Enhanced Sound Recognition and Classification Through Spectrogram Analysis, MEMS Sensors, and PyTorch: A Comprehensive Approach

Alexandros Spournias[1](✉), Nikolaos Nanos[1], Evanthia Faliagka[1], Christos Antonopoulos[1], Nikolaos Voros[1], and Giorgos Keramidas[2]

[1] Electrical and Computer Engineering Department, University of the Peloponnese, Patra, Greece
a.spournias@esdalab.ece.uop.gr

[2] School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

Abstract. The importance of sound recognition and classification systems in various fields has led researchers to seek innovative methods to address these challenges. In this paper, the authors propose a concise yet effective approach for sound recognition and classification by combining spectrogram analysis, Micro-Electro-Mechanical Systems (MEMS) sensors, and the Pytorch deep learning framework. This method utilizes the rich information in audio signals to develop a robust and accurate sound recognition and classification system.

The authors outline a three-stage process: data acquisition, feature extraction, and classification. MEMS sensors are employed for data acquisition, offering advantages such as reduced noise, low power consumption, and enhanced sensitivity compared to traditional microphones. The acquired audio signals are then preprocessed and converted into spectrograms, visually representing the audio data's frequency, amplitude, and temporal attributes.

During feature extraction, the spectrograms are analyzed to extract significant features conducive to sound recognition and classification. The classification task is performed using a custom deep learning model in Pytorch, leveraging modern neural networks' pattern recognition capabilities. The model is trained and validated on a diverse dataset of audio samples, ensuring its proficiency in recognizing and classifying various sound types.

The experimental results demonstrate the effectiveness of the proposed method, surpassing existing techniques in sound recognition and classification performance. By integrating spectrogram analysis, MEMS sensors, and Pytorch, the authors present a compact yet powerful sound recognition system with potential applications in numerous domains, such as predictive maintenance, environmental monitoring, and personalized voice-controlled devices.

Keywords: Sound recognition · Machine Learning · PyTorch · Environmental Monitoring · Spectrogram · Sound Analysis

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 561, pp. 3–17, 2024.
https://doi.org/10.1007/978-3-031-54521-4_1

1 Introduction

The ability to accurately recognize and classify sound plays a crucial role in numerous applications, including predictive maintenance, environmental monitoring, surveillance systems, natural language processing, and human-computer interaction. Traditional methods of sound recognition and classification often rely on handcrafted features and shallow learning models, which can be limited in their capacity to capture complex patterns within audio signals. Consequently, there has been a surge of interest in developing more sophisticated methods that can effectively recognize and classify a wide range of sound types.

In recent years, deep learning techniques have demonstrated remarkable success in various pattern recognition tasks, including sound recognition and classification. These techniques excel in their ability to automatically learn discriminative features from raw data, which has led to significant improvements in recognition and classification performance. In this context, the PyTorch deep learning framework [1] has emerged as a popular choice for researchers, due to its flexibility, ease of use, and strong community support. Spectrogram analysis, which provides a visual representation of the time-frequency characteristics of audio signals, has also been widely employed in sound recognition and classification tasks. The rich information contained within spectrograms enables the identification of distinctive patterns that can be used to differentiate between various sound sources. In parallel, advances in sensor technology, particularly Micro-Electro-Mechanical Systems (MEMS) sensors [2], have facilitated the development of high-quality audio data acquisition systems. MEMS sensors offer numerous advantages over conventional microphones, such as reduced noise levels, low power consumption, and increased sensitivity.

In this paper, the authors propose a novel sound recognition and classification method that includes spectrogram analysis [3], MEMS sensors, and the PyTorch deep learning framework. The primary objectives of this work are to develop a comprehensive approach for sound recognition and classification that leverages the rich information contained in audio signals, and to demonstrate the effectiveness of the proposed method through extensive experimental validation. The remainder of this paper is organized as follows: Section 2 provides a literature review; Sect. 3 an overview of the proposed methodology, including data acquisition, feature extraction, and training- classification stages; Sect. 4 presents the experimental results and performance evaluation; and Sect. 5 concludes the paper with a discussion on future research directions and potential applications.

2 Literature Review

The field of sound recognition and classification has attracted significant attention from researchers due to its wide range of applications and the inherent challenges in processing and understanding audio data. This section provides a review of the relevant literature, focusing on sound analysis, spectrogram representations, MEMS sensors, and the application of PyTorch at sound recognition and classification.

2.1 Sound Analysis and Spectrogram Representations

Sound analysis techniques play a vital role in extracting meaningful information from raw audio signals. Traditional methods, such as Mel-Frequency Cepstral Coefficients (MFCC) and Linear Predictive Coding (LPC), have been extensively used to capture the spectral characteristics of audio signals (Davis & Mermelstein, 1980 [4]; Rabiner & Schafer, 1978 [5]). However, these handcrafted features often fail to capture complex patterns present in the data.

Spectrogram representations, which display the time-frequency characteristics of audio signals, have emerged as an effective alternative to traditional methods (Fulop & Fitz, 2006 [6]). By converting raw audio signals into spectrograms, researchers can visualize and analyze the spectral content and temporal patterns of sounds, enabling more efficient feature extraction and classification.

2.2 MEMS Sensors for Audio Data Acquisition

The quality of audio data acquisition plays a crucial role in the performance of sound recognition and classification systems. Traditional microphones have certain limitations, such as high noise levels and power consumption. MEMS sensors, on the other hand, offer numerous advantages, including reduced noise levels, low power consumption, and increased sensitivity (Koickal et al., 2006 [7]). The use of MEMS sensors in sound recognition and classification tasks has led to significant improvements in data quality and system performance.

2.3 Deep Learning and PyTorch in Sound Recognition and Classification

Deep learning techniques have revolutionized pattern recognition tasks, including sound recognition and classification. Convolutional Neural Networks (CNNs) [8] and Recurrent Neural Networks (RNNs) [9] have been widely played a role in this domain due to their ability to automatically learn discriminative features from raw data (Hinton et al., 2012 [10]; LeCun et al., 2015 [11]). The Pytorch deep learning framework has become a popular choice for implementing deep learning models, thanks to its flexibility, ease of use, and strong community support (Paszke et al., 2019 [12]).

Several studies have explored the use of deep learning and Pytorch in sound recognition and classification tasks. For example, Aytar et al. (2016) [13] used CNNs for environmental sound classification, while Zhang et al. (2017) [14] applied Long Short-Term Memory (LSTM) networks for speech recognition. These studies have demonstrated the effectiveness of deep learning models in recognizing and classifying various sound types.

In summary, the literature highlights the potential of combining spectrogram analysis, MEMS sensors, and PyTorch-based deep learning models for sound recognition and classification tasks. This paper aims to develop a novel method that integrates these techniques, aiming to achieve improved performance and applicability across a wide range of domains such as a predictive maintenance and environmental monitoring.

3 Methodology

This section describes the proposed methodology for sound recognition and classification, which integrates spectrogram analysis, MEMS sensors, and the PyTorch deep learning framework. The proposed system is quite general and can be adapted for various data sources (e.g. image, video, audio). However, we consider the case where the data we get from the sensors is audio data. A block diagram of our model is shown in (see Fig. 1). The methodology consists of three main stages: data acquisition, feature extraction, and training - classification.

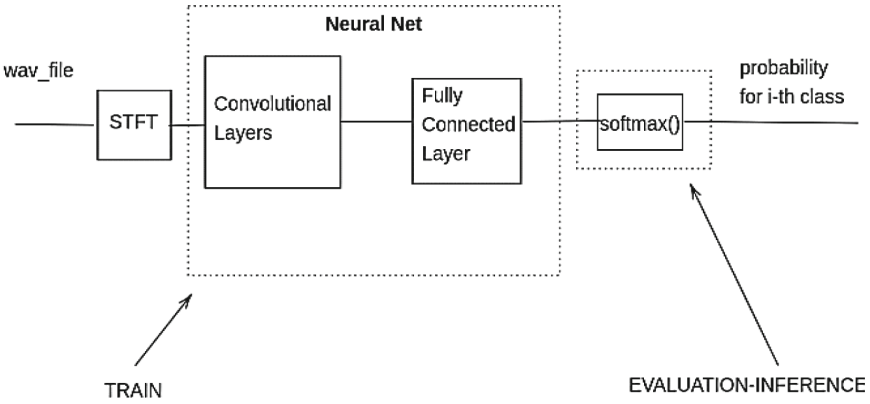

Fig. 1. Proposed model block diagram.

3.1 Data Acquisition

In the data acquisition phase of the study, the authors employed a Raspberry Pi 4 [15], a Raspberry Pi Pico [16], and an Adafruit MEMS microphone [17] to effectively gather the necessary data. The Raspberry Pi 4, a versatile and powerful single-board computer, served as the central processing unit, managing data acquisition and communication with the Raspberry Pi Pico via a USB connection.

The Raspberry Pi Pico, a microcontroller board based on the RP2040 microcontroller chip, was utilized for real-time signal processing and data transfer. The Pico was connected to the Adafruit MEMS a high-performance and compact digital microphone, responsible for capturing and converting acoustic signals into digital data. The microphone's wide frequency response and low noise ensured accurate representation of the collected audio data (see Fig. 2).

To facilitate communication between the Raspberry Pi 4 and the Raspberry Pi Pico, the authors developed also, a custom data acquisition system using the Python programming language. This system allowed for the efficient coordination and exchange of data between the two devices via the USB connection. The Raspberry Pi Pico was chosen for its affordability, simplicity, and compatibility with the Raspberry Pi 4.

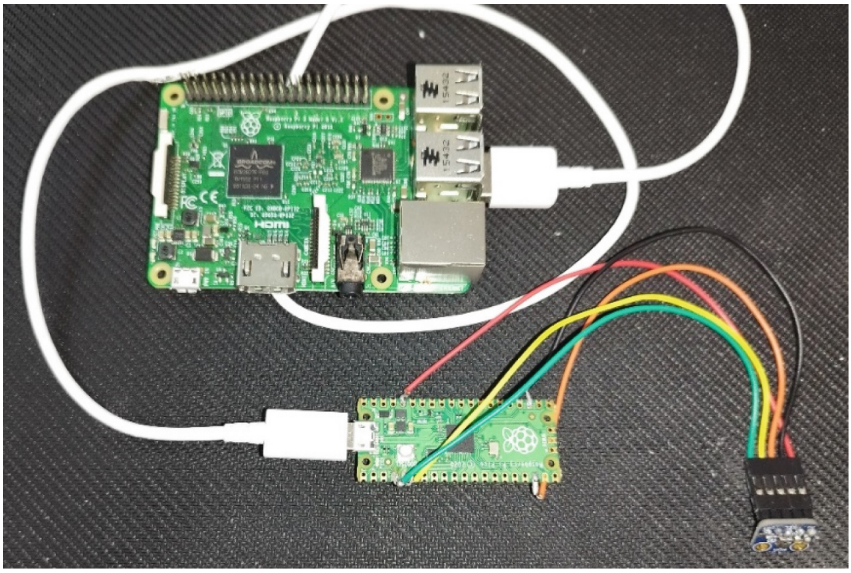

Fig. 2. System for Data Acquisition

The Adafruit MEMS microphone, played a crucial role in capturing high-quality audio data. Authors selected the Adafruit MEMS microphone for its impressive performance characteristics and compatibility with the Raspberry Pi ecosystem.

The collected data is focused on four distinct sound classes, namely *Motor_single, Motor_gran_no_chain, seatrak_all_elements,* and *kinhsh_koble*, which collectively comprise the dataset. Each sound class is represented by a set of WAV files that were recorded within the laboratory environment. To efficiently manage the files, the authors utilized Audacity software [18] to edit the length of each WAV file for every class, subsequently allocating the files into their respective folders (see Fig. 3), including test and valid, based on their corresponding labels.

3.2 Feature Extraction

The feature extraction stage involves the conversion of raw audio signals into spectrograms, which provide a visual representation of the frequency, amplitude, and temporal characteristics of the audio data. Spectrogram analysis allows for the identification of salient features that are conducive to sound recognition and classification.

To generate spectrograms, the Short-Time Fourier Transform (STFT) [19], is applied to the preprocessed audio signals, resulting in a time-frequency representation that captures the spectral content and temporal patterns of the sounds. The Short Time Fourier Transform (STFT) is a well-established technique for examining the time-varying frequency content of a signal. The STFT is computed by taking the Fourier Transform of a signal multiplied by a window function that is translated over time. The mathematical formula for the STFT is expressed as follows (Eq. 1):

$$STFT(x(t), w(t), \tau) = X(\omega, \tau) = \int x(t) * w(t - \tau) * e^{-j\omega t} dt \tag{1}$$

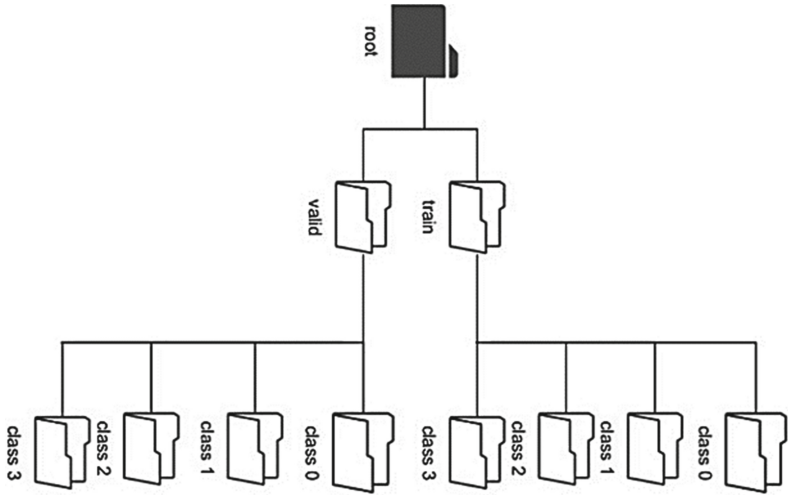

Fig. 3. Dir structure of the Wav files

Here, *x(t)* represents the input signal as a function of time '*t*', and *w(t)* denotes the window function as a function of time '*t*'. The parameter τ is the time shift or translation parameter, while ω is the angular frequency ($\omega = 2\pi f$, with '*f*' being the frequency in Hz). The STFT output, $X(\omega, \tau)$, is a complex-valued function of both angular frequency 'ω' and time shift 'τ'. Lastly, *j* is the imaginary unit *(j^2 = −1)*.

By varying the time shift parameter 'τ', the authors obtain the time-frequency representation of the signal, which offers valuable insights into the frequency content of the signal at different time intervals. In the subsequent figure (see Fig. 4), the authors present a comparative visualization of the previously mentioned fields for a linear Chirp audio signal [20], often referred to as the "linear Chirp."

Employing a sound spectrogram, a technique for visualizing audio signals is elucidated, which affords an understanding of the integral frequencies of the auditory input. Within the spectrogram, the ordinate signifies frequency, while the abscissa corresponds to the temporal aspect. Each individual pixel in the spectrogram discloses the intensity of a distinct frequency at a specific moment, thus granting a thorough comprehension of the time-frequency association present in the audio signal. As previously noted, the gathered data was segregated into four distinct categories, with the corresponding spectrograms exhibited in subsequent Figure (see Fig. 5).

3.3 Training - Classification

The training - classification stage employs a custom deep learning model implemented in PyTorch, a popular deep learning framework known for its flexibility, ease of use, and strong community support. The model is designed to automatically learn discriminative features from the spectrograms and effectively classify the sound sources (see Fig. 6).

The proposed model consists of a combination of convolutional and recurrent layers, which enables the simultaneous learning of spectral and temporal features from

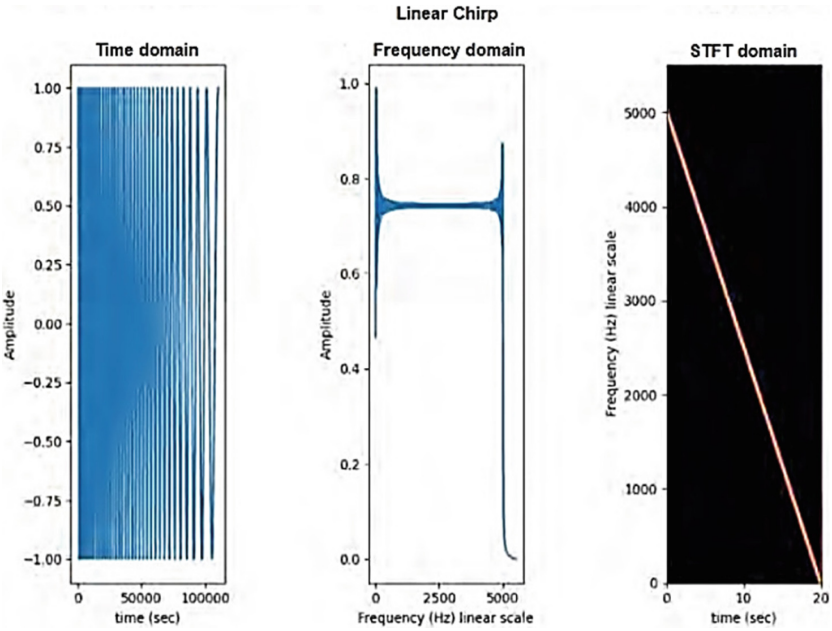

Fig. 4. Linear Chirp in different domains

the spectrograms. The spectrogram representation of every individual class undergoes processing through Convolutional Layers to extract features of the input spectrogram. It is anticipated that, following the Convolutional Layers, the problem under investigation transforms into a linearly separable one. Consequently, these features are transmitted to the Fully Connected layers.

Ultimately, the unnormalized Neural Network responses are modeled as probabilities employing the widely recognized softmax non-linearity. The softmax activation function [21], a generalized variant of the sigmoid function, serves as a methodical selection at the output stage of a Neural Network addressing a multi-class classification issue. The attainable values range between [0, 1], with the aggregate of probabilities equating to 1. The element within the output vector exhibiting the highest probability additionally signifies the class of the input waveform. The softmax function is depicted schematically in the subsequent figure (see Fig. 7).

The convolutional layers are tasked with detecting local patterns, whereas the recurrent layers are responsible for capturing long-range dependencies and temporal dynamics. The model undergoes training using a diverse dataset of spectrogram frames, with the aim of minimizing classification error.

During the training phase, the dataset is partitioned into 70% for training, 15% for validation, and 15% for testing. The CNN classifies and subsequently predicts the input, which is a wave file divided into blocks of duration *seq_dur* (5 s), as well as the class it belongs to (out of the four classes). The output of the CNN is a matrix of dimensions *(nb_blocks, nb_classes)*, with each row representing the unnormalized responses for the four classes for one block. The objective function employed for minimization is the

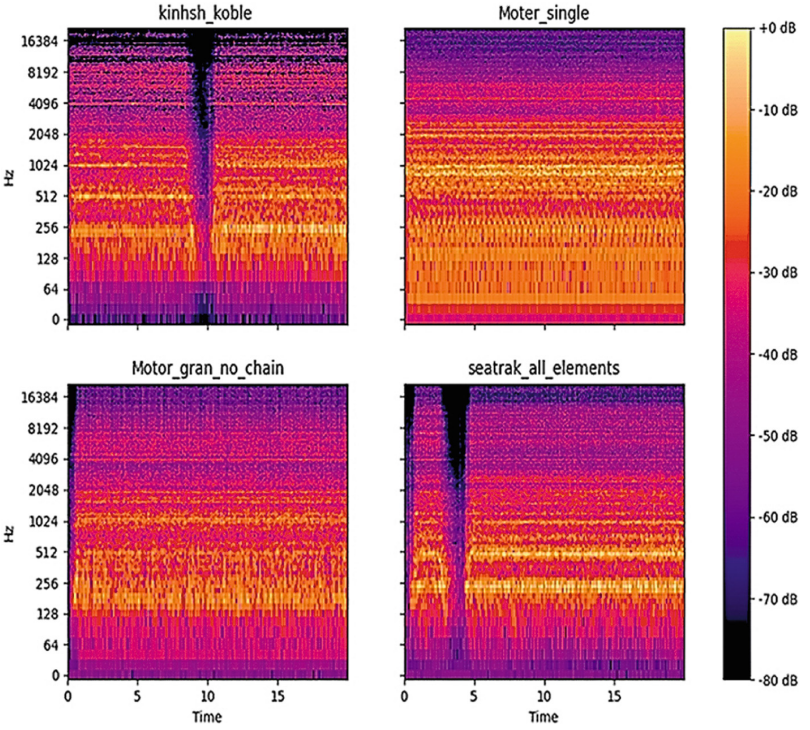

Fig. 5. Spectrograms of sound classes

Fig. 6. Training – classification

cross-entropy function available in PyTorch's CrossEntropyLoss [22], where weights are also utilized for each class to balance the input, as the dataset is typically unbalanced. The training phase is depicted in the subsequent figure (see Fig. 8).

In the following figure (see Fig. 9), the training error is represented by the blue line, which is considerably small from the second epoch onwards, while the validation error, illustrated by the red line, remains low from the first epoch.

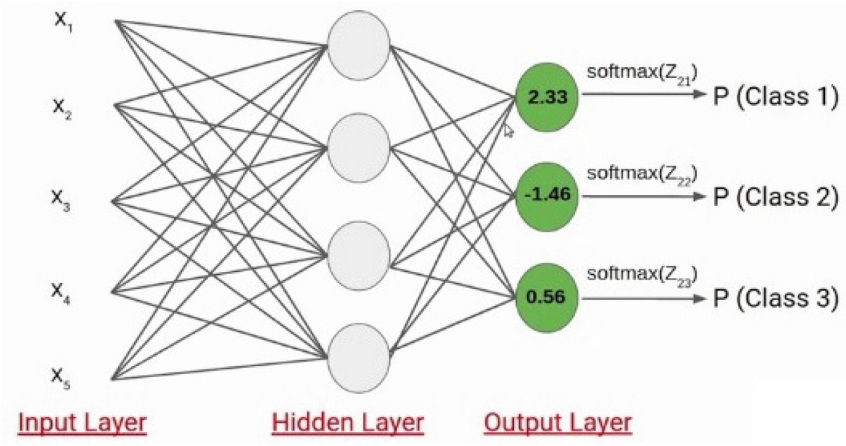

Fig. 7. Softmax layers

```
c:\ML1\ML_pipeline_code_multiple_classes_v1>python3 train.py --root c:\ML1\ML_pipeline_code_multiple_classes_v1\Spectrograms_
tst --nb_classes 4 --output c:\ML1\ML_pipeline_code_multiple_classes_v1\Spectrograms_tst\pretr_model --epochs 10
Using GPU: True
the model will be running on cuda device
  0%|                                                                                                          | 0/10 [00:00<?, ?it/s]=
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Net                                      [16, 4]                   --
├─Sequential: 1-1                        [16, 16, 257, 145]        --
│    └─Conv2d: 2-1                       [16, 16, 257, 145]        416
│    └─BatchNorm2d: 2-2                  [16, 16, 257, 145]        32
│    └─LeakyReLU: 2-3                    [16, 16, 257, 145]        --
├─Sequential: 1-2                        [16, 32, 129, 73]         --
│    └─Conv2d: 2-4                       [16, 32, 129, 73]         12,832
│    └─BatchNorm2d: 2-5                  [16, 32, 129, 73]         64
│    └─LeakyReLU: 2-6                    [16, 32, 129, 73]         --
├─Sequential: 1-3                        [16, 64, 65, 37]          --
│    └─Conv2d: 2-7                       [16, 64, 65, 37]          51,264
│    └─BatchNorm2d: 2-8                  [16, 64, 65, 37]          128
│    └─LeakyReLU: 2-9                    [16, 64, 65, 37]          --
├─Sequential: 1-4                        [16, 128, 33, 19]         --
│    └─Conv2d: 2-10                      [16, 128, 33, 19]         204,928
│    └─BatchNorm2d: 2-11                 [16, 128, 33, 19]         256
│    └─LeakyReLU: 2-12                   [16, 128, 33, 19]         --
├─Sequential: 1-5                        [16, 256, 17, 10]         --
│    └─Conv2d: 2-13                      [16, 256, 17, 10]         819,456
│    └─BatchNorm2d: 2-14                 [16, 256, 17, 10]         512
│    └─LeakyReLU: 2-15                   [16, 256, 17, 10]         --
├─Linear: 1-6                            [16, 4]                   595,084
==========================================================================================
Total params: 1,684,972
Trainable params: 1,684,972
Non-trainable params: 0
Total mult-adds (G): 8.45
==========================================================================================
Input size (MB): 9.52
Forward/backward pass size (MB): 300.87
Params size (MB): 6.74
Estimated Total Size (MB): 317.13
==========================================================================================
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 52.70it/s, loss=0.529]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.47it/s, loss=0.001]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.34it/s, loss=0.001]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.73it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.21it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.07it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 64.87it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 65.44it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.03it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.58it/s, loss=0.000]
Training epoch: 100%|████████████████████| 10/10 [00:12<00:00,  1.29s/it, train_loss=0.000143, val_loss=0.0791]

c:\ML1\ML_pipeline_code_multiple_classes_v1>
```

Fig. 8. Training phase

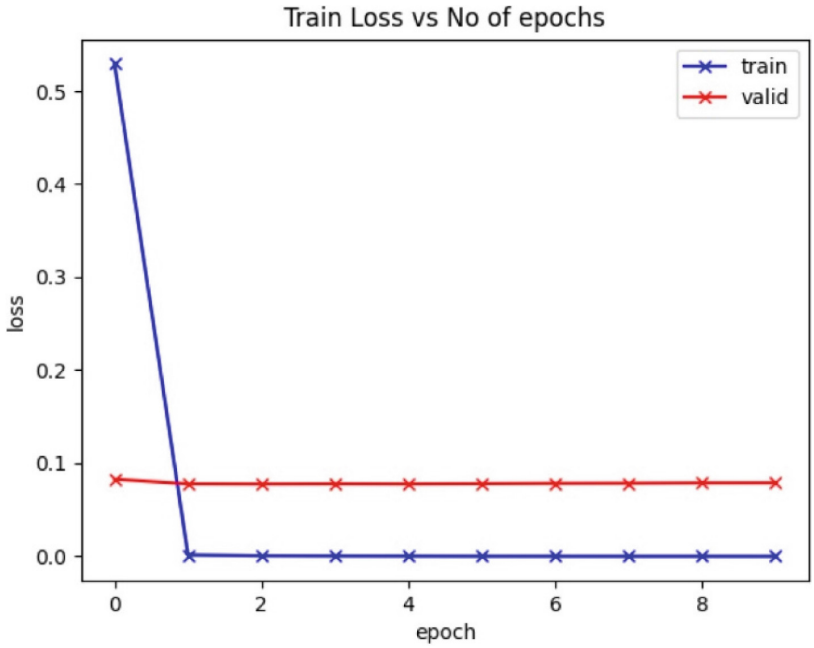

Fig. 9. Training - validation error (Color figure online)

4 Experimental Results - Performance Evaluation

The methodology consists of three main stages: Block of input-wav, Overall classification performance of input-wav, Real-time classification performance of a sound.

4.1 Block of Input-Wav Classification Performance

In order to evaluate the performance of the proposed model, the network's accuracy in classifying the provided blocks into various classes is assessed. The softmax non-linearity is applied to the unnormalized network response (a 4-element vector corresponding to the number of classes) to determine the class to which a block belongs.

This procedure is executed to transform these unnormalized responses into four probabilities with values ranging from 0 to 1. The index of the highest probability (within the vector) represents the class predicted by the network for the specific batch. The model is then tested on an independent set of spectrogram frames, which have not been utilized during training. A range of performance metrics, including accuracy, precision, recall, and F1-score, are employed to evaluate the model's efficacy in identifying and classifying distinct sound sources. The evaluation phase of a single sound class block *(kinisi_koble)* is depicted in the following figure (see Fig. 10 and Table 1).

4.2 Overall Classification Performance of Input-Wav

During the inference stage, to ascertain the class to which the input-wav belongs, the same step as in the evaluation is executed on the network's output. However, the aim

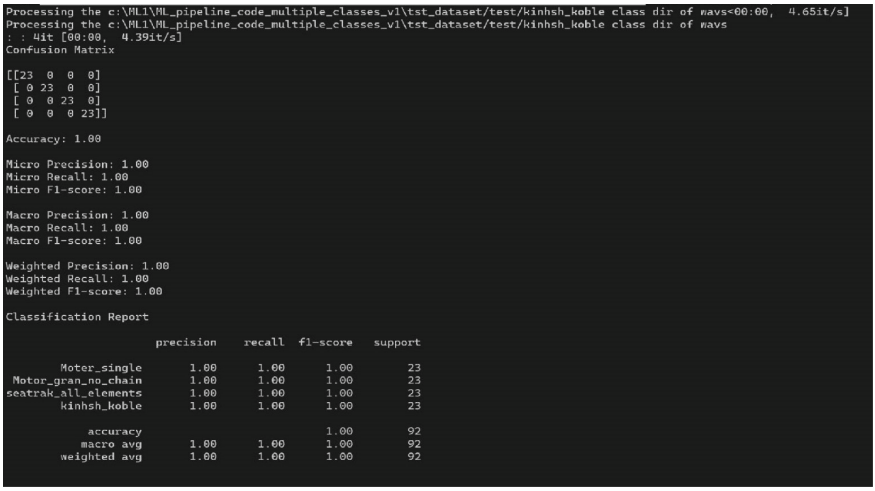

Fig. 10. Evaluation phase of a single sound class

Table 1. Evaluation performance

Name of Class	Precision	Recall	F1-Score	Support
Moter_single	1.00	1.00	1.00	23
Motor_gran_no_chain	1.00	1.00	1.00	23
seatrak_all_elements	1.00	1.00	1.00	23
Kinisi_koble	1.00	1.00	1.00	23

here is to determine the class of the entire input-wav, rather than merely its blocks. Consequently, the decision is made in favor of the class with the most occurrences, akin to consulting a histogram. It could be argued that the class to which a block belongs is a random variable, and the distribution of this variable is calculated through this histogram. The evaluation phase of a whole sound class (kinisi_koble) is depicted in the following figure (see Fig. 11).

4.3 Real-Time Classification Performance of a Sound

In the real-time inference process, to determine the class to which the input-wav belongs, the same step as previous in the evaluation is executed on the network's output. However, the goal here is to decide the class of the entire input-wav, not just its blocks. As a result, the decision is made in favor of the class with the most occurrences, as if consulting a histogram (It could be posited that the class to which a block belongs is a random variable, and the distribution of this variable is calculated through this histogram). The evaluation phase of a whole three sound classes in real-time, is depicted in the following figure (see Fig. 12).

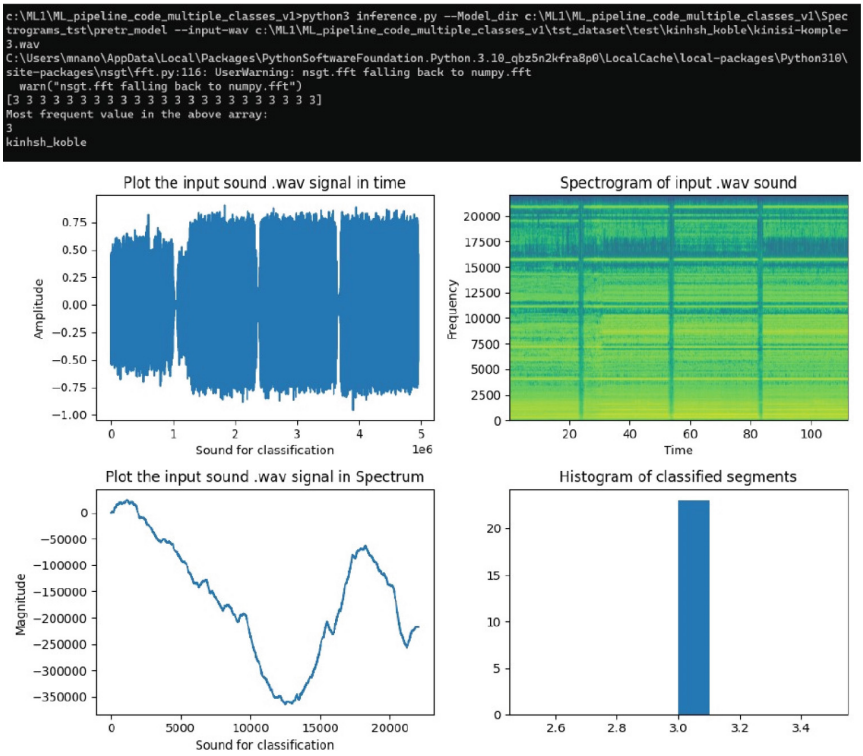

Fig. 11. Overall classification performance of input-wav

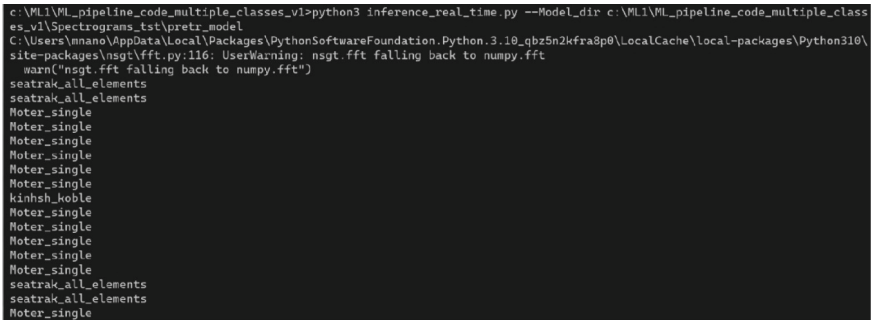

Fig. 12. Real-time classification performance

For the experimental process, 15 recognition trials were conducted for each class. The results demonstrated that recognition was highly successful in the class representing a unique sound with a specific pattern. On the other hand, for complex sounds that included a mix of noises from various components, the recognition performance was equally good, despite a slight increase in false recognitions.

In summary, the following graph (see Fig. 13) present the experimental results and performance evaluation of the proposed method, demonstrating its effectiveness in recognizing and classifying various sound types.

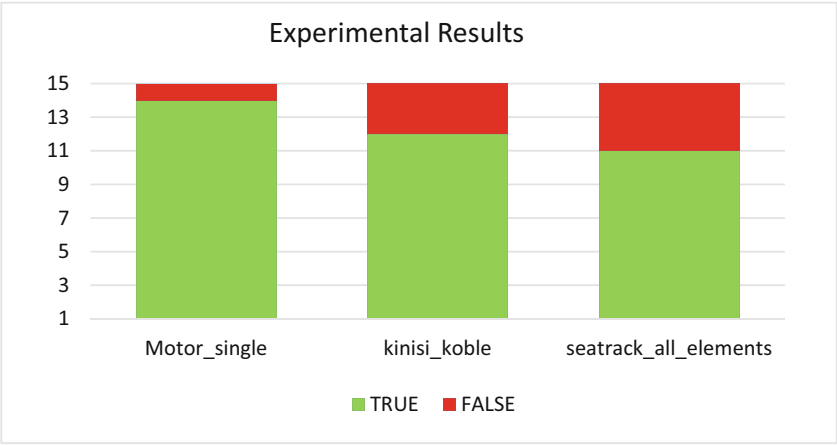

Fig. 13. Experimental results.

5 Conclusion – Future Work and Potential Applications

In conclusion, this study has demonstrated the potential of machine learning-based classification techniques in sound recognition tasks. The findings indicate that such methods can effectively classify unique sounds and even complex mixtures of noises from various components. However, there is still room for improvement in terms of reducing false recognitions.

Future research directions could explore the incorporation of more advanced deep learning architectures, such as attention mechanisms, to enhance the model's ability to focus on the most salient features within the input data. Additionally, the development of more robust and diverse datasets would contribute to the generalization capabilities of the models, making them more applicable to real-world scenarios.

Potential applications of sound recognition via machine learning classification span across various domains, including environmental monitoring, healthcare, industry, and smart cities. For instance, in environmental monitoring, these techniques could be employed to track and identify the presence of specific species, monitor ecosystems, or detect signs of pollution. In healthcare, sound recognition models could be used for early detection of diseases or monitoring patients' conditions through the analysis of sounds produced by the human body. In industrial settings, these techniques can be applied to monitor equipment performance, identify malfunctions, and predict maintenance requirements. Lastly, in the context of smart cities, sound recognition models could contribute to the optimization of traffic management and the enhancement of public safety by detecting unusual or potentially dangerous events based on sound patterns.

Overall, the advancements in machine learning classification for sound recognition hold great promise for numerous practical applications across diverse sectors. Continued research in this area will undoubtedly contribute to the development of more efficient, accurate, and versatile sound recognition systems, addressing the ever-growing needs of various industries and society.

Acknowledgment. This work Funded by the European Union under the Grant Agreement No. 101087257. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

1. PyTorch 2.0: Deep learning framework. https://pytorch.org
2. Micro-Electro-Mechanical Systems. https://www.mems-exchange.org/MEMS/what-is.html
3. Spectrogram. https://en.wikipedia.org/wiki/Spectrogram
4. Davis, S., Mermelstein, P.: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. Speech Signal Process. **28**, 357–366 (1980). https://doi.org/10.1109/TASSP.1980.1163420
5. Rabiner, L.R., Schafer, R.W.: Digital Processing of Speech Signals. Prentice Hall, Upper Saddle River (1978). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1456137
6. Fulop, S.A., Fitz, K.: A Spectrogram for the Twenty-First Century, January 2006. https://www.researchgate.net/publication/243716460_A_Spectrogram_for_the_Twenty-First_Century
7. Koickal, T.J., Hamilton, A., Tan, S.L., Covington, J.A., Gardner, J.W., Pearce, T.C.: Analog VLSI circuit implementation of an adaptive neuromorphic olfaction chip. Circuits Syst. (2007). shorturl.at/qNQW3
8. Convolutional Neural Networks. https://en.wikipedia.org/wiki/Convolutional_neural_network
9. Recurrent Neural Networks. https://en.wikipedia.org/wiki/Recurrent_neural_network
10. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. https://doi.org/10.48550/arXiv.1207.0580
11. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning (2015). https://www.nature.com/articles/nature14539
12. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. https://papers.nips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf
13. Aytar, Y., Vondrick, C., Torralba, A.: SoundNet: learning sound representations from unlabeled video. https://proceedings.neurips.cc/paper/2016/file/7dcd340d84f762eba80aa538b0c527f7-Paper.pdf
14. Zhang, Z., et al.: A framework for quantifying the impacts of sub-pixel reflectance variance and covariance on cloud optical thickness and effective radius retrievals based on the bi-spectral method (2017). https://aip.scitation.org/doi/abs/10.1063/1.4975502
15. Raspberry Pi 4. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
16. Raspberry Pi Pico. https://www.raspberrypi.com/products/raspberry-pi-pico/
17. Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H. https://www.adafruit.com/product/3421
18. Audacity, Free, open source, cross-platform audio software. https://www.audacityteam.org

19. Short-Time Fourier Transform (STFT). https://www.dsprelated.com/freebooks/sasp/Short_Time_Fourier_Transform.html
20. Linear Chirp waves. https://en.wikipedia.org/wiki/Chirp
21. Softmax activation function. https://deepai.org/machine-learning-glossary-and-terms/softmax-layer
22. Cross Entropy Loss. https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html