MosAIc: A Classical Machine Learning Multi-Classifier Based Approach against Deep Learning Classifiers for Embedded Sound Classification

Lancelot Lhoest [1,*] , **Mimoun Lamrini** [2,*] , **Jurgen Vandendriessche** [1] , **Nick Wouters** [1] , **Bruno da Silva** [1,3,*] , **Mohamed Yassin Chkouri** [2] and **Abdellah Touhafi** [1,3,*]

[1] Department of Engineering Sciences and Technology (INDI), Vrije Universiteit Brussel (VUB), 1050 Brussels, Belgium; Jurgen.Vandendriessche@vub.be (J.V.); Nick.Wouters@vub.be (N.W.)
[2] Department of Computer Engineering (SIGL-Lab), University Abdelmalek Essaadi (UAE), Tetuan 93000, Morocco; mychkouri@uae.ac.ma
[3] Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB), 1050 Brussels, Belgium
* Correspondence: lancelot.charles.lhoest@vub.be (L.L.); lamrini.mimoun-etu@uae.ac.ma (M.L.); bruno.da.silva@vub.be (B.d.S.); abdellah.touhafi@vub.be (A.T.)

Abstract: Environmental Sound Recognition has become a relevant application for smart cities. Such an application, however, demands the use of trained machine learning classifiers in order to categorize a limited set of audio categories. Although classical machine learning solutions have been proposed in the past, most of the latest solutions that have been proposed toward automated and accurate sound classification are based on a deep learning approach. Deep learning models tend to be large, which can be problematic when considering that sound classifiers often have to be embedded in resource constrained devices. In this paper, a classical machine learning based classifier called MosAIc, and a lighter Convolutional Neural Network model for environmental sound recognition, are proposed to directly compete in terms of accuracy with the latest deep learning solutions. Both approaches are evaluated in an embedded system in order to identify the key parameters when placing such applications on constrained devices. The experimental results show that classical machine learning classifiers can be combined to achieve similar results to deep learning models, and even outperform them in accuracy. The cost, however, is a larger classification time.

Keywords: machine learning; environment sound recognition; Convolutional Neural Network; embedded system; audio feature extraction; multi-class classification; one-vs-all

1. Introduction

Environmental Sound Recognition (ESR), and more specifically urban sound recognition, has been rapidly evolving over the last few years. As cities try to become smart, ESR is a valuable sensing capability to monitor urban environments. ESR has a wide range of applications, both inside and outside of urban environments. First of all, ESR can help with identifying the sources of sounds responsible for sound noise pollution, which has a negative impact on health [1,2]. These sources of noise pollution range from barking dogs to car traffic to overflying planes. ESR helps to identify what type of noise is dominant, allowing specific solutions for each specific situation.

Currently, Closed-Circuit Television (CCTV) is used widely to monitor street activity in cities. However, video surveillance systems are limited in range and are expensive. ESR is used in sound surveillance systems to reduce the amount of blind spots and to increase the covered range. For instance, they are used to monitor urban traffic [3] or criminal activities [4] since sounds such as gun shots, alarms, dog barks, . . . are all indicators of possible criminal activity.

The relevance of ESR goes beyond urban environments. In agriculture, birds can damage crops; here, the ESR system detects birds so that they can be repelled to save

crops [5]. Repelling birds is not only important to save crops, but also for saving the lives of the birds. Although wind farms are a green source of electricity, they have a negative impact on bird populations. Research estimates that up to half a million birds do not survive a collision with a wind turbine every year in the U.S. alone [6]. An ESR system near the wind farm can activate a collision avoidance procedure when a bird is too close to prevent a collision. Although the focus of this work is on urban sound, some of the datasets also contain sounds that are not related to urban environments, yet, they are relevant for other ESR applications.

ESR is often performed on embedded systems, which are deployed in the area being monitored to perform audio recognition. Such systems provide limited computational power and memory capacity. Moreover, they are in general battery powered, which limits the use of complex algorithms, which can drastically reduce the autonomy of the system. Although recent works have proposed solutions [7], these limitations have slowed the deployment of ESR on embedded devices.

Deep Learning (DL) approaches have been proposed in recent years [8] which outperform the existing classical Machine Learning (ML) solutions in the field of Artificial Intelligence (AI). However, DL models have been growing in complexity, like in number of parameters, in order to deliver higher accuracy, leading to a higher inference time. The authors believe that classical ML based alternatives, which present a lower complexity, and therefore, a lower execution time, can still be competitive against DL models for ESR. This is especially interesting considering that they are used in constrained embedded devices.

The presented work exploits classical ML by combining different well-known ML classifiers, which is called Mesh of Sound AI-based Classifiers or MosAIc for short. This MosAIc approach is the ultimate attempt to exploit classical ML techniques to achieve similar or better accuracies than when using DL. Together with this novel approach, a lightweight DL approach is also proposed to overcome the limitations of embedded devices. Both approaches are compared and evaluated in different platforms, including a normal PC and a popular embedded device. Several audio datasets used in the related literature are used in order to provide a fair comparison and evaluation. Parameters such as accuracy and execution time are used as the metrics in evaluation.

The most relevant contributions of the presented work are:

- A methodology for a windowing-based audio feature evaluation to reduce the overall execution time;
- A novel combination of classical ML classifiers capable of outperforming DL is proposed and evaluated for ESR;
- A lightweight DL-based Convolutional Neural Network (CNN) is proposed and evaluated to be deployed on embedded devices for ESR.

This paper is organized as follows: Section 2 presents related work. The background is set in Section 3, where the methodology used for the evaluation is also presented. Section 4 describes the evaluation flow of the experiments and the experimental setup that is used. The evaluation of the feature extraction can be found in Section 5. Section 6 contains the results of the experiments using classical ML techniques, including the description of the novel MosAIc approach and the experimental results. Section 7 presents the proposed lightweight CNN model and the experimental results of its evaluation. In Section 8 a comparison between both approaches is done. Finally the conclusions are drawn in Section 9.

2. Related Work

ESR has gained more and more attention over the last few years. Most of the recent work focuses on DL approaches to improve the accuracy of the ML models. Some DL models are using the raw waveforms to identify the sound, while others use spectrograms or Mel-Frequency Cepstrum Coefficients (MFCC) as input for the Artificial Neural Networks (ANN) or CNN based classifiers [9–11].

The authors of [12] compare Supported Vector Machines (SVM) against two CNN models, GoogLeNet and AlexNet, trained on spectrograms from three datasets: ESC-10,

ESC-50 and UrbanSound8k. The authors compare not only the accuracy of the models but also their resiliency against adversarial attacks. Adversarial attacks are attacks where the input of the model is altered so that a human still labels the the altered input correctly while the models fail. The authors also propose an approach for increasing the robustness of the SVM model against such types of attacks.

A voting system to classify human activity based on sound is used in [13]. The audio is split into frames for which the MFCCs are calculated. These MFCCs are then used by a random forest classifier to predict the human activity on a frame by frame basis. Using a Non-Markovian ensemble voting system these predictions over time are combined to make a final prediction of the present activity.

A similar principle is used in [14] where a CNN is used to classify sound clips of 5 s. The input of the CNN are raw audio clips of T-seconds long that are generated using a sliding window. During testing, the outputs of the softmax are summed to classify the test data, which is similar to a voting system. The authors also combined this model with other CNN models that were trained using log mel spectrograms and/or delta log mel spectrograms. Each model was trained individually and the outputs of the models were averaged before applying softmax to classify the sounds.

The authors of [8] combined DL and manual feature extraction to classify urban sound. A CNN is used to extract deep features from the spectrograms. These deep features are then combined with manually selected features like zero crossing rate as input to a SVM or Random Forest. Although multiple metrics are used to compare the performance of both classifiers, they are not combined to form one classifier.

In [15] the authors combine three different ML techniques with three different sets of features to see which combinations perform the best. However, their definition of best performing only considers accuracy. Although accuracy is an important metric, in real-time applications, the time to perform a prediction is also important. This is especially true when using embedded devices such as Field-Programmable Gate Arrays (FPGAs) or microcontrollers.

The authors of [16] do consider using embedded devices. They describe a special function to calculate the cost of embedding the model based on the resources needed and the price of these resources. Three different models based on Gaussian mixture models (GMM), SVM and Deep Neural Networks were considered for classifying sounds from the smoke alarm dataset and the baby cry dataset. The input of the models was a combination of MFCC, spectral centroid, spectral flatness, spectral rolloff, spectral kurtosis and zero crossing rate. The authors do not consider how much time each model needs to classify a sound.

A novelty detector for detecting sound events using a GMM model is implemented on a beagle bone in Reference [17]. The authors decided to use a feature similar to MFCC: Power Normalized Cepstral Coefficients (PNCC) together with the first derivative of the PNCC as input for the model. More recently, the authors of [18] compared performance, training and testing time for four different machine learning techniques, SVM, k-Nearest Neighbour (k-NN), bootstrap aggregation and Random Forest, on a Raspberry Pi (RPi) Zero W. To train the models, 3042 audio clips from UrbanSound8k and Sound Events and 8 different classes are used. Although the SVM and k-NN had the highest accuracy, F_1 score, recall and precision, they also have the highest test time with 1.99 s and 0.5 s for the SVM and k-NN, respectively.

Microcontrollers have also been used for audio classification tasks, as in [19]. A small CNN with only 7867 parameters was trained to classify a recording as indoor, outdoor or in-vehicle. The network is trained using log mel spectrograms of 1024 ms long recordings from a proprietary dataset consisting of over 29 h of recordings. The trained model is implemented on a sensorTile development kit using 32 bit floating point (FP) and 8 bit integer by quantizing the model. Both the 32 bit FP model and the 8 bit integer model operate in real-time with an accuracy of 90.16% and 89.17% in 81.505 ms and 36.022 ms respectively.

3. Methodology

In this section, the background of audio classification is first set, before being linked to the methodology that is used in this work. After that, the selected datasets are presented, followed by the chosen audio features, the selected ML techniques and finally the metrics used to evaluate the classifiers.

The goal of the experiments is to compare classical ML techniques to CNN classification and see if combining classical ML techniques can outperform CNNs. Like many other pattern classification tasks, audio classification is made up of three fundamental components:

- Sensing: measuring the sound event or signal;
- Audio signal processing: extracting the characteristic features of the measured sound signal; and finally,
- Classification: recognizing the context of the sound event.

The audio signal processing part mainly deals with the extraction of features from the recorded audio signal. The various methods of time-frequency analysis developed for processing audio signals, in many cases originally developed for speech processing, are used. That is, feature extraction quantizes the audio signal and transforms it into various characteristic features. This results in an N dimensional feature vector often representing each audio frame. A classifier then takes this feature vector and determines what it represents—that is, it determines the context of the audio event. The classification technique has two phases: the training phase and the classification phase. During the training phase the system receives prerecorded audio data from labeled training sets to train a representative model. In the classification phase, the system receives its audio inputs from different prerecorded audio data or directly from a microphone. The classification phase uses the models generated during the training phase to match and determine the type of audio received by the microphone. Figure 1 is a representation of this process.
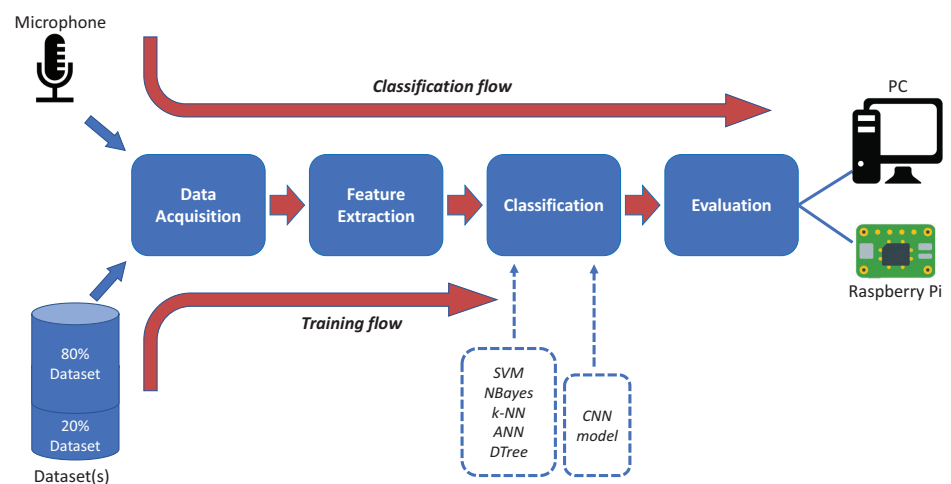

Figure 1. Different steps of the audio classification.

3.1. Datasets

Sound classifiers using ML techniques are used for automatic sound recognition. To be able to recognize a specific type of sound, these classifiers need to be trained. For such training, audio datasets of labelled sounds are necessary.

The well-known open source datasets for ESR used in the experiments are presented in Table 1. Not all classes in these datasets are directly urban related, but they contain enough representative information to be used for urban sound classification. Notice that altering these datasets by removing the audio clips that are not related to urban sounds makes comparison with other work impossible. Furthermore, the datasets are heterogeneous enough to be representative for realistic audio environments.

- **BDLib**: This dataset consists of 12 classes representing real-life situations. It was created by the authors in [20], collecting audio from sources like BBC Complete Sound Effects Library, Sony Pictures Sound Effects Series and the online sound repository called Freesound [21]. Every class contains ten different audio files of 10 s each.
- **UrbanSound**: This large dataset of 12 GB contains around 18.5 h of labelled audio [22]. It is composed of ten classes, all representing only urban sounds. Just like BDLib, UrbanSound was made by manually filtering and labelling every recording from the online sound repository called Freesound [21]. There is also a larger subset of 4-s audio clips called UrbanSound8K, based on this dataset, but it is not considered here.
- **ESC**: ESC-50, presented in [23], is an annotated collection of more than 2000 short audio clips belonging to 50 different classes. The classes of this dataset are not all related to urban sound, but also include environmental recordings such as animal sounds, domestic sounds or natural soundscapes. In this work, the smaller version ESC-10 is used, which contains 10 audio categories.

Table 1. Details of the open source datasets used in the evaluation.

BDLib Dataset		ESC-10 Dataset		UrbanSound Dataset	
Categories	Total Time (s)	Categories	Total Time (s)	Categories	Total Time (s)
Airplane	100	Dog barking	200	Air conditioner	6577
Alarms	100	Baby crying	200	Car horn	4819
Applause	100	Clock tick	200	Children playing	13,454
Birds	100	Person sneezing	200	Dog bark	8399
Dogs	100	Helicopter	200	Drilling	4864
Footsteps	100	Chainsaw	200	Engine idiling	3604
Motorcycles	100	Rooster	200	Gun shot	7865
Rain	100	Fire cracking	200	Jackhammer	4328
Sea waves	100	Sea waves	200	Siren	4477
Wind	100	Rain	200	Street music	6453
Rivers	100				
Thunderstorm	100				

3.2. Audio Features

The use of raw audio as input for classical machine learning models is not possible. Instead, audio features are extracted from the raw audio and serve as input. There is a wide variety in audio features and they can be broadly classified based on their semantic interpretation as perceptual and physical features. Perceptual features approximate properties that are perceived by human listeners such as pitch, loudness, rhythm and timbre. In contrast, physical features describe audio signals in terms of mathematical, statistical and physical properties. Based on the domain of representation, physical features are further divided as temporal features and spectral features. In this section, the physical features that are used during the implementation of these experimental studies are introduced. All of the audio features detailed below are provided by a specialized audio library called *librosa* [24].

- **Zero Crossing Rate (ZCR)**: The ZCR is the most common type of zero crossing based audio features [25]. It is defined as the number of time-domain zero crossings within a processing frame and indicates the frequency of signal amplitude sign change. ZCR allows for a rough estimation of dominant frequency and spectral centroid [26].
- **Root Mean Square (RMS)**: The RMS value is a measure of energy in a signal. The RMS value is defined to be the square root of the average of a squared signal [24].
- **Spectral Centroid (SC)**: SC [25] represents the "balancing point", or the midpoint of the spectral power distribution of a signal. It is related to the brightness of a sound. The higher the centroid, the brighter (high frequency) the sound is. A spectral centroid

provides a noise-robust estimate of how the dominant frequency of a signal changes over time.

- **Spectral Bandwidth (SB)**: Usually defined as the second-order statistic of the signal spectrum, it helps to discriminate tonal sounds (with low bandwidths) from noise-like sounds (with high bandwidths) [24].

- **Spectral Roll off Point (SRP)**: The spectral roll off point is the *N*% percentile of the power spectral distribution, where N is usually 85% or 95% [27].The spectral roll off point is the frequency below which *N*% of the magnitude distribution is concentrated. It increases with the bandwidth of a signal.

- **Mel Frequency Cepstral Coefficients (MFCC)**: MFCCs have been largely employed in the speech recognition field but also in the field of audio classification, due to the fact that their computation is based on a perceptual-based frequency scale in the first stage (the human auditory model which is inspired on the frequency Mel-scale). After obtaining the frame-based Fourier transform, outputs of a Mel-scale filter bank are logarithmized and finally decorrelated by means of the Discrete Cosine Transform (DCT). Only first DCT coefficients (usually from 8 to 13) are used to gather information that represents the low frequency component of the signals spectral envelope (mainly related to timbre) [24].

- **Chroma related features (CHROMA)**: Chroma are an interesting and powerful representations in which the entire spectrum is mapped into 12 cells representing the 12 semitones (or chroma) of the musical octave. It is also called a chromagram, and can be calculated using the short-term log Fourier transform of the sound signal. Another characteristics based on chrominance are the normalized chroma energy distribution statistics, which are used to identify the similarity between different interpretations of the given music [28].

- **Spectral Contrast (SCT)**: SCT is used for extracting spectral peaks, valleys and their differences in each sub-band while MFCCs sum up the Fast Fourier Transform (FFT) amplitudes. Thus, the spectral contrast function represents the relative spectral characteristics. Spectral contrast includes more spectral information than MFCCs because MFCCs only involve average spectral information [29].

- **Tonnetz (TZ)**: The Harmonic network or Tonnetz that are shown calculate the characteristics of the tonal centroid [30]. It is a well-known planar representation of pitch relations first attributed to Euler [31], and later widely used by 19th century music theorists such as Riemann and Oettingen and in recent years by Neo-Riemanninan Music theorists [31–33].

- **Mel Spectrogram (MEL)**: The non-linear transformation of the frequency scale which is based on the perception of locations is called the mel scale. This transformation calculates an energy spectrogram on the mel scale [30], so that two pairs of frequencies separated by a delta in the mel scale are perceived by humans to be equidistant [34].

For this paper, it has been decided to use the *librosa* package to extract useful features from sound information. This Python package is chosen because of its support of ARM-based processors and to be able to use the python package scikit-learn.

3.3. Classifiers

There are several classical ML techniques, which enable effective sound classifiers. For this evaluation, supervised ML techniques, which use correctly labelled data during the classifiers' training, are used. They are known to deliver a good performance when classifying acoustic events. These are the selected supervised ML techniques:

- **k-Nearest Neighbour (k-NN)**: The key idea behind this classifier is that, given a set of patterns (unknown feature vector), X, its k-nearest neighbors in the training set are detected and the count of those that belong to each class is calculated. The feature vector is assigned to the class which has the highest number of neighbors.

- **Naive Bayes**: The goal of this classifier is to decide which class a novel instance belongs to, by calculating the posterior probability of each class, given the feature

values present in the instance. The instance is assigned to the class with the highest probability.

- **Artificial Neural Network (ANN)**: An ANN consists of a set of processing elements, also known as neurons or nodes, which are interconnected. It can be described as a directed graph in which each node i performs a transfer function fi.
- **Support Vector Machine (SVM)**: Given a set of points which belong to either of two classes, a SVM finds a hyper plane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyper plane. SVM performs pattern recognition between two classes by finding a decision surface that has maximum distance to the closest points in the training set which are termed support vectors.
- **Decision Tree**: The decision tree algorithm tries to solve the problem by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

3.4. Metrics

In order to evaluate the different classifiers, several metrics are used for accuracy and timing. On the one hand, accuracy metrics, such as F_1 score [35–37], are used for a fair comparison of the classifiers. On the other hand, additional metrics, such as the execution time, are also considered since they are relevant for real-time applications, becoming a critical parameter for embedded devices.

3.4.1. Accuracy

The evaluation of the accuracy uses the F_1 score, which is a metric well-suited for multi-classifiers. The F_1 score is the harmonic mean of two other metrics: precision and recall, both metrics are based on the true positives, the false positives and the false negatives. Precision (Equation (1)) measures the ability of a classifier to identify only the correct instances for each class and recall (Equation (2)) is the ability of a classifier to find all correct instances per class. They are expressed as follows:

$$\text{Precision} = \frac{t_p}{t_p + f_p} \tag{1}$$

$$\text{Recall} = \frac{t_p}{t_p + t_n}, \tag{2}$$

where t_p and t_n are the number of true positives and true negatives respectively, while f_p is the number of false positives. Both parameters are used to obtain the F_1 score as follows:

$$F_1 \text{ score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3}$$

The value of the F_1 score is normalized between 0 and 1, a score of 1 indicating a perfect balance as precision and recall are inversely related [38]. Moreover, the F_1 score takes into account imbalanced class distribution, such as that which occurs with the one-vs-all approach and the MosAIc approach evaluated in Section 6.

The F_1 score can be averaged in three different ways: the micro average (F_1-micro), the macro average (F_1-macro) and the weighted average (F_1-weighted). The F_1-micro uses the individual true positives, false positives and false negatives to calculate the performance. Since the F_1-micro is not sensitive to the accuracy of individual classes, it can be a misleading metric in cases where the class distribution is imbalanced such as the UrbanSound dataset. A classifier obtaining a high micro average means that it performs well overall. The F_1-macro is calculated using larger groups, namely the performance of individual classes. Therefore it is more suitable when the data have an imbalanced class distribution. A high F_1-macro indicates that the classifier performs well for each individual class. Like the F_1-macro, the F_1-weighted is also calculated with the performance of

individual classes, but it alters the result to account for label imbalance [39]. All three different averages of the F_1 score are used in this evaluation of the classifiers.

3.4.2. Timing

Faster and accurate classification are features that are desirable when looking for real-time audio recognition. Consequently, the classification time is also used as a parameter in the evaluation of the classifiers. This classification time is obtained by measuring the time that is needed to predict to which class one audio sample belongs. The value has a different range based on what type of platform is used to evaluate the classifiers, becoming a valuable parameter when selecting the target platform.

4. Evaluation and Platforms

In this section, an overview of the pre-processing of the audio, the training process and of the embedded platforms that are evaluated is given. The evaluation flow is depicted in Figure 2. The BDLib, ESC-10 and UrbanSound datasets are used in this evaluation. In order to have enough training data, a windowing technique is applied during the feature extraction.
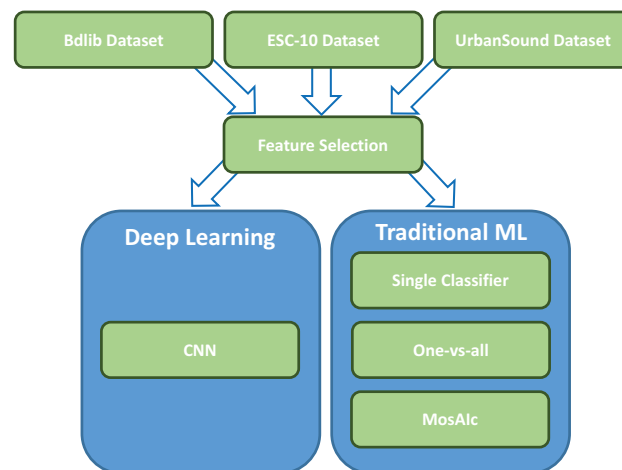

Figure 2. Evaluation flow detailing the steps performed for a fair comparison of classical ML-based approaches and a DL-based approach.

The audio features are extracted per audio frame, each of which is treated individually. The attributes belonging to the same feature are aggregated to obtain a single kind of feature vector. In this case, the statistical means are used as aggregators. Nonetheless, the median, the sum or GMM can also be used to perform feature aggregation. This aggregation is done to reduce the data and to characterize each audio frame sample with a single vector of features.

Based on the selected datasets, a feature selection is performed by identifying the relevant features. The most relevant features are used to compare classical ML based approaches against a DL approach. All results would then be compared to a DL approach using a CNN-based classifier. Both approaches are further detailed and discussed in the sections below.

4.1. Dataset Preprocessing: Windowing

The BDLib dataset contains 1800 audio files saved as wav files, ESC-10 has 400 ogg audio files and UrbanSound 1302 audio files. The files of BDLib and ESC-10 are 10 s long, the ones of UrbanSound are of variable length, between 1 s and several minutes. Furthermore, the audio files in the UrbanSound dataset have different file formats such as wav, .mp3, .ogg, ... To facilitate the experiments, all audio files in the UrbanSound

dataset are first converted to wav. This file format is used because most of the audio files in UrbanSound are already stored in this format and it is lossless as opposed to mp3 format.

To generate more data from these audio files, a windowing technique is used. Figure 3 depicts how the windowing process works. An audio file is loaded and the first 4 s are used for the windowing process. These 4 s audio are then divided into seven audio frames of 1 s, with a 50% overlap. These frames are only saved temporarily in an array, the time that features are extracted out of them. After that the next audio file of the dataset is loaded and the process is repeated. This way, seven times more feature groups are generated and assure enough training data. For example, for the ESC-10 dataset, 2800 feature groups are obtained with windowing instead of 400 without. It should be noted that windowing is only used to generate more training samples, and multiple feature groups are not to combined together to generate an averaged feature group per audio file.
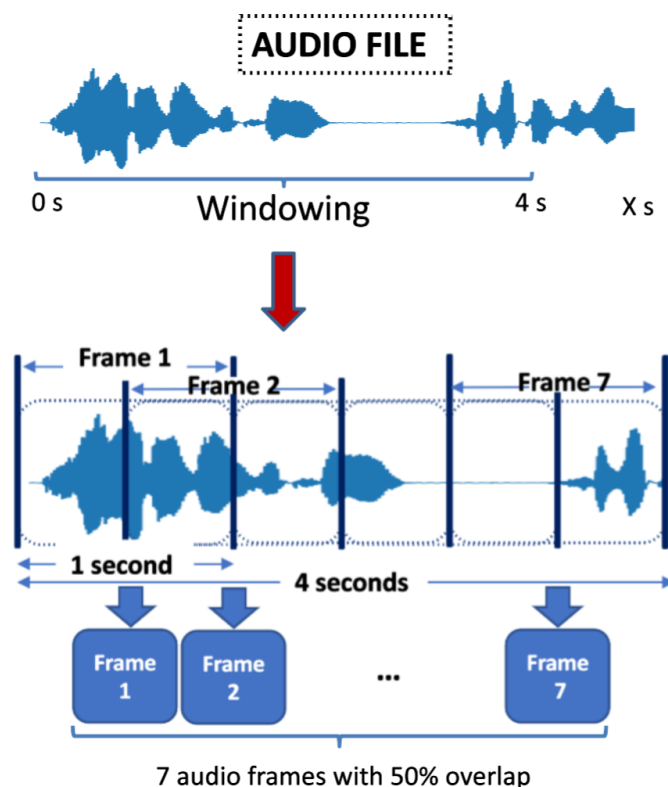

Figure 3. Visual representation of the windowing process to extract the seven audio frames from the four seconds of the audio file used for training.

The audio files of the UrbanSound dataset are different from those of the other datasets, in the sense that they are not exactly 10 s long, they can be shorter or longer, and the relevant audio does not always start at the beginning of the audio file. Therefore, every audio file comes with a CSV file containing the metadata of the associated audio file, including the start time. That start time is then used as offset to define where the 4 s used for windowing will start. For example if the start time of audio file X is 2 s, then for seconds 2 to 6 the windowing process will be used.

One feature group is generated per frame. The feature groups coming from the first 4 s of the audio file are used for the training and the testing of the classifiers, except for the testing of the MosAIc approach. This is due to the fact that all the feature groups coming from the first 4 s of the audio files, are used to train the ten binary classifiers which are used for the MosAIc approach. So in order to avoid using the same data for the training and the testing, another part of the audio files is used as test data for the MosAIc approach: a 4 s long audio frame starting after the fourth second of the audio file (plus a potential offset if working with the UrbanSound dataset). This is also shown in Figure 3.

4.2. Dataset Processing

The classification is performed by splitting the data into 80% training and 20% testing. Notice that the feature extraction and the sound classification are operations independent of the approaches since both use the same extracted features for the sound classification at all stages. Therefore, the experiments in this work target the profiling of the sound classifiers in terms of accuracy and timing.

To test the models, 5 fold cross validation was used. The CNN models were trained using four of the five folds, with an 80–20% distribution between training and validation data.

4.3. Platforms and Experimental Setup

The feature extraction is performed using the Python 2.7 package *librosa* 0.7.2. The classical ML classifiers are implemented using the Python library scikit-learn 0.20.4. The experiments are executed in two different platforms. The experiments performed in a desktop PC are used as a reference. The desktop PC includes a 2.6 GHz Intel Core i7 running Ubuntu O.S. 18.4. Additionally, a RPi 4B+ running Raspbian GNU/Linux 10 is used as representative of an embedded system. One of the objectives of this work is to be able to run the MosAIc on an embedded system maintaining a high accuracy and a low classification time while performing a fair comparison against DL approach.

5. Feature Selection for both ML Approaches

Tools for feature extraction such as *librosa* offer a large variety of features that can be extracted from audio files. In reality, however, a large set of features do not necessarily lead to an accurate prediction. In order to find the importance of each audio attribute, algorithms for feature selection are utilized [40]. The WEKA data mining tool [41] is a useful tool for the evaluation of the audio features. Instead of using all the attributes of the features in the audio classification, only relevant attributes are involved. A lower number of features reduces the processing time as well as increases the performance of the data mining task. Therefore, features and attributes selection is performed before applying data mining tasks such as classification, clustering and outlier analysis. The WEKA features selection uses the ranker algorithm InformationGain, which gives the set of the most relevant features per dataset based on their information gain or entropy with respect to other attributes. This entropy increases with the relevance of the attribute, which can then be used to select the most relevant feature set [7]. The ultimate objective is to achieve the highest accuracy while using the smallest number of features as possible. For this evaluation, only the Spectral features supported by *librosa* [24] are considered. The features are: MFCCs, RMS, CHROMA, MEL, SC, SB, SCT, SRP, ZCR, and TZ.

5.1. Features Information Gain

Figure 4 contains the most relevant features of the three datasets. These features are evaluated using the InformationGain algorithm, which uses entropy as a metric. The entropy values are used to identify the most relevant attributes of the selected features. For the ESC-10 dataset, a decrement below 0.3 occurs from the attribute *mel 84* til a value of 0.05 for the attribute *mel 117*. Similar results are achieved when evaluating the BDLiB dataset. For that dataset, these attributes range from 0.32 up to 0.1 for the attribute *mel 123*. The number of relevant attributes is strongly linked to the dataset. While the ESC-10 and BDLib datasets present similar ranking for the evaluated features, the UrbanSound dataset presents a lower number of relevant attributes. As a result, Figure 4 shows how the relevance of valuable attributes for the ESC-10 or BDLIb datasets decreases for UrbanSound dataset. In fact, most of the attributes present an entropy lower than 0.3. Figure 4 allows us to conclude that there is a decrease from the attribute *mel 38* in all cases, with a higher degree of decline for the UrbanSound dataset.
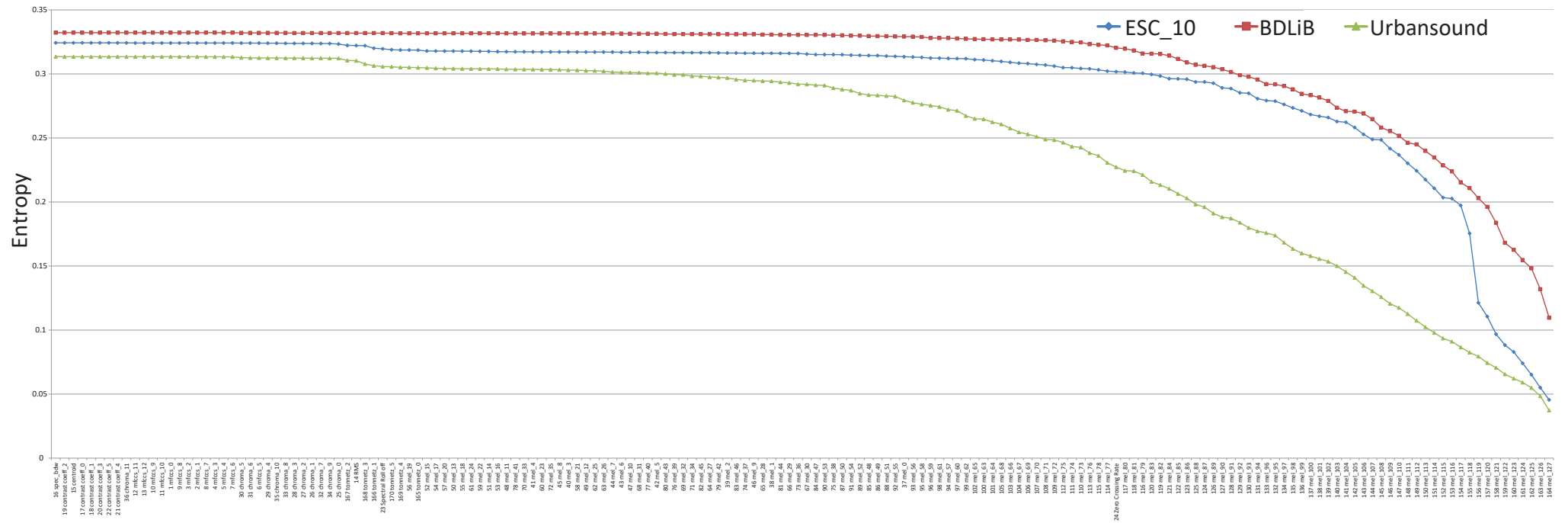
Figure 4. Information gain of the attributes for the ESC-10, BDLib and UrbanSound datasets.

This feature analysis is used in the comparison of classifiers to evaluate the impact of the set of features based on the classifiers and the datasets. Based on the *librosa* available features, a total of up to 170 attributes can be used when ten features are considered. Nonetheless, for a complete analysis towards its consideration for embedded systems, a reduced set of seven features (up to 24 attributes) is also considered, expecting a reduced feature extraction time. Figure 5 summarizes which features are used and how many dimensions they have. All the features used are from the *librosa* library and are spectral features. Even though certain features are multi-dimensional (MEL, SCT, MFCCs, CHROMA and TZ), they are transformed into one-dimensional arrays by taking the average over the frequencies. These one-dimensional arrays are then put together in 1 large one-dimensional array, which represents the feature group of one audio frame. These one-dimensional arrays are the input for the classifiers. Notice that each array is used as a single training or test sample and not combined with other arrays from the same audio file to generate a combined feature group per audio file.

Figure 5. Summary of features used when using ten features or only seven features, and the amount of dimensions they have.

5.2. Experimental Features Extraction Time

The selection of the features can be determined by the feature extraction time. Based on the previous feature analysis, this timing information can be used to select either seven features or ten features. Experimental timings of the feature extraction for the selected datasets in different platforms are depicted in Figure 6. Note the high impact that the selection of the platform has. Due to the limited resources of embedded devices, the feature extraction time is substantially higher when the feature extraction occurs in the RPi. This increment is in fact, independent of the dataset, since it affects of them equally. Similarly, it is interesting to notice how the number of features affects the feature extraction time. As expected, the use of the 7 most relevant features is up to five times faster than using all the ten spectral features available in *librosa*. Since the feature extraction affects both evaluated ML approaches, this experimental evaluation can be used to tune the selection of the platform and ML approaches towards the faster and most accurate approach. For instance, this selection is used in Section 6.5 to accelerate the MosAIc approach when ported to embedded devices. Further discussion about the selection of the features is done in Section 8.
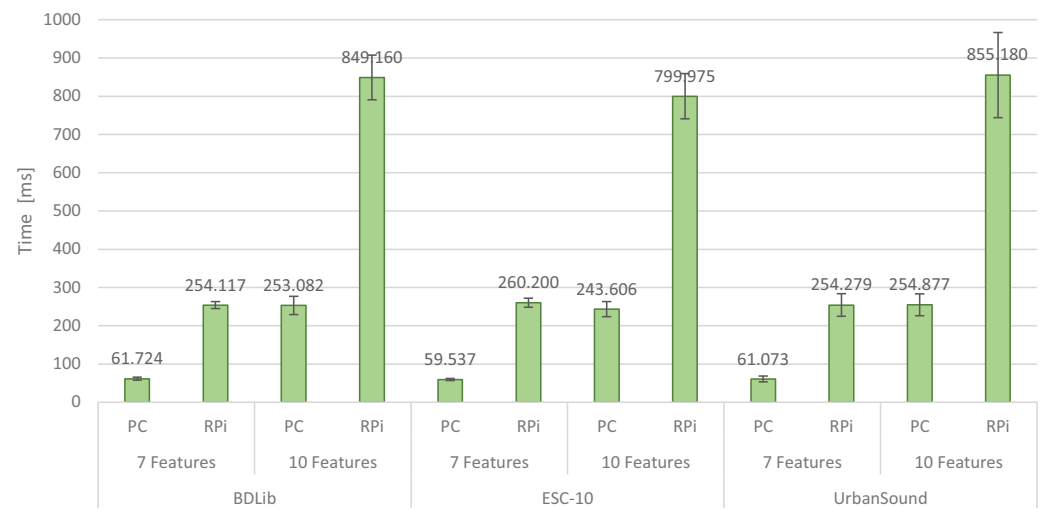
Figure 6. Experimental feature extraction timings measured in different platforms.

6. Evaluation of Classical Machine Learning for ESR: The MosAIc Approach

In this section the experimental results using different classical machine learning techniques are presented. The ultimate goal is to evaluate how classical ML techniques can be combined to achieve a higher accuracy. A novel combination of classifiers, called MosAIc, is proposed and evaluated for multiple datasets and platforms.

The experimental results presented in this section are divided into three parts: the classical approach, the one-vs-all classifiers and the novel MosAIc approach. Initially, the most common ML classifiers are evaluated for the BDlib, ESC-1 and UrbanSound datasets. The results are compared with the performance reported in the literature. As a second step, a one-vs-all approach is used to evaluate the potential of combining multiple classifiers. It is well-know that the accuracy decreases when increasing the number of categories. With the one-vs-all approach, the number of categories is reduced to a minimum in order to achieve the highest performance. The natural evolution of the one-vs-all approach is the MosAIc approach. The structure is described and the experimental accuracy achieved with this novel approach are presented. Finally, all solutions are evaluated on an embedded device. A solution for a faster MosAIc approach when embedded is also discussed.

6.1. The Classical Approach

Classical ML solutions for ESR involve the use of well-known classifiers such as those described in Section 3.3. These classifiers have been evaluated in [7] and the results were compared with those reported in the literature (Table 2). The experimental measurements are summarized in Tables 3–5 for the BDLib, ESC-10 and UrbanSound datasets respectively. Notice that the values depicted in Table 2 are accuracy, and not F_1 scores. Nonetheless, the accuracy values can be compared to the F_1-micro score. The evaluation for the BDLib dataset reflects that the k-NN classifier outperforms not only the original accuracy reported in the literature but also the other classifiers. Nonetheless, such a performance has a cost reflected in its high classification time. Something similar occurs when evaluating the classifiers for the ESC-10 dataset, where the k-NN classifier obtains the highest accuracy but it is also the most time demanding. The SVM classifier is the classical ML technique which offers higher accuracy when using the UrbanSound dataset. The best classifiers in terms of accuracy and timing will be selected for the MosAIc approach.

Table 2. Classifiers' accuracy per dataset reported in the literature (Source: [7]).

Classifier	BDLib [42]	ESC-10 [43]	UrbanSound [44]
k-NN	45.5%	66.7%	-
Naive Bayes	45.9%	-	-
ANN	54.0%	-	-
SVM	53.7%	67.5%	$\approx$70%
Decision Trees	-	-	$\approx$48%

Table 3. F_1 scores and classification time using the BDLib dataset with a classical approach and 10 features. The solution with the highest F_1 score is highlighted.

Classifier	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
SVM	0.002 $\pm$ 0.000	0.921	0.920	0.922
Naive Bayes	0.020 $\pm$ 0.005	0.578	0.554	0.559
k-NN	**0.232 $\pm$ 0.013**	**0.981**	**0.981**	**0.981**
ANN	0.003 $\pm$ 0.001	0.928	0.927	0.928
Decision Trees	0.005 $\pm$ 0.000	0.482	0.417	0.412

Table 4. F_1 scores and classification time using the ESC-10 dataset with a classical approach and 10 features. The solution with the highest F_1 score is highlighted.

Classifier	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
SVM	0.001 $\pm$ 0.000	0.844	0.841	0.843
Naive Bayes	0.016 $\pm$ 0.000	0.516	0.512	0.514
k-NN	**0.597 $\pm$ 0.023**	**0.876**	**0.872**	**0.873**
ANN	0.002 $\pm$ 0.001	0.821	0.820	0.821
Decision Trees	0.003 $\pm$ 0.001	0.451	0.385	0.389

Table 5. F_1 scores and classification time using the UrbanSound dataset with a classical approach and 10 features. The solution with the highest F_1 score is highlighted.

Classifier	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
SVM	0.001 $\pm$ 0.000	0.624	0.609	0.622
Naive Bayes	0.017 $\pm$ 0.000	0.250	0.251	0.252
k-NN	**2.135 $\pm$ 0.132**	**0.932**	**0.935**	**0.932**
ANN	0.002 $\pm$ 0.000	0.656	0.631	0.654
Decision Trees	0.001 $\pm$ 0.000	0.353	0.214	0.291

6.2. The One-vs-All Approach

One strategy to improve accuracy could be the reduction of the classes to be recognized. The minimal number of classes can be reduced to only two classes, which can be used to determine if an audio belongs to one associated class or not. This approach, called one-vs-all [45], is evaluated here. To set this approach, the features are extracted and labeled in a binary way: either "the-associated-class" or "not-the-associated-class". This is done separately for each class. The classifiers are then trained in a binary way. For instance, the classifiers are trained to distinguish the "the-associated-class" *dog bark* from the UrbanSound dataset and audio labelled as the "not-the-associated-class" *no dog bark*. The number of audio samples labelled "not-the-associated-class" will be always higher than "the-associated-class". In the case of the ESC-10 and BDLib datasets, this number is exactly nine times more. This creates imbalance in the training data, causing the classifier to actually learn to recognize "not the class" instead of "this class". Note that, due to its original imbalance, the UrbanSound dataset is the most sensible one to suffer this effect.

Figure 7 depicts the experimental results of F_1 score using the one-vs-all approach with the different ML techniques using the UrbanSound dataset. This dataset is selected due to the relatively low accuracy obtained in the previous experiments. As depicted in Figure 7, the accuracy of the classifiers using the one-vs-all approach increases in comparison to the classical approach. As a result, the one-vs-all approach offers a major improvement in accuracy, with the F_1 scores almost doubling for every ML technique.

Due to the fact that the classifiers have to use the same set of features as in the traditional approach, their classification time is not altered. The exception is Naive Bayes, for which the classification time is reduced to 0.011 ms, half of the classification time measured in the previous section. Although this approach is limited to a binary classification, it reflects how much the accuracy can increase when the number of classes are reduced.
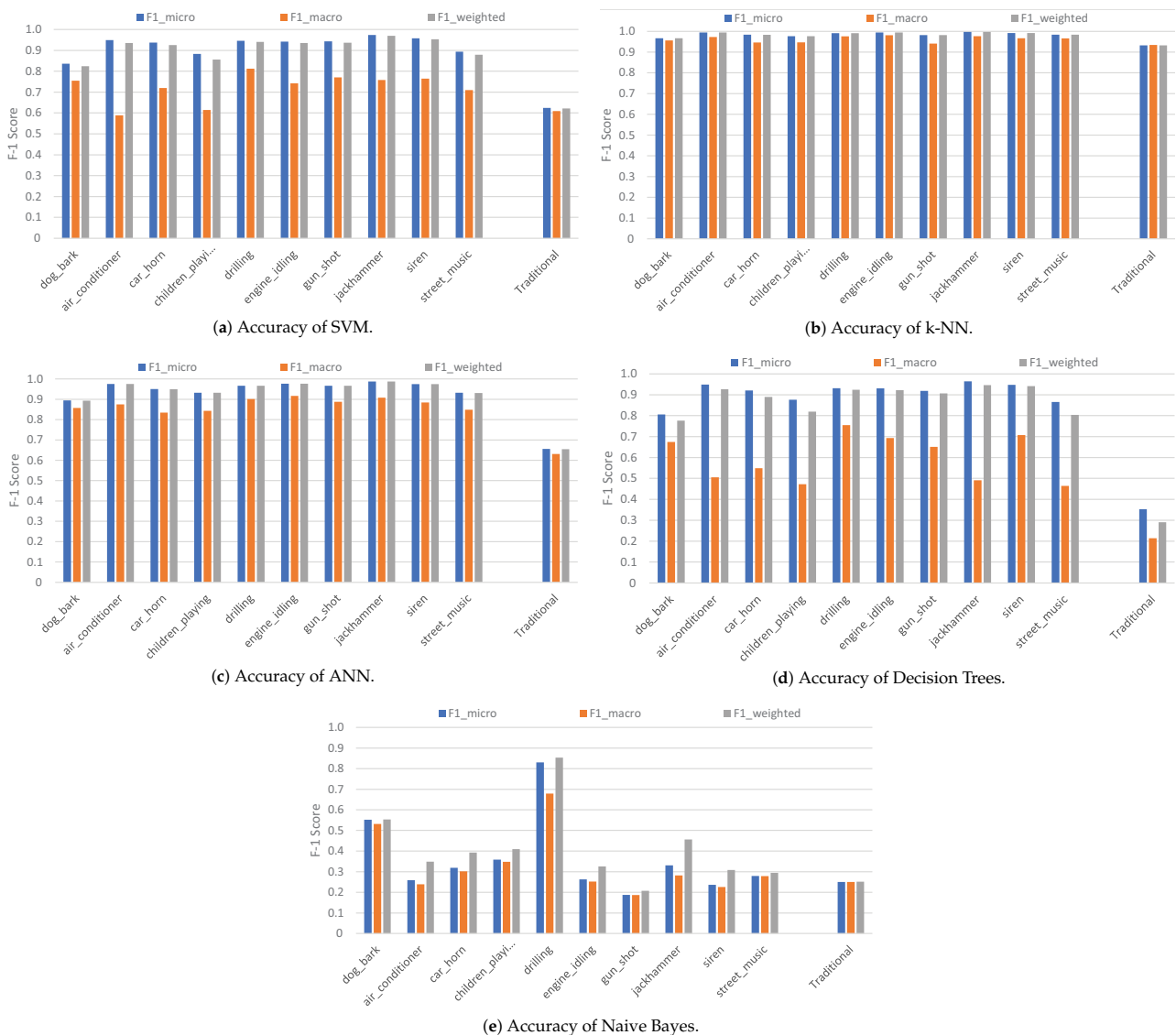

(**a**) Accuracy of SVM.

(**b**) Accuracy of k-NN.

(**c**) Accuracy of ANN.

(**d**) Accuracy of Decision Trees.

(**e**) Accuracy of Naive Bayes.

Figure 7. Accuracy of the classical classifiers for one-vs-all approach obtained using the UrbanSound dataset.

6.3. The MosAIc Approach

The idea behind the MosAIc approach is to improve classical ML techniques by combining multiple classifiers and create, in this way, a "mosaic" of classifiers (Figure 8). A one-vs-all approach uses one binary classifier per class, by training it with the samples of that class as positive samples and all other samples as negatives. If a dataset with ten classes is used for the training, ten binary classifiers are necessary for this approach. The

MosAIc approach combines these ten classifiers with a voting system. Nonetheless, one single type of classifier is not enough. The MosAIc approach uses three different type of ML classifiers selected based on the previous results Because the classification times are significantly lower than the feature extraction times, the accuracy is used as the parameter for the selection of ML techniques for the MosAIc approach. As a result, the SVM, k-NN and ANN techniques are the ML classifiers selected for the MosAIc approach. The configuration of the MosAIc approach depicted in Figure 8 has three groups of one-vs-all classifiers trained with SVM, k-NN and ANN techniques, with a one-vs-all classifier per class in every group.
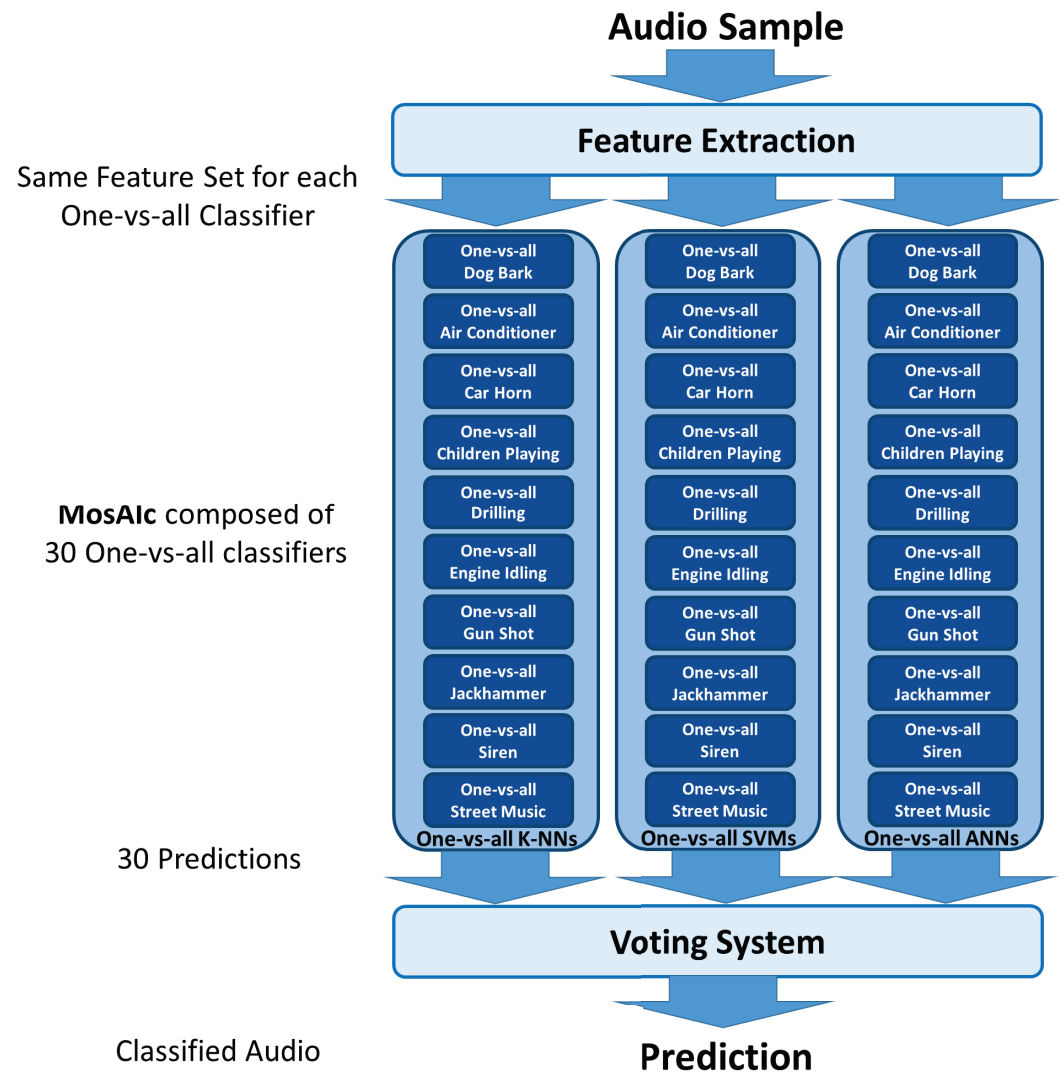

Figure 8. A schematic representation of the MosAIc approach for UrbanSound.

The MosAIc approach is the natural evolution of the one-vs-all approach. When an audio sample is classified, this results in three predictions per class. The voting system will then consider the prediction that has at least two votes out of the three as the final prediction of the three binary classifiers associated with one same class. This means that the classification of an audio sample using the MosAIc approach can result in more than one prediction. An additional benefit of this approach is that it could identify multiple classes in one audio sample.

Other than the improved accuracy obtained with this approach, a reason to use it is the fact that it optimizes the usage of the time available during the feature extraction, assuming the feature extraction and the classification are done in parallel. Feature extraction can indeed be up to one thousand times slower than the classification.

For the evaluation of the MosAIc approach, the same data are used to train all the classifiers associated with the different classes. The labels of the feature groups is the only change between the classes. With the one-vs-all approach, to select test data, a random split is done just before the training, 80% for the training, 20% for the testing. Those 20% are then used to measure the accuracy of the trained classifier. However in the case of the MosAIc approach, the same audio samples are being used for all of the classes, but they are being split differently each time. Therefore, to assure that no test audio samples are used to train the classifiers, the test data are generated from the same audio files, but with the features from the audio between 4 s and 8 s. For this experiment, 30 binary classifiers work together, three per class, one SVM, one k-NN and one ANN. The voting system is applied using the 30 predictions, isolating the majority of every group of three classifiers, ten predictions remain per test audio, one per class. Those ten predictions represent the classification of an audio sample. In the ideal case, if there is only one class to identify, nine of those ten predictions are "not-the-associated-class" and one is "the associated class". As part of the evaluation, the total classification time of MosAIc is calculated by adding up the classification times of the 30 binary classifiers.

As explained in Section 3, the F_1 score is the metric used to evaluate the MosAIc approach. It is a metric calculated with precision and recall, two metrics based on the true positives, the false positives and the false negatives. It is important to use the F_1 score, a metric that does not take the true negatives into account, to evaluate the MosAIc approach, because in the ideal case, 90% of the results of this experiment should be true negatives, which would skew the accuracy if calculated differently.

The different F_1 scores obtained with this experiment can be found in Table 6. This table highlights that F_1 scores around 0.96 can be achieved combining those 30 binary classifiers using classical ML techniques. This is the case when using the BDLib or ESC-10 datasets, but not UrbanSound where the accuracy oddly drops back to the range of the classical approach. As expected, the cost of combining 30 classifiers is reflected in the classification time, which significantly increases. Nonetheless, this increment is dominated by the k-NN ML technique, which demands significantly more time than the other techniques as shown in previous experiments. The use of a different ML technique than k-NN, with a similar accuracy but with a lower classification time, would lead to a faster MosAIc approach.

Table 6. Performance evaluation of MosAIc using 10 features.

Datasets	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
BDLib	3.249 ± 0.099	0.973	0.932	0.974
ESC-10	6.579 ± 0.057	0.964	0.871	0.959
UrbanSound	24.432 ± 1.029	0.574	0.506	0.572

6.4. Evaluation on an Embedded Platform

A high-end embedded platform, a RPi 4B+, is used to evaluate the different approaches. The selection of this embedded platform is motivated by the fact that the experiments performed on a PC can be replicated on this embedded platform without major modifications. This would not be possible using other embedded platforms since it would demand the use of proprietary tools, ML support and different ML implementations if using open source alternatives (e.g., different libraries) or simply not be supported, which is the case for several classical ML techniques.

The main concern when porting the mentioned approaches are the accuracy and the execution time. On the one hand, the accuracy is expected not to change, as already demonstrated in [7]. On the other hand, the overall execution time is expected to increase significantly. This last parameter is, in fact, crucial for real-time recognition and, partially relevant in the selection of the audio frame size.

6.4.1. Evaluation of the Classical Approach on an Embedded Platform

The evaluation of the standalone classical ML techniques for the considered datasets are summarized in Tables 7–9. The values in bold correspond to the classifiers with the highest accuracy. As expected, the achieved accuracy has minor differences when compared to the values obtained using the PC. The classification time on the other hand, is three to four times higher, mainly due to the limited computing power of the embedded system.

Table 7. F_1 scores and classification time using the BDLib dataset with a classical approach and 10 features on a RPi. The solution with the highest F_1 score is highlighted.

Datasets	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
SVM	0.009 ± 0.001	0.927	0.928	0.928
Naive Bayes	0.064 ± 0.004	0.560	0.542	0.544
k-NN	$\mathbf{0.674 \pm 0.034}$	**0.981**	**0.981**	**0.981**
ANN	0.013 ± 0.000	0.931	0.930	0.931
Decision Trees	0.013 ± 0.001	0.499	0.428	0.431

Table 8. F_1 scores and classification time using the ESC-10 dataset with a classical approach and 10 features on a RPi. The solution with the highest F_1 score is highlighted.

Datasets	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
SVM	0.011 ± 0.000	0.836	0.834	0.835
Naive Bayes	0.092 ± 0.043	0.518	0.515	0.516
k-NN	$\mathbf{1.526 \pm 0.038}$	**0.872**	**0.870**	**0.870**
ANN	0.014 ± 0.001	0.826	0.825	0.826
Decision Trees	0.006 ± 0.000	0.447	0.383	0.381

Table 9. F_1 scores and classification time using the UrbanSound dataset with a classical approach and 10 features on a RPi. The solution with the highest F_1 score is highlighted.

Datasets	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
SVM	0.013 ± 0.002	0.619	0.603	0.617
Naive Bayes	0.064 ± 0.023	0.252	0.251	0.250
k-NN	$\mathbf{5.538 \pm 0.176}$	**0.934**	**0.935**	**0.934**
ANN	0.015 ± 0.002	0.653	0.629	0.652
Decision Trees	0.003 ± 0.000	0.354	0.210	0.289

6.4.2. Evaluation of the MosAIc Approach on an Embedded Platform

Similarly, the MosAIc approach is also evaluated on the embedded platform. Although the accuracy is expected to be similar to the values on the PC, the execution time can become prohibitive when porting the approach to an embedded device. Table 10 shows the accuracy and classification times obtained repeating the MosAIc approach experiment on the RPi with ten features. Here again the accuracy remains the same as the one measured on the PC, while there is an increment in the classification time. However, unlike with the classical ML standalone, where timings increase three to four times on RPi, the MosAIc classification time is only two to three times higher.

Table 10. F_1 scores and classification times of the different datasets with the MosAIc approach using 10 features on a RPi.

Datasets	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
BDLib	8.349 ± 0.008	0.968	0.917	0.969
ESC-10	14.512 ± 0.018	0.961	0.861	0.955
UrbanSound	71.781 ± 1.236	0.575	0.511	0.573

6.5. Exploiting the Feature Selection

The overall classification time using the MosAIc approach on the embedded device is significantly high, which can limit the support for real-time sound recognition on such a platform. One alternative to accelerate the MosAIc approach would be to replace the k-NN classifier by a faster ML classifier. Although it would reduce the overall time, its impact would be limited to the classification time. A more ambitious optimization can be considered. Based on the experimental values depicted in Figure 6, the classification time is negligible when compared to the feature extraction time. Moreover, it is also reasonable to expect that the classification time can be reduced if the number of features decreases.

Figure 9 depicts the classification time of the MosAIc approach on the PC and on the RPi when the number of features is reduced. As is observed, the classification time when using only seven features drastically decreases. In fact, although the classification time does not change using the UrbanSound dataset, it becomes four times lower using the BDLib dataset and seven times lower using ESC-10. Such a time reduction is especially interesting for the RPi, achieving lower classification time using seven features than running on the PC using ten features.
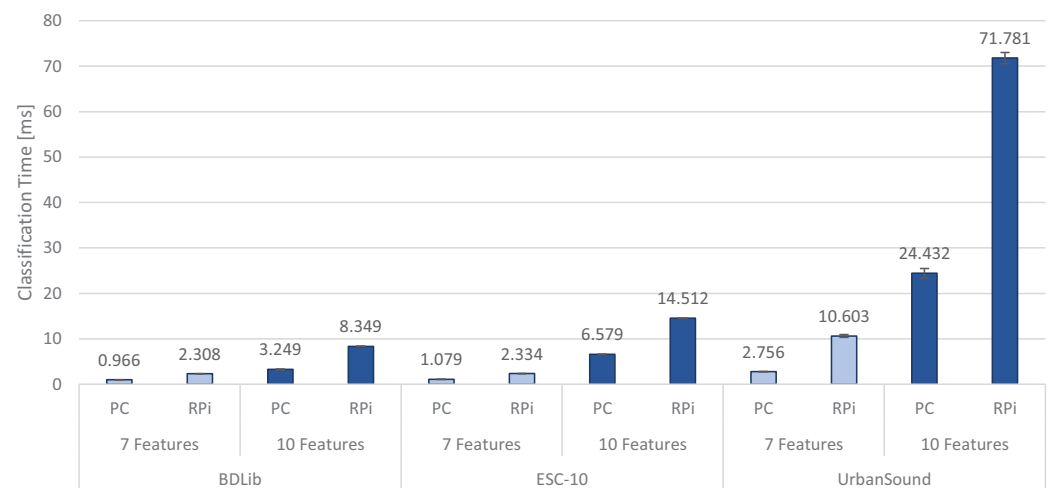

Figure 9. Classification time of the MosAIc approach based on the feature sets and the platforms.

Perhaps, the most interesting result is related to the accuracy. Figure 10 shows how the reduction in the number of features does not lead to accuracy reduction. This fact applies for all datasets and platforms. The feature evaluation depicted in Figure 4 provides insights about the reasons. Through the evaluation of the relevance of the features, one can observe how they do not contribute equally in terms of information gain. This is especially evident for the feature evaluation done for the UrbanSound dataset. Therefore, when the less relevant features are discarded, the accuracy does not need to drop necessarily. It is therefore important to select the most relevant features according to the dataset used to achieve the highest accuracy while reducing the overall feature and classification time.
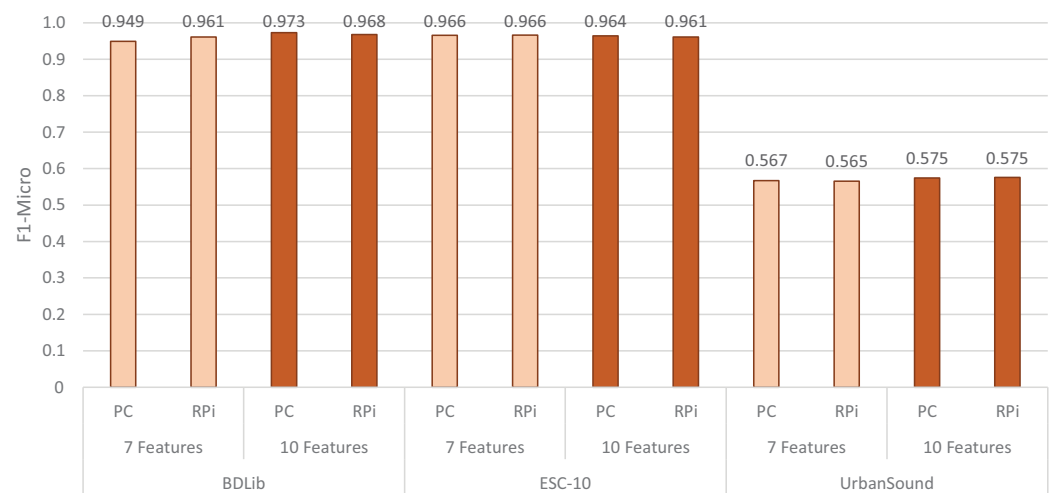
Figure 10. Accuracy (F_1 *micro score*) of the MosAIc approach based on the feature sets and the platforms.

7. Evaluation of Deep Learning for ESR: A Convolutional Neural Network Approach

The deep CNN has experienced advancement through its use in computer vision, recognition, and in language modeling among other fields. It is proven that the CNN-based architecture is more efficient than the conventional methods in various classification tasks [46]. Its use for automatic sound event recognition has lead to very good results in recent years. Using CNNs for the classification of distinct audible sounds had a remarkable increase in the last decade [47] and many researchers implemented their sound classification models by using different techniques on CNNs [48,49]. In order to properly evaluate the MosAIc approach against the state-of-the-art, a novel lightweight CNN is proposed and evaluated using the same features for the BDlib, ESC-10 and UrbanSound datasets.

7.1. Description of the Model

Our proposed lightweight CNN model depicted in Figure 11 is designed to use the same set of features as the MosAIc in order to provide a fair comparison in terms of accuracy and execution time. Nonetheless, that fact implies the use of Conv1D layers. Such a Conv1D CNNs in audio processing present one important challenge which is the fact that the length of the input sample must be fixed while the sound captured from the environment may have a variable duration. Therefore, it is necessary to adapt a CNN to be used with audio signals of different lengths. Splitting the audio signal into several frames of fixed length using a sliding window of appropriate width is a way to circumvent this constraint imposed by the CNN input layer. In the proposed approach, an audio frame of variable width is used to refurbish the audio signal to the 1D CNN input layer. Furthermore, successive audio frames have a 50% of overlapping, which aim is to maximize the use of information while preserving the frame's independence.
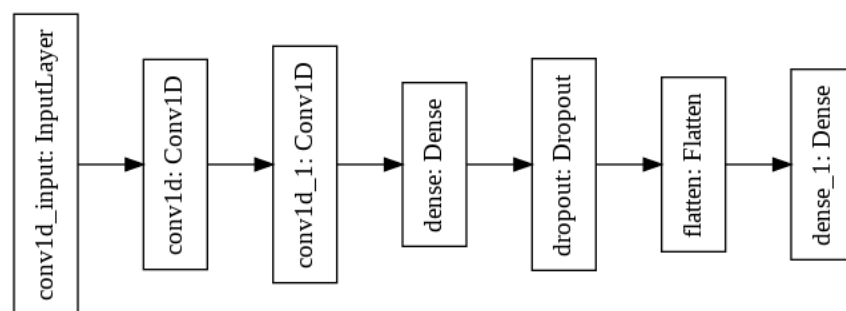

Figure 11. Architecture and structure of our proposed CNN model.

In this experiment, the remaining crucial parameters/mechanisms involved are: the usage of the Adam optimizer, the total number of epochs is 100 for each analysis, the batch size is 32 uniformly, the L2 norm regularizer (0.001) is used, the Rectified Linear Unit (ReLU) activation function is used for the first three layers, and the Softmax activation function is used for the last layer of each model. The characteristics of our proposed CNN architecture are summarized in Table 11. The architecture is composed of the following layers:

- **L1**: The first layer contains 12 filters with a kernel size equal to five. The regularizer L2 norm with a value of 0.001 is used. The activation function utilized is ReLU.
- **L2**: The second layer contains 28 filters with a kernel size equal to five. The L2 norm regularizer is also used. The padding for all the feature extraction involves in this layer is "valid" and ReLU as the activation function.
- **L3**: The third layer is the 1st dense layer that consists of 30 hidden units with the ReLU activation function. A dropout rate of 0.5 is used to avoid over-fitting.
- **L4**: The last layer is the 2nd dense layer. It consists of the output units. These are equal to the number of classes used in the dataset. The Softmax activation function is used in the last layer.

Table 11. Architecture of our proposed CNN model and parameter information used in each layer.

Type of Layer	Output Shape	Number of Parameters
conv1d (Conv1D)	(None, 166, 12)	72
conv1d_1 (Conv1D)	(None, 162, 28)	1708
dense (Dense)	(None, 162, 30)	870
dropout (Dropout)	(None, 162, 30)	0
flatten (Flatten)	(None, 4860)	0
dense_1 (Dense)	(None, 10)	48,610
Total :		51,260
Trainable :		51,260
Non-trainable :		0

7.2. State-of-the-Art CNNs for ESR

Several DL approaches for ESR have been proposed in recent years. The performance of the proposed method is compared against these state-of-the-art solutions for the same dataset. Since most recent work has been evaluated on the ESC-10 dataset, this comparison includes results obtained for this dataset. Nonetheless, the results for the BDLib and the UrbanSound datasets are also included in the following section.

The classification accuracy of our proposed CNN model compared with the best recent solutions from the literature is shown in Table 12. The highest accuracy is achieved by the model proposed in [47], which presents a CNN without max-pooling function and using Log-Mel audio feature extraction (CNN-Model-2 (No-maxpooling) + Log-Mel + augmentation). In [50], the application of the Masked Conditional Neural Networks for sound classification (MCLNN) to the problem of music genre classification, and ESR is presented. Both proposed models achieve high accuracy, but they are large models demanding prohibitive memory requirements for embedded platforms. Their inference time is also expected to be higher than the proposed solution. Slightly smaller models are proposed in [8], where the authors present a compact and effective model capable of characterizing different urban sounds based on deep and handcrafted features combining two models.

The proposed model improves the classification accuracy to 90.71% with only 51 K parameters, around 60 times lower than the number of parameters other methods. Such a model is expected to be not only significantly faster than the state-of-the-art solutions but also more suitable for embedded devices due to its lower memory demand.

Table 12. Recent CNNs-based sound classifiers evaluated with ESC-10 dataset.

| References	Year	Accuracy [%]	Number of Parameters
[47]	2020	94.94	3 M
[50]	2020	85.50	3 M
[8]	2021	83.50	1.48 M
[8]	2021	80.25	1.48 M
Proposed	**2021**	**90.71**	**51 K**

7.3. Experimental Results

The proposed model for environmental sound classification is evaluated with a K-fold cross-validation scheme (K = 5) on the ESC-10, BDLib and UrbanSound datasets. The single training fold is used as the validation set for parameter tuning, following the approach in [51]. A cross validation is performed five times over the three datasets to evaluate the CNN's performance. The results of this performance evaluation are shown in Table 13. The classification time of the proposed CNN demands is relatively low, ranging from 0.156 ms to 0.413 ms depending on the dataset. The F_1 scores is relatively high for the BDLiB dataset and the ESC-10 dataset, while lower for the Sound dataset. These results are consistent with the accuracies measured using classical ML in the previous section.

Table 13. Evaluation of our proposed CNN model for the different datasets.

Datasets	Classification Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
BDLiB	0.413 $\pm$ 0.003	0.972	0.971	0.971
ESC-10	0.254 $\pm$ 0.004	0.907	0.907	0.907
UrbanSound	0.156 $\pm$ 0.001	0.830	0.837	0.830

The graphs in Figure 12 show the accuracy (F_1 *micro*) and validation values according to the number of epochs in the training phase of the CNN model. The accuracy of ESC-10 is illustrated in Figure 12a, where after the 30th epoch, the model at the training stage reaches the optimum performance approximately. However, the accuracy of BDLiB as shown in Figure 12b reaches the optimum performance after the 50th epoch and the Accuracy of UrbanSound presented in Figure 12c does not reach these optimal performances until the 80th epoch.

(**a**) *ESC-10*　　　　　　　　　　(**b**) *BDLiB*　　　　　　　　　　(**c**) *UrbanSound*
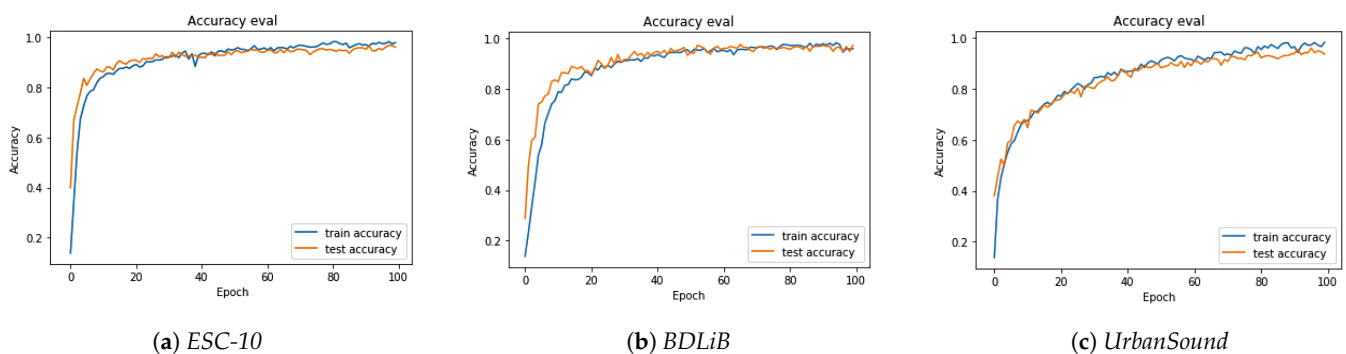
Figure 12. Evaluation of the accuracy (F_1 *micro*) for all datasets using our proposed CNN model.

7.4. Experimental Results on an Embedded Platform

For the evaluation on the embedded platform, the CNN needs to be converted to TensorFlow Lite [52], a framework designed to deploy ML algorithms on embedded devices. TensorFlow Lite (TFLite) converts an already trained TensorFlow model to a more lightweight version, suitable for embedded deployment. Table 14 shows the performance of the full-precision TFLite CNN on the RPi 4B+. Notice that the accuracy does not

change respect to the measurements on the PC. An interesting result is the inference time. Although it would be expected to increase due to being executed on a limited embedded device, the experimental measurements show that the inference time slightly change. This is justified by the fact that high-end embedded devices such as the RPi 4B+ have been optimized in the last years to support DL-based models. As a result, there is no time or accuracy cost when porting CNN models to such an embedded devices.

Table 14. RPi with TFLite without quantization.

Datasets	Inference Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
BDLib	0.381	0.976	0.976	0.976
ESC-10	0.386	0.911	0.905	0.911
UrbanSound	0.369	0.836	0.838	0.837

The converted model can be a full-precision, 32-bit floating point model, or it can be quantized to 8-bit integers using post-training quantization. In fact, many limited embedded devices or hardware accelerators (e.g., TensorFlow Processing Units) only support quantized models. Such a quantized models have the advantage to be smaller and potentially faster than their non-quantized counterparts. However, during the quantization process, some information is inevitably lost that can influence the resulting accuracy. Table 15 shows the performance when our proposed model is quantized to 8-bit integers. Although there is a minor accuracy drop after quantization, the speedup is a minor acceleration compared to the non-quantized TFLite models.

Table 15. RPi with TFLite and post training quantization (using 8bit).

Datasets	Inference Time [ms]	F_1 Micro	F_1 Macro	F_1 Weighted
BDLib	0.275	0.970	0.970	0.970
ESC-10	0.275	0.909	0.903	0.909
UrbanSound	0.275	0.830	0.831	0.831

8. Discussion

The MosAIc approach demonstrates how the combination of different classical classifiers can lead to high accuracy for environmental sound classification when compared to an equivalent DL approach such as the one proposed in Section 7. Figure 13 summarizes the achievable accuracies for both approaches for two different platforms. Notice the strong dependence on the dataset. While the MosAIc approach achieves the highest accuracy for ESC-10 dataset, the novel lightweight CNN outperforms the classical ML approach for the UrbanSound dataset. Nonetheless, both perform similarly for the BDLib dataset.

The cost to achieve such an accuracy with classical ML is reflected in the classification time. Figure 14 depicts the classification time and the inference time for the classical ML and the DL approach respectively. The time cost of the MosAIc approach to achieve such an accuracy is significantly high, especially when considering embedded platforms, reaching up to 28 times higher than our proposed lightweight CNN. Besides such a timing difference, the overall execution time for audio recognition, the feature extraction time plus the classification or the inference time, is still dominated by the feature extraction time as shown in Figure 6. The lowest feature extraction time is several times higher than the maximum MosAIc classification time.
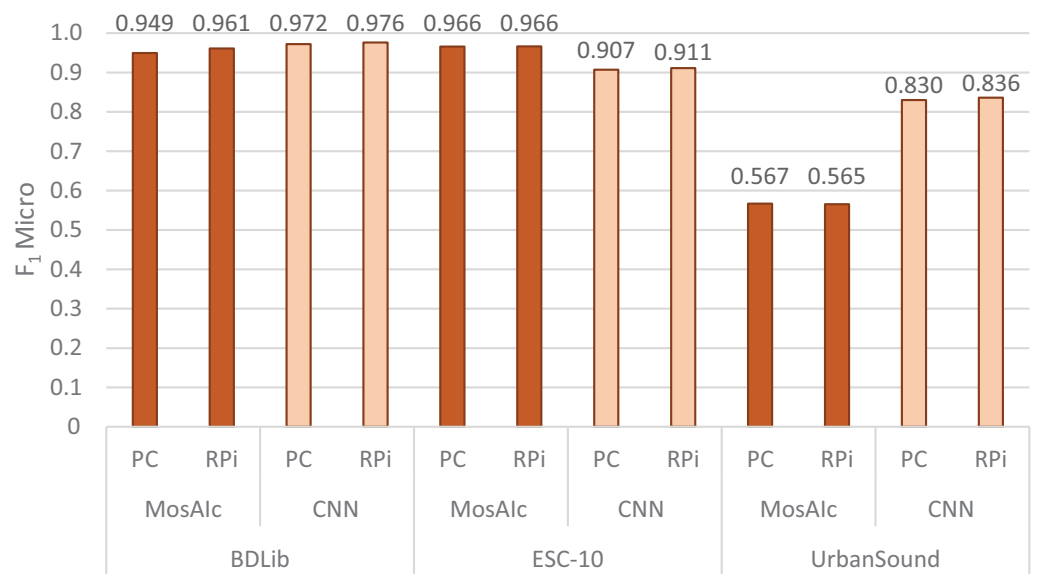
Figure 13. Accuracy (F_1 *micro score*) of the MosAIc approach vs. our proposed lightweight CNN based on the feature sets and the platforms.

The overall execution time has a direct impact in the selection of the audio frame size when windowing. Assuming that the execution of the audio recognition is done in streaming and an overlap of 50% between each audio frame due to windowing, based on the experimental timings, the feature extraction time defines the minimum windowing. In fact, based on the timings in Figures 6 and 14, the feature extraction time for ten features on the RPi is higher than an audio frame of 500 ms, which is the condition when considering audio frames as short as 1 s. As a result, short audio frame sizes are defined by the feature extraction time and the selected platform. Nonetheless, from the embedded perspective, larger audio frame size can lead to power savings since the feature extraction time and the classification or inference time would be lower than the audio frame size, enabling the use of power saving modes present on embedded devices.
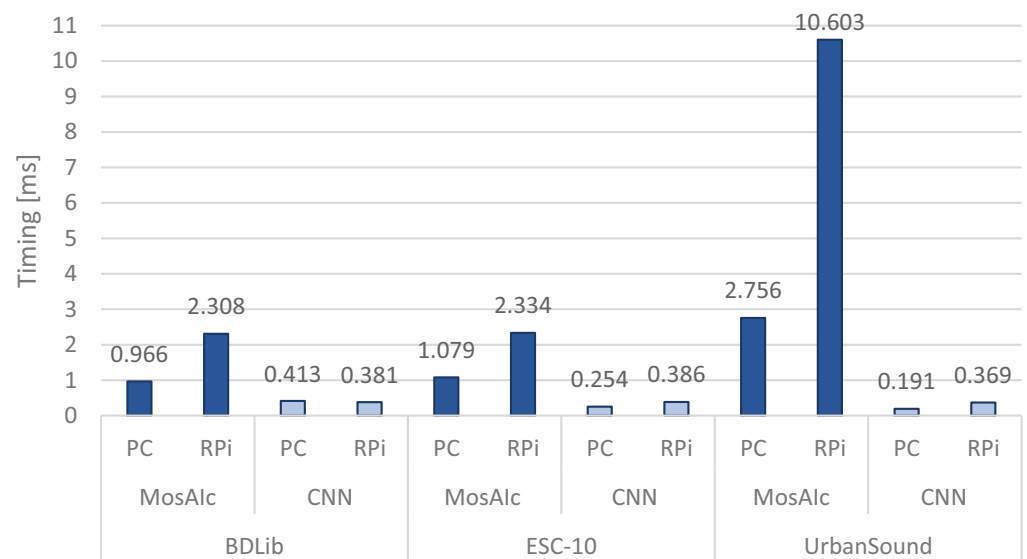

Figure 14. Classification time of the MosAIc approach vs. Inference time of our proposed lightweight CNN based on the feature sets and the platforms.

9. Conclusions

The proposed MosAIc approach demonstrates that classical machine learning can be competitive in terms of accuracy for environmental sound recognition, and is capable of outperforming DL approaches for certain datasets. The cost, however, is a higher classification time. Although such a time cost is affordable on non-embedded platforms, it drastically scales when considering computational constraints for devices such as a RPi. As a solution, our lightweight CNN model drastically reduces the number of parameters of the model, with an inference time in orders of microseconds and is suitable for embedded devices due to its resources demand. Nonetheless, our experiments have shown that the overall execution time for the audio classification is dominated by the feature extraction time, which can determine the length of the audio frame when applying windowing. As a result, the classification of the inference time of the classical ML approach or the DL approach becomes negligible when compared to the feature extraction time.

References

1. López, J.M.; Alonso, J.; Asensio, C.; Pavón, I.; Gascó, L.; de Arcas, G. A Digital Signal Processor Based Acoustic Sensor for Outdoor Noise Monitoring in Smart Cities. *Sensors* **2020**, *20*, 605. [CrossRef] [PubMed]
2. Tsalera, E.; Papadakis, A.; Samarakou, M. Monitoring, profiling and classification of urban environmental noise using sound characteristics and the KNN algorithm. *Energy Rep.* **2020**, *6*, 223–230. [CrossRef]
3. Paris Testing Noise Radar System That Can Identify and Ticket Loud Cars. Available online: https://www.techtimes.com/articles/245203/20190902/paris-testing-noise-radar-system-that-can-identify-and-ticket-loud-cars.htm (accessed on 1 June 2021).
4. Ozkan, Y.; Barkana, B.D. Forensic Audio Analysis and Event Recognition for Smart Surveillance Systems. In Proceedings of the 2019 IEEE International Symposium on Technologies for Homeland Security (HST), Woburn, MA, USA, 5–6 November 2019; pp. 1–6. [CrossRef]
5. Nagy, K.; Cinkler, T.; Simon, C.; Vida, R. Internet of Birds (IoB): Song Based Bird Sensing via Machine Learning in the Cloud: How to sense, identify, classify birds based on their songs? In Proceedings of the 2020 IEEE SENSORS, Rotterdam, The Netherlands, 25–28 October 2020; pp. 1–4. [CrossRef]
6. Gradolewski, D.; Dziak, D.; Martynow, M.; Kaniecki, D.; Szurlej-Kielanska, A.; Jaworski, A.; Kulesza, W.J. Comprehensive Bird Preservation at Wind Farms. *Sensors* **2021**, *21*, 267. [CrossRef] [PubMed]
7. da Silva, B.; Happi, A.W.; Braeken, A.; Touhafi, A. Evaluation of classical machine learning techniques towards urban sound recognition on embedded systems. *Appl. Sci.* **2019**, *9*, 3885. [CrossRef]
8. Luz, J.S.; Oliveira, M.C.; Araújo, F.H.; Magalhães, D.M. Ensemble of handcrafted and deep features for urban sound classification. *Appl. Acoust.* **2021**, *175*, 107819. [CrossRef]
9. Huzaifah, M. Comparison of time-frequency representations for environmental sound classification using convolutional neural networks. *arXiv* **2017**, arXiv:1706.07156.
10. Papadimitriou, I.; Vafeiadis, A.; Lalas, A.; Votis, K.; Tzovaras, D. Audio-Based Event Detection at Different SNR Settings Using Two-Dimensional Spectrogram Magnitude Representations. *Electronics* **2020**, *9*, 1593. [CrossRef]
11. Zinemanas, P.; Cancela, P.; Rocamora, M. End-to-end convolutional neural networks for sound event detection in urban environments. In Proceedings of the 2019 24th Conference of Open Innovations Association (FRUCT), Moscow, Russia, 8–12 April 2019; pp. 533–539.
12. Esmaeilpour, M.; Cardinal, P.; Koerich, A.L. A robust approach for securing audio classification against adversarial attacks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 2147–2159. [CrossRef]

13. Stork, J.A.; Spinello, L.; Silva, J.; Arras, K.O. Audio-based human activity recognition using non-markovian ensemble voting. In Proceedings of the 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, Paris, France, 9–13 September 2012; pp. 509–514.

14. Tokozume, Y.; Harada, T. Learning environmental sounds with end-to-end convolutional neural network. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 2721–2725.

15. Bonet-Solà, D.; Alsina-Pagès, R.M. A Comparative Survey of Feature Extraction and Machine Learning Methods in Diverse Acoustic Environments. *Sensors* **2021**, *21*, 1274. [CrossRef]

16. Sigtia, S.; Stark, A.M.; Krstulović, S.; Plumbley, M.D. Automatic environmental sound recognition: Performance versus computational cost. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 2096–2107. [CrossRef]

17. Bonfigli, R.; Ferroni, G.; Principi, E.; Squartini, S.; Piazza, F. A real-time implementation of an acoustic novelty detector on the BeagleBoard-xM. In Proceedings of the 2014 6th European Embedded Design in Education and Research Conference (EDERC), Milan, Italy, 11–12 September 2014; pp. 307–311.

18. Alsouda, Y.; Pllana, S.; Kurti, A. IoT-based urban noise identification using machine learning: Performance of SVM, KNN, bagging, and random Forest. In Proceedings of the International Conference on Omni-Layer Intelligent Systems, Crete, Greece, 5–7 May 2019; pp. 62–67.

19. Naccari, F.; Guarneri, I.; Curti, S.; Savi, A.A. Embedded Acoustic Scene Classification for Low Power Microcontroller Devices. In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), Tokyo, Japan, 2–3 November 2020; pp. 105–109.

20. Zhan, Y.; Kuroda, T. Wearable sensor-based human activity recognition from environmental background sounds. *J. Ambient Intell. Humaniz. Comput.* **2014**, *5*, 77–89. [CrossRef]

21. Mendoza, J.M.; Tan, V.; Fuentes, V.; Perez, G.; Tiglao, N.M. Audio Event Detection Using Wireless Sensor Networks Based on Deep Learning. In *International Wireless Internet Conference*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 105–115.

22. AbeBer, J.; Gotze, M.; Kuhnlenz, S.; Grafe, R.; Kuhn, C.; ClauB, T.; Lukashevich, H. A distributed sensor network for monitoring noise level and noise sources in urban environments. In Proceedings of the 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 6–8 August 2018; pp. 318–324.

23. Siamwala, A.; Lochhead, Z.; Abdulla, W. Environmental Noise Monitoring Using Distributed IoT Sensor Nodes. In Proceedings of the 2019 International Conference on Electronics, Information, and Communication (ICEIC), Auckland, New Zealand, 22–25 January 2019; pp. 1–10.

24. LibROSA 0.6.3. Available online: https://librosa.org/doc/latest/index.html (accessed on 22 February 2021).

25. Carey, M.J.; Parris, E.S.; Lloyd-Thomas, H. A comparison of features for speech, music discrimination. In Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP99 (Cat. No. 99CH36258), Phoenix, AZ, USA, 15–19 March 1999; Volume 1, pp. 149–152.

26. El-Maleh, K.; Klein, M.; Petrucci, G.; Kabal, P. Speech/music discrimination for multimedia applications. In Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing, (Cat. No. 00CH37100), Istanbul, Turkey, 5–9 June 2000; Volume 4, pp. 2445–2448.

27. Sethares, W.A.; Morris, R.D.; Sethares, J.C. Beat tracking of musical performances using low-level audio features. *IEEE Trans. Speech Audio Process.* **2005**, *13*, 275–285. [CrossRef]

28. Sharma, G.; Umapathy, K.; Krishnan, S. Trends in audio signal feature extraction methods. *Appl. Acoust.* **2020**, *158*, 107020. [CrossRef]

29. Jiang, D.N.; Lu, L.; Zhang, H.J.; Tao, J.H.; Cai, L.H. Music type classification by spectral contrast feature. In Proceedings of the IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland, 26–29 August 2002; Volume 1, pp. 113–116.

30. Aslam, M.A.; Sarwar, M.U.; Hanif, M.K.; Talib, R.; Khalid, U. Acoustic classification using deep learning. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 153–159. [CrossRef]

31. Cohn, R. Introduction to neo-riemannian theory: A survey and a historical perspective. *J. Music Theory* **1998**, 167–180. doi:10.2307/843871. [CrossRef]

32. Hyer, B. Reimag (in) ing Riemann. *J. Music Theory* **1995**, *39*, 101–138. [CrossRef]

33. Lewin, D. *Generalized Musical Intervals and Transformations*; Oxford University Press: New York, NY, USA, 2011.

34. Docs.Nvidia. Available online: https://docs.nvidia.com/deeplearning/dali/user-guide/docs/examples/audio_processing/spectrogram.html (accessed on 22 February 2021).

35. Goutte, C.; Gaussier, E. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In *European Conference on Information Retrieval*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 345–359.

36. Lipton, Z.C.; Elkan, C.; Narayanaswamy, B. Thresholding classifiers to maximize F1 score. *Stat* **2014**, *1050*, 14.

37. Fujino, A.; Isozaki, H.; Suzuki, J. Multi-label text categorization with model combination based on f1-score maximization. In Proceedings of the Third International Joint Conference on Natural Language Processing, Hyderabad, India, 8–10 January 2008; Volume II.

38. Evaluating Multi-Class Classifier. Available online: https://medium.com/apprentice-journal/evaluating-multi-class-classifiers-12b2946e755b (accessed on 18 February 2021).

39. Performance Measures for Multi-Class Problems. Available online: https://www.datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems/ (accessed on 18 February 2021).

40. Harb, H.M.; Moustafa, M.A. Selecting Optimal Subset of Features for Student Performance Model. *Int. J. Comput. Sci. Issues* **2012**, *9*, 253–262.

41. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *ACM SIGKDD Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]

42. Bountourakis, V.; Vrysis, L.; Papanikolaou, G. Machine learning algorithms for environmental sound recognition: Towards soundscape semantics. In Proceedings of the Audio Mostly 2015 on Interaction with Sound, Thessaloniki, Greece, 7–9 October 2015; p. 5.

43. Piczak, K.J. ESC: Dataset for environmental sound classification. In Proceedings of the 23rd ACM International Conference on Multimedia, Brisbane, Australia, 26–30 October 2015; pp. 1015–1018.

44. Salamon, J.; Jacoby, C.; Bello, J.P. A dataset and taxonomy for urban sound research. In Proceedings of the 22nd ACM International Conference on Multimedia, San Francisco, CA, USA, 15–20 October 2014; pp. 1041–1044.

45. Rifkin, R.; Klautau, A. In defense of one-vs-all classification. *J. Mach. Learn. Res.* **2004**, *5*, 101–141.

46. Ahmed, M.; Robin, T.I.; Shafin, A.A. Automatic Environmental Sound Recognition (AESR) Using Convolutional Neural Network. *Int. J. Mod. Educ. Comput. Sci.* **2020**, *12*, 5. [CrossRef]

47. Mushtaq, Z.; Su, S.F. Environmental sound classification using a regularized deep convolutional neural network with data augmentation. *Appl. Acoust.* **2020**, *167*, 107389. [CrossRef]

48. Valero, X.; Alias, F. Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification. *IEEE Trans. Multimed.* **2012**, *14*, 1684–1689. [CrossRef]

49. Cotton, C.V.; Ellis, D.P. Spectral vs. spectro-temporal features for acoustic event detection. In Proceedings of the 2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 16–19 October 2011; pp. 69–72.

50. Medhat, F.; Chesmore, D.; Robinson, J. Masked Conditional Neural Networks for sound classification. *Appl. Soft Comput.* **2020**, *90*, 106073. [CrossRef]

51. Salamon, J.; Bello, J.P. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Process. Lett.* **2017**, *24*, 279–283. [CrossRef]

52. TensorFlow Lite. Available online: https://www.tensorflow.org/lite/guide (accessed on 3 June 2021).

53. Brandalero, M.; Ali, M.; Le Jeune, L.; Hernandez, H.G.M.; Veleski, M.; da Silva, B.; Lemeire, J.; Van Beeck, K.; Touhafi, A.; Goedemé, T.; et al. AITIA: Embedded AI Techniques for Embedded Industrial Applications. In Proceedings of the 2020 International Conference on Omni-Layer Intelligent Systems (COINS), Barcelona, Spain, 31 August–2 September 2020; pp. 1–7. [CrossRef]