amount of coefficients but reducing the necessary steps and computational complexity with respect to the MFCC as follows.

1) Division of the audio into windows ranging from 256 to 4096 audio samples without overlapping.
2) Application of the Hann windowing function for signal smoothing at the window edges.
3) Application of DFT to each window for signal frequency domain conversion.
4) Computation of the magnitude of the spectrogram.
5) Calculation of the mean value for each frequency bin across all chunk duration. This step results in a number of input values for the classifier that depends on the window size $n_{\text{bins}} = \frac{m}{2} + 1$, where m represents the window size.

D. Classifiers

In this study, we conducted a comparison between two distinct classifiers for the binary classification task: one is a NN implemented through the Keras library [35], and the other is a SVM implemented using the scikit-learn (sklearn) library [36]. We focused on these two algorithms without considering convolutional networks that are used in other works [21], [29] because our intention is to keep the complexity of these algorithms low, with the aim of deploying them in microcontroller systems with low hardware resources.

1) NN Classifier: Represents a powerful and adaptable machine learning model with the capability to identify intricate patterns in data. The presented approach involved constructing a multilayer perceptron architecture with fully connected layers. There are two hidden layers, the first with eight neurons and the second with four neurons. This architecture is used in all the experiments except for the one in which different NN sizes are compared (Section III-B). Each node is implemented with rectified linear unit (ReLU) activation function, while the output layers employ a sigmoid function. The final step involved converting the output to a binary value using a threshold of 0.5. During the training process, we implemented early stopping with a patience of 5 to prevent overfitting and identify the optimal number of epochs necessary for the training.

2) SVM Classifier: It is a widely adopted and powerful supervised learning algorithm designed for classification tasks. Its functionality involves identifying an optimal hyperplane that maximizes the separation between data points associated with distinct classes. In our implementation of SVM, we employed the sklearn library [36], which offers a robust set of tools for machine learning in Python. During our experimentation, we specifically worked with the radial basis function kernel function and fine-tuned the C parameter for optimal performance.

These classifier parameters are chosen in order to replicate the architecture used in [29], but reducing the complexity of the models removing the convolution layers and scaling down the architecture size and obtaining a lighter model.

E. Evaluation Metrics

At the end of each experiment, obtaining a set of parameters is crucial to compare different characteristics and provide a good estimation of which model performs better in detecting the presence of the queen bee. In particular, this work adopts the technique of crossvalidation by dividing the training set into tenfolds. The metrics extracted include both the mean value and the standard deviation, which are extracted across ten models. These models are trained with nine folds and tested with the remaining fold, with a variation in the selected folders for each iteration. The metrics presented are the most common and widely used for binary classifiers considering true positive (TP) and true negative (TN) the correctly predicted values that can be positive (queen presence) or negative (queen absence), and the false positive (FP) and false negative (FN) the wrong predicted samples as follows.

1) *Precision [positive predicted value (PPV)]:* It is the ratio of correctly predicted positive observations to the total predicted positives. It provides insights into the accuracy of positive predictions

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{1}$$

2) *Recall [true positive rate (TPR)]:* It is the ratio of correctly predicted positive observations to the total actual positives. It assesses the ability of the model to capture all the relevant instances

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2}$$

3) *Accuracy:* Measures the overall correctness of the classification model. It is the ratio of correctly predicted instances to the total instances

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}. \tag{3}$$

4) *F1 score:* The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, considering both FP and FN

$$F_1\text{score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}. \tag{4}$$

5) *Receiver operating characteristic area under the curve (ROC AUC):* ROC AUC represents the area under the ROC curve, which is a plot of the TRP against the FP rate. It evaluates the model's ability to distinguish between classes.
6) *Confusion matrix:* Provides a detailed breakdown of TP, TN, FP, and FN predictions. It is especially useful for understanding the model's performance on the final test.

By evaluating these metrics, it becomes possible to comprehensively assess the classification performance of different models, considering various aspects. The use of cross-validation ensures a robust evaluation, and reporting both mean values and standard deviations adds a measure of the stability of the model's performance across different folds.

F. Framework Improvements

In this work, we enriched our Python framework introduced in [25]. Specifically, we integrated the possibility of managing additional datasets, and in particular the dataset presented