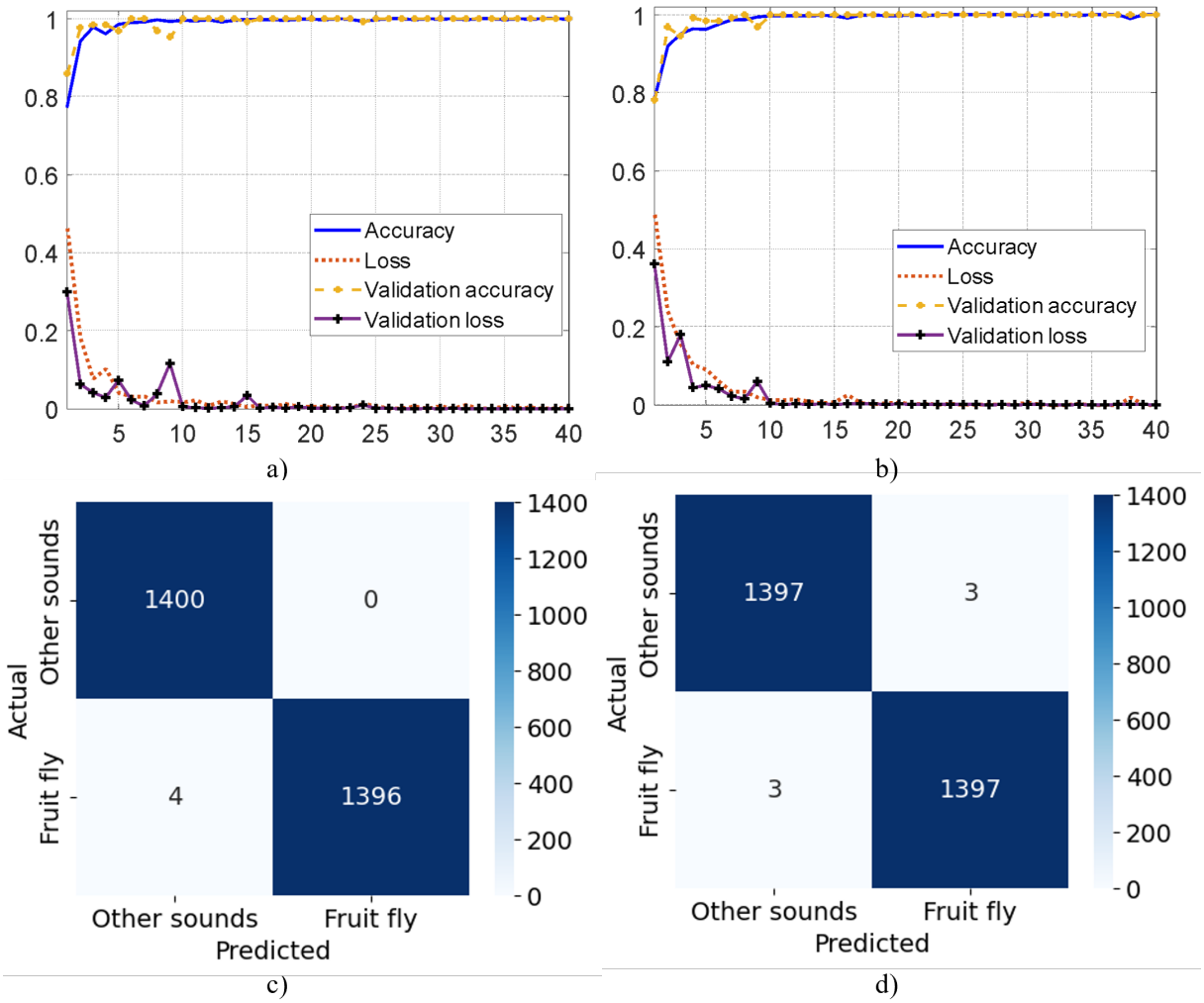

Figure 5 Training and testing results of the proposed models: a) Performance of the model 1 (using the Mel-filter spectrogram) and (b) Performance of the model 2 (using the average filter spectrogram)

The model using the Mel-filter spectrogram misclassified four wingbeat samples while correctly identifying 100% of the remaining sounds. In contrast, the model using the average-filter spectrogram misclassified three samples in each class. Table 3 summarizes the model sizes and evaluation metrics. Both models have sizes suitable for deployment on the ESP32 microcontroller after quantization. The results indicate that the Mel-filter spectrogram-based model achieves higher accuracy while maintaining a smaller size. The quantized models are subsequently evaluated on the MCU platform, including tests with both the prepared test dataset and real-time generated spectrograms.

In addition to the proposed models, LeNet-4, LeNet-5, and MobileNetV2 were also imple-

mented on both datasets for comparison, and the corresponding training results are reported in Table 3. Models with the suffix "a" were trained on the 20×100 spectrogram dataset, whereas those with the suffix "b" were trained on the 20×128 dataset. The models were evaluated in terms of classification performance, memory footprint, and inference latency from an embedded deployment perspective. All three baseline architectures exhibit larger model sizes and higher parameter counts after quantization than the proposed models. MobileNetV2 has the largest footprint, making it unsuitable for deployment on the ESP32 MCU. In terms of classification performance, LeNet-4a and LeNet-5a achieve results comparable to those of the proposed models; however, their parameter counts are significantly higher, approximately 6.5 times larger. Due to memory constraints, only LeNet-4a, LeNet-4b, and LeNet-5a could be executed on the ESP32, and these models exhibit higher inference latency, with the fastest requiring approximately 112 ms, which is approximately four times slower than the proposed model. This increased latency, combined with larger memory requirements, reduces their suitability for real-time, low-power microcontroller-based applications. As illustrated in Figure 6, the comparative visualization further highlights these trade-offs by presenting model accuracy, inference latency, memory footprint, and deployment feasibility. The proposed models are located in the upper-left region of the plot, indicating a favorable balance between high accuracy, low inference time, and compact model size, whereas larger baseline models either exhibit substantially higher latency or cannot be deployed on the target microcontroller due to memory constraints. Overall, these results highlight the trade-off between model complexity and deployment efficiency, demonstrating that the proposed CNN architecture achieves a more favorable balance between accuracy, memory usage, and inference speed for resource-constrained embedded environments.

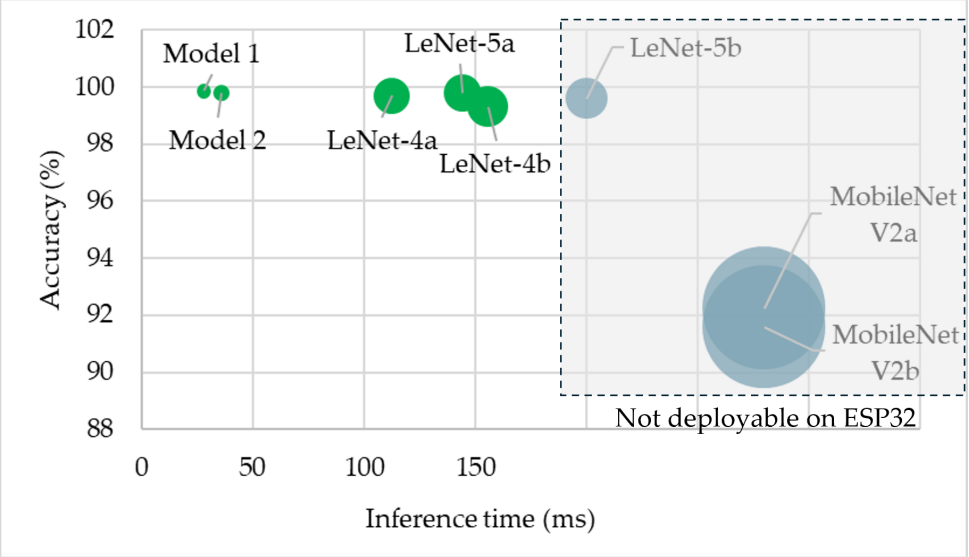

Figure 6 Model accuracy, inference time, and memory footprint comparison. The proposed models achieve a superior trade-off between performance and efficiency, whereas larger models, such as LeNet-5b and MobileNetV2, are not deployable on the ESP32 due to memory constraints

Table 3 Evaluation metrics of the proposed models compared with other architectures

Indicator	Model 1	Model 2	LeNet-4a	LeNet-4b	LeNet-5a	LeNet-5b	MobileNet V2a	MobileNet V2b
Size (kB)	143.76	185.76	945.63	1208.13	988.38	1250.88	9461.76	9461.76
Size after quantization (kB)	41.32	51.78	244.92	310.55	258.40	324.00	2830.20	2830.20
Number of parameters used	36802	47554	242082	309282	253026	320226	2422210	2422210
ACC (%)	99.86	99.79	99.71	99.32	99.79	99.61	92.25	91.60
MP (%)	99.86	99.79	99.72	99.32	99.79	99.61	93.29	92.81
MR (%)	99.86	99.79	99.71	99.32	99.79	99.61	92.25	93.75
$MF1$ (%)	99.86	99.79	99.71	99.32	99.79	99.61	92.20	93.74

3.3 Evaluation of fruit fly recognition capability directly using MCU

The two post-quantization CNN models are embedded directly on the electronic trap in the ESP32 microcontroller. To assess the stability and effectiveness of these models, two testing modes were used: store test spectrograms on a memory card, similar to testing performed on a personal computer. In the field test mode, the models perform real-time inference on spectrograms generated from live audio captured directly by the microphone. The field experiments were conducted using the same hardware setup as shown in Figure 1, where the electronic module was integrated into a conventional trap and deployed directly in an orchard environment. This setup enables evaluation under practical operating conditions, including environmental noise and natural insect activity. Figure 7 shows the algorithm of the CNN model testing program after quantization. With the static test, 1400 spectrograms were used to infer and save the results for the images in the test set. In contrast to field tests, the system must continuously record sounds and change spectrograms in real time. The sound data are streamed from the microphone and stored in the DMA buffer of ESP32. Then, the data are retrieved and converted into a spectrogram using the algorithm described in the previous section. No filtering is applied to the real-time data. The average amplitude of the spectrogram was also calculated for comparison with the preset threshold. If the spectrogram's average amplitude exceeds the predefined threshold, the model proceeds with inference and saves the classification result. Conversely, if the average amplitude falls below the threshold, the system continues to collect data and regenerate the spectrogram until the condition is met or the field test is concluded. This thresholding mechanism is implemented to prevent the system from performing inference on background noise present at the trap location, thereby reducing unnecessary inference on background noise during field operation.

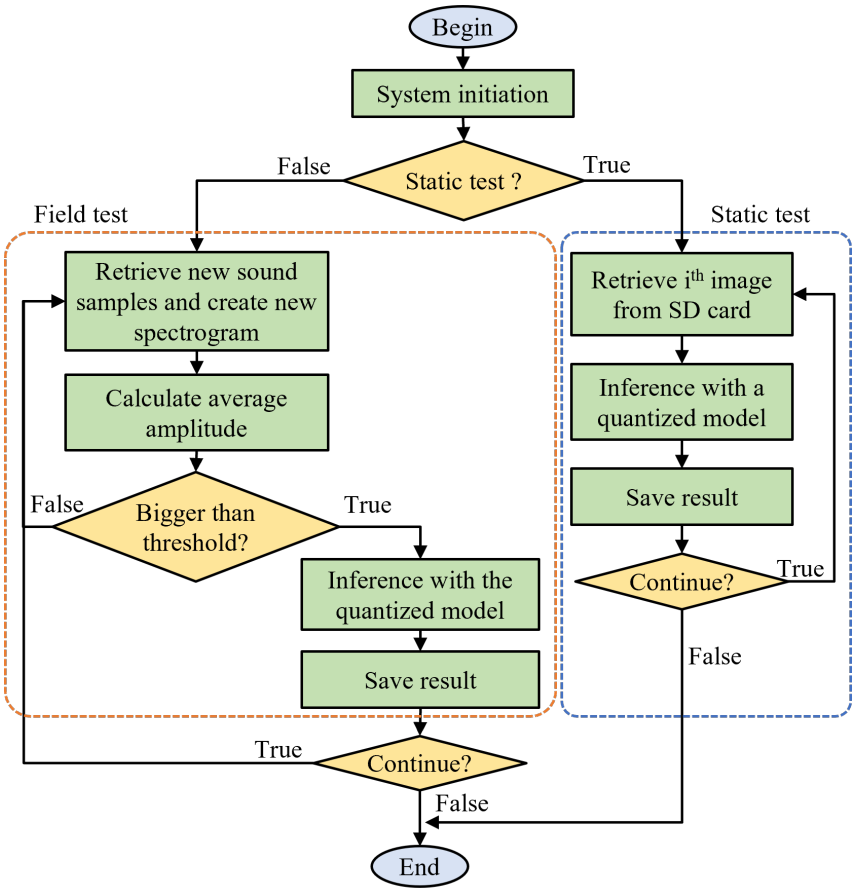

Figure 7 Flowchart of the proposed system's operation in field test and static test modes

Figure 8 presents the confusion matrices of both models under static testing conditions. The results indicate that Model 1 misclassifies three samples for each class, whereas Model

2 misclassifies only a single sample from the wingbeat spectrogram category. The evaluation metrics obtained from the static test for both quantized models are summarized in Table 4. Model 2 consistently outperformed Model 1 across all evaluation metrics, demonstrating slightly superior classification performance following quantization. Model 2 shows an improvement of 0.17 percentage points in ACC, MP, MR, and MF1 compared with Model 1. This improvement may be attributed to the use of average-filter spectrograms in Model 2, which preserves both low- and high-frequency components. Nevertheless, given the very low overall error rate, additional testing with larger or varied datasets is necessary to draw statistically significant conclusions.

A formal statistical test (e.g., t-test or McNemar's test) could not be performed because inference was executed directly on the microcontroller using a fixed static dataset. The total number of misclassified samples was negligible (< 10), resulting in distinct outcomes that do not meet the statistical significance testing assumptions. Therefore, the observed differences should be interpreted as practically consistent improvements rather than statistically validated differences.

Table 4 Metrics of the static test of Models 1 and 2

Metric	Model 1	Model 2
ACC (%)	99.79	99.96
MP (%)	99.79	99.96
MR (%)	99.79	99.96
$MF1$ (%)	99.79	99.96

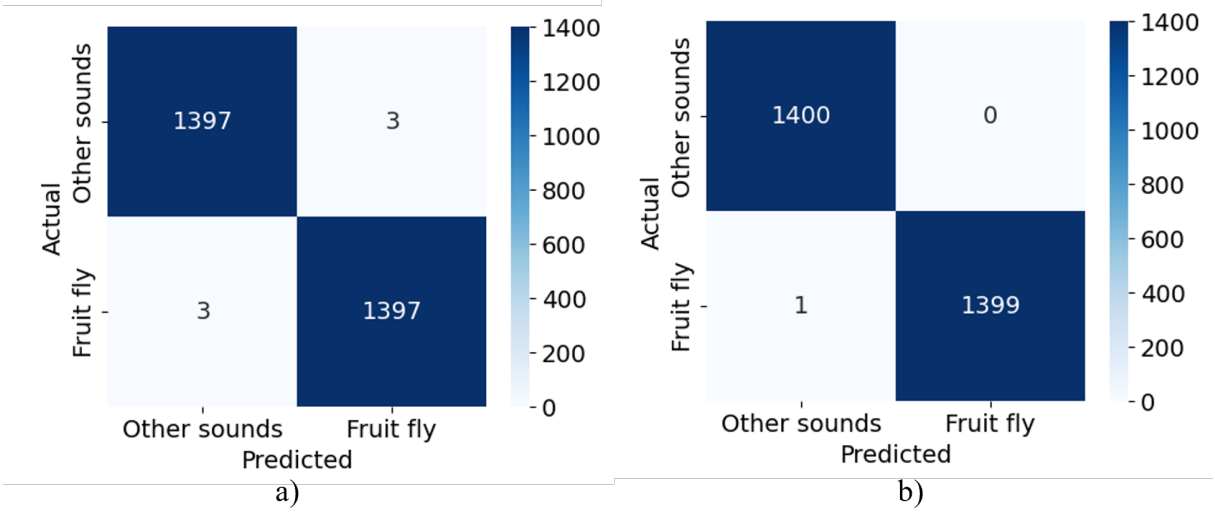

Figure 8 Confusion matrices of the static test on the MCU of (a) Model 1 and (b) Model 2

For the field test, the system was operated for one day, during which fruit fly wingbeat sounds and a few other noises were recorded. This evaluation focused on assessing the recognition rate of wingbeat spectrograms. After one day of operation, Models 1 and 2 inferred 1622 and 884 spectrograms, respectively. Figure 9 (supplementary file) illustrates an example data segment of approximately 4 s containing multiple wingbeat events. Figure 9a shows three distinct clusters corresponding to flies entering the trap, and Figure 9b presents the associated spectrogram with the frames inferred by the model. Solid red frames followed by dashed ones indicate continuous inference, whereas a single red frame denotes an isolated inference. Figure 9c shows the corresponding inference results, including the average amplitude, inference time, and inference scores for both wingbeat and non-wingbeat labels, logged by the MCU to a text file on the memory card. Using a threshold of 25 for wingbeat recognition, frames 1-7 were correctly identified as wingbeat sounds, and frame 8 as non-wingbeat. The lower confidence score of frame 8 resulted from the presence of background noise between two wingbeat clusters.

All recorded segments from the one-day test were manually reviewed to ensure the accuracy of the evaluation results.

Table 5 summarizes the performance of Models 1 and 2 at different detection thresholds. The model becomes more selective in identifying spectrograms corresponding to wingbeat signals as the threshold increases; however, this leads to a slight decrease in overall accuracy. This behavior can be attributed to signal amplitude variations, as wingbeat sounds may weaken when the fruit fly is farther from the microphone, resulting in lower-intensity spectrograms and reduced prediction scores. As shown in Table 5, Models 1 and 2 achieve their highest accuracies at a threshold of 25, reaching 96.86% and 96.38% for Models 1 and 2, respectively. Model 1, which uses Mel-spectrogram inputs, demonstrates slightly better performance than Model 2 under these conditions.

Figures 10a and 10b (in supplementary file) illustrate the classification behavior at the selected threshold through the confusion matrices obtained from field testing for Models 1 and 2, respectively. The majority of samples are correctly classified for both models, while the numbers of false positives and false negatives remain relatively low, consistent with the high classification accuracy reported in Table 5. Model 1 shows slightly better classification performance than Model 2, which is consistent with its marginally higher accuracy reported in Table 5. In addition, the representative misclassification examples in Figures 10c and 10d provide qualitative evidence that some errors are associated with spectrogram patterns that are difficult to distinguish from background sounds or weakened wingbeat signals under practical field conditions. These examples illustrate the types of errors encountered by the system and provide additional insight into model behavior beyond the aggregate performance metrics.

Although the thresholding mechanism helps reduce false detections caused by low-amplitude background noise under the tested conditions, it remains a relatively simple approach and does not explicitly account for more complex acoustic scenarios. The current evaluation does not include systematic testing under conditions such as overlapping insect events, multi-species acoustic mixtures, seasonal variability, or more diverse orchard sound environments. Therefore, the robustness and generalizability of the proposed approach under such conditions have not yet been fully established and require further investigation through more comprehensive and diverse real-world experiments.

Table 5 Evaluation metrics of the field tests of Models 1 and 2

Threshold	Model 1					Model 2				
	TP	TN	FP	FN	ACC	TP	TN	FP	FN	ACC
100	90.63%	0.43%	0.19%	8.75%	90.95%	94.46%	0%	0.45%	5.09%	94.46%
75	93.90%	0.43%	0.19%	5.49%	94.33%	95.93%	0%	0.45%	3.62%	95.93%
50	95.56%	0.43%	0.19%	3.82%	95.99%	96.27%	0%	0.45%	3.28%	96.27%
25	96.42%	0.43%	0.19%	2.96%	**96.86%**	96.38%	0%	0.45%	3.17%	**96.38%**

3.4 Computational analysis time

In addition to accuracy, the execution time of real-time tasks is a very important factor when applying models in practice. Table 6 presents the results of evaluating the execution time of the system's main tasks. The generation times of the two types of spectral images are comparable. Specifically, the 20×100 image requires 14.0 ± 0.2 ms, which is approximately 1.0 ± 0.3 ms faster than the 20×128 image. The inference times of both models are below 37 ms, with Model 1 being approximately 8 ms faster than Model 2, while the LeNet-4 model exhibits significantly higher latency, reaching 112.6 ms for the 2×100 input and 155.8 ms for the 2×128 input. In addition to inference, the system records wingbeat sounds and corresponding results to an SD card for verification, resulting in data transfer and storage times of 156.8 ± 29.8 ms and 161.1 ± 21.4 ms for Models 1 and 2, respectively, with additional inference result storage times of 74.4 ± 7.1 ms and 76.3 ± 8.4 ms, respectively. Consequently, the total average processing time is 273.2

ms for Model 1 and 288.4 ms for Model 2. The slightly longer time of Model 2 is attributed to its larger spectral image size. Despite this, both models satisfy real-time requirements, as the total processing time remains below the 400-ms window for generating a spectral image. After each inference, the system requires only an additional 126.8 and 111.6 ms of new data from the DMA buffer to generate the next input for Models 1 and 2, respectively. It is also observed that SD card operations account for the majority of the processing time (Table 6); however, this step is primarily required for data collection and evaluation and can be removed during deployment to reduce latency and enable integration of additional functionalities.

3.5 Discussions

This study demonstrates the feasibility of applying embedded artificial intelligence for fruit fly detection based on wingbeat sounds:

- Spectral analysis revealed that wingbeat signals exhibit more distinct characteristic frequencies than other sounds, making them suitable for identification tasks. Two datasets were generated using the MCU to support the deployment and testing of recognition models directly on edge devices.

- A CNN-based deep learning architecture was proposed and implemented to classify wingbeat sounds. The model was directly deployed and tested on a microcontroller unit (MCU) under real-world field conditions. Despite environmental variability, the model demonstrated high classification performance, achieving a maximum accuracy of 96.86%, confirming the viability of AI integration into edge computing devices.

- The entire preprocessing pipeline, including real-time signal processing and spectrogram generation, was successfully embedded into the MCU, enabling autonomous operation without reliance on external computing resources, such as PCs or cloud servers. The performance evaluation further indicates that the MCU retains sufficient computational headroom to support additional features on the electronic trap, such as fruit fly counting, communication control, and integration with IoT platforms.

However, fruit fly identification based on wingbeat sounds still faces several challenges. First, the current dataset was collected from a single orchard environment over a limited duration, which may not fully capture the variability of real-world acoustic conditions. Background sounds in natural environments can be highly diverse and dynamic, requiring larger and more representative datasets and longer-term data collection to improve model robustness and generalizability. Second, because the system is integrated into a compact electronic trap, flies may crawl rather than fly within the enclosure, resulting in the absence of wingbeat sound data and limiting the applicability of acoustic-based identification.

Table 6 Computation time of the proposed CNN models.

Processing Stage	Model 1 (ms)	Model 2 (ms)
Retrieving sound data and storing to the SD card	156.8 ± 29.8	161.1 ± 21.4
Spectrogram imaging	14.0 ± 0.2	15.0 ± 0.5
Model inference	28.0 ± 0.5	36.0 ± 0.04
Save the inference results to an SD card	74.4 ± 7.1	76.3 ± 8.4
Total processing time	273.2	288.4

Table 7 summarizes a comparison of this study with previous research on fruit fly identification using wingbeat sounds. Although earlier studies (Khalid et al., 2025; Kalfas et al., 2022; Mankin et al., 2006) achieved high accuracy under controlled conditions, they focused on different fly species and used PC- or mini-PC–based platforms. In contrast, the recognition

system was implemented directly on a microcontroller unit (MCU) and tested under real-world field conditions. This embedded approach indicates promising potential for future IoT-based field applications to minimize bandwidth, latency, and power consumption. However, further validation with a larger and more diverse dataset, longer testing duration, and a wider range of environmental noises is required to confirm this practical applicability.

Table 7 Comparison of related studies.

Study	Species	Sensor type	Features	Method	Platform	Testing area	Result
Mankin et al., 2006	*Ceratitis capitata*	Microphone	Power spectral density	Manually performed using spectral image analysis software	PC	Anechoic chamber and field	In chamber: most triggers captured flights; in the field: only 4 out of 200 triggers
Kalfas et al., 2022	*D. suzukii* and *D. melanogaster*	IR transceiver	Time series, power spectral density, and spectrogram	CNN-based deep learning network	PC	Small insect cage	92.1%, inference time: 4–29.8 ms
Khalid et al., 2025	Guava, melons, blue bottle, and mosquitoes	Microphone	FFT, HoG of FFT, and MFCC	Machine learning (SVM, KNN, RBF, and DT)	Raspberry Pi	Laboratory	90% at night and 82% during the day
This study	*B. dorsalis*	Microphone	Spectrogram	CNN-based deep learning network	ESP32 MCU	Field	96.86%, inference time: 28 and 36 ms

The results demonstrate that controlled environments yield high recognition rates, but field conditions can significantly reduce the performance of the proposed method. For example, Mankin et al., 2006 correctly identified only 4 of 200 samples under real conditions. In our field tests, the CNN model achieved 96.86% accuracy on real-time data streams, demonstrating the feasibility of embedded AI for fruit fly detection. In addition, the proposed CNN model is lightweight, resulting in short inference times of approximately 28 and 36 ms for Models 1 and 2, respectively, when executed on the MCU. Although Kalfas reported inference times ranging from 4 to 29.8 ms on a PC platform under different testing conditions, this comparison highlights that the proposed embedded model operates within a similar real-time range. Therefore, the proposed system demonstrates that real-time fruit fly identification on low-power MCU hardware is feasible.

These results provide an initial foundation for future research. Potential directions include expanding the dataset, improving model robustness to environmental noise, integrating automatic counting mechanisms, and using sound-based identification to extend the approach to other insect species or wildlife. Although the system demonstrates effective audio processing and real-time CNN inference in the current experimental setting, its applicability beyond this setting remains to be validated through more diverse datasets and longer-term field deployments.

4. Conclusions

Many studies have developed electronic traps using visual or infrared sensors to detect fruit flies. However, few have explored identification based on wingbeat sounds, and practical field validations remain limited. In this study, we investigated the use of an acoustic sensor to detect fruit flies' wingbeat sounds, a method that has not been widely explored in the existing literature. Using a deep learning approach with spectrograms, we achieved a classification accuracy of 96.86% in identifying fruit fly wingbeats. The fast inference time (28 ms) further demonstrates the practicality of this approach for real-world applications. The deployment of the recognition model on an MCU enables the creation of compact, low-power traps, providing a basis for the further development of fruit fly monitoring systems. However, our study has several limitations. First, the dataset used for training and testing was limited to a specific

experimental site, which may not fully represent the acoustic environment of other regions or conditions. Second, while the model performs well in controlled settings, its robustness in real-world scenarios, such as varying background noise or environmental factors, still requires further investigation. Future studies should focus on enhancing the model's robustness by incorporating a broader range of environmental variables and expanding the dataset to include more diverse conditions and species. Moreover, integrating acoustic and infrared sensing approaches could improve counting and identification accuracy. Finally, further investigation of real-time data processing and potential integration with agricultural monitoring systems is necessary, as its applicability beyond the current experimental setting requires validation under more diverse and long-term deployment conditions.

Acknowledgements

The authors acknowledge the financial support from the Ministry of Education and Training of Vietnam under the research project B2024-TCT-21.

Author Contributions

Conceptualization, V.-K. Nguyen and V.-K. Tran; methodology, V.-K. Nguyen, V.-K. Tran, and C.-N. Nguyen; formal analysis, V.-K. Nguyen and V.-K. Tran; investigation, V.-K. Nguyen and V.-K. Tran; resources, V.-K. Nguyen; data curation, V.-K. Tran and C.-N. Nguyen; writing—original draft preparation, V.-K. Nguyen and V.-K. Tran; writing-review and editing, V.-K. Nguyen and C.-N. Nguyen; supervision, V.-K. Nguyen and C.-N. Nguyen; funding acquisition, V.-K. Nguyen. All authors have read and approved the published version of the manuscript.

Conflict of Interest

The authors declare no conflicts of interest.

Declaration of AI

Artificial intelligence tools (e.g., ChatGPT) were used only to improve the grammar, wording, and clarity of the manuscript. The authors have reviewed and validated all the content. No AI tool was used for data analysis, interpretation, or scientific conclusion generation.

Supplementary Materials

Figure 9 and 10 in supplementary file.

References

Abdel-Hamid, O., Deng, L., & Yu, D. (2013). Exploring convolutional neural network structures and optimization techniques for speech recognition. *Interspeech*, *2013*, 1173–1175. https://doi.org/10.21437/Interspeech.2013-744

Abdul, Z., & Al-Talabani, A. (2022). Mel frequency cepstral coefficient and its applications: A review. *IEEE Access*, *10*, 122136–122158. https://doi.org/10.1109/ACCESS.2022.3223444

Ali, M., Sayeed, M., & Abdul Razak, S. (2025). HDL-Net: Hybrid deep learning and IoT network-based system for pest detection using pest sound analytics. *Discover Applied Sciences*, *7*(10), 1155. https://doi.org/10.1007/s42452-025-07747-y

Audacity Team. (2017). The name Audacity® is a registered trademark of Dominic Mazzoni [http://audacity.sourceforge.net].

Bogerding, M. (2017). KissFFT library [https://github.com/mborgerding/kissfft].

Espressif Systems. (2021). ESP32 series datasheet [https://www.espressif.com].

Google. (2024). Post-training quantization [https://ai.google.dev/edge].

Hahn, F., Valle, S., Rendón, R., Oyorzabal, O., & Astudillo, A. (2023). Mango fruit fly trap detection using different wireless communications. *Agronomy*, *13*(7), 1736. https://doi.org/10.3390/agronomy13071736

Hendel, F. (1912). H. sauter's formosa-ausbeute: Genus dacus (diptera). *Supplementa Entomologica*, *1*, 13–24.

Hien, N., Trang, V., Thanh, V., Lien, H., Thang, Xuyen, L., & Pereira, R. (2019). Fruit fly area-wide integrated pest management in dragon fruit in Binh Thuan province, Viet Nam. In *Area-wide management of fruit fly pests* (pp. 343–347). CRC Press.

Huang, R., Yao, T., Zhan, C., Zhang, G., & Zheng, Y. (2021). A motor-driven and computer vision-based intelligent e-trap for monitoring citrus flies. *Agriculture*, *11*(5), 460. https://doi.org/10.3390/agriculture11050460

Jaffar, S., Rizvi, S., & Lu, Y. (2023). Understanding the invasion, ecological adaptations, and management strategies of *Bactrocera dorsalis* in China: A review. *Horticulturae*, *9*(9), 1004. https://doi.org/10.3390/horticulturae9091004

Kalfas, I., De Ketelaere, B., Beliën, T., & Saeys, W. (2022). Optical identification of fruit fly species based on their wingbeats using convolutional neural networks. *Frontiers in Plant Science*, *13*, 812506. https://doi.org/10.3389/fpls.2022.812506

Khalid, A., Anjum, M., Naveed, S., & Hussain, W. (2025). Whispers in the air: Designing acoustic classifiers to detect fruit flies from afar. *Smart Agricultural Technology*, *10*, 100738. https://doi.org/10.1016/j.atech.2024.100738

Kibira, M., Affognon, H., Njehia, B., Muriithi, B., Mohamed, S., & Ekesi, S. (2015). Economic evaluation of integrated management of fruit fly in mango production in Embu County, Kenya. *African Journal of Agricultural and Resource Economics*, *10*(4), 343–353. https://doi.org/10.22004/ag.econ.211846

Le, A., Pham, D., Pham, D., & Vo, H. (2021). AlertTrap: A study on object detection in remote insect trap monitoring system using on-the-edge deep learning platform. *arXiv preprint arXiv:2112.13341*. https://doi.org/10.47852/bonviewJCCE42023264

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

Liquido, N., McQuate, G., Birnbaum, A., Hanlin, M., Nakamichi, K., Inskeep, J., Ching, A., Marnell, S., & Kurashima, R. (2017). A review of recorded host plants of oriental fruit fly, *Bactrocera (Bactrocera) dorsalis* (Hendel) (Diptera: Tephritidae), version 3.0 [USDA CPHST Online Database. https://coffhi.cphst.org/].

Long, K., Nghiep, H., & Oanh, N. (2022). Seasonal population dynamics of the oriental fruit fly, *Bactrocera dorsalis* (Hendel), in mango orchards, Cao Lanh City, Dong Thap Province. *Tap chi Khoa hoc DH Dong Thap*, *11*(5), 85–92.

Mankin, R., Machan, R., & Jones, R. (2006). Field testing of a prototype acoustic device for detection of Mediterranean fruit flies flying into a trap. *Proceedings of the 7th International Symposium on Fruit Flies of Economic Importance*.

Martins, V., Freitas, L., de Aguiar, M., de Brisolara, L., & Ferreira, P. (2019). Deep learning applied to the identification of fruit fly in intelligent traps. *2019 IX Brazilian Symposium on Computing Systems Engineering (SBESC)*. https://doi.org/10.1109/SBESC49506.2019.9046088

Molina-Rotger, M., Morán, A., Miranda, M., & Alorda-Ladaria, B. (2023). Remote fruit fly detection using computer vision and machine learning-based electronic trap. *Frontiers in Plant Science*, *14*, 1241576. https://doi.org/10.3389/fpls.2023.1241576

Muriithi, B., Affognon, H., Diiro, G., Kingori, S., Tanga, C., Nderitu, P., Mohamed, S., & Ekesi, S. (2016). Impact assessment of integrated pest management (IPM) strategy for suppression of mango-infesting fruit flies in Kenya. *Crop Protection*, *81*, 20–29.

Mutamiswa, R., Nyamukondiwa, C., Chikowore, G., & Chidawanyika, F. (2021). Overview of oriental fruit fly, *Bactrocera dorsalis* (Hendel) (Diptera: Tephritidae) in Africa: From

invasion, bio-ecology to sustainable management. *Crop Protection*, *141*, 105492. https://doi.org/10.1016/j.cropro.2015.11.014

Nanda, M., Seminar, K., Nandika, D., & Maddu, A. (2018). Discriminant analysis as a tool for detecting the acoustic signals of termites *Coptotermes curvignathus* (Isoptera: Rhinotermitidae). *International Journal of Technology*, *9*(4), 840–851. https://doi.org/10.14716/ijtech.v9i4.455

Navarro-Llopis, V., & Vacas, S. (2014). Mass trapping for fruit fly control. In *Trapping and the detection, control, and regulation of tephritid fruit flies: Lures, area-wide programs, and trade implications* (pp. 513–555). https://doi.org/10.1007/978-94-017-9193-9_15

Nguyen, V., Tran, V., & Nguyen, C. (2025). Towards fruit fly automatic counting: Electronic trap design and long-term feature data acquisition. *International Journal of Applied Science and Engineering*, *22*, 2025102. https://doi.org/10.6703/IJASE.202509_22(3).004

Oanh, N., & Duc, H. (2020). An initial investigation of pest species on Dai Loan mango planting in Cao Lanh City, Dong Thap Province, Vietnam. *Tap chi Khoa hoc DH Dong Thap*, *9*(5), 68–76.

Opoku, E., Haseeb, M., Rodriguez, E., Steck, G., & Cabral, M. (2025). Economically important fruit flies (Diptera: Tephritidae) in Ghana and their regulatory pest management. *Insects*, *16*(3), 285. https://doi.org/10.3390/insects16030285

Oppenheim, A., & Schafer, R. (2009). *Discrete-time signal processing* (3rd). Pearson.

Potamitis, I., Rigakis, I., & Fysarakis, K. (2014). The electronic McPhail trap. *Sensors*, *14*(12), 22285–22299. https://doi.org/10.3390/s141222285

Sandrini-Moraes, F., Edson Nava, D., Scheunemann, T., & Santos da Rosa, V. (2019). Development of an optoelectronic sensor for detecting and classifying fruit fly (Diptera: Tephritidae) for use in real-time intelligent traps. *Sensors*, *19*(5), 1254. https://doi.org/10.3390/s19051254

Shetty, M., & Kumar, Y. (2025). Audio-based classification of insect species using machine learning models: Cicada, beetle, termite, and cricket. *arXiv preprint*. https://doi.org/arXiv:2502.13893

Tran, T., & Nguyen, T. (2023). Determination of species composition and effect of plant extracts to prevent the eggs-laying of fruit flies, *Bactrocera spp.*, infesting jackfruit. https://doi.org/10.22271/j.ento.2023.v11.i3a.9195

Trombik, J., Ward, S., Norrbom, A., & Liebhold, A. (2023). Global drivers of historical true fruit fly (Diptera: Tephritidae) invasions. *Journal of Pest Science*, *96*(1), 345–357. https://doi.org/10.1007/s10340-022-01498-0

Varma, A., Bateshwar, V., Rathi, A., & Singh, A. (2021). Acoustic classification of insects using signal processing and deep learning approaches. *2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)*, 1048–1052. https://doi.org/10.1109/SPIN52536.2021.9566121

Warden, P., & Situnayake, D. (2019). *TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers*. O'Reilly Media.

Wijekoon, C., Ganehiarachchi, M., Wegiriya, H., & Vidanage, S. (2024). The variation of oviposition preference and host susceptibility of the oriental fruit fly (*Bactrocera dorsalis* Hendel) (Diptera: Tephritidae) on commercial mango varieties. *Advances in Agriculture*, *2024*, 7490120.

Zeng, Y., Reddy, G., Li, Z., Qin, Y., Wang, Y., Pan, X., Jiang, F., Gao, F., & Zhao, Z. (2019). Global distribution and invasion pattern of oriental fruit fly, *Bactrocera dorsalis* (Diptera: Tephritidae). *Journal of Applied Entomology*, *143*(3), 165–176. https://doi.org/10.1111/jen.12582