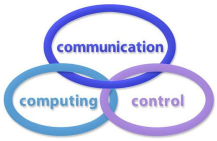
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 19, Issue: 4, Month: August, Year: 2024
Article Number: 6632, https://doi.org/10.15837/ijccc.2024.4.6632



CCC Publications

IoT Embedded Smart Monitoring System with Edge Machine Learning for Beehive Management

M. Doinea, I. Trandafir, C. Toma, M. Popa, A. Zamfiroiu

Mihai Doinea
Department of Informatics and Economic Cybernetics
Bucharest University of Economic Studies, Romania
mihai.doinea@ie.ase.ro
*Corresponding author: mihai.doinea@ie.ase.ro

Ioana Trandafir
Bucharest University of Economic Studies, Romania
trandafirioana16@stud.ase.ro

Cristian Toma
Department of Informatics and Economic Cybernetics
Bucharest University of Economic Studies, Romania
cristian.toma@ie.ase.ro

Marius Popa
Department of Informatics and Economic Cybernetics
Bucharest University of Economic Studies, Romania
marius.popa@ie.ase.ro

Alin Zamfiroiu
Department of Informatics and Economic Cybernetics
Bucharest University of Economic Studies, Romania
National Institute for Research & Development in Informatics
alin.zamfiroiu@csie.ase.ro

Abstract

The need of an automated support system that helps beekeepers maintain and improve beehive population was always a very stressing aspect of their work considering the importance of a healthy bee population. This paper presents a proof of concept, further referred as a PoC solution, based on the Internet of Things technology which proposes a smart monitoring system using machine learning processes, diligently combining the power of edge computing by means of communication and control. Beehive maintenance is improved, having an optimal state of health due to the Deep Learning inference triggered at the edge level of devices which processes hive's noises. All this is achieved by using IoT sensors to collect data, extract important features and a Tiny ML network for decision support. Having Machine Learning inference to be performed on low-power microcontroller devices leads to significant improvements in the autonomy of beekeeping solutions.

Keywords: Machine Learning, Monitoring Solution, Beehive Support, Internet of Things.

1 Introduction

The concept of Edge Machine Learning (Edge ML) is part of the AI – Artificial intelligence and coagulates all models and techniques used for inferences/classifications/pattern detections on the devices which are on the edge of the platform/solution and not in cloud. If we refer to AI Deep Learning that involves complex models and neural networks training for pattern recognitions in sound processing or visual computing, then we can choose from a wide variety of options for training and inference [23]. The training stage of a neural network often occurs within a Machine Learning (ML) Cloud environment, utilizing various hardware types like CPUs, GPUs, or TPUs across different ML frameworks, including Keras, TensorFlow, among others [15]. This phase encompasses tasks such as data pre-processing, transformations, and training, which, while possible to conduct anywhere, are typically performed on ML Cloud platforms equipped for such operations. Due to their significant resource demands, many machine learning algorithm processes benefit from the powerful computing capabilities of these platforms. However, one particular process, inference, requires comparatively fewer resources, offering advantages in scenarios where the execution of this process remotely or in isolated locations is necessary. This is especially relevant for machine learning edge devices, where stable connections to ML Cloud systems may not be available, allowing for direct inference [23]. Efficient use of such systems has been demonstrated through the analysis of diverse sounds from various sources [30], [31]. Conducting sound analysis close to the source enhances sound classification by eliminating noise, thereby reducing the computational resources required for pre-processing the input data. In [17] the sounds sources are divided in three important categories, with each one having different characteristics, so different ways of analyzing and interpreting the results:

(1) Natural – sounds created by natural forces like animal, water or nature.

(2) Mechanical – sounds created by cars or other machines used in transportation or construction.

(3) Human – all sounds produced by human people: music, walking sounds, speaking and so on.

For being able to distinguish between different sound sources an Acoustic Scene Classification framework for machine learning models was used. According to [50] the ML is used with a higher accuracy than genetic algorithms on designing metasurface for regional control of sound fields. In [33] an end-to-end convolutional neural network is proposed for detecting emergency situations by analyzing urban noise. All these sound processing systems that make use of machine learning techniques have a common denominator when trying to reveal patterns that would otherwise be unable to detect: data generation, training the model and optimizing the phase gradient according to desired functionality to fit on an edge computing system.

1.1 Tiny Machine Learning on Edge

Machine Learning on Edge includes personal computers, smartphones and other devices that could power machine learning algorithms with all included stages. By going further and trying to put machine learning inside embedded devices that have extremely low-power microcontrollers we experience what is known as Tiny Machine Learning, or Tiny ML systems. Until recent years, building machine learning applications meant an exclusive usage of large and complex models that give a very good performance, but are also expensive to build, use and maintain. These large models require a remarkable amount of computing capabilities, so they are running in dedicated data centers, or they are translated to the cloud. This causes latency, wastes resources, puts applications at the mercy of connection speed and generates security concerns. Tiny ML is a cutting-edge technology that takes advantage of the power brought by machine learning models and the performance of running optimized models on embedded low-power devices that consume only milliwatts of power. An example of this technology can be seen in smart speakers like Google Home, where a digital signal processor continuously listens for the "Ok Google" or "Hey Google" wake words and analyses the sound with a tiny machine learning model that runs locally, on the device [18], [32]. Traditionally, IoT devices send the collected data to the cloud, where machine learning models generate insights based on identified patterns. Tiny ML makes it possible to deploy a trained model on the embedded device. Therefore, inference is executed on the device itself, without sending raw data to the cloud. By performing the inference on the embedded device, Tiny ML enables greater reliability and reduces the latency associated with sending

large amounts of data to the cloud over a crowded internet network. Reducing data transmissions over the network has other benefits as well, including energy savings (the cost associated with data transmission is higher than the cost of inference) and increased privacy [34]. The efficiency of Tiny ML opens up new ways to analyze and make sense of the substantial amount of data collected by IoT devices and brings new opportunities that were previously too costly to be implemented, such as wildlife tracking. Tiny ML offers the same advantages presented above for a smart beekeeping audio analyzer. By classifying the audio stream on-device and sending over Wi-Fi only the result of the classification, we save a lot of energy and bandwidth. Furthermore, audio recordings that can contain the beekeeper's voice are not transmitted or stored in any servers, which enhances the privacy of the beekeeper. This new technology also presents some challenges. Specifically, the models must be small in order to fit within the limitations of MCU class devices (limited CPU power and memory), consequently limiting the number of neural network layers and nodes or requiring the implementation of more lightweight machine learning algorithms. These hardware constraints also mean that a Tiny ML application needs to tackle a specific use case. A full speech recognition model cannot be deployed on a tiny microcontroller, no matter how much we try to shrink the model, but a model that is able to recognize certain words is small enough to fit withing the hardware constraints. Another challenge usually encountered when building Tiny ML models is related to the quantization technique. This is the process of converting the neural network parameters (weights and biases) from floating-point values to a lower memory storage such as 8-bit integers for improving the memory footprint, [4]. As a result, the computational cost and memory requirements of running the neural network are drastically reduced. This process can lead to a decrease in accuracy, but usually the loss is not significant and the trade-of is worth trying it. The Edge devices have features which make them suitable for neural networks inference but not for training. For instance, it is challenging to train an AI neural network in a device with a few KB of RAM, MCU on 8/16 bits and storage of a few hundreds of KB or 1 MB, versus a dedicated cloud where there is virtualization, several GB of RAM, CPU on 64 bits and storage of order of several TB. Therefore, on the Edge devices of an IoT solution there is not common to have ML dedicated hardware or ML dedicated software and the consequence is that on these kinds of devices only AI neural network inference can take place [52].

1.2 Related Studies and Motivation

Bee colonies all over the world are an important piece in nature's equilibrium [37] and are constantly affected by an increasing number of threats [43], most of them caused by humans. Any stress factor that affects the hive, such as parasites, disease or absence of the queen can have a fatal impact if it's not treated in time. Taking this into consideration, it is vital for beekeepers to improve the methods they use to monitor and take care of their beehives to maximize their productivity and to protect the bees of these constant threats. Manually inspecting the health and state of the hive disrupts the bees' work and can stress them [7]. Furthermore, this activity is time consuming for the beekeeper and he/she may accidentally hurt the worker bees or the queen during frame manipulations. The limitations of this approach are felt especially during the winter season, when the beekeeper can observe the beehive only from outside by checking the surroundings or by analyzing the buzz produced by bees after a gentle tap on the hive. Continuously monitoring the bee colony state is critical for the beekeeper, so that he can take the right actions as soon as possible in case of any threats. Smart beehive monitoring system [8], [35], were designed by combining technologies such as Internet of Things (IoT) and Machine Learning (ML), that can remotely monitor the beehive and process the acquired parameters to detect its state [28]. This automated system has the potential to help scientists minimize losses in bee colonies and beekeepers to successfully manage the hive activity, allowing them to easily monitor the conditions inside the hives, even when they own dozens, and take the appropriate action on time. Various beehives monitoring systems have been proposed over the years, mostly relying on different sensors such as temperature, humidity, weight, gas [51]. These parameters are important and can reveal very useful information about the hive activity and health, but the most promising parameter is the sound emitted by bees. Additionally, the sound can be correlated with other parameters and provide patterns that can differentiate between a normal behavior and a critical situation for the honeybee colony. Bees use vibrations and sound signals generated by movement and muscle contractions to

communicate inside the beehive [20], [22]. Evaluating beehive acoustics can be also used for detecting swarm activity which requires an immediate response from the beekeepers. In [49], authors proposed a machine learning technique based on Hidden Markov models and Gaussian Mixture models to model the beehive behavior. Although results identified the swarming period with above 80% accuracy when using 32 Gaussian mixtures per state, this came with higher computation costs compared to 2 Gaussian mixtures per state which provided only 59% accuracy and 8 Gaussian mixtures per state having 75% accuracy. Sound analysis can reveal valuable information about the state of the colony and can detect sudden variations in the beehive in a noninvasive manner. Microphones can be placed in specific positions inside the hive to capture the sound produces by honeybees. Accelerometers can also be used as an alternative to record the hive vibrations [9]. Various events can be detected based on the acoustic signature of the beehive at a given time [47]:

- Swarming - Swarming usually occurs when the bee colony exceeds the capacity of the hive, which becomes overcrowded. The bees produce a younger queen, and the old queen leaves the hive with a large portion of the worker bees to search for a new home [5], [14].
- Queen hatching – Young queens announce their hatching using tooting and quacking noises. They make quacking vibrations to announce they are ready to hatch and after they emerge from their cells, they stop quacking and start tooting [5].
- Missing queen – A hive without a queen is referred to as queen less. Detecting this event is very important for the beekeeper because a colony can't survive for a long time without a queen. Many bee colonies succeed in producing a new queen, but some fail. In this scenario, the beekeeper can save the colony by installing another queen in the hive [10].
- Sick colony – Honeybees are vulnerable to numerous diseases and some of them are very contagious. It's important for a beekeeper to be able to detect the signs of diseases in a hive as soon as possible so he can treat the colony in time.
- Exposure to toxic substances – Honeybees are negatively affected by agrochemicals such as pesticides, insecticides, or fertilizers. They come in contact with these chemicals when collecting nectar from plants and they can cause poisoning, disorientation and abnormal behavior [7].

Among all these events, this paper focuses on detecting if the queen is present in the beehive or if the colony is queen less. This information is valuable for the beekeeper as undetected queen less state can lead to the colony loss. The queen bee is vital to a beehive because she is the only bee that lays fertilized eggs and produces pheromones that help to maintain the unity of the colony [19]. Studies have revealed if the level of pollutants have impacted or not the beehive wellbeing based on the background noise coming from the hive [35]. Honeybees, Apis Mellifera [37], are the most significant pollinators of food crops, playing a vital role in maintaining food security, ecological balance and biodiversity in the world. Researchers estimated that a third of the food production depends upon pollination and that bees are the main contributors in this process [6]. These amazing insects can also be seen as natural indicators of the health of our environment, and they produce healthy food (honey, pollen, and royal jelly) and medicinal products that are being used in healthcare (venom, propolis). Unfortunately, the bee population is on the decline, and this can result in alarming consequences for the ecosystem. Their extinction is an extremely serious concern and is affecting both managed and wild colonies alike. According to [12], in most recent years, beekeepers have reported abnormal colony losses in many parts of the world and there is evidence that wild bee species are also at risk. The decline in the population of bees has been attributed to numerous causes including the use of pesticides in agriculture, global warming, the loss of natural habitats and environmental pollution [27]. As the bee population continues to drop, researchers have started to work on solutions that aim to monitor and analyze the health of bees with the help of technology. Most losses occur during the winter, when the bee colony is weaker, and the beekeeper cannot inspect the hive because of the low temperature that could kill the colony. During this season, it's a lot more difficult for the beekeeper to understand what goes on inside his beehives. The beekeeper can observe the hive only from the

outside by checking the surroundings or analyze the buzz produced by bees after a gentle tap on the hive. The sounds emitted by honeybees can give important clues about the state of the hive, and one important state that can be analyzed is the presence of the queen bee in the hive. The loss of the queen bee is a very dangerous event in the beehive, and it requires the intervention of beekeepers. Usually, the beekeeper needs to manually search for the queen in the beehive during regular inspections. This activity is very time consuming and can disrupt the normal activity of the hive. The purpose of this paper is to build an innovative solution that can analyze the sound produced by honeybees using deep learning in an energy efficient and secure way to determine if the queen bee is present in the hive, [20]. This solution can help beekeepers to identify vulnerable hives that lost their queens, while also reducing the unnecessary stress for the honeybees and the effort of the beekeeper to perform manual inspections. This automated decision helps the beekeeper to maintain the hive in an optimal state of health and gives it opportunities to grow without too many interventions.

1.3 Proposed solution

The innovation of this paper lies in the integration of a suite of technologies and algorithms aimed at serving a greater good, by employing Tiny ML to analyze the audio signatures of beehives. In this paper we analyze and describe how the implementation of an IoT architecture powered by an Edge ML network can boost the beehive population and help beekeepers act as fast as possible when it comes to beehive problems. First section focuses on existing solutions that are making use of edge machine learning, Edge ML, for image and sound processing in general and bee sound noise analysis. The second section describes the use of convolutional neural networks for beehive management decision support and the network configuration with all its underlying layers used for the PoC implementation. Beekeepers are assisted by an Edge ML architecture presented in the third section and used for analyzing the sound coming from beehives to improve bees' wellbeing and optimize decision making processes. In the fourth section we address implementation challenges of this smart IoT embedded system that uses Edge ML for beehive noise analysis. The data feeds are processed locally, at the beehive-level, using machine learning techniques and neural networks, to help take decisions without relying on Cloud resources for additional processing. The last section presents conclusions and future work, highlighting the importance of the Edge ML.

2 System Architecture for IoT Monitoring Solutions

This section outlines the hardware specifications and the system architecture utilized in the proof of concept (PoC) solution discussed in this paper, demonstrating the effective deployment of edge machine learning within the Internet of Things (IoT) domain. By leveraging IoT sensors and devices, we enhance beehive maintenance, ensuring an optimal health state through deep learning inference applied directly to the hive noises at the edge. This achievement is facilitated by employing TinyML to develop the application's audio analysis functionality. This technology enables Deep Learning - Machine Learning inference to be performed on a low-power microcontroller device, potentially leading to a significant improvement in the autonomy of beekeeping solutions. The monitoring system implements a wireless node that collects data about several parameters of the hive to assess the internal conditions of a bee colony. The acquired parameters were the temperature and humidity inside the colony and the weight of the beehive. A microphone was also added to the system, and it was used to measure the amplitude for certain frequencies of the sounds produced by honeybees. It's hard for a beekeeper to correlate the variations in recorded amplitudes with changes produced inside a beehive. Therefore, a feature that can analyze the sound inside the hive and assess its health status would be very useful for a beekeeper. The idea is to upload the data collected by all the IoT sensors to a cloud data center, where it's further processed by large Artificial Intelligence models. Latest technological advances enable Machine Learning algorithms to run at the edge, on tiny, resource-constrained devices. This approach brings a lot of benefits such as energy efficiency, low bandwidth, and enhanced security, but it also requires developers to be aware of certain device constraints when building the application. Machine Learning model needs to be small enough to fit the memory constraints of the device it will

be deployed on. The purpose of this study is to analyze the sound produced by honeybees using a deep learning model that runs on an embedded, low-power device.

2.1 Hardware and Software components

The proposed honeybee monitoring solution is designed to collect relevant data about the beehive, including the sound emitted by honeybees, analyze it using a tiny neural network that runs on the acquisition system, and display the status of the hive to the beekeeper. The system allows the beekeeper to act at the right time to maintain the honeybees healthy, so the solution must be accurate and reliable. The proposed architecture comprises of five major components:

A. Sensor system – The natural flow of this solution begins in the hive, where the acquisition system collects relevant data, including temperature, humidity, weight, and sound samples. The sound is analyzed using an embedded machine learning model and the processed information is published using the MQTT protocol [26], to a topic.

B. MQTT Broker – The broker receives the payload and transmits it further to the subscribed clients (the web service).

C. Web Service – The web service subscribes to the topic, receives information that describes the internal condition of the hive and stores it in the relational database. The web service is also used by the mobile application to access hive data or to perform other operations on the database (saving or updating information about the hives).

D. Relational Database – A relational database is used for permanent storage of hive conditions and for keeping track of the hives, inspections and harvests performed by the beekeeper.

E. Android Application – The mobile application is used by the beekeeper to monitor the beehive internal conditions, to check the queen's presence and to record relevant information about the beekeeper's activity.

Figure 1 presents the main components and the flow of the system architecture.

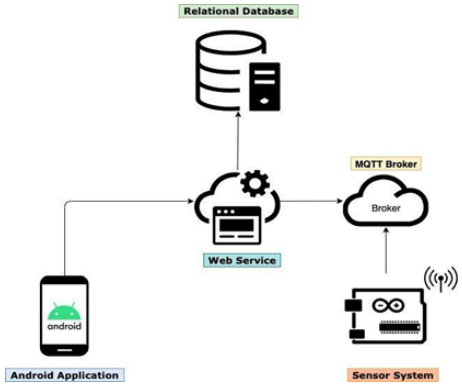


Figure 1: IoT Monitoring Solution PoC Architecture diagram for the beehive

In terms of hardware components for the sensors system, a set of IoT Embedded devices have been deployed. The low-power microcontroller chosen for this project is the Arduino Nano 33 BLE Sense [2]. This board features a 64 MHz CPU, 1 MB of flash memory, 256 RAM and it comes with a selection of embedded sensors, including a digital microphone, a temperature sensor, and a humidity sensor. Furthermore, the key feature of this tiny microcontroller is the ability to run Edge Computing applications on it using Tiny ML [46]. Arduino Nano collects the sensor data, processes the audio recording, and sends the results to the Wi-Fi Module through the serial port. The Wi-Fi module is used to publish the data received from the main microcontroller to an MQTT topic. The Wi-Fi module checks first if it's connected to the broker before trying to send any data. The hive parameters are transformed back to floating point values and a json payload is created using the sensor data and a product key. The product key is a unique identifier for the sensor system placed in each hive. SparkFun Load Cell Amplifier is used for measuring the weight of the beehive. By connecting the amplifier to the Arduino board, changes in the resistance of load cells can be read [38]. Precise weight measurements were obtained after the scale was carefully calibrated.

The MQTT or Message Queuing Telemetry Transport is a lightweight messaging protocol frequently used with IoT devices due to its efficiency and optimized network bandwidth [26]. The protocol is based on the publish/subscribe pattern, where a client publishes messages on a topic (message queue), and consumers subscribe to the MQTT topic to receive those messages. The clients don't communicate with one another directly, but instead connect to a broker. In this case an open-source broker implementation, called Mosquitto, has been used.

There is also NodeJS application acting as a backend component that plays two major, important roles:

- It's a RESTful web service that acts as an intermediary between the Android application and the MySQL database. To perform certain operations on the database like saving or retrieving hive information, the mobile application sends a request to the NodeJS service. The service receives the request, performs the operation on the database, and sends a response back to the mobile application.
- It subscribes to the MQTT topic to receive updates regarding the hive conditions and stores them in the relational database; the server first connects to the Mosquitto broker, subscribes to the topic, and listens for incoming messages on the subscribed message queue.

Each time the server receives a message, it verifies the integrity and authenticity of the received payload by computing the HMAC value based on a previously established secret key (the secret key is the same in the Huzzah module and the NodeJS server). The hashing function from the algorithm assures the integrity of the payload and the secret key guarantees the authenticity. If the payload would have been secured with only a hash value, an attacker could change the payload and recompute the hash during transmission, so the hash value would still be valid on the server side. The query returns a row for each parameter type, which corresponds to a value returned by the sensor system. One by one, each value is saved in the database along with the hive id and the parameter type id. The hive management information provided by the beekeeper in the mobile application is private and needs to be properly secured during transmission. Therefore, the system ensures an encrypted connection between the RESTful web server and the mobile application by implementing an SSL/TLS handshake protocol communication. Figure 2 shows the activity that displays a list of beehives that are part of an apiary.

Figure 2: Mobile dashboard for the beehive status

A panel which shows the measured parameters of the bee colony is displayed for each hive that is equipped with a smart sensor system. Besides the temperature and humidity inside the hive and the measured weight of the hive, the panel used to display the amplitude for a lower frequency interval between 200-220 Hz and the amplitude for a higher frequency interval between 400-440 Hz. It would be difficult even for an experienced beekeeper to correlate the variations in the displayed amplitudes

with changes produced inside the monitored beehive. Now that the sensor system is smart enough to return a certain result based on the sound analysis, the panel displays the scores for each possible result: the probability that the queen is present in the hive and the probability that the queen is missing from the hive based on its acoustics. If the score related to the queen less status is high (over 70%) it means that the beekeeper should inspect that specific beehive and see if the queen is missing from the hive. If the decision turns out to be a real issue, then the beekeeper can take the proper action and provide a new queen bee for that beehive.

2.2 Model deployment on edge

Embedded system programming - after converting the model with the compiler provided by Edge Impulse, the resulted Arduino library had to be deployed on the board. The library was integrated into an Arduino project that collects the data from the sensors, runs inference on the audio recordings captured in the hive and sends the results over serial communication to the Wi-Fi module. Every Arduino sketch is comprised of two main sections: the initialization phase placed in the setup() function and the main loop that runs continuously and is placed in the loop() function. In the initialization phase, the serial communication with Feather Huzzah module is established and the transmission rate is specified at 9600 bits per second. The humidity/temperature sensors and the scale are also initialized. The scale is calibrated using two parameters: calibration factor and zero factor. These two values were previously selected to make sure the recorded weight is accurate. Audio processing is also prepared by initializing the structures used at inference and configuring the Arduino's microphone. Finally, the function is verifying if the Feather Huzzah module is successfully connected to Wi-Fi by checking a flag received from the module. After setup is complete, the sketch runs the loop() function continuously, where all the operational logic takes place. The sensors collect the data, which is then processed and sent to the Wi-Fi module. This section of code is repeated every 30 seconds. Once the microphone has been configured to sample audio at 16 kHz rate, this rate was also used for the dataset collection, it will continuously read the analog voltage signals that are coming through the microphone and gets a digital representation into the audio buffer. The readAndAnalyzeSound() method first waits until the audio buffer is filled up with raw data by the microphone. The microphone captures a 2 second audio recording from the hive that is further preprocessed and analyzed by the tiny convolutional model. The buffer is then used to extract features from the audio signal, so the model can perform the inference process on the generated spectrogram features. The preprocessing steps happens before the inference process, which is effectively invoking the Tiny ML model. The classification results are stored in a resulting array structure. Once the system has made an inference, the classification array is traversed to store the results for each array item associated with the "queen" and the "queen less" label. The scores are transformed into integer values before being sent to the Wi-Fi module over the serial port, also known as UART. They are first collected in a byte array using bitwise operators and then transmitted one byte at a time. Bitwise operators can only be applied to integer values, and this is why the collected data was transformed into integer after reading. A flag is transmitted first, to make sure the Huzzah module is synchronized with the Arduino board and that it doesn't start reading halfway through the payload. The setup() function in the Wi-Fi module initializes the UART communication, tries to connect to the Wi-Fi network and transmits to the Arduino board if the Wireless communication was successfully established. The role of the Feather Huzzah board is to simply receive the sensor data and to publish the information to an MQTT topic, to communicate with the NodeJS server. The Huzzah board must establish the Wi-Fi network communication with an access point first to connect to the broker and to publish the sensor data to an MQTT topic. Each network can be identified by a Service Set Identifier (SSID), which is essentially the name of the access point. The easy way to connect the board to an access point is to hard code the SSID and the password of the Wi-Fi network in the Arduino sketch. Even though this approach is straightforward and quite simple, it has certain drawbacks. Each sensor system needs to be re-programmed with the credentials of the user's Wi-Fi network. The loop() function receives the data transmitted by the Arduino board, verifies if the module is still connected to the Wi-Fi network and tries to automatically reconnect otherwise. If the Wireless communication is still active, a payload that contains the sensor data is published to an MQTT topic.

3 Solution Engineering for Machine Learning Decision Support

3.1 Feature engineering

Determining whether the hive is queenright or queen less is a binary classification problem that falls under the supervised learning techniques [29]. Data engineering is an important component of supervised learning and consists of collecting relevant data and preprocessing it to extract relevant features that can predict the variable of interest. Data collection is a critical and time-consuming step that can imply active data collection or curation of information from external sources, such as pre-existing datasets.

There is an existing public dataset [30], for beehive sound recognition that contains audio samples for queen less and queenright beehives, but we chose to collect our own recordings and go through various steps like collecting, labeling, verifying, and curating the recordings from a beehive. The bees were recorded when the queen was present in the hive, and after it was manually removed by a beekeeper. The bees can recognize if the queen is missing after a few hours because the level of queen pheromones starts to drop off, which causes a change in their behavior. Therefore, the honeybees were recorded in different days.

Edge Impulse platform was used during the ML workflow, starting from data collection. This platform is used for building end-to-end an embedded machine learning model and provides various features such as automated data recording, low-code access to preprocessing blocks and leading algorithms, testing, live classification, and deployment [11].

To get started on building the prototype, a dataset with approximately one hour of audio samples from both classes has been created: “queen” and “queen less”, that was automatically split between the training and test datasets. Audio data was collected using the MP34DT05 digital microphone mounted on the Arduino Nano 33 BLE Sense board. The recordings have different sample lengths, but they will be further preprocessed in smaller equally chunks, so it doesn’t impact in any way the process, maintaining the balance between the two classes.

The audio samples were recorded with a sampling rate of 16 kHz, which means that 16 000 data points were picked up in a second. Training and inference are being performed on 2 second audio data chunk and not all that information needs to be fed to the network. The raw audio data needs to be preprocessed before being served as input to the model in the training step. A spectrogram processing block is used to transform the audio signal received from the hive into an input that is easily digested by the CNN model when training it. A spectrogram is a better input for a deep learning model compared to a raw audio signal as the critical features are already extracted and so the neural network doesn’t need to learn how to do that on its own. The spectrogram is a visual representation of the signal that shows the frequencies present in the sound and how they are changing over time in amplitude. This type of preprocessing block works well with audio data for non-voice recognition applications, as well as any audio signal with continuous frequencies [39]. Figure 3 presents the spectrograms generated from the audio sample of the colony in the queenright (a) and queen less states (b). From the two distinct states, it can be observed that the samples recorded when the queen was present in the hive have a stronger signal than the samples recorded when the hive was queen less.

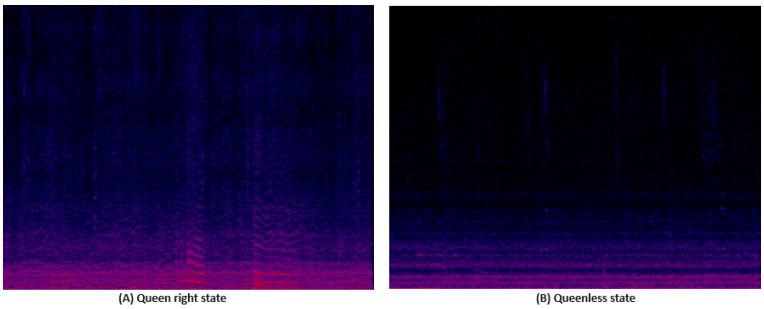


Figure 3: Spectrogram for (A) Queen right state / (B) Queen less state

To build the spectrogram, each window is first split into several overlapping frames. The length

of each frame was set to 0.02 seconds and the stride between successive frames was 0.01 seconds. The digital signal in the time domain, time on the X-axis and amplitude on the Y-axis, is transformed into the frequency domain, frequency on the X-axis and amplitude on the Y-axis, using the Fast Fourier Transformation (FFT) algorithm [48] where each frame is divided into 128 frequency bands. The FFT is computed for the whole window to construct the spectrogram. By applying the FFT through a sliding window procedure the intensity of each frequency can be analyzed over time. The X-axis of the spectrogram plot represents time, and the Y-axis represents the frequency. The Z-axis, that is often represented in colors, is the amplitude of each frequency band. The highest frequencies can be found on the top part of the spectrogram and the lowest frequencies at the bottom.

3.2 Model engineering

In this section, the neural network design is presented, and the model is trained to differentiate between the sound emitted by a queenright colony and the sound of a queen less colony. One of the most popular approaches of machine learning is deep learning, which is inspired by the inner workings of the human brain and works very well on the types of applications suited to microcontrollers [13]. Deep learning algorithms use neural networks, which are comprised of layers of simulated neurons. These layers are trained to find patterns in the input data and model relationships with the classification results. Different architectures of simulated neurons can be used, depending on the task that needs to be performed. A Convolutional Neural Network (CNN) architecture is used for this audio analysis task, as it performs very well at extracting meaning from spectrograms, which are basically images. The particularity of a CNN architecture is given by the convolutional layers, despite the fact that it can have other types of layers as well, that support with filtering and classification. The algorithm feeds the spectrogram into the first layer of neurons, which is then transformed based on the internal state of each neuron. The activations of the first layer are fed into the second layer, and so on, sequentially transforming the spectrogram's input into only two numbers: the probability that the audio input is received from a queenright colony, and the probability that the sound is emitted by a queen less colony. Although larger models are definitely more powerful and capable, they are not appropriate in this situation where the model needs to run on a microcontroller, where on-device memory and processing power are not enough to handle such large neural networks. The board has only 256 KB of RAM memory and 1MB of CPU Flash memory for storage, which brings certain limitations when designing the model [42]. Figure 4 presents the neural network architecture and the types of layers used are described below.

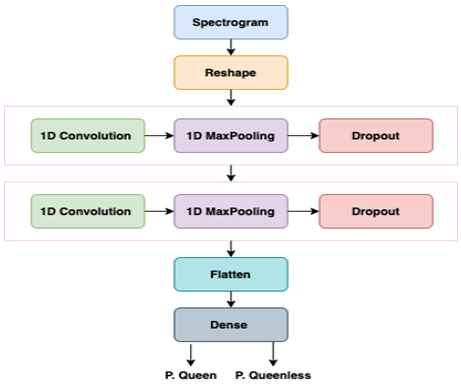


Figure 4: Neural network architecture

The convolution layers are applying filters to an input image to automatically pick out relevant features. A filter (also called kernel) is a matrix with weights that multiplies the pixels in a filter-sized window from the input image. The values resulted from the multiplication are then summed, producing a new value which corresponds to a pixel in the output image. The filter then slides over to the next set of pixels and repeats the same technique for the entire image, resulting in a new filtered image. During the training process, the filter weights are automatically updated so that the model learns what features are most important for distinguishing between the two acoustic signatures. Each

pixel of the output image is passed through the Rectified Linear Unit (ReLU) activation function, which transforms the negative values from the filtered image to zero. This function helps the network to decide if a neuron should be activated or not. 2-D convolutions, that involve traversing the entire image using a small filter, are very time-consuming. The convolutional layers used in this architecture have one dimension, and this means that the filter only moves along the time axis of the spectrogram image. The CNN architecture contains two convolutional layers, and the second layer detects features that are more complex compared to the first layer. The first convolutional layer consists of eight filters and the spectrogram image is copied to each of these filters. After filtering, max pooling and dropout, the resulted images are copied to each node of the second convolutional layer. The second layer has 16 filters, and it will generate 128 different images. The output of the convolutional layer is then passed to a 1-D max pooling layer. A max pooling layer also contains filters, like convolutional layers, but instead of computing the dot product, the highest number from that window is selected. The filtering and max pooling layers reduce the size of each filtered image, which saves memory and processing power. The Dropout layer is added after the max polling layer to prevent overfitting by ignoring some of the outputs from the previous layer during training. As a result, the network will generalize better for new spectrograms. The final part of the network is a dense layer that classifies the features. This layer is expecting a one-dimensional input so the features extracted from the previous layers must be concatenated and flattened into one array. The layer consists of two nodes corresponding to each label. The flattened array is copied to each node, that further computes a weighted sum based on its parameters (weights and biases). The classification layer uses the softmax activation function to output the probability for each class.

4 Decision Support for Beehive Management Using Convolutional Neural Networks

The proposed solution presented in this paper, that was also implemented based on an IoT architecture powered by Machine Learning algorithms proved to be very efficient when used on beehive datasets that are hiding patterns such as if the queen bee is present or not in the hive. The solution uses a convolutional neural network, CNN, that has basically the same configuration as a multilayer perceptron, MLP [24], [36]. The machine learning model is trained in the cloud with data gathered from sensors deployed next to the beehives. Then the model is deployed in a Tiny ML architecture that swiftly analysis beehive parameters and manages to decide on whether it requires assistance or not from the beekeeper. Figure 5 presents the workflow of the Tiny ML implementation, analyzed further in this paper.

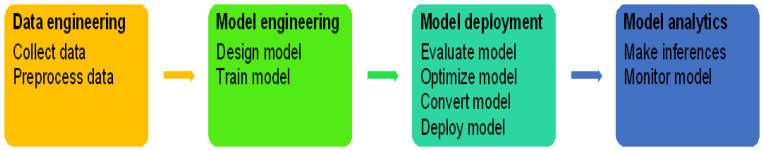


Figure 5: Tiny ML workflow implemented by the PoC

The machine learning model consists of multi layers that are used to extract feature maps from the provided inputs, also known as activation maps [16]. The input is passed through each layer until it's conveyed in a simpler interpretation. CNN models are very efficient against high resolution images compared with MLPs that are not capable of easily mapping so many neurons for each image component. This is the reason why the sound noises coming from the hive are going to be translated into images called spectrograms that will better reflect the sounds inside the CNN model. If we take just a 100x100 pixels rasterized bitmap as input to the first layer of a fully connected MLP, the network will need 10.000 weights for each neuron in the following layer. The structure of a CNN, follows the same principles as an MLP network, consisting of an input layer; some hidden layers, and an output layer. The hidden layers are called convolutional layers, they are used to convolve the input, passing the output to the next layer. Moreover, a CNN can also have a Rectified Linear Unit, RELU [25], that

can help increase the non-linearity of the input image which represents its natural state. Compared to other types of activation functions such as Sigmoid and Tanh, RELU has the following advantages:

(a) Low computational complexity – uses only a max function compared with exponential calculation.

(b) Can output a zero value – Sigmoid ant Tanh only manages to approximate.

(c) It has a linear behavior.

Another important component of the CNN is known as the one that reduces the dimensionality of the feature maps that are created by the convolutional layers, thus reducing the general complexity. The pooling layers, as they are called, take advantage of the hierarchical patterns that were identified as feature maps by the convolutional layers, creating other patterns much smaller and simpler that are embedded in filters. The study employed a four-stage methodology, Figure 6, to develop a model that utilizes deep learning algorithms and IoT systems to classify beehive noises into two distinct categories. The initial stage involved collecting data from beehives using sensors installed at their locations. This data was then collected and pre-processed in the second stage, preparing it for the third stage, which focused on extracting features essential for building a robust deep-learning model. The final stage involved deploying this model on edge machine learning devices to perform inference on newly collected data.

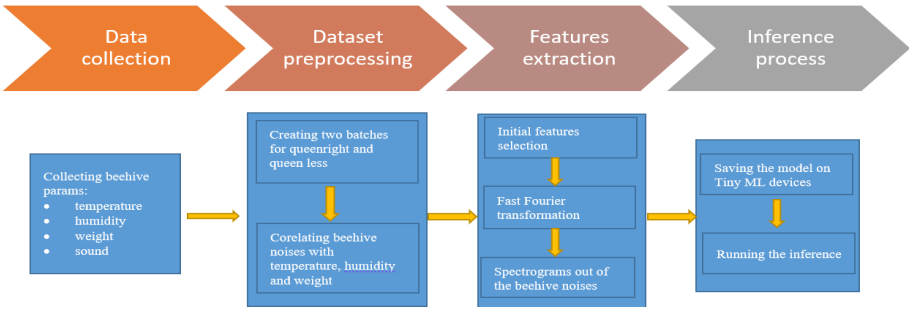


Figure 6: Research methodology

Figure 7 depicts a general model of a CNN, perfectly capable of distinguishing and classifying sound patterns coming from a beehive. The last layer is the prediction layer which gives the classification output for the entire network.

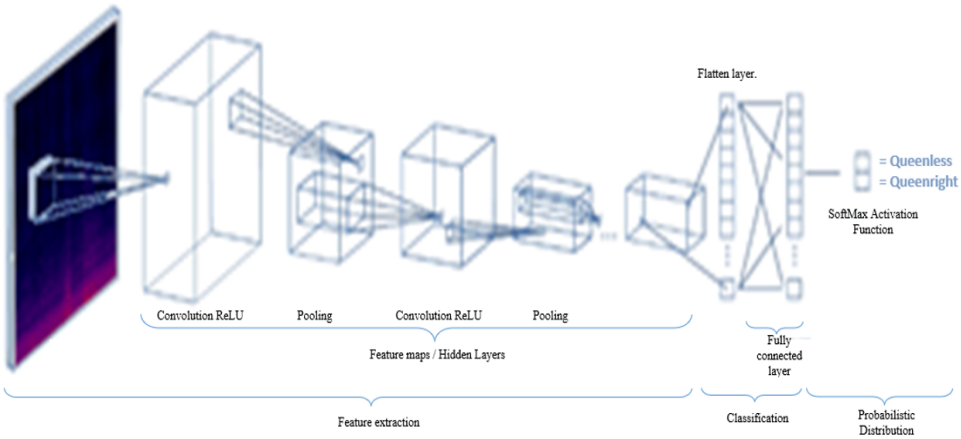


Figure 7: Convolutional Neuronal Network used by the PoC

The primary input into this deep learning architecture consists of the spectrograms of the noises coming from queen less or queenright beehives. The final output tells us if the spectrogram pertains to one of those two cases, so beekeepers could swiftly act and go on preserving the beehive by adding new frames with eggs or larvae until the bees are capable of replacing their lost queen or add a new mated queen. The similarity between Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) in terms of information flow management stems from their shared foundational principles.

Can be said that a CNN represents a regularized version of an MLP network where the disadvantages of the last one, in certain contexts, are efficiently resolved, like vanishing and exploding gradients that may arise in the backpropagation phase of a traditional neural network [3], [44]. In a CNN these problems are solved using regularized weights with a much smaller number of parameters. Because an MLP, known also as a fully connected network, uses layers of neurons where each neuron from one layer connects to all the neurons from the next layer, this makes the network prone to errors, such as an easily over trained network. Regularization techniques address issues such as the accumulation of weight in specific network regions, which can lead to neurons activating falsely or not activating at all. The approach on regularization that CNN models use is that each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field, in this way also lowering the complexity. The dropout component is especially important since it will randomly drop out nodes during training, also known as the pruning process, [16], which is a highly effective regularization method to reduce overfitting and improve generalization of errors [40]. In [21] we can see relevant results of different sound classification techniques based on spectrograms in which CNN leads with around 73% accuracy, against other proposed systems, as depicted in Figure 8.

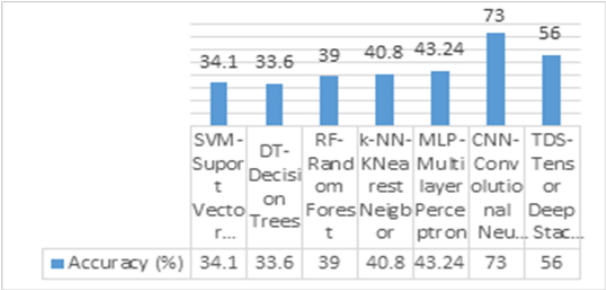


Figure 8: Accuracy of different Machine Learning models

In the proposed architecture a 1D Convolutional Neural Network, Conv1D, is used because of how the data that we want to process is structured. Using Conv1D will allow us to also use less resources than compared with a 2D or a 3D CNN. The spectrograms are basically structured as a 2D input data, the time axes and the amplitude of the sound that is coming from the beehive. This type of input data for non-voice recognition systems is suitable to be fed into a CNN due to its continuous frequencies over a time period. The proposed architecture is composed of two hidden layers that both manage to extract relevant features from the spectrograms that were fed in. The first convolutional layer includes an eight-filter component. To have a look into the convoluting process we need to present a numerical example based on which the mathematical formalism can be deduced. The CNN is fed with an array of spectrograms, images that need to be trained in order to get the corresponding weights for the network. The first layer of the implemented CNN has 8 filters and the second one has 16 filters. The sliding dot product is applied after doing a padding on the initial data both to the beginning and end of the time series so that we won't trim the initial data and take every bit of it in the process of convoluting. Figure 9 reveals a frame window in the convolutional process of the CNN architecture with the following parameters: filter/kernel of size 3, stride value of size 1.

The hyper parameters for the CNN model that is presented are chosen so that in the end, the CNN model will output values for two distinct neurons one for each classification class. The MLP layer for the convolutional neural network will be the one that, after extracting the relevant features from the spectrograms, will be able to tell if the input is pertaining to a queen less or queenright beehive. In order to start the mathematical formalism, we will do so by specifying that the spectrogram, x, is of length m, the filter / kernel, y, is of length n. The output, z, should have the same size as the input.

$$x = [x_0, x_1, x_2, ..., x_m - 1]$$
$$y = [y_0, y_1, y_2, ..., y_n - 1]$$
$$z = [z_0, z_1, z_2, ..., z_m - 1]$$

			0	1	2	3	4	5	6	7	
	input	0	1	0	-50	-13	0	48	0	0	0
1	kernel	2	9	5							
	input	0	1	0	-50	-13	0	48	0	0	0
2	kernel		2	9	5						
	input	0	1	0	-50	-13	0	48	0	0	0
3	kernel			2	9	5					
	input	0	1	0	-50	-13	0	48	0	0	0
4	kernel				2	9	5				
	input	0	1	0	-50	-13	0	48	0	0	0
5	kernel					2	9	5			
	input	0	1	0	-50	-13	0	48	0	0	0
6	kernel						2	9	5		
	input	0	1	0	-50	-13	0	48	0	0	0
7	kernel							2	9	5	
	input	0	1	0	-50	-13	0	48	0	0	0
8	kernel								2	9	5
	Output		9	-248	-515	-217	214	432	96	0	
			0	1	2	3	4	5	6	7	

Figure 9: Frame window in the convolutional process of the CNN with the parameters: filter/kernel of size 3, stride value of size 1

In order to simplify the notations the kernel size can be expressed as 2p+1 and flip it so that in the end we should work with the following kernel notation. This is possible because we chose a kernel that has an odd value.

$$y = [y_{-p}, y_{-p+1}, ..., y_0, ..., y_{p-1}, y_p]$$

So the output, z, is going to be calculated based on the dot product sliding through the input and kernel with the following formula:

$$z_j = \sum_{k=-p}^{p} x_{j-k} y_k$$

Based on the previous output array we can further proceed and apply the max pooling layer having a stride of value 2 with a window of value 2, as shown in the Figure 10.

9	-248	-515	-217	214	432	96	0	max pooling
								stride 2
		9	-217	432	96			window 2

Figure 10: Frame window in the convolutional process of the CNN with the parameters: filter/kernel of size 2, stride value of size 2

For the final computation the CNN with two convolutional layers having the following hyper parameters: filter size of value 3, stride of value 1, and subsampling, one for each convolutional layer, pooling MP1 of value 2 and pooling MP2 of value 2, it produces 16 features arrays, each of 23 values, as depicted in Figure 11.

		filter size - 3 / stride - 1 / MP1=2, MP2=2			
		8 filters		16 filters	
		Conv size	MaxPooling	Conv size	MaxPooling
input size 1x100		8x98	8x49	16x47	16x23

Figure 11: Frame window in the convolutional process of the CNN with the parameters: filter/kernel of size 3, stride value of size 1 with 8 and 16 filters

After getting through all the convolutional layers the 16x23 features arrays are flattened into one single array. The CNN layers are processing the raw 1D audio sample in small chunks, learning how to extract features that are used in the classification process performed by the MLP layers. The last MLP layers help the CNN network to classify data using an array of 16*23=368 input values. These

values are passed to a feed forward neural network having a first layer of 19 neurons, with a matrix of weights of size [368,19], which is fully connected to the output layer with only two neurons for each classification class. The output neurons finalize the process of prediction using the softmax activation function that outputs the probability for each class.

5 Results and discussions

The model was trained in Edge Impulse using the TensorFlow framework. TensorFlow is a machine learning framework developed by Google that is used to build and train especially deep learning models, and to deploy them in the cloud or to on-premises infrastructures [41]. Data was gathered from a beehive for a span interval of 3 months, between March and June 2021, when the hive is at its full throttle. Small 16kHz samples of around 12 seconds each were recorded randomly across different days when queen was present or removed from the beehive. A total of approximately one hour of recording time was done for each beehive consisting of around 300 samples split into 75% for training and 25% used for testing. This accounts roughly for a total of 23 minutes (queen) and 23 minutes (queenless) time in 227 training samples and 10 minutes (queen) and 4 minutes (queenless) time in 73 testing samples. The model is trained for 100 epochs, using a learning rate of 0.005. During the training process, the parameters associated with each neuron are gradually tweaked, so that the layers transform the spectrogram input in the right ways to produce the correct output. This is accomplished by feeding into the network a training sample, determining how far the output is from the correct response, and updating the neuron's parameters to increase the likelihood of producing a correct answer next time. The loss function used in training the model is the categorical cross entropy function, which is applied in classification tasks. Its aim is to calculate the difference between the two probability distributions: the output predictions and the target predictions. The results of the inference process, Figure 12 (right), which was subsequently performed on the beehive received data had similar results with the training process of which pattern distribution is presented in Figure 12 (left). The ratio of correctly predicted positive observations to the total predicted positive, known also as the precision was around 97%. The ratio of correctly predicted positive observations out of all the observation predicted, known as the recall rate was at 98%. Finally, the F1 score, that combines both of the previous metrics was 0.98.

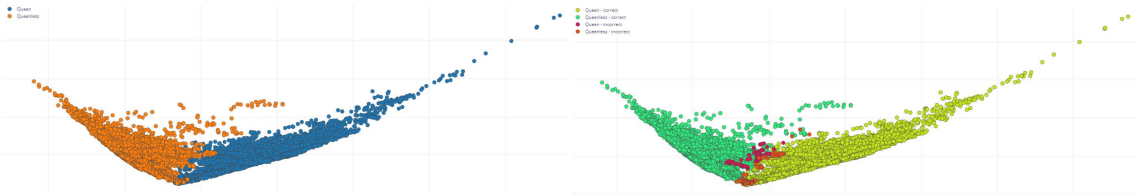


Figure 12: Distribution model for Training (left side) / Inference (right side)

TensorFlow models can run on powerful servers, but they don't fit on tiny microcontrollers. The model can be converted to a TensorFlow Lite format, that was specially designed to run on mobile platforms and embedded devices [46]. Therefore, once the training process is done, the model is converted using the TensorFlow Lite Converter. The converter's job is to compress the model into a format that it's suitable for running on constrained devices. It begins by pruning the model, which entails simplifying the model by removing memory-consuming elements from the network that may not have a major impact in the final result: for example, removing synapses that don't result in a drastic loss of accuracy, [16]. A special optimization called quantization is also applied by the converter to help reduce the size of the model and make it run faster. Quantization is the process of recasting the model's weights and biases from high-memory data storage such as float32 to a lower memory storage such as int8, without a significant loss in accuracy. Figure 12 shows the accuracy of the model after the training processed ended and the model was quantized.

The confusion matrix shows the balance between the audio windows correctly classified and the ones that were incorrectly classified. It represents the performance of the model in terms of number

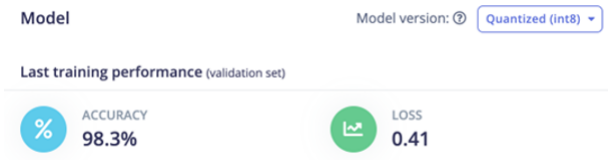


Figure 13: Model accuracy

of false positives and false negatives, Figure 13, stressing very well that the model performs better when the queen is not present which is actually what should be inspected further by the beekeepers compared with when the bee queen is present and the prediction says it is not.

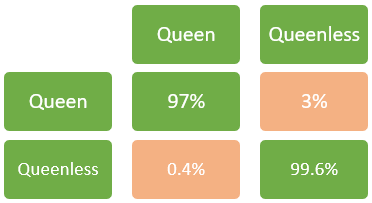


Figure 14: Confusion matrix

Before starting the training process, 25% of the training data was put aside for validation. The validation dataset was used to compute the model's performance after each training step and to fine-tune the model parameters. This score is a more reliable measure since the model was not aware of the data during the training process. The validation results are shown in the panel above, which provides important details about how well the model is performing. The model has an accuracy of 98.3%, which represents the percentage of spectrograms that were correctly classified. The model was also validated against the test dataset, to be sure that it did not overfit both the training and validation samples. The board obtained after classifying the samples in the testing dataset also provide the accuracy and the confusion matrix. According to Figure 14, the model also performed well on the testing audio samples.

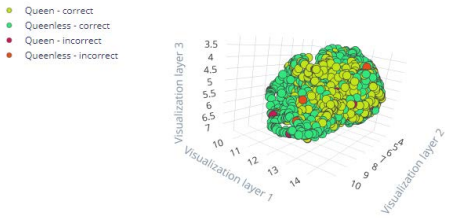


Figure 15: Test model validation

A comparable study was conducted on a low-energy architecture to analyze and monitor beehive acoustics, comparing deep learning and standard machine learning techniques. The results presented in [45] indicate that CNNs performed better on raw audio data compared to spectrogram analysis, while spectrogram analysis outperformed all standard machine learning models such as KNN, SVM, and Random Forest. Notably, the deep learning approach for raw sound analysis required around 80 hours of training, whereas spectrogram analysis included an additional step of converting raw sound into images. Despite this, the inference process on a low-energy system showed that spectrogram analysis outperformed traditional machine learning models but was comparable with raw sound classification. Because of their ability to bypass the learning of critical features, those being extracted when creating the spectrogram, the CNN with spectrogram analysis is an advantageous method for its ability to reduce background noise through image preprocessing and filtering before training and classification is done. After converting the model in an optimized format and evaluating its performance, the trained neural network was converted into an Arduino Library using Edge Impulse. The neural network was

compiled to C++ source code using the EON Compiler. This new compiler uses TensorFlow Lite for Microcontrollers under the hood and reduces the RAM and flash memory needed to run the model on the embedded device. The inference time for analyzing two seconds of data (estimated by default for ST IoT Discovery Kit) was 51ms and the model required 28KB of RAM to perform the inference. Therefore, the converted model fits quite well the restrictions of the Arduino microcontroller.

6 Conclusions

Hive acoustics can reveal useful information about the health status of bees. This study used the sound produced by a honeybee colony to build a system that can automatically determine whether the queen is present in the hive, allowing the beekeeper to easily monitor its state and take the proper action in a timely manner. A deep learning model was trained and optimized using Tiny ML, providing a very good performance in accuracy, size, and inference time. This paper presented every step of the workflow, from dataset collection until model deployment and integration with the server and mobile application. The study confirms that the audio signatures of queenright and queen less colonies can be distinguished by a smaller model, deployed on a tiny microcontroller. This approach is energy efficient since the system doesn't constantly send audio recordings to cloud and enhances the privacy of the beekeeper. However, building an accurate and reliable model that will perform well in production is quite challenging since hives are complex environments. For example, some analyzed beehives can have undetected issues that can cause the model to misinterpret the data. Another possible issue is the human error during dataset collection or the labeling of audio samples, since the person responsible for these tasks must have beekeeping experience and should recognize the behavior of queenright and queen less colonies. Honeybee colonies don't react in the same way when they lose their queen bee, and their behavior can vary from hive to hive, and even from one day to another. This could prove to be an issue in the presented study since the data was collected from one single hive for a not so long amount of time. This solution was designed as a proof of concept, but it can be improved by devoting more time and effort to developing a more reliable classification model. This can be achieved by collecting more data from different hives, which will improve its ability to generalize on new, unseen hives. Additionally, a better approach would be to record the hives automatically and to send the recordings to a server for a longer period of time, maybe even a year. This will help to gather a wider distribution of training samples, that would improve the resilience of the network to differences in the sound produced by the colony caused by seasonal changes in the honeybees' behavior. Another possible challenge would be to integrate additional parameters in the model, such as the temperature and the relative humidity. In order to see if these parameters are significant in detecting the beehive status, data should be collected with these additional factors and the model can be retrained and compared to the previous model. If the new model performs betters, then the additional parameters are relevant, and they should be kept in the model. If the parameters are not relevant, the model needs to learn to ignore the irrelevant information, and it may be prone to making false associations – maybe the hive lost its queen during winter when the temperature was lower. A better microphone can be used, that is more sensitive to the frequencies of the honeybees' sound and maybe a pass-band filter that blocks the frequencies which are outside of a specified limit, focusing on the range of interest. Additionally, the system can learn to detect other critical states of the beehive like swarming or the exposure to toxic substances. Therefore, more data can be collected from hives that present various anomalies to create a model able to provide a general health status. Technology platform wise the solution can be improved to use distributed training of neural network with Apache Spark. For this action, the solution exports the neural network model from Python Keras in Java deeplearning4j and use Apache Spark for training. After the training process is successfully accomplished, the neural network model can be imported back in Jupyter, in order to translate with Tensorflow Lite into the model for inference on C/C++ Arduino. As a final thought, the presented system can be seen as an affordable solution for analyzing the acoustic signature of a honeybee colony in order to detect its state and to help beekeepers react promptly by replacing the queen in case of need.

References

[1] Abadade, Y.; Temouden, A.; Bamoumen, H.; Benamar, N.; Chtouki, Y.; Hafid, A. S.; (2023). A comprehensive survey on tinyml, 2023 *IEEE Access*

[2] [Online]. Arduino, Available: https://store.arduino.cc/arduino-nano-33-ble-sense. [Accessed 9 April 2024]

[3] Balas, V. E., Kumar, R., & Srivastava, R. (Eds.). (2020). Recent trends and advances in artificial intelligence and internet of things. *Cham: Springer International Publishing.* ISBN 978-3-030-32644-9. 2020.

[4] Banner, R., Hubara, I., Hoffer, E. & Soudry, D. (2018). Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*

[5] Bencsik, M., Bencsik, J., Baxter, M., Lucian, A., Romieu, J., & Millet, M. (2011). Identification of the honey bee swarming process by analysing the time course of hive vibrations. *Computers and electronics in agriculture* 76(1), 44-50. 2011.

[6] Berkeley, U. o. C. (2006). Pollinators Help One-third Of The World's Food Crop Production. ScienceDaily. [Online]. Available: www.sciencedaily.com/releases/2006/10/061025165904.htm, [Accessed 9 April 2024]

[7] Bromenshenk, J. J., Henderson, C. B., Seccomb, R. A., Rice, S. D., & Etter, R. T. (2009). *U.S. Patent and Trademark Office. Patent US20070224914A1* U.S. Patent No. 7, 549,907. Washington, DC

[8] Cecchi, S., Spinsante, S., Terenzi, A., & Orcioni, S. (2020). A smart sensor-based measurement system for advanced bee hive monitoring. *Sensors.* 20(9), 2726. 2020.

[9] Cecchi, S., Terenzi, A., Orcioni, S., Riolo, P., Ruschioni, S., & Isidoro, N. (2018, May). A preliminary study of sounds emitted by honey bees in a beehive. 2018 *In Audio Engineering Society Convention 144. Audio Engineering Society.*

[10] Cejrowski, T., Szymański, J., Mora, H., & Gil, D. (2018). Detection of the bee queen presence using sound analysis. *Proceedings of Intelligent Information and Database Systems* 10th Asian Conference, ACIIDS 2018, Dong Hoi City, Vietnam, March 19-21, 2018, Part II 10 (pp. 297-306). Springer International Publishing. 2018.

[11] Edge Impulse Democratizes Machine Learning for All Developers on NVIDIA Jetson Edge AI Platform, Edge Impulse. [Online]. Available: https://www.prnewswire.com/news-releases/edge-impulse-democratizes-machine-learning-for-all-developers-on-nvidia-jetson-edge-ai-platform-301269308.html. [Accessed 9 April 2024]

[12] Europa.eu, "What's behind the decline in bees and other pollinators?," [Online]. Available: https://www.europarl.europa.eu/news/en/headlines/society/20191129STO67758/what-s-behind-the-decline-in-bees-and-other-pollinators-infographic. [Accessed 18 April 2024]

[13] Feed Forward Neural Webpage – "Deep Learning: Feedforward Neural Network. [Online]. Available: https://medium.com/hackernoon/deep-learning-feedforward-neural-networks-explained-c34ae3f084f1. [Accessed 9 April 2024]

[14] Ferrari, S., Silva, M., Guarino, M., & Berckmans, D. (2008). Monitoring of swarming sounds in bee hives for early detection of the swarming period. *Computers and electronics in agriculture* 64(1), 72-77. 2008.

[15] Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. *O'Reilly Media, Inc.*. Release Date: September 2019, ISBN: 9781492032649. 2022.

[16] Gholami, Amir; Kim, Sehoon; Zhen, Dong; Yao, Zhewei; Mahoney, Michael; Keutzer, Kurt. (2022). A Survey of Quantization Methods for Efficient Neural Network Inference. *Low-Power Computer Vision* 36 pg., eISBN 9781003162810, doi: 10.1201/9781003162810-13.

[17] Green, M.; Murphy, D. (2020). Environmental sound monitoring using machine learning on mobile devices. *Applied Acoustics*. 159, 107041, 2020.

[18] Google A.I. Cloud. [Online]. Available: https://cloud.google.com/ai-platform. [Accessed 21 March 2024]

[19] Honey Bee Queens: Evaluating the Most Important Colony Member. BEE-HEALTH, 18 August 2015. [Online]. Available: https://bee-health.extension.org/honey-bee-queens-evaluating-the-most-important-colony-member/. [Accessed 21 March 2024]

[20] Howard, D. Duran, O. Hunter G. and Stebel K. (2013). Signal processing the acoustics of honeybees (APIS MELLIFERA) to identify the "queen less" state in Hives. *Proceedings of the Institute of Acoustics.* 35. 290-297.

[21] Khamparia, A.; Gupta, D.; Nguyen, N. G.; Khanna, A.; Pandey, B.; Tiwari, P. (2019). Sound classification using convolutional neural network and tensor deep stacking network. *IEEE Access* 7, 7717-7727., Date of Publication: 08 January 2019, Electronic ISSN: 2169-3536. 2019.

[22] Kirchner, W. H. (1993). Acoustical communication in honeybees *Apidologie* 24(3), 1993.

[23] Koul, A., Ganju, S.; Kasam, M. (2019). Practical deep learning for cloud, mobile, and edge: real-world AI & computer-vision projects using Python, Keras & Tensorflow. *O'Reilly Media* Release Date: October 2019, ISBN: 9781492034865.

[24] Liang, J., Zhao, X., Li, M., Zhang, Z., Wang, W., Liu, H.,; Liu, Z. (2023, April). MMMLP: multi-modal multilayer perceptron for sequential recommendations. *Proceedings of the ACM Web Conference* (pp. 1109-1117). 2023.

[25] Mao, X.; Xiang, Y.; Lu, J. (2024). An efficient nonlinear adaptive filter algorithm based on the rectified linear unit. *Digital Signal Processing* 104373. 2024

[26] MQTT: The Standard for IoT Messaging. [Online]. Available: https://mqtt.org/. [Accessed 28 March 2024]

[27] Murphy, F. E., Magno, M., Whelan, P.; Vici, E. P. (2015). b+ WSN: Smart beehive for agriculture, environmental, and honey bee health monitoring—Preliminary results and analysis. *In 2015 IEEE sensors applications symposium (SAS)* pp. 1-6, IEEE 2015

[28] Murphy, F. E., Popovici, E., Whelan, P.; Magno, M. (2015). Development of an heterogeneous wireless sensor network for instrumentation and analysis of beehives. *In Proceedings 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)* pp. 346-351. IEEE. 2015

[29] Nettleton, D. F., Orriols-Puig, A., Fornells, A. (2010). A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review* 33, 275-306. 2010.

[30] Nolasco, I.; Benetos, E. (2018). To bee or not to bee: An annotated dataset for beehive sound recognition. [Data set], *Zenodo* https://doi.org/10.5281/zenodo.1321278, 2018.

[31] Nolasco, I., Benetos, E. (2018). To bee or not to bee: Investigating machine learning approaches for beehive sound recognition. *Workshop on Detection and Classification of Acoustic Scenes and Events*

[32] Online Doc for A.I. Cloud, [Online]. Available: https://cloud.google.com/ai-platform/docs/technical-overview. [Accessed 16 April 2024]

[33] Park, J., Yoo, T., Lee, S., Kim, T. (2023). Urban Noise Analysis and Emergency Detection System using Lightweight End-to-End Convolutional Neural Network. *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL* Vol. 18(5), DOI: 10.15837/ijccc.2023.5.5814

[34] Perry, L. (2020). IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics and security. *Packt. Packt Publishing Ltd* 2020.

[35] Qandour, A., Ahmad, I., Habibi, D.; Leppard, M. (2014). Remote beehive monitoring using acoustic signals. *Australian Acoustical Society* 42. 204-209.

[36] Ruck, D. W., Rogers, S. K.; Kabrisky, M. (1990). Feature selection using a multilayer perceptron. *Journal of neural network computing* 2(2), 40-48. 1990.

[37] Sarton, G. (1943). The Feminine Monarchie of Charles Butler, 1609 *The University of Chicago Press* Vol. 34(6), 469-472. 1943

[38] SparkFun Load Cell Amplifier - HX711, [Online]. Available: https://www.sparkfun.com/products/13879. [Accessed 21 March 2024]

[39] Spectrogram, Edge Impulse, [Online]. Available: https://docs.edgeimpulse.com/docs/spectrogram. [Accessed 11 March 2024]

[40] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I.; Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research.* 15(1), 1929-1958., ISSN 1532-4435. 2014.

[41] TensorFlow. [Online]. Available: https://www.tensorflow.org/. [Accessed 21 March 2024]

[42] TensorFlow Model Optimization Toolkit — Pruning API. Tensorflow, [Online]. Available: https://blog.tensorflow.org/2019/05/tf-model-optimization-toolkit-pruning-API.html. [Accessed 9 March 2024]

[43] Varol, E.; Yücel, B. (2019). The effects of environmental problems on honey bees in view of sustainable life. *Mellifera.* Vol. 19(2), 23-32. 2019

[44] Venkatesan, R.; Li, B. (2017). Convolutional neural networks in visual computing: a concise guide. *CRC Press.* ISBN 978-1-351-65032-8. 2017

[45] Vladimir K., Sarbajit M., Prakhar A. (2018). Toward Audio Beehive Monitoring: Deep Learning vs. Standard Machine Learning in Classifying Beehive Audio Samples. *APPLIED SCIENCES-BASEL.* Vol 8(9). eISSN: 2076-3417

[46] Warden, P.; Situnayake, D. (2019). Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers. *O'Reilly Media.* ISBN: 9781492052043. 2019

[47] Woods, E. F. (1957). Means for Detecting and Indicating the Activities of Bees and Conditions in Beehives. *U.S. Patent and Trademark Office.* Patent US2806082A, 10 September 1957.

[48] Yu, D., Zhan, X., Yang, L. J.; Jia, Y. (2023). Theoretical description of logical stochastic resonance and its enhancement: Fast Fourier transform filtering method. *Physical Review E.* Vol. 108(1), 014205. 2023

[49] Zgank, A. (2020). Bee Swarm Activity Acoustic Classification for an IoT-Based Farm Service. *Sensors* Vol. 20(1); https://doi.org/10.3390/s20010021.

[50] Zhao, T., Li, Y., Zuo, L.; Zhang, K. (2021). Machine-learning optimized method for regional control of sound fields. *Extreme Mechanics Letters.* Vol. 45, 101297, 2021.

[51] Zhao, Y., Deng, G., Zhang, L., Di, N., Jiang, X.; Li, Z. (2021). Based investigate of beehive sound to detect air pollutants by machine learning. *Ecological Informatics.* Vol. 61, 101246, 2021.

[52] Zou, Z., Jin, Y., Nevalainen, P., Huan, Y., Heikkonen, J.; Westerlund, T. (2019). Edge and fog computing enabled AI for IoT-an overview. *Proceedings of 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)* pp. 51-56. IEEE. doi: 10.1109/AICAS.2019.8771621. 2019

Copyright ©2024 by the authors. Licensee Agora University, Oradea, Romania.
This is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial 4.0 International License.
Journal's webpage: http://univagora.ro/jour/index.php/ijccc/

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

Cite this paper as:

Doinea, M.; Trandafir, I.; Toma, C..; Popa, M.; Zamfiroiu, A. (2024). IoT Embedded Smart Monitoring System with Edge Machine Learning for Beehive Management, *International Journal of Computers Communications & Control*, 19(4), 6632, 2024.
https://doi.org/10.15837/ijccc.2024.4.6632