Journal of the Brazilian Computer Society, 2025, 31:1, doi: 10.5753/jbcs.2023.4210

Environmental Monitoring with Low-Processing Embedded AI through Sound Event Classification

Bernardo Feijó Junqueira [**Rio de Janeiro State University** | ***bernardo.junqueira@eng.uerj.br***]
Rafael Greca Vieira [**Unidade Embrapii CPQD** | ***rgrecav@cpqd.com.br***]
Erika Costa Alves [**Unidade Embrapii CPQD** | ***erikaa@cpqd.com.br***]
Bruno Bilhar Karaziack [**Unidade Embrapii CPQD** | ***bilhark@cpqd.com.br***]
Marcelo Vieira dos Santos [**GreenBug** | ***marcelo.vieira@greenbug.com.br***]

Rio de Janeiro State University (UERJ), Department of Mechanical Engineering, 121 Fonseca Teles Street, Rio de Janeiro, RJ, 20940-200, Brazil

Received: 06 March 2024 • **Accepted:** 22 April 2025 • **Published:** 05 August 2025

Abstract In this work, we propose an embedded low-processing Machine Learning solution designed to assist in environmental acoustic monitoring. The pre-processing stage employs the Wavelet Packet Transform, generating low-dimensional features that serve as inputs to a Gradient Boosting model for the near-real-time classification of relevant sound events. Subsequently, we introduce an event filter that checks if there is any relevant event occurring at the moment before sending the features to the model or ignores them until any sound event is detected. This approach enhances the robustness of our solution, making it resilient to noise and wind-contaminated samples while optimizing memory, battery, and computational power usage. Finally, we converted the processing pipeline and trained model to the C programming language, successfully embedding them into the Nordic Thingy:53, a low-power hardware device equipped with a built-in digital Pulse Density Modulation microphone (VM3011 from Vesper). To evaluate the efficacy of our proposed method, we compared it with a convolutional neural network approach using Mel-frequency cepstral coefficients and conducted tests using audio recordings of bird species found in forests located in the central and western regions of Brazil, as well as samples of human activity-related sounds. The favorable classification scores obtained, in conjunction with the embedded solution's substantial battery life capacity, have the potential to greatly reduce the necessity for extensive environmental monitoring field surveys.

Keywords: Embedded AI, Environmental Monitoring, Sound Event Classification, Machine Learning, Wavelet Packet Transform, Acoustics

1 Introduction

Environmental sustainability is intrinsically linked to the biodiversity of animal species and the impact of human activities on diverse ecosystems. However, effectively monitoring the health of these ecosystems poses significant challenges, often requiring exhaustive field surveys [Burivalova *et al*., 2019, 2018; Deichmann *et al*., 2018]. The study of environmental sound has great potential to enhance biodiversity monitoring by analyzing acoustic signals to detect relevant sound events, thereby providing valuable insights into ecosystem dynamics. Nonetheless, establishing best practices for data collection and audio recording analysis remains an open task [Bradfer-Lawrence *et al*., 2019]. Furthermore, there is an increasing trend towards implementing Machine Learning (ML) models on end-devices, such as microcontrollers (MCUs), to perform diverse tasks in the field, particularly in scenarios where access to energy supply is limited or unavailable. However, these devices often have limited resources, which poses challenges and limitations in the development of embedded applications [Branco *et al*., 2019]. In this regard, the present work proposes a low-processing embedded ML application for the near-real-time classification of significant sound events, aimed at aiding in the biodiversity monitoring of ecosystems.

Birds are increasingly being used as valuable bioindicators in ecosystem monitoring due to their responses to variations in habitat, serving as an early warning of climatic or environmental changes [Mekonen, 2017]. Studies conducted by Morelli *et al.* [Morelli *et al*., 2021] in Europe and Wotton *et al.* [Wotton *et al*., 2020] in Africa have demonstrated the use of birds as bioindicators, observing that the presence of bird species varies according to the demographic and urban characteristics of each region within the countries. Furthermore, monitoring the decline of species may provide important insights about ecosystem health, as it can lead to consequences such as the loss of natural pest control, reduction or extinction of dependent species, pollinator limitation, and changes in cultural practices [Şekercioğlu çağan H *et al*., 2004]. Additionally, Burivalova *et al.* [Burivalova *et al*., 2018] used the impact of human presence as an indicator, which, depending on the severity, might cause the reduction of vocalizing biodiversity, consequently eliminating the dusk and dawn choir. This whole monitoring process can be quite costly and time-consuming, especially due to the need for data collection, labeling, and availability, which often require extensive expert man-hours, such as those of ornithologists.

To overcome the issues associated with bird species recognition, extensive investigations have been conducted in the development of automatic audio signal pattern recognition algorithms. Moreover, many researchers have explored the adaptation of speech recognition algorithms to effectively

perform this classification task [Cai *et al*., 2007]. In [Bardeli *et al*., 2010], the authors introduce an automated bird vocalization detector based on the analysis of temporal patterns within specific frequency bands by inspecting spectrograms. The proposed algorithm is evaluated on two endangered bird species. Potamitis *et al.* [Potamitis *et al*., 2014] propose an automated pattern recognition algorithm using Perceptual Linear Prediction cepstral coefficients (PLP-CC) in conjunction with hidden Markov models (HMMs) and related techniques. More recently, Kahl *et al.* [Kahl *et al*., 2021] developed a deep learning approach utilizing spectrograms as inputs to a Convolutional Neural Network (CNN) based on residual networks (ResNets). This approach shows considerable advancements in bird species recognition, potentially identifying up to 984 North American and European bird species by sound. In a more computationally cost-efficient approach, Yang *et al.* [Yang *et al*., 2021] utilized the energy derived from Wavelet Packet Decomposition coefficients of bird species syllables in conjunction with shallow learners, achieving high classification scores when tested on 12 distinct bird species' sounds. While significant progress has been made in environmental acoustic monitoring algorithms, there is a persistent demand for implementing these solutions in affordable devices to enable conservation researchers to conduct large-scale monitoring. However, developing onboard applications for such devices presents difficulties due to limitations in energy consumption, storage capacity, and processing power [Prince *et al*., 2019].

In that sense, the main contribution of this work is to propose a low-processing and affordable environmental monitoring tool capable of detecting relevant bird species' sounds specific to a particular ecosystem, as well as identifying sound events related to undesirable human activities. Similarly to the approach used by Yang *et al.* [Yang *et al*., 2021], we utilize energy features derived from Wavelet Packet Decomposition coefficients. However, in contrast, we employ these features as inputs to a Gradient Boosting model, which is computationally efficient and enables the ML model to be executed on low-processing devices. For this study, we embedded our solution in the Nordic Thingy:53, a low-power hardware device featuring an integrated Vesper VM3011 microphone. The proposed method is tested using audio recordings of bird species found in forests located in the central-western and northern regions of Brazil, as well as sounds related to human activities.

The remaining of this work is organized as follows: In Section II, we present the methodology used in the development of the application. In Section III, we introduce the data used to develop the ML model. In Section IV, we present the results for both the Python and the converted embedded solution. In Section V, we discuss the results. Finally, in Section VI, we draw the conclusions.

2 Methodology

In this section, we describe each step of the methodology employed to compose the proposed solution. It is worth mentioning that the ML model experiments were conducted using the hyperparameter tuning (HPT) framework Optuna with the Tree-structured Parzen Estimator (TPE) sampler (a Bayesian optimization method) and the MedianPruner, which minimize the objective function with the median stopping rule [Bergstra J. and B., 2011]. This approach allowed us to choose the best combination of input features and ML model parameters. Additionally, we discuss the integration of the solution into the selected hardware for this study.

2.1 Preprocessing and Feature Extraction

Firstly, it should be pointed out that there are some processing steps regarding only the dataset used to train the ML models. These procedures, when performed during the training stage, eliminate the need for additional processing steps within the hardware, saving the available processing memory. These steps are presented as follows:

- The available bird vocalization data is manually divided into syllables using the software Audacity, as can be seen in Figure 1. This process aims to capture different vocalization patterns of different species (increasing the variance of the data for each species) and avoid including chunks with no sound events when further splitting the signal [Yang *et al*., 2021].
- The data is converted to mono and downsampled to a sample rate of 16kHz, which aligns with the hardware-integrated microphone sample rate and the configured number of channels. It should be acknowledged that some bird species have vocalizations that exceed the 8kHz frequency range. For species whose vocalizations are entirely above this range, classification using the proposed approach may not be feasible. We recognize this as a limitation due to the used hardware (Nordic Thingy:53). However, Figure 2 shows the average Power Spectral Density (PSD) for each considered sound event before downsampling, demonstrating that all the species considered in this study can be characterized within the specified frequency range, with some loss of information observed only for the Orange-Headed Tanager, Pale-breasted Thrush, White-eyed Parakeet, and especially the White-chinned Sapphire.
- The resulting audio samples are then split into chunks of 0.5 seconds, a time length supported by the hardware, considering all the processing memory demand. During the experimentation procedure with the HPT, this duration was found to be close to the optimal time length.

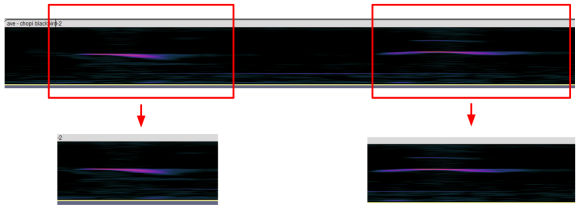

Figure 1. Example of bird syllable splitting. The top image shows the spectrogram of the entire audio sample, while the bottom image displays the syllables manually separated from the original spectrogram, based on areas with noticeable spectral energy density corresponding to vocalization patterns. After this process, the original audio is discarded.

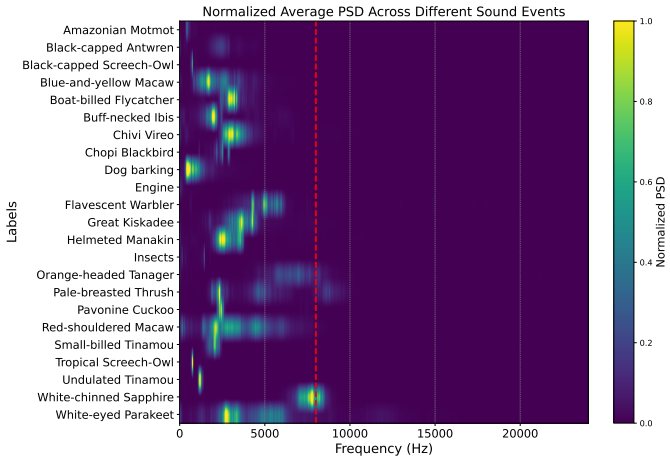

Figure 2. The normalized average PSD for each considered sound event before the downsampling to 16kHz. The red vertical dashed line represents the 8kHz frequency, which corresponds to the Nyquist frequency resolution for a sample rate of 16kHz.

The remaining processing steps are present in both the model training stage and the hardware-integrated solution and are described as follows:

- The audio sample is normalized to a range between -1 and 1.
- The Wavelet Packet Transform (WPT) at level 4 with a Daubechies wavelet filter of length six is performed in order to decompose the signal into a set of sixteen coefficients (C), defined as follows [Coifman *et al.*, 1994]:

$$C^k_{j,m}(x) \cong \left\langle x(t), 2^{-j/2} W_m(2^{-j}t - k) \right\rangle \quad (1a)$$

$$W^k_j(x) \cong 2^{-j/2} \int_{-\infty}^{\infty} x(t)\psi^*(2^{-j}t - k)dt, \quad (1b)$$

where $L^2(R),\ W_m(t),\ m \in N$, such that

$$\int_{-\infty}^{\infty} W_0(t)dt = 1, \quad (2)$$

and for all $k \in Z$,

$$2^{-1/2} W_{2m}\left(\frac{t}{2} - k\right) = \sum_{i=\infty}^{\infty} h_{i-2k} W_m(t-k) \quad (3a)$$

$$2^{-1/2} W_{2m+1}\left(\frac{t}{2} - k\right) = \sum_{i=-\infty}^{\infty} g_{i-2k} W_m(t-k), \quad (3b)$$

where $x(t)$ denotes the original signal, ψ^* represents the complex conjugate of the mother wavelet function, m is the decomposition level, j indicates the jth node of a specific level, and h and g stands for the impulse response of lowpass and highpass paraunitary Quadrature mirror filters (QMF), respectively [Gokhale *et al.*, 2010]. That is, the WPT decomposes both low and high-frequency bands of the signal, which overcomes the difficulty in differentiating detailed transient components, giving a good time-frequency resolution of the entire decomposed signal [Gao and Yan, 2010; Frusque and Fink, 2022]. It should be noticed that this processing approach is increasingly being used in acoustic classification tasks, demonstrating promising results, as can be seen in [Yang *et al.*, 2021; Bianchi D., 2015].

- Following the WPT procedure, the energy of each coefficient (ε) is extracted as follows [Yen and Lin, 2000; Yang *et al.*, 2021]:

$$\varepsilon(m,j) = \frac{1}{n_m} \sum_{l=1}^{n_m} C^2_{m,j}(l),\ 1 \le j \le 2^m, \quad (4)$$

where n_m denotes the number of data points for each coefficient, and $C_{m,j}(i)$ represents the i-th entry of the j-th coefficient at level m. This procedure results in a total of sixteen one-dimensional attributes for the audio signal under analysis.

- Lastly, the most informative features are selected from the sixteen generated energy attributes. This step was performed with the aid of HPT, where the input features were evaluated as one of the parameters in each model experiment. The results indicated that the optimal set of input features encompasses the energies extracted from coefficients 0 to 12.

It should be noted that even though the entire pre-processing procedure may not be straightforward or computationally inexpensive, it results in only thirteen informative one-dimensional input features. In comparison, using a Mel spectrogram feature approach, which is commonly addressed in the literature, would generate high-dimensional features that could potentially be used as inputs for a complex architecture CNN [Lasseck, 2018; Ntalampiras and Potamitis, 2021; Xie *et al.*, 2019; Zhang *et al.*, 2021]. However, such an approach would make the solution computationally costly and thus impractical to deploy on an end-device with limited processing and memory capacity. By utilizing the feature approach proposed in the present work, we were able to drastically reduce the computational cost of the ML model employed to perform the acoustic classification task. The complete data pre-processing procedure is illustrated in Figure 3.

2.2 Machine Learning Method

As mentioned in the previous section, the ML algorithm used to develop the model should be viable in terms of computational cost and yet perform with a satisfactory classification score for the target sound events. Dealing with embedded AI models is a difficult task, specifically when it comes to handling the trade-off between the artificial intelligence (AI) method and the pre-processing step [Situnayake and Plunkett, 2023]. This consideration is quite relevant since embedded solutions tend to have few memory resources and must run fairly quickly.

Considering that the data pre-processing algorithm is computationally expensive, yet extracting informative sound event features which simplifies the model inputs, we decided to use shallow learner methods rather than Deep Learning (DL) methods. Typically, DL methods are the preferred choice when dealing with problems that involve unstructured

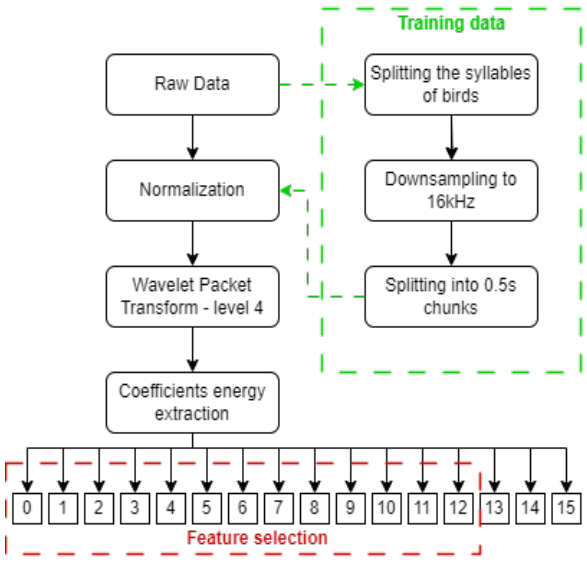

Figure 3. Flowchart of the entire pre-processing procedure. The green, dashed square represents the pre-processing stage, which is applied only to the raw training data; the testing data does not undergo this step. Next, either the processed training data or the raw test data is normalized before calculating the WPT and its coefficient energy. The red, dashed square indicates the indexes of the coefficients selected during the HPT step.

data, as these methods have the capability to perform feature extraction themselves [Goodfellow *et al.*, 2016]. However, in this work, all feature extraction is done during the pre-processing stage, enabling the use of a simpler and more cost-effective ML model, such as a shallow learner.

To perform a study to determine which shallow learner would be a better fit for this project, we conducted experiments involving a collection of classical algorithms, such as Gradient Boosting (XGBoost and LightGBM), Support Vector Machine (SVC, LinearSVC and NuSVC), Random Forest, Decision Trees, and Naive Bayes. The models were evaluated using the HPT method, in which each selected ML method was trained with different hyperparameters. After all the training and evaluation procedures, we compared the performance of the top-scoring models from each method. The results indicated that the Random Forest achieved the highest classification score. However, due to the large size of the saved trained model, it is not suitable for memory-constrained devices. Thus, we selected the second-highest scoring model, XGBoost, which not only delivers good performance but is also affordable in terms of memory consumption. Table 1 provides a brief analysis of all the shallow learners tested in this study. A more in-depth analysis of all tested models' classification performance can be found in Appendix B, while a detailed analysis of the selected shallow learner (XGBoost) is presented in Section 4.2.

2.3 Event Filter

To analyze only potential target sound events, the development of an event filter was crucial to remove audio samples containing only ambient noises or with significant wind noise contamination. As for ambient noises, we straightforwardly discard samples with maximum amplitude below a threshold of 1500 μPa in terms of sound pressure, which is equivalent to approximately 37.5 dB. However, discard samples highly contaminated with wind noises proved to be more challenging, since wind noise can easily be mistaken for the events of interest. Introducing another complex processing step to the solution would be impractical, as it could significantly increase the computational cost, making its hardware-integration unfeasible.

Table 1. Comparison of the shallow learners in terms of computational and classification performance, including peak processing memory, file size, and weighted F1 score, sorted by classification performance.

Approach	Peak processing (KB)	Size (KB)	F1 (%)
Random Forest	1466	45697	84.1
XGBoost	752	358	82.8
LightGBM	808	509	80.6
Decision Trees	746	733	73.7
SVC	740	2707	50.1
NuSVC	1325	3161	47.8
LinearSVC	689	4	32.0
Naive Bayes	982	6	19.9

To address this challenge, we used the already implemented data pre-processing algorithm, since WPT-based filters have shown its effectiveness in similar tasks [Juodakis and Marsland, 2022; Bianchi D., 2015]. Therefore, in this study, we employed a combination of energies from WPT coefficients, zero-crossing rate, and amplitude analysis. This allowed us to develop a wind filter that effectively identifies and addresses samples contaminated with wind noise. The methodology is presented as follows:

- Firstly, the algorithm checks if the maximum amplitude exceeds a threshold of 4000 μPa in terms of sound pressure, which is equivalent to approximately 46 dB, indicating a sample containing more than an ambient noise.
- Then, since wind noises are more active in lower frequencies, it is expected that the zero-crossing rate (the rate at which the signal changes sign) would be small in such samples [Nelke *et al.*, 2016; Honkakunnas, 2021]. Therefore, we established a superior threshold of 10% for the zero-crossing rate relative to the signal length, which corresponds to a maximum of 800 sign changes in a 0.5s chunk with a 16kHz sampling rate (a signal with 8000 time points), indicating a low zero-crossing rate.
- Lastly, it was observed that samples contaminated with wind noises tend to exhibit high energies in the two first WPT coefficients, while showing practically zero energy in coefficients related to higher frequencies. Hence, by analysing the mean and the standard deviation of wind-contaminated samples, we established an inferior threshold of 0.5 in the first coefficient energy (coefficient number 0), and a superior threshold of 0.5 for the sum of energies encompassing coefficients 4 to 15 (see Figure 2).

If the sample successfully passes through all of these verification steps, it is considered acceptable for further process-

ing in the shallow learner inference stage. It should be noticed that this procedure also contributes to reduce the overall processing time, since samples that do not pass through the ambient noise filter are not processed through the entire pre-processing procedure, and samples that pass through the ambient noise filter but not through the wind filter are not processed by the shallow learner. This selective processing approach optimizes computational efficiency by avoiding unnecessary computations for irrelevant samples and reduces the likelihood of misclassifications by the shallow learner.

2.4 Deploying the Solution into the Hardware

We selected the Nordic Thingy:53 as an on-field end-device for environment acoustic monitoring. This is a low-power hardware for Internet of Things (IoT) prototyping that has 512 KB of Static Random Access Memory (SRAM), 1 MB of Flash, and operates with a system clock of 128 MHz by default. To develop application's code, the C programming language and Zephyr Real-Time Operating System (RTOS) Software Development Kit (SDK) on Visual Studio code (VS code) were used. Additionally, the Nordic nRF SDK extension for VS Code was used to build the firmware.

The Nordic Thingy:53 is equipped with a built-in digital Pulse Density Modulation (PDM) microphone VM3011 from Vesper, whose drivers were developed using Zephyr Digital PDM microphone API. It is important to note that not all functionalities and customization settings of the sensor drivers are necessary to execute the proposed solution. Therefore, only the MONO operation mode was implemented with maximum gain and at an acquisition rate of 16kHz. Since it is a digital microphone, it collects 16 bit precision integer type data, whose amplitudes range from -32768 to +32767.

This built-in microphone serves as the data acquisition source for this work and is fully capable of recording more than 5 seconds of audio in SRAM with no failure. However, since the WPT is a very complex algorithm and is implemented with 32 bit precision float type data, it uses a lot of memory resources. Then, to avoid memory crashes, a recording time of 500 milliseconds was selected, with an extra 260 milliseconds added. It is important to note that this recording duration (0.76s) differs from the one used during the training procedure (0.5s); this adjustment was made to increase the likelihood of capturing relevant sound events. Since the model is already trained, this change only affects event capture and not classification, as the features extracted from the WPT are not influenced by the duration of the data due to their nature. It was considered to use Flash memory to overcome this memory bottleneck, however the execution time is not as efficient as using SRAM.

All the pre-processing steps, including normalization, WPT, and energy bin calculation, execute approximately in 940 milliseconds, considering a time record of 760 milliseconds. It is a very fast performance, considering that all these steps use float data type, which is computationally expensive for limited hardware. Thus, for this work, a short recording time is perfect to avoid memory crashes and, furthermore, for real-time or near real-time execution.

Tiny model deployment can be a very difficult task depending on which method was used, it highly depends on the trade off between it and the pre-processing step, since time execution is a very important factor in such applications. For this work, the deployment was relatively simple since the XGBoost method is a rule-based algorithm. After training the model in Python, the EloquentML library was used to convert and generate the model in C. EloquentML is a library that converts machine learning models from Python to C. It is capable of converting various types of algorithms, including both classic machine learning models and deep learning models. In this work, an XGBoost model was trained using the Scikit-Learn API and then converted into a C file using EloquentML. As the model is a decision tree-based model, the generated C file consists of a series of nested conditionals, where votes are made to select the class with the highest probability. A detailed description of the entire conversion process, along with an example, can be found in Appendix A. The generated model is a header file composed of conditional structures, enabling an execution time of approximately 1 millisecond. In total, the complete application executes in approximately 1760 milliseconds based on Thingy:53's default system clock, this information was obtained using the Zephyr's API. Furthermore, the whole application utilizes 33% of Flash memory and 40% of RAM.

After the inference, the top three highest-rated classes are sent to the device's USB serial port, which can be sent via any wireless communication. If the device is located in an area with Ethernet or Wi-Fi access, or if there is 4G/5G traffic coverage, the data can be directly sent to the nearest server. In cases where these options are not available, its possible to send the data via satellite. Figure 4 provides an overview of the hardware-integrated solution.

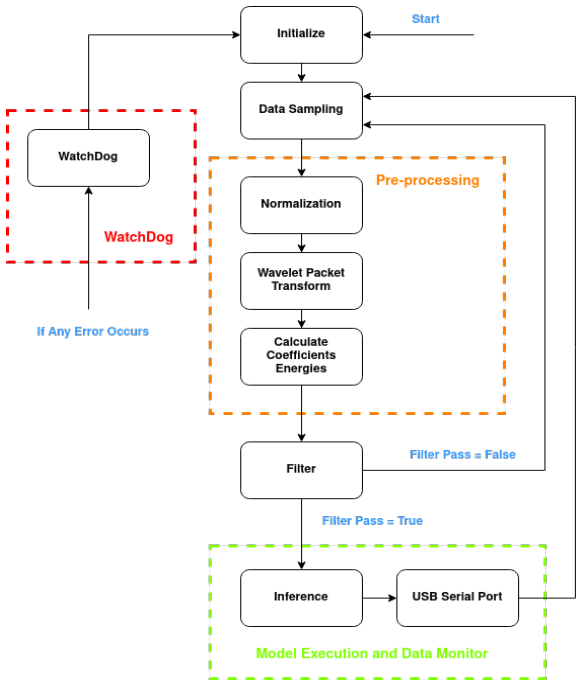

Figure 4. Flowchart of the hardware-integrated solution.

Besides all the implementations previously described, the WatchDog function is used to automatically reboot the device in case an unwanted action occurs. This procedure occurs so quickly that it does not interfere with monitoring perfor-

mance. It is also important to clarify that no sensitive data (e.g., human voice) is saved or sent. All the data is processed in SRAM and then discarded.

3 Data

Our database is composed of 23 classes, including 20 distinct bird species, general insects sounds, and human-related activities, such as dog barking and engine noises (including chainsaw and car engine sounds). The selection of these events was based on feedback and insights from both locals and researchers, ensuring their relevance to the study.

Initially, data was recorded using the XIXI SPY microphone. However, after more than 900 hours of recording time, we encountered insufficient data for each class, particularly for events that has rare occurrence, such as specific bird species' vocalizations. To address this issue, we incremented the dataset by including additional data from the ESC50, xeno-canto and FreeSound databases. It is important to emphasize that for xeno-canto, only the audios that were recorded in the central-western and northern regions of Brazil were selected. Similarly, for ESC50 and FreeSound, only the insects that are native to the same region were chosen. Furthermore, it is essential to emphasize that this work is solely dedicated to research, and as such, the data collected from these databases were strictly intended for research purposes only. The total number of samples per class is shown in Table 2.

Table 2. Total number of samples per class.

Class	Number of samples
Amazonian Motmot	21
Black-capped Antwren	20
Black-capped Screech-Owl	20
Blue-and-yellow Macaw	31
Boat-billed Flycatcher	23
Buff-necked Ibis	33
Insects	218
Chivi Vireo	38
Chopi Blackbird	24
Dog Barking	249
Engine	304
Flavescent Warbler	24
Great Kiskadee	27
Helmeted Manakin	25
Orange-headed Tanager	28
Pale-breasted Thrush	34
Pavonine Cuckoo	27
Red-shouldered Macaw	22
Small-billed Tinamou	34
Tropical Screech-Owl	41
Undulated Tinamou	37
White-chinned Sapphire	29
White-eyed Parakeet	20

Following the manual syllable splitting procedure described in Section II.A, the resulting total number of samples for each bird species, as well as for the other sound events, is shown in Table 3.

Table 3. Total number of samples per class after the syllable splitting procedure.

Class	Number of samples
Amazonian Motmot	85
Black-capped Antwren	73
Black-capped Screech-Owl	55
Blue-and-yellow Macaw	93
Boat-billed Flycatcher	84
Buff-necked Ibis	95
Insects	218
Chivi Vireo	103
Chopi Blackbird	87
Dog Barking	249
Engine	304
Flavescent Warbler	98
Great Kiskadee	98
Helmeted Manakin	98
Orange-headed Tanager	94
Pale-breasted Thrush	112
Pavonine Cuckoo	106
Red-shouldered Macaw	89
Small-billed Tinamou	105
Tropical Screech-Owl	113
Undulated Tinamou	99
White-chinned Sapphire	95
White-eyed Parakeet	81

During the data collection procedure, it was identified that some bird species and insects syllables were larger compared to the mean duration of their respective classes, increasing the class overall standard deviation, as presented in Table 4. This can be attributed to the variations in vocalization size and complexity among different species [Kershenbaum *et al*., 2016]. However, the mean duration of the classes is still relatively short, indicating that the majority of the syllables has a shorter time duration. On the other hand, the engine class has the highest mean value, which was already expected because the majority of the files were from the ESC50 dataset, where each one has a duration of 5 seconds.

After the data collection stage, the gathered data is processed using the methodology described in Section II.A, which involves splitting each audio into 0.5s chunks. The final number of samples per class is shown in Table 5, where it can be seen that our dataset is highly unbalanced.

Table 4. Statistical analysis of the data time duration.

Class	Mean (s)	Standard deviation (s)
Birds	1.453	2.005
Insects	2.831	5.955
Dog barking	0.593	0.385
Engine	4.133	1.401

It is important to emphasize that the data is split into training, validation, and test sets before the syllable and time window splitting procedures (see Table 2). This ensures that data

Table 5. Total number of syllables for each class after the pre-processing step.

Class	Number of syllables
Amazonian Motmot	443
Black-capped Antwren	698
Black-capped Screech-Owl	2469
Blue-and-yellow Macaw	527
Boat-billed Flycatcher	294
Buff-necked Ibis	564
Insects	3235
Chivi Vireo	293
Chopi Blackbird	411
Dog barking	600
Engine	6681
Flavescent Warbler	405
Great Kiskadee	336
Helmeted Manakin	712
Orange-headed Tanager	528
Pale-breasted Thrush	568
Pavonine Cuckoo	1372
Red-shouldered Macaw	743
Small-billed Tinamou	258
Tropical Screech-Owl	378
Undulated Tinamou	661
White-chinned Sapphire	1097
White-eyed Parakeet	462

leakage is avoided, preventing any artificial inflation of the models' performance.

4 Results

In this section, we present the results of a deep learning approach combining Mel-frequency cepstral coefficients (MFCC) with a CNN to compare computational cost and classification performance with that of the chosen shallow learner. We also assess the performance of the proposed models with and without the use of the Synthetic Minority Over-sampling Technique (SMOTE) for data augmentation. The SMOTE is a popular method for addressing class imbalance in datasets by generating synthetic samples for the minority classes by first selecting a minority class instance at random and finding its k nearest minority class neighbors. A synthetic instance is then created by randomly choosing one of the k nearest neighbors β and forming a line segment between α and β in the feature space. The synthetic instance is generated as a convex combination of α and β. This helps improve the performance of machine learning models, especially in scenarios where the number of samples in the minority classes is limited [Chawla *et al*., 2002]. It should be noted that SMOTE is applied solely to the training dataset during the experiments conducted in this study to prevent data leakage. Additionally, a dedicated subsection compares both approaches in terms of classification performance, processing cost, and memory usage. Finally, we discuss the results and execution performance of the selected approach when embedded in the Nordic Thingy:53.

4.1 Deep Learning Approach

As mentioned previously, the selected deep learning approach involves extracting MFCC features from each split audio sample and using them as input to a CNN. For the sake of comparison with the shallow learner, this approach was chosen due to its proven effectiveness, its success in bird classification and sound event detection, and its continuous use in numerous studies in the literature, as can be seen in [Lasseck, 2018; Ntalampiras and Potamitis, 2021; Xie *et al*., 2019; Zhang *et al*., 2021; Carvalho and Gomes, 2023; Kamarajugadda *et al*., 2024; Saad, 2020]. The use of transformers has shown promising results in bird sound detection as well, although their high computational cost [Rauch *et al*., 2023; Zhang *et al*., 2023] makes them prohibitive for deployment on the Nordic Thingy:53, and thus they are not considered in the current work. Additionally, more complex CNN architectures are excluded from this study due to their increased computational cost. Consequently, a simpler CNN architecture is used, which is presented in Figure 5. This architecture was achieved through HPT combined with a Monte Carlo cross-validation approach, with a limited set of simple architectures and hyperparameters. The implemented Monte Carlo algorithm consists of conducting 100 experiments for each model evaluated by the HPT. In each experiment, the dataset is randomly split into training and test data in a stratified 80/20 ratio, respectively. The Adaptive Moment Estimation (ADAM) optimizer was employed for training. The extracted MFCC features, used as inputs of the CNN, have a shape of (128, 16). This corresponds to 128 mel bins, a hop length of 512, and an FFT size of 2048 applied to each audio sample following the window split procedure to standardize the sample size. Specifically, each sample, prior to MFCC extraction, has a fixed size of 8000 time points. Table 6 presents the classification results for the considered classes using the deep learning approach. All the steps of this procedure were developed in Python 3.11.3.

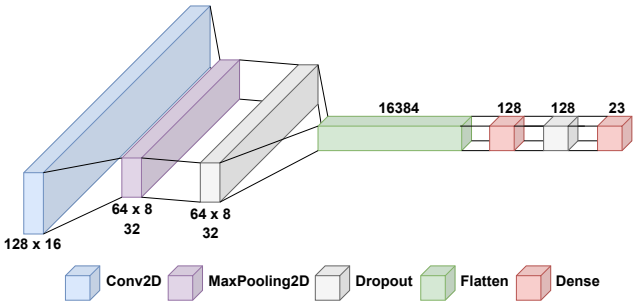

Figure 5. The CNN architecture. ReLU activation functions are used for both the convolutional and intermediate dense layers, while the final dense layer uses a softmax activation function. The dropout rate is set to 0.25.

From Table 6, it can be observed that the model exhibits an overall performance exceeding 78% in F1 score. However, there are notable variations in classification performance across different classes. This disparity is particularly pronounced when comparing the top and bottom performers. For instance, the Black-capped Screech Owl demonstrates an F1 score of 91%, while the Great Kiskadee shows an F1 score below 51%. To improve the model's performance, several data augmentation techniques were tested, including tra-

Table 6. Performance metrics for the trained CNN model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal, while the bottom three are shown in red. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Black-capped Screech Owl	**98.1**	**84.9**	**91.0**
Engine	**84.1**	**97.9**	**90.5**
Tropical Screech-Owl	**88.4**	**91.0**	**89.7**
White-chinned Sapphire	92.7	84.4	88.4
Chopi Blackbird	78.2	97.4	86.8
Pavonine Cuckoo	91.8	81.7	86.4
White-eyed Parakeet	83.3	76.1	79.5
Amazonian Motmot	78.6	76.7	77.6
Dog barking	75.7	67.5	71.3
Flavescent Warbler	74.4	68.1	71.1
Insects	60.9	82.0	69.9
Orange-headed Tanager	83.8	59.6	69.7
Helmeted Manakin	82.0	60.3	69.5
Undulated Tinamou	97.2	53.8	69.3
Buff-necked Ibis	87.5	53.8	66.7
Black-capped Antwren	67.2	65.0	66.1
Chivi Vireo	69.7	59.0	63.9
Boat-billed Flycatcher	77.3	53.1	63.0
Red-shouldered Macaw	78.9	46.2	58.3
Blue-and-yellow Macaw	61.5	49.0	54.5
Small-billed Tinamou	**58.6**	**47.2**	**52.3**
Pale-breasted Thrush	**78.6**	**38.6**	**51.8**
Great Kiskadee	**64.0**	**42.1**	**50.8**
Weighted Average	80.2	79.3	78.5

ditional SMOTE, adding random noise, pitch-shifting, and applying frequency and time masking to the spectrograms [Ferreira-Paiva *et al*., 2022]. However, none of these methods resulted in significant improvements. Although traditional SMOTE helped reduce class imbalance, it did not lead to a notable performance boost. One potential explanation is that the high-dimensional input feature space may limit SMOTE's effectiveness, as discussed in [Blagus and Lusa, 2013]. Furthermore, SMOTE is primarily designed for numerical vectors and is not inherently suited for time-series or sequential data, where the relationship between samples is crucial. In time-series data (such as audio or other signal-based data), preserving the temporal structure is essential for accurate model interpretation [Zhao *et al*., 2022]. Due to its design, SMOTE can mix matrices (representing sequences of data) without considering their temporal dependencies. This process may introduce distortions or misalignments in the time-series data, generating noise rather than meaningful synthetic examples, which could negatively impact model performance. Additionally, the pitch-shifting technique caused a significant decrease in model performance, increasing confusion between bird species and reducing the overall F1 score by more than 10%.

To address this issue, Temporal-oriented SMOTE (T-SMOTE) was employed to the raw training data after the syllable splitting with the goal of generating synthetic data points by interpolating between instances of the minority class while preserving their temporal relationships. Given a set of minority class samples $P = \{X_1, X_2, \ldots, X_n\}$, where each sample X_i is a one-dimensional vector in the problem under analysis, $X_i \in \mathbb{R}^T$, with T representing the sequence length after the syllable splitting (i.e., the number of timestamps), with x_j^i being the raw audio segment value at time step j for sample X_i. In summary, T-SMOTE works as follows [Zhao *et al*., 2022]:

1. T-SMOTE generates a set of candidate samples with different leading time l, X_i^l, near the class boundary, where a subsequence of length w (with $w < T$) is selected from the sequence X_i. The subsequence starts at timestamp $T-l-w+1$ and ends at timestamp $T-l$. Mathematically, the subsequence is:
$$X_i^l = \{x_i^{T-l-w+1}, \ldots, x_i^{T-l}\}.$$
2. T-SMOTE then generates synthetic samples by interpolating the subsequences. For a given subsequence X_i^l, a new synthetic sample X_{new} is created by linearly interpolating between X_i^l and its temporal neighbor X_i^{l+1}. The interpolation is done along the temporal dimension, preserving the structure of the time-series data:
$$X_{\text{new}} = \alpha X_i^l + (1-\alpha) X_i^{l+1},$$
where α is a random value on the interval [0, 1] generated by Beta distribution.
3. Finally, T-SMOTE uses a weighted sampling technique to select the generated synthetic samples. This helps reduce noise and prevent overfitting by emphasizing more meaningful synthetic samples near the decision boundary, based on their relevance to the minority class.

In other words, for each sample in the minority class, synthetic samples are created by linearly interpolating the generated segments along the time axis. After this, the time window splitting procedure is applied, followed by MFCC feature extraction, thereby maintaining the structure and integrity of the feature space in the time and spectral (frequency) dimensions while increasing the volume of the training data. With this approach, all classes were balanced to 3000 samples per class in the training data. The results following this procedure are shown in Table 7.

From Table 7, it can be observed that the model's classification performance improved overall by approximately 4% in the weighted F1 score, resulting in an F1 score above 82%. However, T-SMOTE had a negative impact on some classes (e.g., Small-Billed Tinamou), which could be attributed to overfitting in certain minority classes or boundary confusion during the T-SMOTE procedure. Furthermore, using

Table 7. Performance metrics for the trained CNN model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal, while the bottom three are shown in red. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Black-capped Screech-Owl	**97.6**	**91.6**	**94.5**
Tropical Screech-Owl	**95.3**	**91.0**	**93.1**
Engine	**87.1**	**97.5**	**92.0**
Chopi Blackbird	81.8	98.5	89.4
White-chinned Sapphire	93.6	81.1	86.9
Pavonine Cuckoo	88.5	84.4	86.4
White-eyed Parakeet	92.5	80.4	86.0
Insects	85.1	85.7	85.4
Flavescent Warbler	88.6	66.0	75.6
Helmeted Manakin	86.5	66.2	75.0
Amazonian Motmot	74.4	74.4	74.4
Dog barking	78.1	68.7	73.1
Orange-headed Tanager	69.0	76.9	72.7
Black-capped Antwren	64.9	80.0	71.6
Chivi Vireo	82.8	61.5	70.6
Buff-necked Ibis	67.3	71.2	69.2
Undulated Tinamou	61.7	76.9	68.5
Boat-billed Flycatcher	64.7	68.8	66.7
Red-shouldered Macaw	83.7	55.4	66.7
Great Kiskadee	77.8	55.3	64.6
Blue-and-yellow Macaw	**73.5**	**51.0**	**60.2**
Pale-breasted Thrush	**64.3**	**47.4**	**54.5**
Small-billed Tinamou	**50.0**	**30.6**	**37.9**
Weighted Average	83.0	83.0	82.4

the MFCC approach to reduce the number of Mel bins and working with smaller architectures has been shown to reduce the classification performance of the DL model.

4.2 Shallow Learner Approach

As mentioned in Section 2.2, the ML algorithm chosen to develop the shallow learner is XGBoost, and its optimal set of hyperparameters was determined by performing the HPT in conjunction with a Monte Carlo cross-validation approach. Analogously to the experiments conducted using the deep learning approach, the implemented Monte Carlo algorithm consists of conducting 100 experiments for each model evaluated by the HPT. In each experiment, the dataset is randomly split into training and test data in a stratified 80/20 ratio, respectively.

This approach was implemented to assess the robustness of each model and evaluate its performance under different training and test data scenarios. All the steps of this procedure were developed in Python 3.11.3, and the resulting optimal set of hyperparameters is presented in Table 8. Furthermore, the performance metrics of a single trained XGBoost model, evaluated on the same test data used for the deep learning approach, are summarized in Table 9.

Table 8. Hyperparameters of the XGBoost model.

Parameter	Value
Number of estimators	5
Max depth	6
Learning rate	0.712

Table 9. Performance metrics for the trained XGBoost model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal, while the bottom three are shown in red. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**92.3**	**95.9**	**94.1**
Black-capped Screech-Owl	**87.9**	**92.5**	**90.1**
White-chinned Sapphire	**82.2**	**90.9**	**86.3**
Engine	80.0	92.0	85.6
Tropical Screech-Owl	78.8	88.2	83.2
Insects	79.4	86.2	82.7
White-eyed Parakeet	72.4	68.5	70.4
Pavonine Cuckoo	68.7	71.2	69.9
Undulated Tinamou	83.2	59.8	69.6
Red-shouldered Macaw	68.3	65.1	66.7
Dog barking	72.2	58.3	64.5
Flavescent Warbler	65.3	60.5	62.8
Small-billed Tinamou	74.3	50.0	59.8
Black-capped Antwren	65.5	52.9	58.5
Buff-necked Ibis	62.6	50.4	55.9
Boat-billed Flycatcher	58.5	52.5	55.4
Blue-and-yellow Macaw	69.7	43.8	53.8
Helmeted Manakin	62.9	46.5	53.4
Amazonian Motmot	61.4	39.8	48.3
Orange-headed Tanager	55.0	41.5	47.3
Pale-breasted Thrush	**53.7**	**37.7**	**44.3**
Great Kiskadee	**45.8**	**40.3**	**42.9**
Chivi Vireo	**37.8**	**28.8**	**32.7**
Weighted Average	76.1	77.2	76.2

From Table 9, one can notice that the model exhibits an overall performance above 76% in terms of the F1 score, slightly below the deep learning approach. Nonetheless, it

is clear that, similarly to the deep learning approach, the model's performance varies within different sound events, as evidenced by the difference in results for the Chopi Blackbird and the Chivi Vireo. To improve the model's performance, some data augmentation techniques, such as SMOTE and T-SMOTE, were applied to evaluate the model's improvements over the classification scores. The traditional SMOTE technique showed a increase of performance of approximately 3%. Since the features used in this approach do not explicitly exhibit temporal dependencies, they can be treated as numerical vectors, making them suitable for the SMOTE technique. However, the procedure using the T-SMOTE overall yielded better results, by increasing the weighted F1 score by more than 6%. With this approach, all classes were balanced to 3000 samples per class in the training data. The results following this procedure are shown in Table 10.

Table 10. Performance metrics for the trained XGBoost model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal, while the bottom three are shown in red. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**99.2**	**99.5**	**99.4**
Black-capped Screech-Owl	**93.7**	**93.1**	**93.4**
Tropical Screech-Owl	**90.0**	**94.7**	**92.3**
White-chinned Sapphire	92.1	90.9	91.5
Insects	90.9	86.9	88.9
Engine	93.9	79.1	85.9
Undulated Tinamou	80.4	84.1	82.2
Pavonine Cuckoo	81.9	74.1	77.8
Red-shouldered Macaw	77.3	77.9	77.6
Black-capped Antwren	75.2	77.9	76.5
Flavescent Warbler	68.0	86.4	76.1
White-eyed Parakeet	69.5	79.3	74.1
Dog barking	65.1	79.2	71.4
Boat-billed Flycatcher	63.0	78.0	69.7
Amazonian Motmot	61.3	73.9	67.0
Buff-necked Ibis	61.5	73.5	66.9
Pale-breasted Thrush	61.7	71.9	66.4
Orange-headed Tanager	58.1	70.8	63.8
Great Kiskadee	58.0	70.1	63.5
Helmeted Manakin	68.3	59.2	63.4
Blue-and-yellow Macaw	**50.0**	**79.0**	**61.3**
Small-billed Tinamou	**55.4**	**78.8**	**65.1**
Chivi Vireo	**48.7**	**64.4**	**55.5**
Weighted Average	84.1	82.3	82.8

From Table 10, it can be observed that the model's classification performance improved overall, resulting in an F1 score above 82%. In this case, the T-SMOTE data augmentation had a greater effect than in the deep learning approach. In that case, the T-SMOTE contributed to reducing the performance discrepancy between classes.

4.3 Approaches Comparison

In this section, for the sake of comparison between the DL and shallow learner approaches, both methods are analyzed in terms of their classification performance (see Tables 7-10). Additionally, since the goal of this work is to embed the solution in the Nordic Thingy:53 hardware, the processing costs and memory usage of both approaches are also compared. Given that the DL approach typically has relatively high memory consumption and model size, quantization techniques were applied to assess the feasibility of embedding this approach into the target hardware.

For this purpose, post-training quantization was performed, which involves representing the weights and activations of the CNN, as well as the operations involved in the MFCC preprocessing, with lower-precision data types. In this study, model conversion to TensorFlow Lite (TFLite) and MFCC function conversion were tested using different quantization formats, including 16-bit floating point (float16), 8-bit integer (int8), and 8-bit unsigned integer (uint8). Quantization reduces the solution size and memory footprint, making it more suitable for embedded systems, though it may lead to a slight decrease in model accuracy. Previous works, such as those by [Jacob *et al.*, 2018], have demonstrated the effectiveness of these techniques for reducing the computational cost while maintaining an acceptable level of accuracy in embedded systems. The results for the comparison of the shallow learner approach and the DL approach, considering all the processing steps and different types of quantization, can be seen in Table 11.

Table 11. Comparison of the chosen shallow learner (XGBoost) and DL approaches in terms of computational and classification performance, including peak processing memory, file size, and weighted F1 score. For the DL approach, results are presented for the non-converted model, the TensorFlow-to-TFLite converted model, and three types of quantization.

Approach	Peak processing (KB)	Size (KB)	F1 (%)
Shallow Learner	752	358	82.8
DL	2801	25166	82.4
DL (TFLite)	2411	8176	82.4
quantized DL (float16)	1578	4099	82.4
quantized DL (int8)	1187	2065	81.8
quantized DL (uint8)	1054	2045	56.4

As can be observed in Table 11, the shallow learner approach still maintains the best balance between classification performance, memory size, and computational cost, achiev-

ing the highest weighted F1 score, the smallest memory footprint, and the lowest processing cost. Even when compared to the deep learning approach with integer quantization, its computational cost remains lower.

4.4 Embedded Solution

During the process of transferring the pre-processing code from Python to the C programming language, a slight loss was observed in the WPT step due to a necessary decrease in floating point precision. However, this loss was insignificant and had no impact on the XGBoost performance. Therefore, no loss was observed during the conversion of the trained model from Python to C, as it achieved the exact same results presented in Table 9 when evaluated with the same data.

The sound event filter was developed following the methodology described in Section 2.3. The main objective was to optimize the retention of relevant sound event samples, including birds, insects, dog barking, and engine noises, while simultaneously minimizing the number of undesirable samples, such as wind and ambient noise, that pass through the filter. In essence, the challenge lies in striking a well-balanced trade-off between capturing relevant events and rejecting unwanted ones. To validate the proposed wind filter algorithm, wind audio samples, along with other sound events such as chainsaw, engine, dog barking, insects, human laughing, and siren, from the ESC-50 dataset [Piczak, 2015] were used to verify whether there is a low rate of wind-contaminated samples passing through the filter in comparison to other sound events. The results are shown in Table 12. Furthermore, during the development process, an additional set of 1000 audio samples contaminated by wind noise and over 3000 samples of only environmental ambient noises (collected from the area of interest) were included, along with the samples presented in Table 5, to thoroughly test the filter's performance within the solution development dataset. The results are presented in Table 13.

Table 12. The sound event filter validation results using the ESC-50 dataset wind samples.

Class	% of samples passing through the filter
Wind	25.00
Other Events	69.60

Table 13. The sound event filter results using the dataset employed to develop the embedded solution.

Class	% of samples passing through the filter
Birds	75.55
Insects	76.06
Dog barking	75.83
Engine	80.02
Wind	18.92
Ambient Noise	0.27

From Tables 12 and 13, it can be observed that the filter performs as intended, showing slightly better results for the test dataset compared to the validation dataset. At a minimum, the filter retains approximately 70% of the relevant sound event samples while removing around 75% of wind-contaminated samples and almost entirely eliminating samples with only ambient noises.

It is important to mention that the collected data is sent to a hardware gateway that connects sensors to the cloud through Long Range Wide-Area Network (LoRaWAN), a standard Low-Power Wide-Area Network (LPWAN) communication protocol for low-power wide-area networks that works on Long Range (LoRa), which relays the result of inferences to our platform. In order to evaluate the hardware's battery runtime, we conducted an experiment in which the solution ran continuously in an infinite loop with different power levels for the LoRa communication until the battery was fully depleted. The entire experiment was conducted indoors with no external peripherals, and only an LED was used to indicate that the hardware was still operating. Table 14 presents the results of the battery runtime experiment.

Table 14. Battery runtime of the Nordic Thingy:53 embedded solution at different LoRaWAN transmission power levels.

Transmission Power	Hardware Runtime (h)
Without LoRaWAN	75
Low Power	41
Medium Power	28
High Power	11

From Table 14, it can be seen that the hardware's battery runtime demonstrated a duration ranging from 11 to 75 hours, depending on the power level of the wireless communication. These results reaffirm the viability of the proposed solution for field use, particularly when utilizing low-power LoRaWAN for transmitting the solution's inference results. It should be highlighted here that the onboard nPM1100 provides the Thingy:53 with advanced battery management that not only has Li-Ion charging capabilities but can also help to extend battery life. The battery provided with the Thingy:53 has a capacity of 1350mAh and can be charged over a USB C cable.

5 Discussion

The results of sound event classification provide a comparison between a shallow learning approach, which uses wavelet coefficient energy as input features, and a more conventional CNN approach, which uses MFCCs as input features. It is important to note that the nature of these two types of input features could potentially lead to biased conclusions regarding the performance of each model. However, the main objective of this comparison is not to evaluate the models' performance directly but rather to assess the entire pipeline — ranging from feature extraction to the classification model — and determine whether it can be embedded into the proposed hardware, considering computational cost, while still maintaining acceptable classification performance. The results showed that the deep learning approach achieved a weighted F1 score above 82% with the use of

the T-SMOTE technique on the training dataset. Additionally, as highlighted in Section 4.3, the CNN model's size and memory consumption make it unfeasible for embedding into the Nordic Thingy:53, even with the use of post-quantization techniques. Despite only a negligible decrease in classification performance (except for the uint8 approach), the memory reduction is still insufficient to fit within the hardware constraints.

It is important to note that more complex CNN architectures would likely yield better overall performance, as demonstrated in previous studies (e.g., [Lasseck, 2018; Ntalampiras and Potamitis, 2021; Xie *et al*., 2019; Zhang *et al*., 2021; Carvalho and Gomes, 2023; Kamarajugadda *et al*., 2024; Saad, 2020]). However, since even the simple architecture tested in this study was too large in terms of memory to be embedded in the Nordic Thingy:53, further experimentation with more complex architectures would not be feasible and would go beyond the scope of this study.

In contrast, the shallow learner approach, while slightly underperforming compared to the deep learning approach without data augmentation, showed a significant improvement with the use of T-SMOTE. Its performance increased to over 82%, surpassing the CNN + MFCC approach by 0.4% in the overall weighted F1 score. Consequently, the XGBoost model, based on the shallow learner approach, was selected to compose the solution embedded in the Nordic Thingy:53, as discussed in Section 4.4. Figure 6 presents the confusion matrix for the chosen (best) model, which is the one embedded in the hardware.

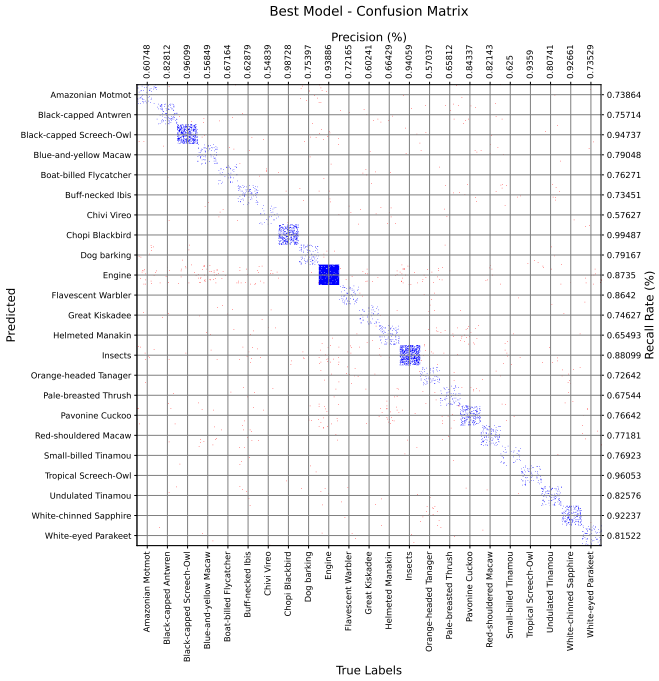

Figure 6. The confusion matrix for the model embedded in the Nordic Thingy:53 hardware. This model is the XGBoost, utilizing the shallow learner approach, where the input space consists of WPT coefficient energy. The blue dots refer to sound event test samples that were accurately predicted, while the red dots represent false positives or false negatives, indicating incorrect predictions.

From Figure 6 and Table 10, one can notice that the XGBoost model performance ranges from 99.4% to 55.5% in terms of F1 score, performing particularly well for human activity-related events and some bird species, such as the Chopi Blackbird, Black-capped Screech-Owl, and Tropical Screech-Owl. However, certain discrepancies in the results for other bird species (see Table 10) could be attributed to various factors:

- Firstly, the presence of inharmonic calls can pose a challenge in precisely characterizing the vocalizations of some species [Yang *et al*., 2021].
- Moreover, it is important to note that the pre-processing of the disposable data resulted in a very unbalanced training dataset, which could lead to biases in the model's predictions, affecting its performance. This issue is partially addressed with the use of T-SMOTE, which improves the model's overall performance.
- Additionally, some bird species may have a significantly larger vocabulary of syllables compared to others [Garamszegi *et al*., 2012] (e.g., Buff-necked Ibis), requiring greater variance and diversity in the training data to classify them adequately Even T-SMOTE could not address this issue, and the only solution would be to acquire more training data for species with a broader range of vocabulary.
- Lastly, some audio samples were contaminated by the calls of other birds or background sound events (e.g., some engine audios had bird calls in the background, and a few bird audios had calls from different species occurring at the same time), which could potentially prejudice the model's decision-making process. This is evidenced by the confusion matrix shown in Figure 6, which indicates that the class most often confused by the model is bird species with engine sounds.

In summary, the proposed shallow learner demonstrates potential for classifying sound events, especially considering its low computational cost, which is a significant advantage. However, addressing the aforementioned challenges to further enhance its performance and robustness could be a concern, particularly when considering integration into an end-device. For instance, tackling the audio background contamination would require a complex filtering algorithm to isolate the relevant sound event from the rest of the audio. Adding such computational complexity to the algorithm might render the solution unfeasible to embed in resource-constrained devices. This emphasizes the importance of finding a balance between accuracy and computational efficiency.

When integrated into the Nordic Thingy:53, the embedded solution performed as expected. The shallow learner maintained its performance seamlessly when running on the hardware, and the entire solution operated smoothly without encountering any errors. Moreover, the embedded sound event filter proved to be effective, reducing the number of samples contaminated by wind and almost completely eliminating occurrences of samples with only ambient noises (see Table 13). These results are consistent with the study by Juodakis and Marsland [Juodakis and Marsland, 2022], which demonstrated the potential of using WPT to detect wind noise, as well as with the studies [Nelke *et al*., 2016; Honkakunnas, 2021; Guo and Zheng, 2022], which reinforce the idea that wind sound retains energy in the lower frequency range. Regarding battery runtime, this refers to the duration of oper-

ation with low-power wireless communication, in line with the use of LoRaWAN. Specifically, the embedded solution can operate continuously for up to 41 hours.

6 Conclusions

In the present work, we proposed an embedded ML solution aimed at supporting environmental acoustic monitoring. Our approach focuses on tracking both bird biodiversity and human-related sound events using Nordic's Thingy:53 hardware and its built-in Vesper's digital PDM VM3011 microphone. The solution is mainly based on a low-dimensional feature extraction step, which consists of performing the WPT on the audio signal and calculating the energy from its coefficients. These extracted features serve as input for a low-cost Gradient Boosting model responsible for performing the event classification. This approach has demonstrated potential for the assigned task, delivering a near-real-time response and a classification performance above 82% of weighted F1 score even when integrated into Nordic Thingy:53, while our developed sound event filter has proven to be robust against wind and ambient noise. Furthermore, we strongly believe that this approach exhibits remarkable versatility, making it easily adaptable to track and classify not only bird species from distinct regions but also potentially extendable to classify species from other fauna families, such as anurans. Additionally, the embedded solution can operate continuously for approximately 41 hours before requiring a recharge, making it a reliable tool for assisting ecoacoustic researchers during extended field surveys. Moreover, due to challenges of relocating the research team to other biomes and the scope of the project being restricted to the central and western regions of Brazil, future studies should be conducted across diverse ecosystems and a broader range of sound categories to test the versatility of the proposed solution.

Declarations

Acknowledgements

We would like to express our sincere gratitude to Embrapii for their valuable support.

Funding

This research was supported by funding from Embrapii (Brazilian Company of Research and Industrial Innovation).

Authors' Contributions

B.F. Junqueira, R.G. Vieira, E.C. Alves, and M.V. dos Santos contributed to the conception of this study. B.F. Junqueira, R.G. Vieira, E.C. Alves, and B.B. Karaziack performed the experiments. B.F. Junqueira, R.G. Vieira, and E.C. Alves are the main contributors and writers of this manuscript. All authors read and approved the final version of the manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The data that support the findings of this study are available from the corresponding author upon reasonable request and with permission from CPQD and Greenbug.

References

Bardeli, R., Wolff, D., Kurth, F., Koch, M., Tauchert, K.-H., and Frommolt, K.-H. (2010). Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring. *Pattern Recognition Letters*, 31(12):1524–1534. DOI: 10.1016/j.patrec.2009.09.014.

Bergstra J., Bardenet R., B. Y. and B., K. (2011). Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc. Available at: `https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf`.

Bianchi D., Mayrhofer E., G. M. B. G. V. A. (2015). Wavelet packet transform for detection of single events in acoustic emission signals. *Mechanical Systems and Signal Processing*, 64:441–451. DOI: 10.1016/j.ymssp.2015.04.014.

Blagus, R. and Lusa, L. (2013). Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, 14:1–16. DOI: 10.1186/1471-2105-14-106.

Bradfer-Lawrence, T., Gardner, N., Bunnefeld, L., Bunnefeld, N., Willis, S. G., and Dent, D. H. (2019). Guidelines for the use of acoustic indices in environmental research. *Methods in Ecology and Evolution*, 10(10):1796–1807. DOI: 10.1111/cobi.12968.

Branco, S., Ferreira, A. G., and Cabral, J. (2019). Machine learning in resource-scarce embedded systems, fpgas, and end-devices: A survey. *Electronics*, 8(11):1289. DOI: 10.3390/electronics8111289.

Burivalova, Z., Game, E. T., and Butler, R. A. (2019). The sound of a tropical forest. *Science*, 363(6422):28–29. DOI: 10.1126/science.aav1902.

Burivalova, Z., Towsey, M., Boucher, T., Truskinger, A., Apelis, C., Roe, P., and Game, E. T. (2018). Using soundscapes to detect variable degrees of human influence on tropical forests in papua new guinea. *Conservation Biology*, 32(1):205–215. DOI: 10.1111/cobi.12968.

Cai, J., Ee, D., Pham, B., Roe, P., and Zhang, J. (2007). Sensor network for the monitoring of ecosystem: Bird species recognition. In *2007 3rd international conference on intelligent sensors, sensor networks and information*, pages 293–298. IEEE. DOI: 10.1109/ISSNIP.2007.4496859.

Carvalho, S. and Gomes, E. F. (2023). Automatic classification of bird sounds: using mfcc and mel spectrogram features with deep learning. *Vietnam Journal of Computer Science*, 10(01):39–54. DOI: 10.1142/s2196888822500300.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357. DOI: 10.1613/jair.953.

Coifman, R. R., Meyer, Y., Quake, S., and Wickerhauser, M. V. (1994). Signal processing and compression with wavelet packets. *Wavelets and their applications*, pages 363–379. DOI: 10.1007/978-94-011-1028-0_18.

Deichmann, J. L., Acevedo-Charry, O., Barclay, L., Burivalova, Z., Campos-Cerqueira, M., d'Horta, F., Game, E. T., Gottesman, B. L., Hart, P. J., Kalan, A. K., *et al*. (2018). It's time to listen: there is much to be learned from the sounds of tropical ecosystems. *Biotropica*, 50(5):713–718. DOI: 10.1111/btp.12593.

Ferreira-Paiva, L., Alfaro-Espinoza, E., Almeida, V. M., Felix, L. B., and Neves, R. V. (2022). A survey of data augmentation for audio classification. In *Congresso Brasileiro de Automática-CBA*, volume 3. DOI: 10.20906/cba2022/3469.

Frusque, G. and Fink, O. (2022). Learnable wavelet packet transform for data-adapted spectrograms. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3119–3123. IEEE. DOI: 10.1109/ICASSP43922.2022.9747491.

Gao, R. X. and Yan, R. (2010). *Wavelets: Theory and applications for manufacturing*. Springer Science & Business Media. DOI: 10.1007/978-1-4419-1545-0.

Garamszegi, L. Z., Zsebok, S., and Török, J. (2012). The relationship between syllable repertoire similarity and pairing success in a passerine bird species with complex song. *Journal of Theoretical Biology*, 295:68–76. DOI: 10.1016/j.jtbi.2011.11.011.

Gokhale, M., Khanduja, D. K., *et al*. (2010). Time domain signal analysis using wavelet packet decomposition approach. *Int'l J. of Communications, Network and System Sciences*, 3(03):321. DOI: 10.4236/ijcns.2010.33041.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press. DOI: 10.1038/nature14539.

Guo, R. and Zheng, Y. (2022). A wind noise detection and suppression method in digital hearing aid. In *2022 International Conference on Networks, Communications and Information Technology (CNCIT)*, pages 20–24. IEEE. DOI: 10.1109/cncit56797.2022.00012.

Honkakunnas, A. (2021). Characterizing and detecting wind noise in audio recordings. Master's thesis. Available at: `https://trepo.tuni.fi/bitstream/handle/10024/131219/HonkakunnasAapo.pdf;jsessionid=B5D74336894FFD83CDB9C9E213B5B60E?sequence=2`.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713. DOI: 10.1109/cvpr.2018.00286.

Juodakis, J. and Marsland, S. (2022). Wind-robust sound event detection and denoising for bioacoustics. *Methods in Ecology and Evolution*, 13(9):2005–2017. DOI: 10.1111/2041-210X.13928.

Kahl, S., Wood, C. M., Eibl, M., and Klinck, H. (2021). Birdnet: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, 61:101236. DOI: 10.1016/j.ecoinf.2021.101236.

Kamarajugadda, R., Battula, R., Borra, C. R., Durga, H., Bypilla, V., Reddy, S. S., Khan, F. F., and Bhavanam, S. (2024). Optimizing avian species recognition with mfcc features and deep learning models. *International Journal of Information Technology*, 16(7):4621–4626. DOI: 10.1007/s41870-024-02108-1.

Kershenbaum, A., Blumstein, D. T., Roch, M. A., Akçay, □., Backus, G., Bee, M. A., Bohn, K., Cao, Y., Carter, G., Cäsar, C., *et al*. (2016). Acoustic sequences in non-human animals: a tutorial review and prospectus. *Biological Reviews*, 91(1):13–52. DOI: 10.1111/brv.12160.

Lasseck, M. (2018). Acoustic bird detection with deep convolutional neural networks. In *DCASE*, pages 143–147. Available at: `https://dcase.community/documents/challenge2018/technical_reports/DCASE2018_Lasseck_76.pdf`.

Mekonen, S. (2017). Birds as biodiversity and environmental indicator. *Indicator*, 7(21). DOI: 10.3390/electronics8111289.

Morelli, F., Reif, J., Díaz, M., Tryjanowski, P., Ibáñez-álamo, J. D., Suhonen, J., Jokimäki, J., Kaisanlahti-Jokimäki, M.-L., Moller, A. P., Bussiere, R., *et al*. (2021). Top ten birds indicators of high environmental quality in european cities. *Ecological Indicators*, 133:108397. DOI: 10.1016/j.ecolind.2021.1083979.

Nelke, C., Jax, P., and Vary, P. (2016). Wind noise detection: Signal processing concepts for speech communication. *Energy*, 60(40):20. Available at: `https://pub.dega-akustik.de/DAGA_2016/data/articles/000078.pdf`.

Ntalampiras, S. and Potamitis, I. (2021). Acoustic detection of unknown bird species and individuals. *CAAI Transactions on Intelligence Technology*, 6(3):291–300. DOI: 10.1049/cit2.12007.

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. DOI: 10.1145/2733373.2806390.

Potamitis, I., Ntalampiras, S., Jahn, O., and Riede, K. (2014). Automatic bird sound detection in long real-field recordings: Applications and tools. *Applied Acoustics*, 80:1–9. DOI: 10.1016/j.apacoust.2014.01.001.

Prince, P., Hill, A., Piña Covarrubias, E., Doncaster, P., Snaddon, J. L., and Rogers, A. (2019). Deploying acoustic detection algorithms on low-cost, open-source acoustic sensors for environmental monitoring. *Sensors*, 19(3):553. DOI: 10.3390/s19030553.

Rauch, L., Schwinger, R., Wirth, M., Sick, B., Tomforde, S., and Scholz, C. (2023). Active bird2vec: Towards end-to-end bird sound monitoring with transformers. *arXiv preprint arXiv:2308.07121*. DOI: 10.48550/arxiv.2308.07121.

Saad, A. (2020). *Bird species identification using spectrograms and convolutional neural networks*. PhD thesis, Ph. D. dissertation, Dept. Comput. and Microelectronic

Syst., Univ .... Available at: `https://eprints.utm.my/93040/`.

Situnayake, D. and Plunkett, J. (2023). *AI at the Edge*. " O'Reilly Media, Inc.". Book.

Wotton, S., Eaton, M., Sheehan, D., Munyekenye, F. B., Burfield, I., Butchart, S., Moleofi, K., Nalwanga-Wabwire, D., Pomeroy, D., Senyatso, K., *et al.* (2020). Developing biodiversity indicators for african birds. *Oryx*, 54(1):62–73. DOI: 10.1017/S0030605317001181.

Xie, J., Hu, K., Zhu, M., Yu, J., and Zhu, Q. (2019). Investigation of different cnn-based models for improved bird sound classification. *IEEE Access*, 7:175353–175361. DOI: 10.1109/ACCESS.2019.2957572.

Yang, S., Frier, R., and Shi, Q. (2021). Acoustic classification of bird species using wavelets and learning algorithms. In *2021 13th International Conference on Machine Learning and Computing*, pages 67–71. DOI: 10.1145/3457682.3457692.

Yen, G. G. and Lin, K.-C. (2000). Wavelet packet feature extraction for vibration monitoring. *IEEE transactions on industrial electronics*, 47(3):650–667. DOI: 10.1109/41.847906.

Zhang, S., Gao, Y., Cai, J., Yang, H., Zhao, Q., and Pan, F. (2023). A novel bird sound recognition method based on multifeature fusion and a transformer encoder. *Sensors*, 23(19):8099. DOI: 10.3390/s23198099.

Zhang, T., Feng, G., Liang, J., and An, T. (2021). Acoustic scene classification based on mel spectrogram decomposition and model merging. *Applied Acoustics*, 182:108258. DOI: 10.1016/j.apacoust.2021.108258.

Zhao, P., Luo, C., Qiao, B., Wang, L., Rajmohan, S., Lin, Q., and Zhang, D. (2022). T-smote: Temporal-oriented synthetic minority oversampling technique for imbalanced time series classification. In *IJCAI*, pages 2406–2412. DOI: 10.24963/ijcai.2022/334.

Şekercioğlu çağan H, Daily, G. C., and Ehrlich, P. R. (2004). Ecosystem consequences of bird declines. *Proceedings of the National Academy of Sciences*, 101(52):18042–18047. DOI: 10.1073/pnas.0408049101.

A EloquentML

This appendix aims to explain the use of the EloquentML library for converting Scikit-Learn models to C language. The code below provides an example of how to use the Scikit-Learn library to train a Support Vector Machine (SVM) model using the Iris dataset. Through the 'port' method, imported from micromlgen (one of the tools in EloquentML), the trained model is converted into a .h file.

```python
from micromlgen import port
from sklearn.svm import SVC
from sklearn.datasets import load_iris

if __name__ == '__main__':

    iris = load_iris()
    X = iris.data
    y = iris.target
    clf = SVC(kernel='linear', gamma=0.001).fit(X, y)

    with open('SVMLinear.h', 'w') as file:
        file.write(port(clf, classmap={
        0: 'setosa',
        1: 'virginica',
        2: 'versicolor'}))
```

The code generated in the .h file is shown as follows:

```cpp
#pragma once
#include <cstdarg>
namespace Eloquent {
    namespace ML {
        namespace Port {
            class SVM {
                public:
                    // Predict class for features vector
                    int predict(float *x) {
                        float kernels[27] = { 0 };
                        float decisions[3] = { 0 };
                        int votes[3] = { 0 };
                        kernels[0] = compute_kernel(x,  5.1 , 3.3 , 1.7 ,
                        ↪  0.5 );
                        kernels[1] = compute_kernel(x,  4.8 , 3.4 , 1.9 ,
                        ↪  0.2 );
                        (...)
                        + kernels[25] * -0.160296720576
                        - kernels[26];
                        votes[decisions[0] > 0 ? 0 : 1] += 1;
                        votes[decisions[1] > 0 ? 0 : 2] += 1;
                        votes[decisions[2] > 0 ? 1 : 2] += 1;
                        int val = votes[0];
                        int idx = 0;
                        for (int i = 1; i < 3; i++) {
                            if (votes[i] > val) {
                                val = votes[i];
                                idx = i;
                            }
                        }
                        return idx;
                    }
                    // Predict readable class name
                    const char* predictLabel(float *x) {
                        return idxToLabel(predict(x));
                    }
                    // Convert class idx to readable name
                    const char* idxToLabel(uint8_t classIdx) {
                        switch (classIdx) {
                            case 0:
                            return "setosa";
                            case 1:
                            return "virginica";
                            case 2:
                            return "versicolor";
                            default:
                            return "Houston we have a problem";
                        }
                    }
                protected:
                    // Compute kernel between feature vector and support vector
                    ↪  Linear.
                    float compute_kernel(float *x, ...) {
                        va_list w;
                        va_start(w, 4);
                        float kernel = 0.0;
                        for (uint16_t i = 0; i < 4; i++) {
                            kernel += x[i] * va_arg(w, double);
                        }
                        return kernel;
                    } }; } } }
```

Finally, an example of how to import the header file and use the SVM class of our model, and use the predict() method.

```cpp
#include SVMLinear.h"

Eloquent::ML::Port::SVM cls;
float X[] = {...};

void setup() {
}

void loop() {
    float y_pred = cls.predict(X);
}
```

B Shallow Learners Results

From Table 15 to Table 28, we present the classification performance of all shallow learners tested, comparing the results with and without the use of T-SMOTE data augmentation. As in Section 4, the analysis considered the optimal set of hyperparameters for each model, determined through hyperparameter tuning (HPT) combined with a Monte Carlo cross-validation approach. That is, the Monte Carlo algorithm used for this purpose involves conducting 100 experiments for each model evaluated by the HPT. In each experiment, the dataset is randomly split into training and test data in a stratified 80/20 ratio, respectively, with subsequent syllable splits

for bird species and time window segmentation. The metrics presented in this appendix for each shallow learner represent a single trained model with the optimal set of hyperparameters evaluated on the same test data. The results for the XGBoost model, the selected shallow learner to be embedded into the Nordic Thingy:53, are presented in Section 4.

Table 15. Performance metrics for the trained LightGBM model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**92.1**	**95.9**	**94.0**
Black-capped Screech-Owl	**89.2**	**90.3**	**89.7**
White-chinned Sapphire	**83.4**	**87.2**	**85.3**
Engine	76.4	93.9	84.3
Insects	77.6	86.9	82.0
Tropical Screech-Owl	80.8	82.9	81.8
White-eyed Parakeet	81.4	62.0	70.4
Pavonine Cuckoo	69.6	63.5	66.4
Undulated Tinamou	80.0	51.5	62.7
Black-capped Antwren	63.4	59.3	61.3
Red-shouldered Macaw	63.8	59.1	61.3
Flavescent Warbler	69.4	53.1	60.1
Dog barking	63.8	55.8	59.6
Boat-billed Flycatcher	53.1	57.6	55.3
Blue-and-yellow Macaw	65.2	41.0	50.3
Helmeted Manakin	45.0	47.9	46.4
Amazonian Motmot	62.7	36.4	46.0
Buff-necked Ibis	54.1	35.4	42.8
Small-billed Tinamou	51.4	34.6	41.4
Great Kiskadee	60.0	31.3	41.2
Pale-breasted Thrush	46.6	36.0	40.6
Orange-headed Tanager	60.0	25.5	35.8
Chivi Vireo	32.0	13.6	19.0
Weighted avg	73.9	75.2	73.7

Table 16. Performance metrics for the trained LightGBM model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**99.0**	**99.0**	**99.0**
Black-capped Screech-Owl	**96.1**	**89.7**	**92.8**
Tropical Screech-Owl	**89.6**	**90.8**	**90.2**
White-chinned Sapphire	89.3	87.2	88.2
Insects	88.8	81.0	84.7
Engine	91.4	79.3	84.9
Red-shouldered Macaw	80.0	75.2	77.5
Undulated Tinamou	74.1	75.8	74.9
Flavescent Warbler	65.7	85.2	74.2
Black-capped Antwren	77.3	70.7	73.9
Pavonine Cuckoo	79.5	67.9	73.2
White-eyed Parakeet	65.5	80.4	72.2
Boat-billed Flycatcher	66.2	72.9	69.4
Dog barking	61.9	75.8	68.2
Helmeted Manakin	64.9	69.0	66.9
Small-billed Tinamou	58.7	71.2	64.3
Buff-necked Ibis	55.8	72.6	63.1
Blue-and-yellow Macaw	51.9	78.1	62.4
Amazonian Motmot	51.6	72.7	60.4
Orange-headed Tanager	50.3	70.8	58.8
Great Kiskadee	55.3	62.7	58.7
Pale-breasted Thrush	51.7	65.8	57.9
Chivi Vireo	82.1	79.9	80.6
Weighted avg	82.1	79.9	80.6

Table 17. Performance metrics for the trained SVC model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**79.2**	**90.4**	**84.4**
Engine	**62.6**	**79.9**	**70.2**
Black-capped Screech-Owl	**56.5**	**83.2**	**67.3**
Chopi Blackbird	43.1	90.8	58.4
Pavonine Cuckoo	48.0	65.0	55.2
Insects	39.6	40.2	39.9
Tropical Screech-Owl	52.2	31.6	39.3
Undulated Tinamou	70.8	25.8	37.8
Amazonian Motmot	62.5	22.7	33.3
Orange-headed Tanager	75.9	20.8	32.6
Red-shouldered Macaw	51.2	14.8	22.9
Great Kiskadee	24.6	20.9	22.6
Dog barking	78.9	12.5	21.6
Helmeted Manakin	34.4	15.5	21.4
Boat-billed Flycatcher	29.2	11.9	16.9
Buff-necked Ibis	24.5	10.6	14.8
Pale-breasted Thrush	100.0	7.9	14.6
Black-capped Antwren	13.1	7.9	9.8
White-eyed Parakeet	33.3	5.4	9.3
Flavescent Warbler	66.7	2.5	4.8
Chivi Vireo	100.0	1.7	3.3
Blue-and-yellow Macaw	100.0	1.0	1.9
Small-billed Tinamou	0.0	0.0	0.0
Weighted avg	54.9	53.2	47.8

Table 18. Performance metrics for the trained SVC model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**83.7**	**86.8**	**85.2**
Chopi Blackbird	**76.7**	**56.4**	**65.0**
Black-capped Screech-Owl	**52.7**	**78.3**	**63.0**
Tropical Screech-Owl	45.0	94.7	61.0
Engine	84.6	46.8	60.2
Pavonine Cuckoo	67.3	48.2	56.2
Undulated Tinamou	54.3	57.6	55.9
Buff-necked Ibis	36.1	54.0	43.3
White-eyed Parakeet	32.9	58.7	42.2
Orange-headed Tanager	41.9	41.5	41.7
Red-shouldered Macaw	36.6	37.6	37.1
Insects	66.4	24.4	35.7
Boat-billed Flycatcher	22.5	57.6	32.4
Flavescent Warbler	21.1	69.1	32.3
Pale-breasted Thrush	39.1	23.7	29.5
Helmeted Manakin	40.8	21.8	28.4
Small-billed Tinamou	20.0	48.1	28.2
Blue-and-yellow Macaw	19.8	47.6	27.9
Dog barking	20.2	44.2	27.7
Great Kiskadee	21.4	37.3	27.2
Black-capped Antwren	26.5	25.7	26.1
Amazonian Motmot	17.0	43.2	24.4
Chivi Vireo	15.9	16.9	16.4
Weighted Average	59.9	46.7	50.1

Table 19. Performance metrics for the trained LinearSVC model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**77.3**	**91.8**	**83.9**
Chopi Blackbird	**54.7**	**91.5**	**68.5**
Engine	**40.5**	**95.1**	**56.8**
Pavonine Cuckoo	46.1	53.3	49.4
Black-capped Screech-Owl	46.2	28.5	35.3
Orange-headed Tanager	52.6	9.4	16.0
Pale-breasted Thrush	35.0	6.1	10.4
Helmeted Manakin	53.8	4.9	9.0
Insects	11.9	6.0	8.0
Tropical Screech-Owl	66.7	2.6	5.1
Flavescent Warbler	100.0	1.2	2.4
Amazonian Motmot	0.0	0.0	0.0
Black-capped Antwren	0.0	0.0	0.0
Blue-and-yellow Macaw	0.0	0.0	0.0
Boat-billed Flycatcher	0.0	0.0	0.0
Buff-necked Ibis	0.0	0.0	0.0
Chivi Vireo	0.0	0.0	0.0
Dog barking	0.0	0.0	0.0
Great Kiskadee	0.0	0.0	0.0
Red-shouldered Macaw	0.0	0.0	0.0
Small-billed Tinamou	0.0	0.0	0.0
Undulated Tinamou	0.0	0.0	0.0
White-eyed Parakeet	0.0	0.0	0.0
Weighted avg	32.8	43.2	32.0

Table 20. Performance metrics for the trained LinearSVC model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**65.0**	**90.0**	**75.5**
Chopi Blackbird	**86.9**	**66.2**	**75.1**
Black-capped Screech-Owl	**50.5**	**74.9**	**60.3**
Tropical Screech-Owl	39.3	98.7	56.2
Pavonine Cuckoo	59.9	49.6	54.3
Undulated Tinamou	53.5	40.2	45.9
White-eyed Parakeet	35.3	45.7	39.8
Red-shouldered Macaw	55.3	17.4	26.5
Orange-headed Tanager	21.3	34.0	26.2
Great Kiskadee	16.7	34.3	22.4
Boat-billed Flycatcher	13.6	61.0	22.2
Small-billed Tinamou	11.7	67.3	19.9
Pale-breasted Thrush	21.5	14.9	17.6
Flavescent Warbler	9.7	61.7	16.8
Buff-necked Ibis	13.4	19.5	15.9
Chivi Vireo	9.9	30.5	15.0
Engine	82.1	7.2	13.2
Blue-and-yellow Macaw	9.9	14.3	11.7
Amazonian Motmot	5.5	53.4	10.0
Helmeted Manakin	22.6	4.9	8.1
Dog barking	9.7	2.5	4.0
Black-capped Antwren	21.4	2.1	3.9
Insects	6.7	0.3	0.6
Weighted avg	48.3	31.0	28.4

Table 21. Performance metrics for the trained NUSVC model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**84.8**	**84.0**	**84.4**
Chopi Blackbird	**77.2**	**62.6**	**69.1**
Black-capped Screech-Owl	**52.0**	**75.5**	**61.6**
Engine	84.8	41.0	55.3
Undulated Tinamou	57.6	51.5	54.4
Pavonine Cuckoo	67.4	44.5	53.6
Tropical Screech-Owl	29.5	92.1	44.7
White-eyed Parakeet	36.0	53.3	43.0
Orange-headed Tanager	40.8	39.6	40.2
Red-shouldered Macaw	34.5	45.0	39.1
Buff-necked Ibis	30.9	40.7	35.1
Boat-billed Flycatcher	25.6	52.5	34.4
Blue-and-yellow Macaw	21.3	58.1	31.1
Dog barking	26.9	35.0	30.4
Black-capped Antwren	31.0	27.9	29.3
Great Kiskadee	24.5	34.3	28.6
Pale-breasted Thrush	36.1	22.8	28.0
Small-billed Tinamou	18.7	50.0	27.2
Amazonian Motmot	17.4	46.6	25.4
Flavescent Warbler	15.7	66.7	25.4
Insects	37.8	18.7	25.0
Helmeted Manakin	36.2	17.6	23.7
Chivi Vireo	17.9	23.7	20.4
Weighted avg	56.1	45.8	47.0

Table 22. Performance metrics for the trained NUSVC model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**84.5**	**79.5**	**81.9**
Chopi Blackbird	**89.5**	**50.5**	**64.6**
Black-capped Screech-Owl	**54.5**	**77.1**	**63.9**
Engine	82.1	45.0	58.1
Pavonine Cuckoo	63.3	39.1	48.3
Tropical Screech-Owl	30.2	92.1	45.5
Undulated Tinamou	49.1	41.7	45.1
White-eyed Parakeet	33.1	58.7	42.4
Red-shouldered Macaw	41.5	36.2	38.7
Orange-headed Tanager	32.3	39.6	35.6
Buff-necked Ibis	35.5	34.5	35.0
Dog barking	42.0	28.3	33.8
Insects	39.5	28.7	33.3
Boat-billed Flycatcher	24.6	47.5	32.4
Flavescent Warbler	19.6	65.4	30.2
Blue-and-yellow Macaw	17.9	56.2	27.1
Great Kiskadee	20.8	38.8	27.1
Small-billed Tinamou	17.7	50.0	26.1
Amazonian Motmot	18.3	39.8	25.1
Chivi Vireo	21.0	28.8	24.3
Helmeted Manakin	29.3	20.4	24.1
Pale-breasted Thrush	20.8	28.1	23.9
Black-capped Antwren	30.3	19.3	23.6
Weighted avg	56.2	46.0	47.8

Table 23. Performance metrics for the trained Random Forest model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**95.1**	**100.0**	**97.5**
Black-capped Screech-Owl	**92.9**	**95.1**	**94.0**
Tropical Screech-Owl	**89.2**	**97.4**	**93.1**
White-chinned Sapphire	88.5	95.0	91.6
Insects	88.0	91.0	89.5
Engine	82.3	96.6	88.9
Pavonine Cuckoo	80.1	78.1	79.1
Undulated Tinamou	89.4	70.5	78.8
Flavescent Warbler	79.7	77.8	78.7
Dog barking	74.1	71.7	72.9
Red-shouldered Macaw	73.6	71.1	72.4
Black-capped Antwren	74.0	69.3	71.6
White-eyed Parakeet	74.1	68.5	71.2
Blue-and-yellow Macaw	78.0	61.0	68.4
Buff-necked Ibis	73.5	63.7	68.2
Helmeted Manakin	71.1	60.6	65.4
Boat-billed Flycatcher	80.0	54.2	64.6
Orange-headed Tanager	74.0	53.8	62.3
Pale-breasted Thrush	77.8	49.1	60.2
Small-billed Tinamou	77.4	46.2	57.8
Amazonian Motmot	80.9	43.2	56.3
Great Kiskadee	64.6	46.3	53.9
Chivi Vireo	63.4	44.1	52.0
Weighted avg	83.2	83.7	82.9

Table 24. Performance metrics for the trained Random Forest model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**97.5**	**99.5**	**98.5**
Tropical Screech-Owl	**93.7**	**97.4**	**95.5**
Black-capped Screech-Owl	**95.8**	**92.9**	**94.3**
White-chinned Sapphire	90.6	92.2	91.4
Insects	92.8	87.8	90.2
Engine	94.2	82.0	87.6
Black-capped Antwren	84.8	75.7	80.0
Red-shouldered Macaw	75.6	81.2	78.3
Undulated Tinamou	75.4	81.1	78.1
Pavonine Cuckoo	85.3	71.9	78.0
Dog barking	73.0	83.3	77.8
Flavescent Warbler	68.2	90.1	77.7
White-eyed Parakeet	69.1	82.6	75.2
Boat-billed Flycatcher	67.7	74.6	71.0
Helmeted Manakin	73.3	67.6	70.3
Orange-headed Tanager	60.6	75.5	67.2
Buff-necked Ibis	60.4	74.3	66.7
Pale-breasted Thrush	62.3	71.1	66.4
Great Kiskadee	60.5	73.1	66.2
Small-billed Tinamou	56.2	78.8	65.6
Chivi Vireo	58.1	72.9	64.7
Amazonian Motmot	58.7	69.3	63.5
Blue-and-yellow Macaw	51.9	79.0	62.6
Weighted avg	85.3	83.6	84.1

Table 25. Performance metrics for the trained Decision Trees model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**94.7**	**95.9**	**95.3**
White-chinned Sapphire	**85.7**	**87.2**	**86.4**
Black-capped Screech-Owl	**83.5**	**85.2**	**84.4**
Tropical Screech-Owl	76.7	86.8	81.5
Engine	80.5	80.8	80.7
Insects	78.7	79.3	79.0
Undulated Tinamou	63.6	67.4	65.4
Pavonine Cuckoo	57.1	59.9	58.5
Flavescent Warbler	53.7	63.0	58.0
Buff-necked Ibis	56.8	55.8	56.2
White-eyed Parakeet	52.5	57.6	54.9
Red-shouldered Macaw	52.3	54.4	53.3
Boat-billed Flycatcher	50.0	50.8	50.4
Black-capped Antwren	50.0	47.1	48.5
Dog barking	49.5	45.0	47.2
Orange-headed Tanager	45.5	43.4	44.4
Amazonian Motmot	48.6	39.8	43.8
Pale-breasted Thrush	41.3	39.5	40.4
Blue-and-yellow Macaw	39.0	37.1	38.0
Small-billed Tinamou	36.4	38.5	37.4
Helmeted Manakin	42.3	33.1	37.2
Great Kiskadee	38.3	26.9	31.6
Chivi Vireo	29.0	33.9	31.3
Weighted avg	70.2	70.5	70.3

Table 26. Performance metrics for the trained Decision Trees model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
Chopi Blackbird	**95.9**	**96.9**	**96.4**
Tropical Screech-Owl	**91.0**	**93.4**	**92.2**
Black-capped Screech-Owl	**88.2**	**86.4**	**87.3**
White-chinned Sapphire	88.1	84.5	86.2
Insects	87.6	77.6	82.3
Engine	91.1	66.5	76.8
Undulated Tinamou	69.1	72.7	70.8
Red-shouldered Macaw	71.2	69.8	70.5
White-eyed Parakeet	56.9	80.4	66.7
Pavonine Cuckoo	67.8	63.1	65.4
Black-capped Antwren	61.9	68.6	65.1
Flavescent Warbler	50.4	77.8	61.2
Dog barking	54.2	70.0	61.1
Buff-necked Ibis	52.2	62.8	57.0
Blue-and-yellow Macaw	47.7	67.6	55.9
Helmeted Manakin	51.3	56.3	53.7
Pale-breasted Thrush	42.8	54.4	47.9
Amazonian Motmot	38.5	62.5	47.6
Small-billed Tinamou	35.9	63.5	45.8
Orange-headed Tanager	37.8	55.7	45.0
Boat-billed Flycatcher	41.2	47.5	44.1
Great Kiskadee	36.6	55.2	44.0
Chivi Vireo	31.1	47.5	37.6
Weighted avg	76.6	72.5	73.7

Table 27. Performance metrics for the trained Naive Bayes model without SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**76.2**	**87.7**	**81.5**
Tropical Screech-Owl	**59.8**	**64.5**	**62.0**
Chopi Blackbird	**84.4**	**47.2**	**60.5**
White-eyed Parakeet	31.0	42.4	35.8
Pavonine Cuckoo	75.0	21.9	33.9
Orange-headed Tanager	53.1	16.0	24.6
Red-shouldered Macaw	13.3	78.5	22.8
Black-capped Screech-Owl	34.1	14.2	20.0
Great Kiskadee	11.8	43.3	18.6
Dog barking	12.3	27.5	17.0
Boat-billed Flycatcher	11.0	30.5	16.1
Flavescent Warbler	11.9	16.0	13.7
Buff-necked Ibis	45.0	8.0	13.5
Amazonian Motmot	31.8	8.0	12.7
Pale-breasted Thrush	19.2	8.8	12.0
Small-billed Tinamou	5.9	26.9	9.6
Engine	34.6	4.8	8.4
Blue-and-yellow Macaw	4.3	68.6	8.1
Undulated Tinamou	15.0	4.5	7.0
Chivi Vireo	7.9	5.1	6.2
Helmeted Manakin	16.7	2.8	4.8
Insects	23.8	1.5	2.9
Black-capped Antwren	0.0	0.0	0.0
Weighted avg	36.2	20.2	19.9

Table 28. Performance metrics for the trained Naive Bayes model with T-SMOTE data augmentation, sorted by F1 score. The top three performing classes are highlighted in teal. In cases of a tie, the classes are ordered alphabetically.

Class	Precision (%)	Recall (%)	F1 (%)
White-chinned Sapphire	**75.2**	**88.6**	**81.3**
Tropical Screech-Owl	**58.8**	**65.8**	**62.1**
Chopi Blackbird	**87.1**	**46.7**	**60.8**
White-eyed Parakeet	27.8	43.5	33.9
Pavonine Cuckoo	76.6	21.5	33.6
Red-shouldered Macaw	15.6	68.5	25.5
Orange-headed Tanager	40.5	16.0	23.0
Amazonian Motmot	13.3	45.5	20.6
Great Kiskadee	10.6	53.7	17.7
Pale-breasted Thrush	21.8	10.5	14.2
Flavescent Warbler	10.6	14.8	12.4
Buff-necked Ibis	25.7	8.0	12.2
Dog barking	13.8	9.2	11.0
Black-capped Screech-Owl	69.2	5.5	10.1
Boat-billed Flycatcher	7.8	11.9	9.4
Small-billed Tinamou	5.7	19.2	8.8
Blue-and-yellow Macaw	4.4	80.0	8.4
Engine	47.3	4.0	7.3
Chivi Vireo	7.0	6.8	6.9
Helmeted Manakin	16.2	4.2	6.7
Undulated Tinamou	8.8	4.5	6.0
Black-capped Antwren	2.0	2.1	2.1
Insects	10.8	0.6	1.2
Weighted avg	40.5	19.1	18.4