COLLEGE DOCTORAL PAYS DE LA LOIRE — SCIENCES DE L'INGENIERIE ET DES SYSTEMES

Le Mans Université

THÈSE DE DOCTORAT DE

LE MANS UNIVERSITÉ

ÉCOLE DOCTORALE N° 602
Sciences de l'Ingénierie et des Systèmes
Spécialité : *Acoustique*

Par

« Ammar AHMED »

Multi-purpose acoustic sensor for smart home

Capteur polyvalent acoustique pour l'habitat intelligent (CAPAHI)

Thèse présentée et soutenue à « Le Mans », le « 04 Mai 2023 »
Unité de recherche : LAUM UMR CNRS 6613 France
Thèse N° : 2023LEMA1007

Rapporteurs avant soutenance :

Emil NOVAKOV Professeur des Universités - Université Grenoble Alpes.
Mounir SAYADI Professeur des Universités - École Nationale Supérieure d'Ingénieurs de Tunis ENSIT.

Composition du Jury :

Président & Examinatrice :	Mme. Marion BERBINEAU	Directrice de recherche, Université Gustave Eiffel (IFSTTAR).
Dir. de thèse :	M. Kosai RAOOF	Professeur des Universités, Le Mans Université (LAUM).
Co-dir. de thèse :	M. Jean-François DIOURIS	Professeur des Universités, Université de Nantes.
Co-encadrant :	M. Youssef SERESSTOU	PRAG, Le Mans Université (LAUM).

Acknowledgement

I would like to express my gratitude to everyone who has supported me throughout the course of my thesis.

First and foremost, I would like to thank my thesis advisors, Prof. Kosai RAOOF, Prof. Youssef SERRESTOU, and Prof. Jean-François DIOURIS, for their guidance, patience, and encouragement throughout my research. Their expertise and support have been invaluable to me. I will always be grateful for their support during the pandemic and even after the pandemic.

I would also like to extend my appreciation to the faculty and staff of Le Mans Université, École nationale supérieure d'ingénieurs du Mans (ENSIM), and LAUM-Laboratoire d'Acoustique de l'Université du Mans (LAUM) for providing me with the resources and opportunities necessary to pursue my academic goals.

Furthermore, I would like to express my sincere gratitude to my colleagues and friends. Their support, encouragement, and willingness to provide feedback and assistance have been invaluable to me. Their positivity, enthusiasm, and willingness to share their knowledge and experiences have helped me grow both personally and professionally. I am thankful for the collaborative and friendly atmosphere that we have created and the activities we did together. I thank you, Stepan, Saoucene, Patrick, Erwan, Theo M., Theo B., Gildas, Alex, Audrey, Lilit, Manel, Leila, Nadia, Ninon, Remi, and Guillaume for being a part of my journey and your kind made my academic journey even more enjoyable.

Lastly, I would like to express my heartfelt appreciation to my family. My mother, Shabana, and my father, Zahid, my lovely sister, Maryum, my brothers, Waqar, Usman, Sufyan, and the new addition to our family Sara and Aliyan this would not have been possible without them and their constant support. Their unconditional love, unwavering support, and endless sacrifices have been the driving force behind my success. Their encouragement, motivation, and belief in me have kept me going even in the most challenging times. I am deeply grateful for their constant presence, encouragement, and unwavering faith in me.

Thank you all for being a part of my thesis journey and for helping me achieve this milestone.

Acronyms

List of acronyms used in the document:

ESC: Environmental Sound Classification
EMD: Empirical Mode Decomposition
TKEO: Teager Kaiser Energy Operator
DESA: Discrete Energy Separation Algorithm
CNN: Convolutional Neural Networks
DSCNN: Depth Wise Separable Convolutional Networks
RNN: Recurrent Neural Networks
LSTM: Long Short-Term Memory
GRNN: Gated Recurrent Neural Networks
ANN: Artificial Neural Networks
FFNN: Feed Forward Neural Networks
SVM: Support Vector Machines
MCU: Micro Controller Unit
ReLU: Rectified Linear Units
ADAM: Adaptive Moment Estimation
SIGM: Logistic Sigmoid Function
Adagrad: Adaptive Gradient Algorithm
TANH: Hyperbolic Tangent
SFS: Sequential Forward Selection
SBS: Sequential Backward Selection
SFFS: Sequential Forward Floating Selection
SBFS: Sequential Backward Floating Selection
MBE: Mel Band Energies
S-MBE: Signal Trend Removed Mel Band Energies
MFCC: Mel Frequency Cepstral Coefficients
IMFs: Intrinsic Mode Functions
FFT: Fast Fourier Transform

STFT: Short Time Fourier Transform
DFT: Discrete Fourier Transform
PCM: Pulse Code Modulation
MACC: Multiply–Accumulate Operation

Symbols

$\boldsymbol{x(t)}$	Continous time vector
$\boldsymbol{x[n]}$	Discrete time vector
$\boldsymbol{W}$	Matrix
$\boldsymbol{\sigma}$	Activation function, e.g. sigmoid, tanh, ReLU
$\boldsymbol{\theta}$	Model parameters
η	Learning rate
$\hat{\boldsymbol{y}}$	Predicted output vector
$\boldsymbol{y}$	Ground truth output vector
$\mathbb{1}$	Indicator function
$\mathbb{R}$	Real number
$\overrightarrow{\boldsymbol{U}}$	Forward recurrent weights matrix
$\overrightarrow{\boldsymbol{b}}$	Forward recurrent biases vector
$\overleftarrow{\boldsymbol{V}}$	Backward direction recurrent weights matrix
$\overleftarrow{\boldsymbol{b}}$	Backward direction recurrent biases vector
$\boldsymbol{\nabla}$	Gradient
$\boldsymbol{\Delta}$	Rate of change

Abstract

Environmental sounds emanate from a variety of sources, such as human and non-human activities, traffic sounds, birds, rain, and sounds produced by human activity in houses, offices, cafes, and numerous others. The automatic classification of these sounds has recently attracted a lot of interest due to its great potential for application in various domains, such as human-machine interaction, smart homes, robotic hearing, automatic activity recognition, automatic surveillance systems, etc. In the case of a smart home, the heterogeneity of the events to be monitored leads to the use of a large number of sensors of different types, which impacts the cost, energy consumption, and complexity of installation and management, as well as on the volume of data to be processed. The objective of this thesis is to demonstrate that the use of ESC (Environmental Sound Classification) provides a solution to the problem of reducing the number and diversity of sensors by replacing all or part of these sensors with acoustic sensors. The feasibility of this solution has been validated by the development of a system, implemented on an edge device composed of a microcontroller (Cortex M4) with a small memory footprint and low power consumption and a microphone, for the automatic recognition in real-time of four environmental sounds (rain, wind, human footsteps, and passing cars). For better algorithm-architecture matching, different feature extraction and selection methods and different machine-learning models are studied and compared. A new feature extraction approach based on the Instantaneous Amplitude (IA) and Frequency (IF) spectrogram has been proposed and compared to the traditional approach using the Mel spectrogram. To overcome the limitations of the traditional approach, related to the fact that environmental sounds signals are multi-component, non-stationary signals from non-linear systems, the construction of the instantaneous amplitude and frequency spectrogram is carried out by Empirical Mode Decomposition (EMD) coupled with the Teager-Kaiser Energy Operator (TKEO). To generate a model for ESC, a database of selected environmental sounds was constructed. The selected learning models are convolutional neural networks (CNN) and depth-wise separable convolutional neural networks (DSCNN), optimized in terms of the number of parameters and features. The feature size reduction and optimization are carried out using a recurrent neural network (RNN) feature selection method.

This developed automatic recognition system was evaluated with new signals for real-time classification.

Key words: Environmental sound classification (ESC), empirical mode decomposition (EMD), Teager-Kaiser energy operator (TKEO), instantaneous amplitude (IA), instantaneous frequency (IF), convolutional neural networks (CNN), depth-wise separable convolutional networks (DSCNN), recurrent neural networks (RNN).

Resumé

Les sons environnementaux proviennent généralement de sources diverses et variées, telles que l'activité humaine, les objets et la nature. La classification automatique de ces sons suscite récemment un grand intérêt grâce à son grand potentiel d'application dans divers domaines, comme l'interaction home-machine, l'habitat intelligent, l'audition robotique, la reconnaissance automatique d'activités, les systèmes de surveillance automatique, etc. Dans le cas d'un habitat intelligent, L'hétérogénéité des événements à surveiller conduit à l'usage d'un grand nombre de capteurs, de différentes natures, ce qui impacte le coût, la consommation énergétique, la complexité d'installation et de gestion ainsi que l'encombrement et le volume de données à traiter. L'objectif de cette thèse est de démontrer que l'utilisation de la classification automatique des sons environnementaux (ECS pour Environmental Sound Classification en anglais), apporte une solution à cette problématique de réduction du nombre et de la diversité des capteurs en remplaçant tout ou une partie de ces capteurs par des capteurs acoustiques. La faisabilité de cette solution a été validée par le développement d'un système, implémenté sur une carte composée d'un microcontrôleur (Cortex M4) à faible empreinte mémoire et à basse consommation et d'un microphone, pour la reconnaissance automatique en temps réel de quatre sons environnementaux (pluie, vent, pas humain et passage de voiture). Pour une meilleure adéquation algorithme-architecture, différentes méthodes d'extraction et de sélection de caractéristiques et différents modèles d'apprentissage automatique ont été étudiés et comparés. Une nouvelle approche d'extraction de caractéristiques à partir du spectrogramme d'amplitude et de fréquence instantanées (SAFI), a été proposée et comparée à l'approche traditionnelle utilisant le spectrogramme de Mel. Pour s'affranchir des limitations de l'approche traditionnelle, liées au fait que les sons environnementaux sont des signaux multi-composantes, non-stationnaire et issus de systèmes non-linéaires, la construction du spectrogramme d'amplitude et de fréquence instantanées est effectuée par décomposition en modes empiriques (EMD) couplée à l'opérateur d'énergie de Teager-Kaiser (TKEO). Pour générer le modèle d'apprentissage, une base de données de sons environnementaux choisis a été construite. Les modèles d'apprentissage retenus sont les réseaux de neurones convolutifs (CNN) et convolutifs profonds (DSCNN), optimisés en termes du nombre de

paramètres et de caractéristiques. Cette optimisation est effectuée grâce à une méthode de sélection de caractéristiques par réseau de neurones récurrent (RNN). Ce système de reconnaissance automatique développé a été évalué avec de nouveaux signaux pour une classification en temps réel.

Mot clés : Classification Automatique des Sons Environnementaux (ECS), amplitude instantanée, fréquence instantanée, décomposition en modes empiriques (EMD), Teager-Kaiser energy operator (TKEO), réseau de neurones récurrents (RNN), réseau de neurones convolutif (CNN), réseau de neurones convolutif profond (DSCNN).

Table of Contents

Introduction **1**

General Introduction 1
Challenges 3
Problem Formulation 6
Thesis Objectives 8
Contribution 9
Thesis Structure 9
List of Publication 10

I Background **11**

1 Background **13**

1.1 Environmental Sound Classification 13
1.2 Sound Representation / Feature Extraction 14
1.2.1 Steps involved in acoustic feature extraction 15
1.2.2 Spectrogram construction 17
1.2.3 Mel Spectrogram Construction 17
1.2.4 Mel-frequency cepstral coefficients (MFCC) 19
1.2.5 Other feature extraction methods 20
1.3 Machine Learning for Environmental sound classification 21
1.4 Artificial Neural Networks 23
1.4.1 Feed-forward Neural Networks 25
1.4.2 Recurrent Neural Networks 27
1.4.3 Convolutional Neural Networks 31
1.4.4 Depthwise separable convolution networks 34
1.5 Neural network model training and evaluation 36
1.5.1 Training of Model 36
1.5.2 Optimization by Gradient Descent Based Optimization Algorithms 36

1.5.3 Network Regularization 41
1.5.4 Evaluation of sound classification model 42
1.5.5 Performance Metrics 42
1.5.6 Cross Validation 44
1.6 Datasets 45
1.6.1 Acoustic Scene Classification Dataset 45
1.6.2 Low-Complexity Acoustic Scene Classification Dataset 46
1.6.3 Urbansound8k 46
1.6.4 Custom Database 46

II Feature Engineering 47

2 Environmental Sound Classification Systems based on Empirical Features 49
2.1 Introduction 49
2.2 Proposed method 51
2.2.1 Signal representation 52
2.2.2 Empirical Mode Decomposition 52
2.2.3 Sifting Process for IMFs 53
2.2.4 Teager–Kaiser Energy Operator (TKEO) 57
2.3 Feature Extraction 58
2.3.1 Mel Band Energies 58
2.3.2 EMD-Mel Band Energies 60
2.3.3 S-MBE 61
2.3.4 Features 62
2.4 Experimental Setup 63
2.4.1 Datasets 63
2.4.2 Custom dataset creation description 70
2.4.3 Classification Model 71
2.5 Results and Discussion 74
2.6 Conclusion 75

3 Feature Selection 79
3.1 Introduction 79

3.2 Background . . . 81
3.2.1 Feature Selection Method . . . 82
3.3 Method description and Experimental Setup . . . 86
3.3.1 Acoustic Data . . . 86
3.3.2 Feature Extraction . . . 87
3.3.3 Experimental setup description . . . 88
3.4 Results . . . 88
3.5 Conclusion . . . 99

III Hardware Implementation 100

4 Machine Learning on Edge devices 101
4.1 Introduction . . . 101
4.2 Background . . . 103
4.2.1 Hardware Description . . . 103
4.3 Dataset and Pre-processing . . . 104
4.3.1 Custom Dataset . . . 104
4.3.2 Feature Extraction . . . 106
4.3.3 Z-Score Scaling . . . 107
4.3.4 Audio signal accusation and feature extraction . . . 108
4.4 Experimental Setup . . . 108
4.5 Results and Discussion . . . 111
4.6 Conclusion . . . 116

IV Conclusion 117

Conclusion 119

Bibliography 131

List of Tables

2.1 Dataset Recordings 71
2.2 Convolutional neural network model specifications 73
2.3 System performance comparison. 75
2.4 Classification accuracy of each class for the Acoustic Scene Classification Dataset. 76
2.5 Classification accuracy of each class for the Low Complexity ASC Dataset. 76
2.6 Classification accuracy of each class for the Urbansound8k dataset. 76
2.7 Classification accuracy of each class for the custom dataset. 77

3.1 Sound event categories 87
3.2 Neural network construction 89
3.3 Classification accuracy with different subsets 97
3.4 Classification accuracy per category at k = 35 98
3.5 Parameters trained versus accuracy 98

4.1 Neural Network Mode Description. 113
4.2 Convolutional neural networks model description 114
4.3 Depth-wise separable convolutional neural networks model description . . . 114
4.4 CNN and DSCNN computational complexity 115
4.5 CNN and DSCNN parameters, activations and weights 115
4.6 Model Accuracy inference Time and MACCs 115

List of Figures

1 Smart Home. 2
2 Environmental sound classification system block diagram. 3

1.1 Feature extraction stages from an audio signal. 16
1.2 Mel scale vs Hertz scale. 18
1.3 Triangular Mel filter banks. 19
1.4 Log Mel spectrograms of different environmental sounds. 20
1.5 Mel frequency cepstral coefficients of different environmental sounds. . . . 21
1.6 Block diagram of machine learning method with neural networks. 23
1.7 Biological neuron and artificial neuron. 24
1.8 Feed forward neural network. First layer is the input layer, followed by two hidden layers and a output layer at the end. 28
1.9 Recurrent Units in Recurrent neural networks. 29
1.10 LSTM and GRU Cell. 31
1.11 Convolution operation in convolutional neural networks (CNN) 33
1.12 Convolutional neural network (CNN) on a spectrogram. 34
1.13 Depthwise convolution operation on an input image. 35
1.14 Pointwise convolution operation on an input image. 35
1.15 Relu, Tan hyperbolic function, and sigmoid activation function response. . 41
1.16 Multi fold cross-validation setup. 44

2.1 Block diagram of proposed system—feature extraction using EMD-TKEO method and classification using neural networks. 52
2.2 Flow chart of sifting process. 55
2.3 Empirical mode decomposition and intrinsic mode function extraction. . . 56
2.4 Block diagram of smoothing the output of the energy operator. 58
2.5 Empirical mode decomposition-based Mel filter bank energies extraction block diagram. 61
2.6 SMBE feature extraction block diagram. 62
2.7 FFT-MBE spectrogram extracted from a 10-sec audio file of a car passing. 63

2.8 Signal trend removed (S-MBE) spectrogram of a 10-sec audio file of a car passing. . . . 64
2.9 Empirical mode decomposition- Mel band energies (EMD-MBE) of a 10-sec audio file of a car passing. . . . 64
2.10 Signal representation of a person walking . . . 65
2.11 Empirical mode decomposition- Mel band energies (EMD-MBE) of a 10-sec audio file of a person walking. . . . 65
2.12 IMF(s) 1 - 4 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s). . . . 66
2.13 IMF(s) 5 - 8 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s). . . . 67
2.14 IMF(s) 9 - 12 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s). . . . 68
2.15 IMF(s) 13 - 16 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s). . . . 69

3.1 Sequential Forward Floating Selection (SFFS) Algorithm. . . . 85
3.2 Block diagram of feature selection training and testing process. . . . 86
3.3 Plots of All feature selected. (k = 52) for FFT-MFCC-SFS. . . . 91
3.4 Plots of SFS feature selected. (k = 35) and Plots of SBS feature selected. (k = 35) . . . 92
3.5 Plots of All feature selected. (k = 52) for EMD-MFCC-SFS. . . . 92
3.6 Plot of All feature selected. (k = 52) for FFT-MFCC-SFS. . . . 93
3.7 Plot of SFS for EMD-MFCC feature selected (k = 35). . . . 93
3.8 Plot of SFS for S-MFCC feature selected (k = 35). . . . 94
3.9 Plot of SBS for EMD-MFCC feature selected (k = 35). . . . 94
3.10 Plots of SBS for S-MFCC feature selected. (k = 35) . . . 95
3.11 Feature performance of SFS-FFT-MFCC, SFS-EMD-MFCC, and SFS-S-MFCC . . . 95
3.12 Feature performance of SBS-FFT-MFCC, SBS-EMD-MFCC, and SBS-S-MFCC . . . 96
3.13 Feature performance of SFFS-FFT-MFCC, SFFS-EMD-MFCC, and SFFS-S-MFCC . . . 96
3.14 Feature performance of SBFS-FFT-MFCC, SBFS-EMD-MFCC, and SBFS-S-MFCC . . . 97

4.1 SensorTile board size and microprocessor and sensors equipped onboard. . 104
4.2 SensorTile Cradle and onboard components description. 105
4.3 SensorTile board components and connectivity with STM32L4 processor block diagram [133]. 105
4.4 Block diagram of classification process on a microcontroller. 109
4.5 Processing chain for inference on STM32 SensorTile. 110
4.6 Model accuracy over five-fold cross-validation. 112
4.7 Inference rate vs MACCs 113

INTRODUCTION

General Introduction

Environmental sounds emanate from a variety of sources and are a ubiquitous part of our daily lives. The sounds that we hear in our surroundings are generated by both human and non-human activities, such as traffic sounds, birds, rain, and human activities taking place in homes, offices, cafes, and many other places. In recent years, the automatic classification of these sounds has attracted a significant amount of interest due to its potential applications in a wide range of domains including human-machine interaction, smart homes, robotic hearing, automatic activity recognition, automatic surveillance systems, and many others.

The concept of smart homes has been gaining traction in recent years, as more and more people look for ways to improve their living spaces through the use of technology. With the increasing use of smart homes and the rapid advancements in technology, the need for effective and efficient ways of recognizing environmental sounds has become even more pressing. In addition, smart homes typically rely on a large number of sensors to monitor various events within the home, including the presence of people, the state of the environment, and other factors. However, this approach has a number of drawbacks, including high costs, energy consumption, complex installations, and the need to process a large volume of data.

Automatic detection of environmental sounds or simply environmental sound classification (ESC), based on simple acoustic sensors, can resolve the aforementioned problems. Acoustic sensors are able to detect a wide range of environmental sounds, from footsteps to door slams, and can provide a plethora of information about the environment. These sensors are also much easier to install and manage than other types of sensors, as they can be easily integrated into existing systems. Furthermore, acoustic sensors typically consume less power than other types of sensors, which can help to reduce the energy consumption associated with the implementation of smart homes. Another advantage of automatic sound detection is its ability to recognize environmental sounds in real-time. This is a crucial feature for many applications, as it enables the quick and accurate de-

tection of events taking place in the environment. For example, in the case of automatic surveillance systems, the real-time recognition of environmental sounds can be used to quickly detect and respond to potential security threats. Similarly, in the case of robotic hearing, the real-time recognition of environmental sounds enables robots to respond quickly to changes in their environment, such as the approach of a person or the presence of a new object. In human-machine interaction, ESC can improve the overall user experience by creating more intuitive and responsive systems that can recognize and respond to environmental sounds in real-time. In automatic activity recognition, ESC can detect and recognize different activities taking place in the environment and provide relevant information or feedback to the user. ESC systems offer a number of advantages over other approaches to the implementation of smart homes depicted in Figure 1. For example, they can be easily integrated into existing systems and implemented on edge devices with low power consumption. This allows for monitoring the environment without consuming a large amount of energy.

Figure 1 – Smart Home.

Environmental Sound Classification (ESC) involves the application of machine learning that deals with the categorization of sounds found in the environment. Its aim is to detect sounds into various categories automatically, based on their acoustic characteristics. The utilization of machine learning algorithms is a crucial aspect of ESC, as they are employed to model the connections between different sound attributes and the associated

class labels. The general representation of the ESC system is depicted in Figure 2. ESC systems are built using a database of recorded sounds and associated labels. The database is created by recording target environmental sounds and labeling each recording with a corresponding label. The recordings are then processed and transformed into other domains to prepare them for the classification algorithm. The final step is the application of a machine learning algorithm to automatically detect and classify the input sound based on the transformed features.

Figure 2 – Environmental sound classification system block diagram.

The classification capabilities of machine learning algorithms have improved significantly as a result of the re-introduction of old architectures and the development of new architectures in the past decade. Modern machine learning models such as neural networks have significantly improved the system capability over the legacy machine learning modes. The neural network image classification achieved a higher accuracy rate of classification by the introduction of convolutional neural networks (CNN) as compared to legacy machine learning algorithms [1]. Classifying sound is a more complex task as compared to image classification. The sound classification includes the conversion of an audio signal into an image by using time-frequency representation as a spectrogram, the most commonly used representation. Spectrograms as features have proven to be robust to noise and compact. The spectrogram is utilized as input to machine learning models to learn the patterns and perform the classification of different sounds. In this thesis, we will focus on the sounds generated in the environment due to human or nature related activities in the domain of environmental sound classification in the context of a smart home.

Challenges

Environmental sound classification focuses on the detection and classification of sounds present in the indoor or outdoor environment. This presents a huge potential in different types of applications where ESC can be applied. However, the progress in the research

field has been a little slow due to several factors that affect the performance of ESC systems. Some of the challenges faced by ESC systems are:

— **Intra-class and inter-class variability:** Intra-class and inter-class variability are two significant challenges in environmental sound classification (ESC). Intra-class variability refers to the significant variations that can exist within the same class of sounds in terms of their characteristics such as duration, frequency, and amplitude. For example, the sound of a car engine can vary depending on the make and model of the car, and the driving conditions such as if it is on a highway or in a city. This variability makes it challenging to develop a single model that can accurately classify all instances of a specific class of sound. This is because the model needs to be able to adapt to the different variations of the sound in order to classify it accurately. Intra-class variability can be caused by various factors such as the different sources that produce the sound, the environmental conditions, and the recording conditions. On the other hand, Inter-class variability refers to the similarities between different classes of sounds, making it difficult to distinguish them. For example, the sound of a car and a motorcycle may be similar, making it challenging to determine which one is present. This variability can be caused by the similarity in the physical properties of the sound, such as frequency and amplitude, or the similarity in the sources that produce the sound. Inter-class variability can make it challenging to develop models that can accurately classify the different classes of sounds, as the model needs to be able to distinguish between the sounds that are similar in order to classify them correctly.

— **Overlapping Events:** Overlapping events refer to situations where multiple sounds happen at the same time, making it hard to separate and categorize them individually. For example, in a noisy urban setting, the sounds of cars, buses, trains, and pedestrians might overlap making it challenging to pinpoint a specific sound. This is due to the fact that the different sounds might share similar features such as frequency and amplitude, and may originate from different directions, making it difficult to differentiate between them.

— **Noise:** Environmental noise poses a significant challenge for ESC, as it can introduce noise, variability, and distortion into the recordings, making it difficult to accurately hear and classify the sounds of interest. Background noise such as wind or ambient traffic can mask or interfere with the target sound and make it harder to identify the specific characteristics of the sound needed for accurate classification. Additionally,

the quality of the recording equipment and the proximity of the microphone to the sound source can also affect the accuracy of the classification.

— **Underlying Pattern:** Unlike speech or music, which have a clear structure or pattern, environmental sounds do not have a well-defined structure or pattern that can be used to classify them. This makes it challenging to develop models that can accurately identify specific sounds, as the models need to be able to learn the underlying structure of the sounds in order to classify them correctly. In ESC, it makes it difficult to develop models that can accurately identify specific sounds.

— **Limited Feature extraction techniques:** ESC faces a significant challenge in terms of the limited type of feature extraction methods available. Feature extraction is the process of extracting relevant information from a sound signal that can be used to identify its category. However, there are only a limited number of feature extraction methods available, and most of them are based on hand-engineered features that were designed for speech recognition tasks. These methods are often not suitable for ESC, as environmental sounds are much more diverse and complex than speech sounds. The lack of suitable feature extraction methods makes it difficult to capture the full complexity of environmental sounds, which can lead to lower accuracy in classification tasks.

— **Labels availability:** Obtaining labeled data is a significant challenge in ESC as a large dataset of labeled sound samples is required to train a model. However, it can be difficult to obtain labeled data, particularly for rare or specific classes of sounds. The main challenges include the time and resources required to manually label large amounts of sound data, and the difficulties in finding experts who can accurately label specific classes of sounds. Additionally, it can be challenging to obtain a representative sample of sounds for each class, as environmental sounds can vary greatly depending on location, weather conditions, and time of day.

In addition to the challenges of developing ESC systems, the deployment of ESC systems on edge devices also presents additional challenges. These challenges are mostly related to the inherent limitations of edge devices, such as limited processing power, memory, storage, and power consumption. These limitations can make it difficult to run complex ESC algorithms and models on edge devices, and may also affect the overall performance and accuracy of the system. Other challenges may include the need for real-time processing, handling large amounts of data, and dealing with variations in the sound environment. Due to these challenges, developing and deploying ESC systems on edge

devices requires careful consideration and optimization to ensure that the system can effectively operate in the intended environment. Some of the challenges are described below:

— **Limited storage and memory:** Edge devices typically have limited computational resources such as memory and storage which can make it difficult to run large and complex machine-learning models for ESC. The storage constraints in edge devices limit the size of the models and decreases their performance. Also, the memory is scarce and the input to the models i.e. features, are also needed to be tailored to be memory bound.

— **Limited processing power:** Edge devices are generally designed to have limited processing power. Different machine learning models have different computation requirements, only the models that have relatively less computational requirements are eligible to be deployed on the edge devices. This also bounds the research to only use the models that require less processing power.

— **Model compression and optimization:** The computational resources of the edge device are limited, so it is necessary to find ways to optimize the model to make it lightweight and run on low-power devices. In most cases, the compression of the machine learning model also reduces the performance of the model.

— **Adaptability:** During the construction of data-sets, different recording devices record the sounds with different sensitivities. For example, the recording done through mobile phones may be slightly different from professional recording devices due to the internal active filters present in mobile phone microphones. The deployment of the model trained with a different recording device also deteriorates the performance of ESC on the edge device. The sound characteristics, type, and context of the environment may change rapidly, so the model must adapt accordingly.

Problem Formulation

The goal of environmental sound classification is to detect the sound present in the given set of environmental audio recordings and estimate the associated textual label of the sound. The labels are the representation of the sound class present in the recording. ESC can constitutes of two steps: sound representation and classification. The sound representation stage is also called the feature extraction stage. In the sound representation

stage, acoustic features are extracted from the audio recording. In the classification stage, a machine learning model is trained to estimate the probability $p(y_t|x_t, \theta) \in [0, 1]^t$ of the sound class present in the recording. Here, θ represents the model parameters of the classifier that are optimized to learn the acoustic features. The trained acoustic model performs the inferences on the input features x_t and results in an output vector of probabilities y_t, corresponding to the number of classes, the highest value representing the class present in the input class.

To perform the classification between different classes the model learns patterns present in the input. In ESC, the input is represented by sound features, generally time-frequency images. This image contains the information of the sound signal at the of the class to be classified. The machine learning model learns the information presented in the form of a time feature image and generates the output according to the label associated with the input feature. The features plays important role in training the model and its ability to generalization on unseen data. Feature engineering reflects on the aspect of machine learning where the input data to the machine learning model is provided as such that the model's capability of inference should be high. Feature engineering involves techniques such as creating new features from existing ones, altering existing features, or selecting the most useful features. This process is crucial in the machine learning process as it can greatly affect the outcome of the final model.

The input features play an important role in the classification performance of the machine learning model. The most commonly used features are Mel band energies (MBEs) and Mel cepstral coefficients (MFCCs) for training machine learning models. In order to construct the time-frequency representation Fourier transform is applied to an input signal. Fourier spectral analysis requires a linear system with stationary and ergodic data to produce accurate results. However, when analyzing sound, this can be a challenge as the frequency and energy of the signal can vary depending on the source, resulting in non-uniform and non-stationary data. This may require additional components to be added to the analysis, which can cause energy to be spread over a wider frequency range. To analyze non-stationary data, multiple Fourier components are used which increases the range of frequencies affected. Furthermore, the pre-defined basis functions used in Fourier spectral analysis may not accurately capture the characteristics of non-uniform and non-stationary data. Features based on short-time fast Fourier transform (STFT), introduced by Cooley and Tukey in 1965 [2], are predominately used in extracting frequency domain features [3]–[7]. Another problem is that these features require large memory and computational

power to train the network. Additionally, increasing the number of layers in the network also increases the cost in terms of computational power, time, and resources. To make the model more efficient, reducing the dimensionality of the input features and avoiding over-fitting can be done to decrease the computational cost.

The implementation of ESC models in the real world poses several challenges in terms of model capability, scale, and computational resources. Current models are designed to achieve high accuracy and generalization, but they are computationally expensive and costly to deploy in terms of energy and other resources. To address this issue, researchers are focusing on reducing the complexity of models while maintaining accuracy. However, they often overlook the complexity of the feature extraction stage, which can be more complex than the trained model itself. With the growth of the Internet of Things, there is an increasing focus on processing data at the edge nodes, rather than transmitting it to a central hub. This reduces power consumption by avoiding data transfer and allows for the development of applications with a balance between accuracy and complexity, taking into account the limited resources of battery-powered edge nodes.

Thesis Objectives and Motivation

The objective of this thesis is to study feature extraction methods used for environmental sound classification and study the implementation of these classification methods on edge devices in the context of smart homes. Modern machine learning techniques, such as artificial neural networks or deep neural networks, and convolutional neural networks, are used for the detection and classification tasks of sounds present in the environment. In this thesis, we tackled the question related to:

— The impact of different forms of time-frequency representation on the performance of machine learning models for ESC.

— The impact of input feature reduction and impact on the classification model.

— The implementation of sound classification related task models on the edge devices and validation on low processing power micro-controllers.

— Feasibility of running edge ESC by validating the implementation of machine learning model on low processing power edge device.

Contribution

— A novel method of feature extraction to construct a time-frequency image is proposed. Empirical Mode Decomposition (EMD) is used to analyze non-linear and non-stationary signals. EMD is highly adaptive and based on the direct extraction of energy with local time scales. Teager–Kaiser energy operator (TKEO) is used to extract instantaneous frequency and amplitude from the IMFs.

— We proposed the use of sequential feature selection from the family of wrapper-based methods for handpicking the features. The proposed system is composed of LSTM and features extracted from MFCC, by adding another abstraction level for feature extraction and compared with CNN trained with MBEs.

— The edge node based is used to validate the concept of performing sound classification using an artificial intelligence-based model running on a limited computational power micro-controller device. We showed that the low-processing powered devices could be employed to perform environmental sound classification both on recorded audio and through audio acquisition in real-time.

Thesis Structure

In the first part, the background of environment sound classification is presented. The sound representation and machine learning models are discussed. Later in the thesis, the studies on two aspects of the environmental sound classification are presented. In the first part, feature extraction methods for ESC are described and a novel method is presented. In the second part, the implementation related aspect of ESC is presented, where the development of ESC in an edge node is presented. In the last part, the conclusion and perspectives are discussed.

Publication and Conferences

List of Publication

Journal Publication(s):

— **J1:** Ahmed, Ammar, Youssef Serrestou, Kosai Raoof, and Jean-François Diouris. 2022. "Empirical Mode Decomposition-Based Feature Extraction for Environmental Sound Classification" Sensors 22, no. 20: 7717. https://doi.org/10.3390/s22207717.

Conference Publication(s):

— **C1:** A. Ahmed, Y. Serrestou, K. Raoof and J. -F. Diouris, "Sound event classification using neural networks and feature selection based methods," 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 2021, pp. 1-6, doi: 10.1109/EIT51626.2021.9491869.

— **C2:** A. Ahmed, Y. Serrestou, K.Raoof and J.-F.Diouris, "Environmental Sound Classification on Low Power Edge Device," 8th International Conference on Sensors and Electronic Instrumentation Advances (SEIA' 2022).

Conference(s) Attended:

— **CA1:** Detection and Classification of Acoustic Scenes and Events Workshop (DCASE 2022) - Nancy, France.

— **CA2:** 8th International Conference on Sensors and Electronic Instrumentation Advances (SEIA' 2022) - Corfu Greece.

— **CA3:** 16ème Congrès Français d'Acoustique (CFA 2022) - Marseille, France.

— **CA4:** Journées Jeunes Chercheur · se · s en Audition, Acoustique musicale et Signal audio (JJCAAS 2023) - Paris, France.

Part I

Background

Chapter 1

Background

1.1 Environmental Sound Classification

Environmental sound classification (ESC) is the task of detecting and classifying sounds that originate from human or non-human activities, such as sound emanated by the passing of a car, a dog barking, or sound produced by the drops of rain. Environmental sound classification is a general term and includes many fields such as sound event detection (SED) [8], [9], acoustic scene classification (ASC) [10], [11], bird sound detection [12], [13] and many more [14]–[18].

Recently, ESC has been a hot topic for researchers. This increase of interest in ESC is due to the introduction of neural networks and their application in image and speech recognition [19]–[21] and partly because of competitions organized [6], [22]–[24] to improve the performance in the recognition of daily life sound events. Another reason for this interest is the increasing potential application of ESC. These applications of sound event recognition are spread in various domains such as wildlife monitoring, traffic analysis, industrial noise control, smart homes, and smart cities. For example, bird sounds and wildlife sound recognition [25] can be used for the preservation of wildlife in forests, breakage and gunshot detection [26] for security purposes, activity detection in smart homes, fall detection for elderly persons[27]. The use of advanced machine learning techniques and new technologies like deep learning allows us to improve the accuracy and robustness of the classification process, making it more reliable in real-world settings. The detection and recognition of the events in surroundings and their use to ameliorate life experiences and improve automation in different applications.

Environment sound classification systems are composed of sound recording, sound representation, and classifying systems. The sound recording is the phase where different sounds are recorded using a digital microphone and a database is created. The second step, after obtaining audio recordings and labels, is processing the sound recordings for the classification stage. This processing stage is generally called sound representation or

feature extraction. In this stage, the audio recording is processed, and depending on the process the sound is manipulated in other domains, for example, the transformation from a one-dimensional time domain to a two-dimensional time-frequency domain. In this stage, the sound is represented as an image, further discussed in detail in the next section.

The last stage of the ESC system is the classification stage, where the sounds after processing are sent to the classifier with the target label during the learning stage, and in the testing stage only the processed input is applied and the label associated with the sound is obtained. This is the most important stage in the ESC system and hence the most difficult. To classify sounds, several methods have been studied in the past. Machine learning algorithms have performed better on the task of classifying sounds and recently neural networks are being used to classify sounds. The neural networks are part of machine learning algorithms that are based on the neuron in the brain cell. Neural networks are quite handy in the task of classifying images and sounds. Nowadays, machine learning models such as CNN, RNN, and LSTM are considered the first choice and state of the art for various tasks including ESC, due to their ability to model complex relationships and non-linear patterns in data, as well as their ability to handle large and high-dimensional data such as sound [18], [28]–[30].

1.2 Sound Representation / Feature Extraction

Audio signals are discrete streams of bits when recorded with a digital microphone. These signals are generated by several sources and are recorded in real-life environments or are generated synthetically. The signals are represented in the time domain with changing amplitudes with respect to time. The sequence is the lowest possible method for representing any sound events. However, this lowest level representation is extremely difficult for a machine learning model to learn to which category the sound belongs and performs inferences afterward. To address this issue, often the sounds are converted into a different form of representation which the classifier can learn. This representation of sound derived from the lowest level to a different form is known as features or specifically for audio signals acoustic features. The most common method to extract acoustic features consists of converting the time domain signal into a frequency domain. Signals belonging to the same category often share characteristics in the frequency domain. Another advantage of frequency domain representation is that it is more robust to noise and compact and allows more processing as compared to time domain representation. The number of stages

of processing over the time domain signal, to obtain the acoustic features, defines the level of abstraction for the representation of sound. For example, fast Fourier transform-based spectrograms are most commonly used and Mel frequency-based cepstral coefficients (MFCCs) are one abstraction level above the spectrograms. The calculation of MFCCs requires more processing steps in the frequency domain and therefore represents a higher abstraction level as shown in Figure 1.1.

1.2.1 Steps involved in acoustic feature extraction

Fourier transform is the most commonly used method for representing acoustic features in frequency domain[28]–[30]. The algorithm to apply Fourier transform in short widows is called Short-Time Fourier Transform (STFT) and involves three stages: framing, windowing, and frequency spectrum calculation. To obtain the frequency spectrum through STFT, the inherent property of STFT is assumed i.e. the signal is a sum of stationary sinusoids. Therefore, the audio signal is first divided into short time frames and later the frequency spectrum of each frame is calculated. This stage is known as the framing of the signal. In this scenario we are faced with a trade-off between frequency resolution and time resolution, depending on the length of the frame. The larger the frame length, the higher the frequency resolution and consequently the smaller the time resolution. The length of the frame is selected according to the task at hand. For environmental sound classification, generally, the range between 20 to 50 ms is often adopted [8]. To have a smooth transition between frames an overlap of 20% to 50% of the frame is often selected. This process is done in the windowing operation, where each frame is multiplied with a window function to avoid spectral leakage and discontinuities at the edge of each frame, which could lead to the corruption of the estimation of the frequency spectrum. Most popular windowing functions such as Hamming, Hann, and Blackman functions are commonly used for window operations in the feature extraction stage. In the next step, the frequency spectrum representation of each frame, subjected to the windowing function, is calculated using a discrete Fourier Transform. Discrete Fourier Transform calculates the spectral information in the window. The frequency and the power related to each frequency are later stacked on a time scale to obtain a time-frequency representation, commonly known as a spectrogram. Transformation is depicted in Figure 1.1.

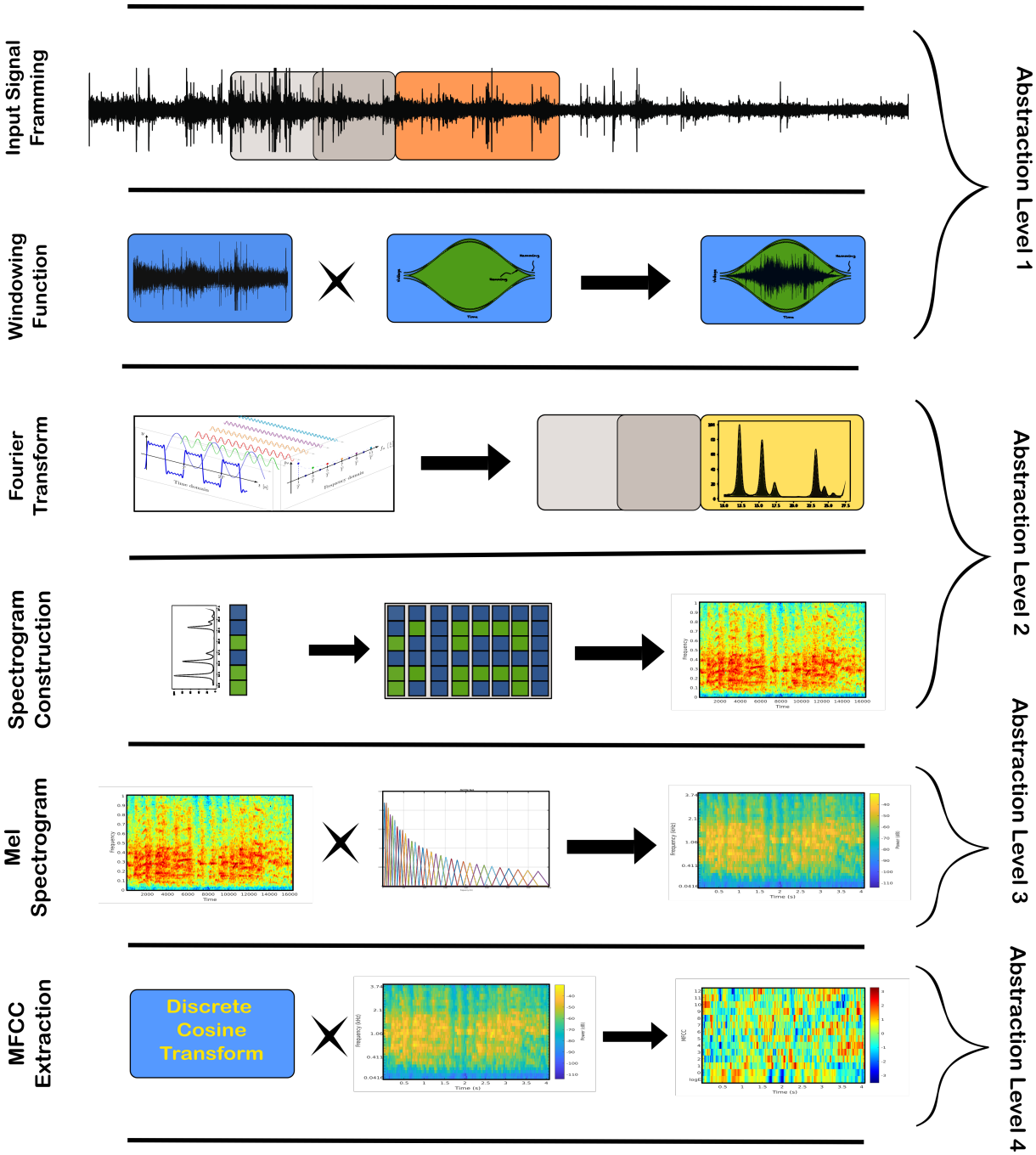


Figure 1.1 – Feature extraction stages from an audio signal.

1.2.2 Spectrogram construction

The spectrogram is a time-frequency feature matrix, obtained by extracting vectors containing spectral components for each consecutive time frame and concatenating them with other vectors according to the time frame of the audio signal. The spectrogram provides the first level of abstraction of the signal as a feature for environmental sound classification. The Fourier transform provides complex values and hence the spectrogram consists of complex values. However, only magnitude is employed and the phase information of the spectrogram is discarded. The reasoning behind such an operation is that the phase information is deemed to be less informative for the classification system. Sound events mostly demonstrate high energy in a lower frequency range and in time frequency feature representation they dominate the lower frequency components. The logarithm can be employed to compress the dynamic range of the magnitude spectrogram by obtaining the log magnitude spectrogram.

Spectrograms are considered to be a useful tool for the classification of sound events, as compared to the raw sound signal. Similar to images, spectrograms are multidimensional and this allows the image classification systems developed for image-based classification applicable to sound classification. In addition, as compared to raw audio signals the spectrograms contain more information about the sound event. The spectrogram manifests information in the time domain as well as the relative distribution in the frequency domain. Spectrograms are also considered to be more robust to noise present in the environment as compared to time domain signal, as the environmental noise generally presents in the lower frequencies, resulting in higher performance of sound classification systems.

1.2.3 Mel Spectrogram Construction

There are several ways to represent sounds as spectrograms that are based on how humans perceive sound. The article [31] has shown that human perception of sound is not linear with respect to frequency and that we are more sensitive to changes in lower frequencies than higher frequencies. One way to represent the sound that takes this into account is the Mel scale, which adjusts pitches so that they are perceived as equally spaced [32]. The Mel scale is used in the creation of Mel spectrograms and Mel frequency cepstral coefficients (MFCCs).

Mel Filter Banks

The Mel scale is a way of measuring the perceived pitch of a sound based on how it is perceived by the human ear. The Mel scale is designed to replicate the way the human ear perceives sound, which is non-linear. It is more sensitive to differences in pitch at lower frequencies and less sensitive at higher frequencies. It is often calibrated so that a pitch of 1000 Mels is equal to 1 kHz as shown in Figure 1.2. To convert a pitch measured in hertz to the Mel scale is defined as [33]:

$$m = 2595 * log_{10}\left(1 + \frac{f}{700}\right) \tag{1.1}$$

$$f = 700\left(10^{\frac{m}{2595}} - 1\right) \tag{1.2}$$

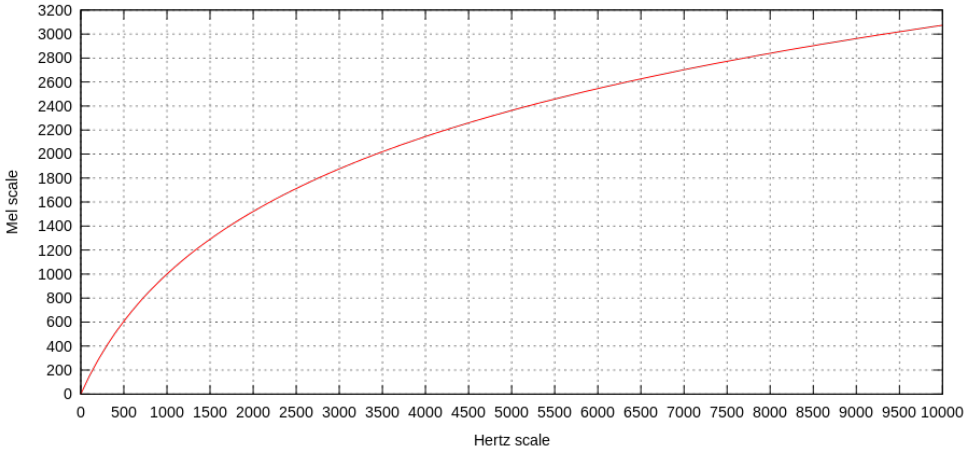


Figure 1.2 – Mel scale vs Hertz scale.

The Mel filter-bank magnitude response denoted as $\mathbf{F} \in \mathbb{R}^{B \times M}$, is made up of coefficients for B triangular band-pass filters that have central frequencies that are evenly spaced on the Mel scale. This results in filters that are closely spaced at lower frequencies and widely spaced at higher frequencies. The coefficients of the triangular filters are chosen from the range [0, 1] and are scaled so that the area under each triangle for the magnitude response is roughly the same across the Mel bands. A visualization of the Mel filter-bank magnitude response can be found in Figure 1.3.

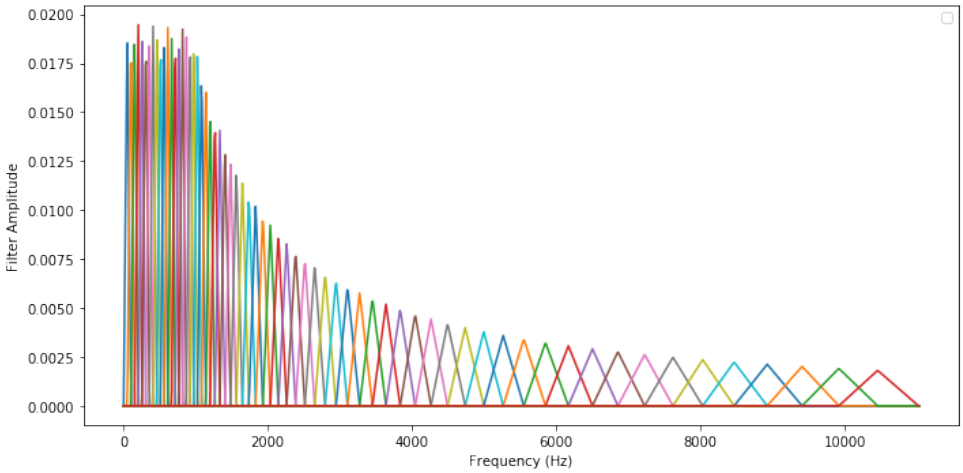


Figure 1.3 – Triangular Mel filter banks.

Mel Spectrogram

Mel spectrograms are matrices that contain Mel band energy feature vectors for consecutive time frames generated by applying a Mel filter bank to the magnitude spectrogram at each time frame. The Mel filter bank uses the Mel scale and consists of triangular filters whose bandwidths increase with the central frequency of the filters. This results in higher frequency resolution in the lower frequency range and lower resolution in the higher frequency range. Log Mel spectrograms, which are created by taking the logarithm of Mel spectrograms, are a popular representation for tasks related to sound classification and have been used in many methods for environmental sound, rare sounds, sound events, and acoustic scenes [28]–[30], [34], [35]. The number of Mel filter banks used in ESC is typically between 30 and 80, which is often smaller than the number of frequency bins used in the Short-Time Fourier Transform (STFT). As a result, Mel spectrograms provide a more compact representation than magnitude spectrograms. Several sound representations are illustrated in Figure 1.4.

1.2.4 Mel-frequency cepstral coefficients (MFCC)

MFCCs, which stand for Mel-frequency cepstral coefficients, are a popular method of representing speech and audio data in processing applications. This method involves converting the Mel spectrogram into a log Mel spectrogram by taking the logarithm and then using the DCT (Discrete Cosine Transform) to calculate cepstral coefficients from the

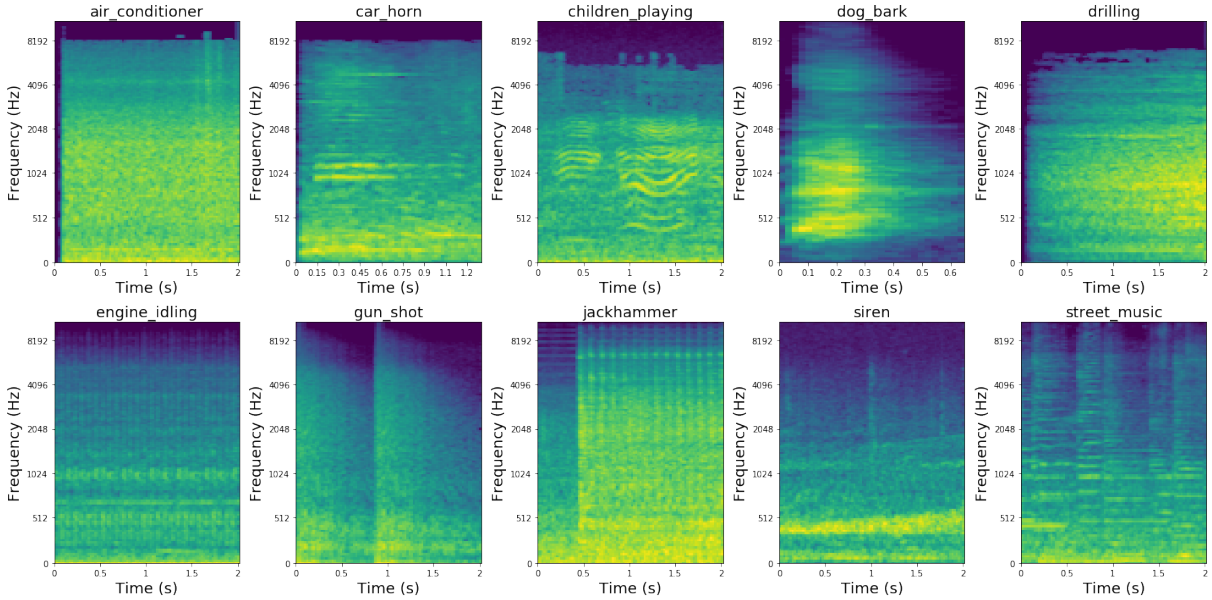


Figure 1.4 – Log Mel spectrograms of different environmental sounds.

Mel-scaled log-power spectrum. DCT is used to decorrelate the features in the adjacent windows obtained in the log Mel spectrogram, as visualized in Figure 1.5. MFCCs are widely used in speech recognition, speaker identification, and music classification tasks.

1.2.5 Other feature extraction methods

There are several methods for representing sounds that use a magnitude spectrogram as the base, a visual representation of the frequency content of a sound over time. One such method is the Gamma-tone spectrogram [36], [37], which has also been used for machine hearing and has been proposed for detecting rare sound events [38]–[40]. It is based on the equivalent regular bandwidth scale and is used to extract acoustic features related to human auditory perception. Another method is the Spectrogram Image Feature (SIF), which involves converting the magnitude spectrogram into an RGB image based on the normalized amplitudes of the spectrogram [41]. This method has been suggested for sound event classification [42]. The histogram of oriented gradients (HOG) is another image processing-based feature extraction method that uses the change in intensity (amplitude) in the spectrogram to classify acoustic scenes and events[43].

In addition to spectrogram-based features, another signal-to-feature transformation is the wavelet transform. The wavelet transform is a windowed Fourier transform in the

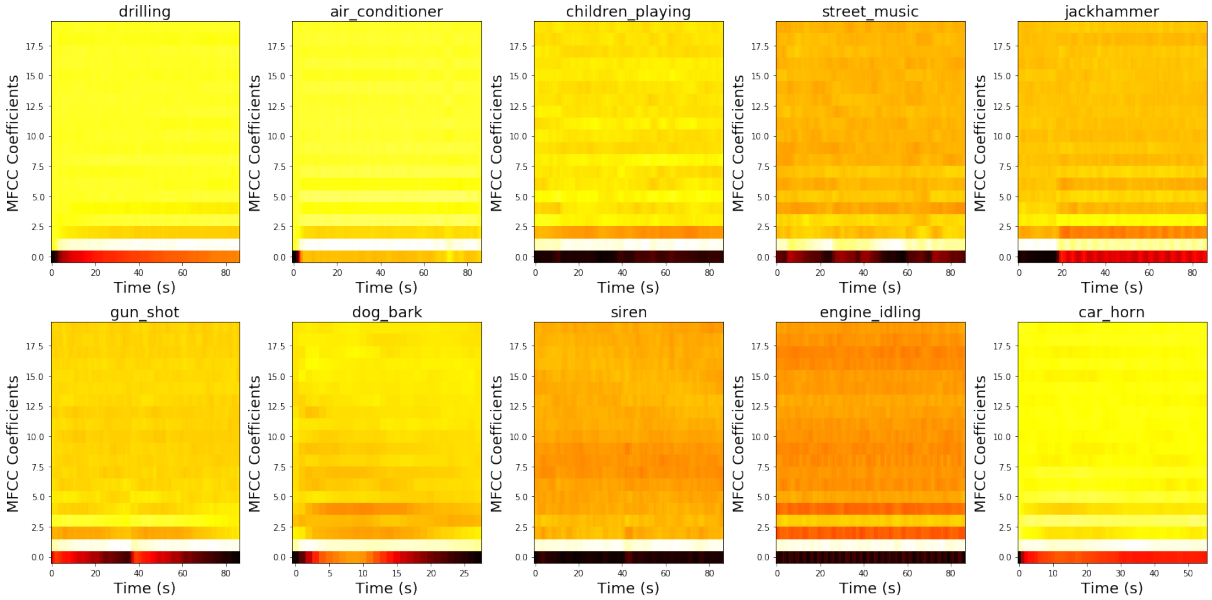


Figure 1.5 – Mel frequency cepstral coefficients of different environmental sounds.

time domain. Wavelet analysis provides the solution for analyzing non-stationary data [44] overcome limitations of STFT because the window is scaled in both time and frequency [45].

1.3 Machine Learning for Environmental sound classification

Machine learning is a part of artificial intelligence that deals with the development of algorithms or models that can learn from data and make predictions on new data. In machine learning, a computer is trained on a dataset and makes predictions or decisions without being explicitly programmed to perform the task. The goal of machine learning is to build models that can be generalized to new, unseen data, rather than just memorizing the patterns in the training data.

Machine learning methods have been widely used for the detection of sound events, which are described by the discrete occurrence of specific noise or sound patterns in an audio signal. These methods are comprised of training a model on a large dataset of audio examples, annotated by humans or machines, where the specific sound event has been identified or labeled. Once the model is trained, it is tested with an unseen audio

example to detect and classify the sound automatically. Machine learning approaches can be highly effective for sound event detection, as they can learn complex patterns and relationships in the data that may be difficult to capture using traditional, rule-based methods. There are a variety of machine learning algorithms that have been applied to sound event detection, including Support Vector Machines (SVM), random forests, and deep neural networks.

A formal definition for machine learning algorithm is provided by [46]:

Machine Learning

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

In this thesis, the task T is globally defined as an environmental sound classification in this thesis. For environmental sound classification, the experience, E, is often represented by the acoustic features accompanied by the labels of the audio recordings in the database. The acoustic features can be represented by an input matrix, X, which comprises of M acoustic features extracted from each frame of the audio recording divided into T time frames. The labels are encoded, generally one hot encoding, as a binary target output matrix Y, with C referring to the total number of classes of sound events. The target output labels are the annotations of the target sound input. If a particular sound event p is present in an audio recording then Y_p is set to 1, otherwise, it is set to 0. The reference labels are used in the machine learning model for classification. Then the performance of this model is evaluated using metrics such as accuracy and error rate discussed in the upcoming section.

The objective of the machine learning task is to develop a function F that can accurately map input data X to target output Y. In sound classification, the input is represented by X, the acoustic feature, and the output target is the label or the target output matrix Y. The function F takes an acoustic feature matrix and produces a probability vector $\hat{Y}$ indicating the likelihood of the sound event being present in that input. During the learning phase, the parameters of this function are adjusted to minimize the difference between the predicted output $\hat{Y}$ and the actual target output Y. In the usage phase, when the target output is either not available or being used for evaluation purposes, the

predicted output $\hat{Y}$ is used to identify the most likely sound event label in a given input. In the case of ESC, this is the sound event label with the highest probability as shown in Figure 1.6.

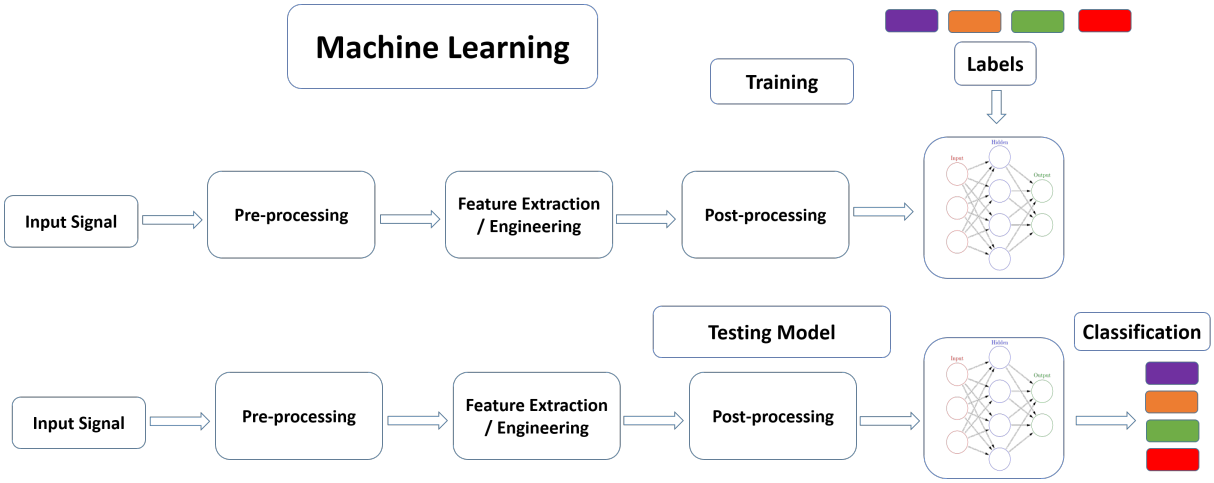


Figure 1.6 – Block diagram of machine learning method with neural networks.

1.4 Artificial Neural Networks

An artificial neural network (ANN) is a type of machine learning algorithm that is inspired by the structure and function of the human brain as depicted in Figure 1.7 [47]. The human brain is composed of billions of interconnected neurons, which are stimulated by various electrochemical signals in order to process and transmit information. ANNs attempt to replicate the structure and function of a biological neural network, but use a simplified set of concepts from biological neural systems. ANNs are specifically designed to simulate the electrical activity of the brain and nervous system and are composed of processing elements (also known as neurons or perceptrons) that are connected to each other in a way that emulates the connections between neurons in the brain. The processing elements in an ANN receive input data, perform a computation on the data, and transmit the output data to other processing elements or external destinations.

Different types of signals, such as sensory, audio, and visual signals, can stimulate different paths of neurons in the brain, and the signal information is processed through a collective set of neuron stimulations. From the moment of birth, the neurons in the brain specialize in processing certain types of signals and continuously improve their ability to

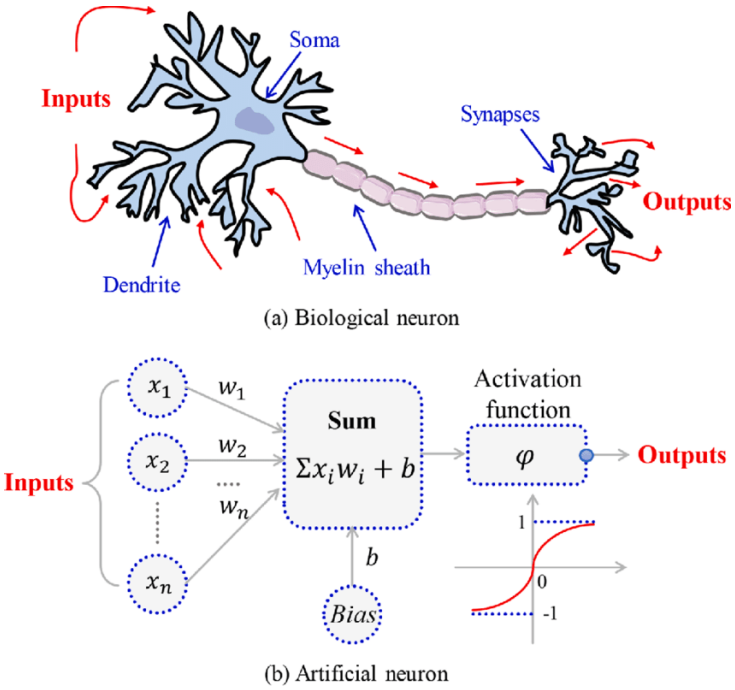


Figure 1.7 – Biological neuron and artificial neuron.

create a mapping between the input signal and its cognitive representation. For example, when we hear human speech, a special kind of audio signal used for communication, it is first transformed into electrochemical signals in the ear and brain. The cochlea is a spiral-shaped structure in the ear that is responsible for converting sound waves into electrical signals. These signals are then processed by the neurons in the brain through a series of stimulations, which map the signals to a set of phonemes, or units of sound, which have a shared cognitive representation for human communication.

ANNs attempt to replicate this process of information processing and mapping by using a set of interconnected processing elements, which receive input data and transmit output data based on the weights of their connections and the activation functions applied to the input data. Each neuron has certain parameters, such as weights and biases, which are adjusted iteratively through a process such as a gradient optimization in order to minimize an error function between the desired output and the estimated output. The layer that receives the input signal is called the input layer, the layer that determines the final output of the network is called the output layer, and the layers in between are called hidden layers. ANNs have a set of hyper-parameters that determine the network architecture, such as the number of hidden units and layers, and the training procedure,

such as the optimization method and regularization parameters. ANNs with multiple hidden layers are often referred to as deep learning or deep neural networks (DNNs). In this context, "deep learning" refers to all ANN methods that utilize multiple hidden layers. Deep learning has become increasingly popular in the field of machine learning due to the availability of large datasets, advanced training techniques, and increased computational power by GPUs.

1.4.1 Feed-forward Neural Networks

Feed-forward neural networks (FNN) are a type of artificial neural network that processes input data and generates output data in a single direction, without looping back. These networks, also known as fully connected networks or multi-layer perceptrons (MLPs), consist of layers of interconnected artificial neurons, or processing elements, that receive input data, apply computation based on the weights of their connections and activation functions and transmit output data to the next layer.

Each neuron in a feed-forward network receives input from multiple other neurons in the previous layer, performs a computation on the input data, and transmits the output to multiple neurons in the next layer. The input layer receives the raw input data, and the output layer generates the final output of the network. The layers in between the input and output layers are called hidden layers, and their purpose is to extract features and patterns from the input data that can be used to make predictions or classifications.

The calculation of FNN is calculated in two stages, first, the weighted sum $\mathbf{z}$ of the output of the previous connected layer of neurons is calculated and an additional bias $\mathbf{b}$ is added, and represented as:

$$z_j^{(l)} = \sum_{1 \leq k \leq n^{(l-1)}} w_{j,k}^{(l)} h_k^{(l-1)} + b_j^{(l)} \tag{1.3}$$

where $W_j^{(l)} = (w_{j,k}^{(l)})_{1 \leq k \leq n^{(l-1)}}$ represents the input weights in the neuron j from the layers *l*, $n^{(l-1)}$ is the number of outputs of the layer $(l-1)$, $h_k^{(l-1)}$ is the output for the neuron k in the layer *l-1*, $z_j^{(l)}$ is the weighted sum for the neuron j in l^{th} layer.

Secondly, a non-linear function is applied to the output of the neuron k in the layer $l-1$, to map the non-linear relationship in the input. In a neural network, non-linear functions are used to introduce non-linearity into the model. This is important because many real-world problems are non-linear in nature, and a model that is only capable of learning linear relationships will not be able to accurately capture the complexity of these

problems. The non-linearity function is defined as σ and the relation is defined as:

$$h_k^l = \sigma(z_k^{(l)}) \tag{1.4}$$

The non-linearity function is most commonly known as *activation function*, and this activation function empowers the neural networks to approximate complex functions. Let $h^{(l-1)} \in \mathbb{R}^{n^{(l-1)}}$ be input to the l^{th} layer (outputs of the $(l-1)^{th}$layer), and $h^{(l)} \in \mathbb{R}^{n^{(l)}}$ is its output, then the output of the neuron k in the layer l is defined as :

$$h_k^{(l)} = \sigma(W_k^{(l)} \left(h^{(l-1)}\right)^T + b_k^{(l)}) \tag{1.5}$$

and $h^{(l)} \in \mathbb{R}^{n^{(l)}}$ is defined as:

$$h^{(l)} = \sigma(\boldsymbol{W}^{(l)} \left(h^{(l-1)}\right)^T + b^{(l)})$$

where

$$\boldsymbol{W^{(l)}} = (w_{j,k}^{(l)}) \quad \begin{matrix} 1 \leq j \leq n^{(l)} \\ 1 \leq k \leq n^{(l-1)} \end{matrix}$$

$$\boldsymbol{b} = (b_j^{(l)})_{1 \leq j \leq n^{(l)}}$$

Non-linear functions that are frequently used include the logistic sigmoid function (*sigm*), the hyperbolic tangent function (*tanh*), and the rectified linear unit function ($ReLU$). These functions are element-wise, meaning that they operate on individual elements in an array or matrix, and their equations are discussed in the following section.

Feed-forward neural networks are characterized by the fact that information flows through the model in a single direction, from the input $\mathbf{x} \in \mathbb{R}^n$ to the output $\mathbf{y} \in \mathbb{R}^m$, without any feedback connections. In these models, the output is determined by the intermediate computations used to define the function, [48]:

$$f : \begin{array}{ccc} \mathbb{R}^n & \rightarrow & \mathbb{R}^m \\ \boldsymbol{x} = (x_1, ..., x_n) & \rightarrow & \boldsymbol{y} = (y_1, ..., y_m) = f(x) \end{array}$$

The goal of a feed-forward network is to find the appropriate parameters (weight and biases: $\boldsymbol{W}$ and $\boldsymbol{b}$) that result in the best function approximation. The weight $\boldsymbol{W}$ and biases $\boldsymbol{b}$ are defined as:

$$\boldsymbol{W} = (w_{j,k}^{(l)}) \quad \begin{matrix} 1 \leq j \leq n^{(l)} \\ 1 \leq k \leq n^{(l-1)} \\ 1 \leq l \leq L \end{matrix}$$

$$\boldsymbol{b} = (b_{j}^{(l)}) \quad \begin{matrix} 1 \leq j \leq n^{(l)} \\ 1 \leq l \leq L \end{matrix}$$

So the variables are the parameters, that we denote for simplification $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b})$ and the feed-forward network can be defined as :

$$\widehat{f}: \begin{array}{ccc} \mathbb{R}^{n+p} & \rightarrow & \mathbb{R}^{m} \\ (\boldsymbol{x}, \boldsymbol{\theta}) & \rightarrow & \boldsymbol{y} = (y_1, ..., y_m) = \widehat{f}(\boldsymbol{x}, \boldsymbol{\theta}) \end{array}$$

where, n is the number of input and p is *parameters* and $\widehat{f}$ represent the neural network model. A feed-forward neural network is illustrated in Figure 1.8. For the model shown in the figure, the model takes the input feature vector, in our case acoustic feature per time frame, as input and calculates the probability for each fourth output nodes, sound classes present in the acoustic feature in our case. If the target output vector y is binary-encoded (common in SED tasks), the weighted sum of each neuron in the output layer is passed through an activation function that bounds the output between 0 and 1, allowing the network output $\hat{\boldsymbol{y}}$ to be interpreted as the estimated probabilities of the sound events being present in the frame.

Feedforward neural networks play a crucial role in the field of machine learning and are the foundation of many practical applications. For instance, *convolutional neural networks*, which are used for object recognition from images, are a type of *feed forward network*. Feedforward networks are a fundamental concept that leads to the development of recurrent neural networks, which are used in natural language processing applications.

When feed-forward networks are extended to include feedback connections, they become recurrent neural networks, which will be discussed in a later section.

1.4.2 Recurrent Neural Networks

Recurrent neural networks (RNN) are a type of artificial neural network that is specifically designed to handle sequential data. This quality has made RNNs popular in the

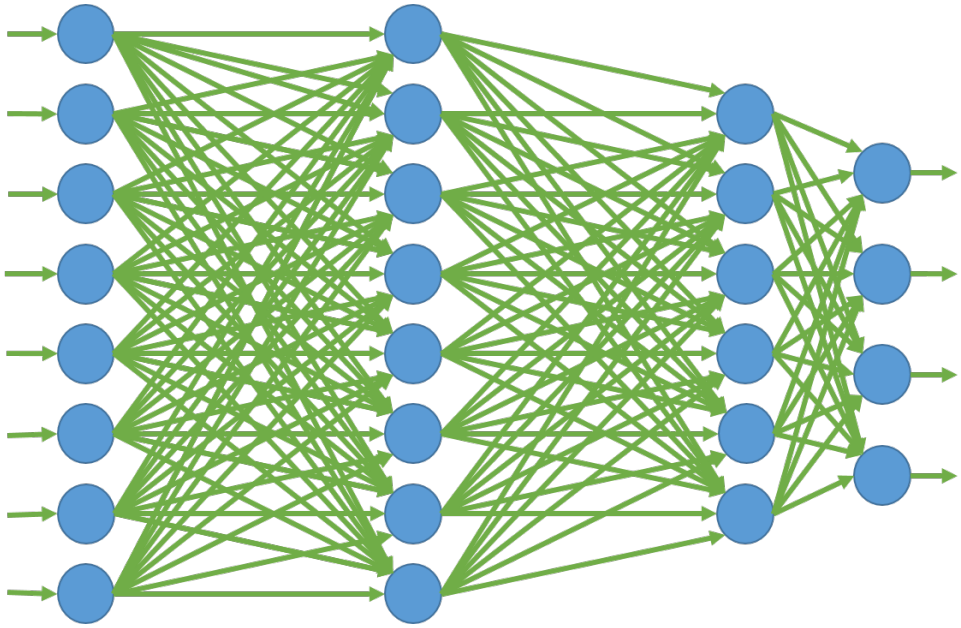


Figure 1.8 – Feed forward neural network. First layer is the input layer, followed by two hidden layers and a output layer at the end.

tasks related to language translation, speech recognition, and natural language processing [49]–[56]. In a deep neural network, the input data is typically transformed into a fixed-length feature vector. This feature vector is passed and processed by the hidden layers before generating class estimation at the output layer. This means that at each time step, the prediction made by the $\mathbf{y}$ is only based on the input $\mathbf{x}$ at present and does not take into account any context from the previous input. This can be problematic for certain models where context is important. In contrast, RNN processes the input data element one at a time while keeping the states of elements passed before in the sequence by using hidden layers or hidden states. In RNN, the value of each hidden layer depends not only on the values of the layers below it at the current time step but also on the value of the same layer at the previous time step. The value of the hidden layer, represented by $h_t^{(l)}$ of the $l-th$ layer at time t, is given by:

$$h_t^{(l)} = \sigma(U^{(l)} h_{t-1}^{(l)} + W^{(l)} h_t^{(l-1)} + b^{(l)}) \tag{1.6}$$

Here, $U^{(l)}$ matrix is the *recurrent weight matrix*. At each time step, the RNN cell processes the current input and the previous hidden state, producing a new output and updating the hidden state with the new information as shown in Figure 1.9 [57]. This

process is repeated for each element in the sequence, allowing the RNN to build up a representation of the entire sequence over time.

RNNS processes an input sequence in a single direction, using information from past contexts to calculate the output for the current time step. In some cases, it may be beneficial to also process the input sequence in the opposite direction in the future context. In this case, *bidirectional RNN* have been proposed [58]. In addition to the forward chain in the hidden layer in RNN, a backward chain is also included. Both chains of the hidden layer are connected to the forward and backward chains in the next hidden layer. Let $\overrightarrow{h_t^{(l)}}$ represent the output of the $l-th$ layer in forward direction, $\overleftarrow{h_t^{(l)}}$ represent the output of the l^{th} layer in backward direction and $h_t^{(l)}$ represents the concatenation of the two. The relation is described by :

$$\overrightarrow{h_t^{(l)}} = \sigma(\overrightarrow{U}^{(l)}\overrightarrow{h}_{t-1}^{(l)} + W^{(l)}h_t^{(l-1)} + \overrightarrow{b}^{(l)}) \tag{1.7}$$

$$\overleftarrow{h_t^{(l)}} = \sigma(\overleftarrow{U}^{(l)}\overleftarrow{h}_{t-1}^{(l)} + W^{(l)}h_t^{(l-1)} + \overleftarrow{b}^{(l)}) \tag{1.8}$$

Here, $\overrightarrow{U}^{(l)}$ and $\overrightarrow{b}^{(l)}$ are the forward recurrent weights and biases. Similarly, $\overleftarrow{V}^{(l)}$ and $\overleftarrow{b}^{(l)}$ are the weights and biases in the backward direction. A bidirectional RNN has access to the entire input sequence when making a prediction at any time step, providing it with unlimited context.

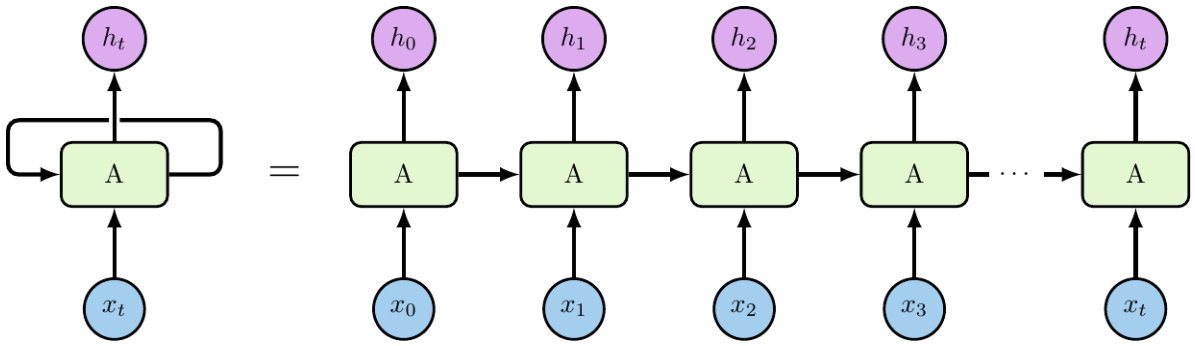


Figure 1.9 – Recurrent Units in Recurrent neural networks.

RNNs face difficulty in modeling long-term dependencies, as the impact of past time steps on the current output tends to decrease exponentially over time due to the *gradients vanishing* problem. This limits the ability of RNNs to effectively incorporate long temporal context, despite their theoretical potential to do so [59].

To address the issues with traditional RNNs, gated recurrent layer methods such as

g*ated recurrent units* (GRUs) and *long short-term memory networks* (LSTMs) were introduced. These methods use units called cells, which combine multiple gate activations to produce their output. LSTMs have external input, forget, and output gates, while GRUs have an update and reset gates. These gates, which are made up of weights and an activation function, allow the cells to accumulate and selectively preserve information from past time steps in a cell state. During training, the gate weights learn how to combine the cell state and the input for the current time step to produce the gated unit output for the current time step.

The cell structure of LSTM and GRU are shown in figure 1.10(a) and 1.10(b) respectively [57]. The three types of gates in an LSTM network are the input gate, forget gate, and output gate. These gates use the following equations to determine how to update the state of the network at each time step:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \tag{1.9}$$

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \tag{1.10}$$

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \tag{1.11}$$

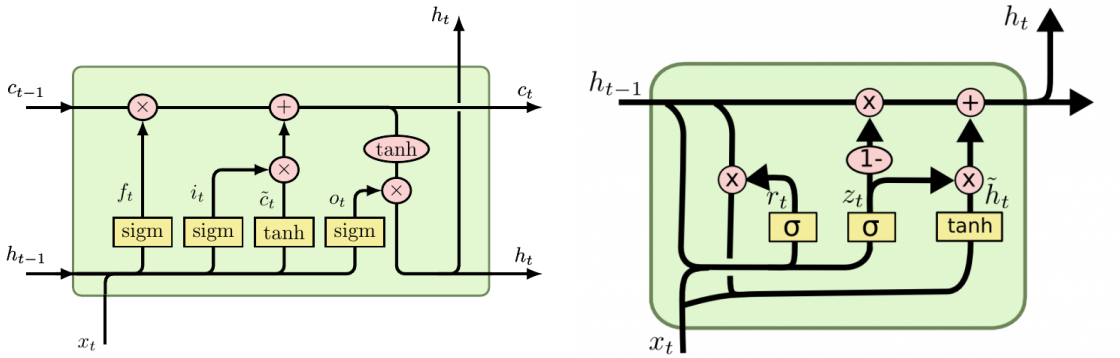
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \tag{1.12}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{1.13}$$

where,

- x_t is the input at time step t
- h_{t-1} is the hidden state at time step $t-1$
- i_t, f_t, o_t are the input, forget, and output gates, respectively, at time step t
- W, U, and b are learn-able weight matrices and biases
- c_t is the cell state at time step t
- c_{t-1} is the cell state at time step $t-1$
- h_t is the hidden state at time step t
- σ is the logistic sigmoid function

Both LSTMs and GRUs are able to capture long-term dependencies in data, however, they differ in a few aspects as shown in Fig.1.10. LSTMs have three types of gates (input, forget, and output gates) while GRUs have only two (reset and update gates). LSTMs



(a) Long short term memory cell

(b) Gated Recurrent unit cell

Figure 1.10 – LSTM and GRU Cell.

have a separate cell state that is updated using the input, forget, and output gates, while GRUs do not have a separate cell state and update their hidden state directly using the reset and update gates. LSTMs have more parameters than GRUs, which can make them more difficult to train and may require more computational resources. In general, LSTMs and GRUs perform similarly on many tasks, but LSTMs may be more suitable for tasks that require long-term dependencies to be captured over longer sequences, while GRUs may be more suitable for tasks that require the model to learn more quickly.

1.4.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep neural network that performs relatively well for image classification and recognition tasks. They are able to learn the important features and objects in an image by assigning weights and biases to different aspects of the input data. The design and architecture of CNNs are inspired by the structure of the visual cortex in human and animal brains. Each neuron is responsible for processing a specific region of input data, known as *receptive field* that is sensitive to the specific patterns in the visual environments. The receptive field of multiple neurons overlaps to cover the entire input data.

The fundamental component of CNN is a *convolutional layer* inter-weaved with *pooling layers*, which are responsible for extracting features from the input data. The convolutional layer contains a collection of filters, also called kernels or weights, which are scanned over the input data and detect specific patterns. Each filter is a small matrix of weights that is applied to a small region of the input data. The output of this process is a *feature map*,

which is the representation of the responses of the filters at each location in the input data.

In a multi-layer convolutional network, the output of one layer serves as the input for the next layer. This output typically consists of the results of multiple convolutions at each position. When working with images, we typically represent the input and output of the convolutional operation as 3-dimensional tensors, with indices for the different channels and the spatial coordinates within each channel. In practice, software implementations of convolutional networks often use batch processing, which involves 4-dimensional tensors with an additional index for the different examples in the batch. For simplicity, we will describe the operations here without considering the batch axis.

The convolution operation between a filter $\mathbf{F}$ and an input image $\mathbf{X}$ with height h, width w, and number of channels c can be represented mathematically as:

$$H_{i,j,k} = \sigma\left(\sum_{c=1}^{C}\sum_{m=1}^{S}\sum_{n=1}^{S} X_{s\cdot i+m,s\cdot j+n,c}\cdot F_{m,n,c,k} + B_k\right) \tag{1.14}$$

Where $H_{i,j,k}$ is the activation at position (i, j) in the output feature map for channel k, C is the number of channels in the input image, S is the size of the filter, s is the stride, B_k is the bias for channel k, and $F_{m,n,c,k}$ is the entry of the filter F at position (m, n) for channel c and channel k. The double summation over m and n is taken over the height and width of the filter, while the summation over c is taken over the number of channels in the input image. The i and j indices indicate the position of the filter in the input image.

The final output feature map has shape (h', w', k), where h' and w' are the height and width of the output feature map, and k is the number of channels in the output feature map. Where σ is the activation function. The kernel slides over the input to extract the local representation and is generally smaller in size as compared to the input as visualized in Figure 1.11.

A pooling layer operates by dividing the input feature map into non-overlapping regions, each containing $m \times n$ pixels. The stride of the pooling layer is equal to $m \times n$. For each region, the pooling layer computes a statistic, which can be either the maximum or the average value. These are the most commonly used statistics in pooling layers.

When applied to image recognition tasks, a neural network typically makes a prediction for an entire image represented as 1 or 3 input feature maps (for gray-scale or color images, respectively). The layers of the network are usually arranged in a way that in-

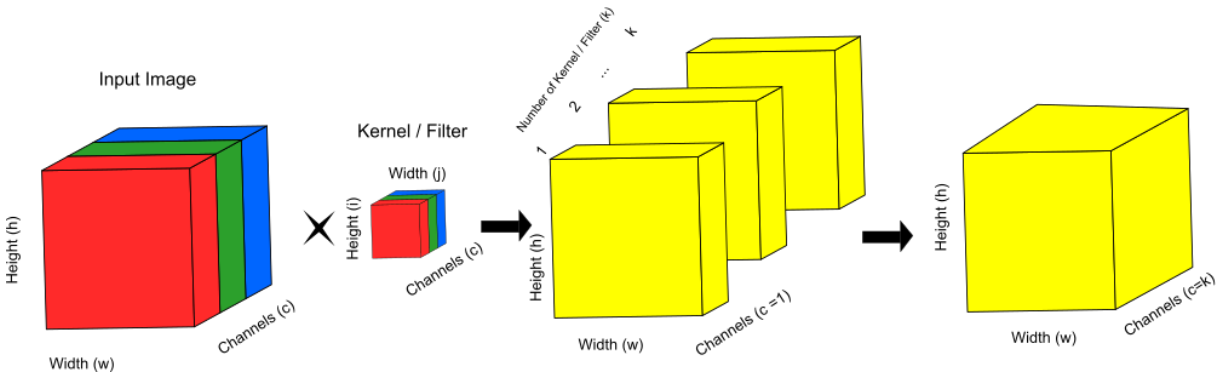


Figure 1.11 – Convolution operation in convolutional neural networks (CNN)

volves increasing the number of feature maps through the use of convolutional layers and decreasing the size of the feature maps through the use of pooling layers. Once the feature maps become small enough, they are often flattened into a single vector and processed by fully connected layers to make the final prediction.

The Convolutional layers are applied to the feature image, time-frequency representation for sound, along with pooling layers. The image is transformed into a sequence by subsequent application of convolutional and pooling layers as shown in Figure 1.12. This sequence is used as input to fully-connected neural networks. In contrast to fully-connected neural networks, where each input feature is connected to a hidden unit in the next layer, convolutional layers use shared parameters among the input features and train the kernel parameters to learn local patterns that can be found anywhere in the input. This is particularly useful for ESC, where a sound event should be detected from a spectrogram regardless of its position in time. Additionally, convolutional layers, which typically consist of tens or hundreds of kernels, are often more memory-efficient than fully-connected neural networks due to the reduced number of weights. Convolutional layers in ANNs are typically used to identify important features. These features are then introduced into another layer in the ANN. If this input needs to be in the form of a vector, such as for use in an RNN, the features from each feature map are combined along the frequency axis to form a single vector for each time step.

CNNs have several advantages, including shift in-variance and locality. Shift in-variance means that the prediction for an image should not change if the object of interest moves within the image. This property is also desirable for audio signals, as a phoneme or sound event should be recognized regardless of its position in the audio signal, and a limited shift

along the frequency axis should not affect the prediction. CNNs achieve shift in-variance by applying the same convolution kernel to all parts of the input. Locality means that the network has a sense of which parts of the input are close to each other and which parts are far apart. This is achieved through the use of neurons that only receive information from neurons representing a neighboring region in the lower layer. As a result of locality, CNNs do not have the unlimited context that RNNs do when applied to audio. However, tasks involving audio may not require unlimited context, as audio usually does not exhibit long-range dependence in the same way that natural language does.

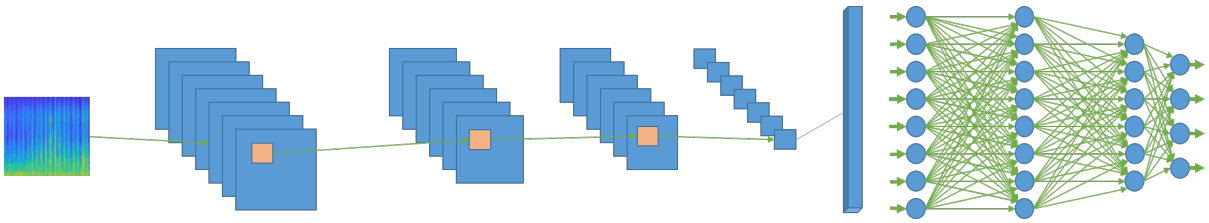


Figure 1.12 – Convolutional neural network (CNN) on a spectrogram.

1.4.4 Depthwise separable convolution networks

A depthwise separable convolutional network (DS-CNN) is a type of CNN architecture that aims to reduce the number of parameters and computation required by a traditional CNN. This is achieved by breaking a standard convolutional layer into two separate layers: a depthwise convolution layer and a pointwise convolution layer [60].

The depthwise convolution layer applies a single filter to each input channel, rather than applying the same filter across all channels as in a traditional CNN. This results in a set of feature maps, one for each input channel. Let the input image tensor be denoted as $\mathbf{X}$ with dimensions (h, w, c) where h is the height, w is the width, and c is the number of channels. The first step of the DSCNN operation is the depthwise convolution operation, which is defined as follows:

$$\mathbf{Z}_{i,j,k} = \sum_{m=0}^{k-1} \sum_{n=0}^{f_h-1} \sum_{p=0}^{f_w-1} \mathbf{X}_{i-n,j-p,m} \cdot \mathbf{F}_{n,p,m,k} \tag{1.15}$$

Where $\mathbf{Z}$ is the intermediate feature map with dimensions (h, w, c), $\mathbf{F}$ is the depthwise filter with dimensions $(f_h, f_w, c, 1)$, i and j are the indices for the height and width respectively, k is the channel index, and m, n, and p are the indices for the filter height, width, and channel, respectively. As shown in Figure 1.13.

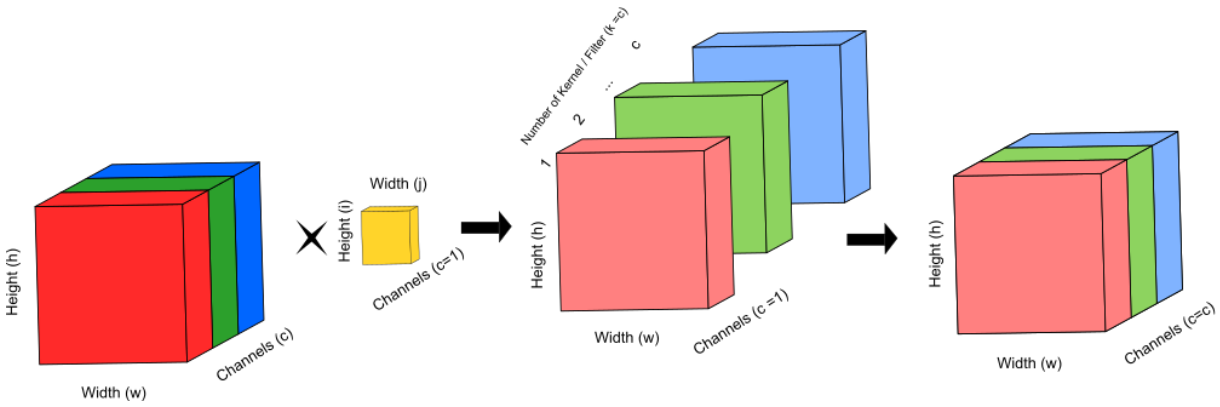


Figure 1.13 – Depthwise convolution operation on an input image.

The pointwise convolution layer then applies a 1x1 convolution to the set of feature maps output by the depthwise convolution layer. This combines the information across all channels and produces a single output feature map, which is defined as follows::

$$\mathbf{Y}_{i,j,k} = \sum_{m=0}^{c-1} \mathbf{Z}_{i,j,m} \cdot \mathbf{G}_{m,k} \tag{1.16}$$

Where $\mathbf{Y}$ is the output feature map with dimensions (h, w, k), $\mathbf{G}$ is the pointwise filter with dimensions (c, k), and k is the index for the number of output channels, depicted in Figure 1.14.

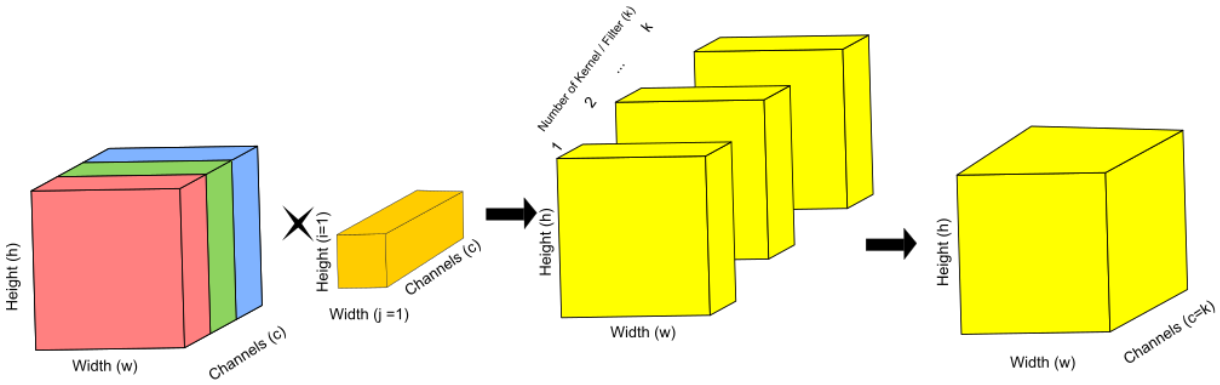


Figure 1.14 – Pointwise convolution operation on an input image.

The use of depthwise separable convolutions can greatly reduce the number of parameters in a CNN, as the number of parameters in the depthwise layer is equal to the number of input channels, while the number of parameters in the pointwise layer is equal

to the number of input channels times the number of output channels. This can lead to faster training and better performance on some tasks.

1.5 Neural network model training and evaluation

1.5.1 Training of Model

Training of a neural network model involves the adjustment of the model's weights and biases to minimize the error between the predicted output and the true output based on the dataset presented. This is typically done using an optimization algorithm, such as stochastic gradient descent (SGD), which adjusts the weights and biases iteratively to minimize the loss function. The loss function is a measure of the difference between the predicted output and the true output, and the optimization algorithm attempts to find the best combination of weights and biases that minimizes this difference.

The parameters of the neural network θ are typically initialized with small, randomly generated values drawn from a normal distribution. The input data is then passed through the network, and the output of the network is calculated $\hat{y} = f(x, \theta)$. This process is known as forward pass or forward propagation. To evaluate how close the predicted output $\hat{y}$ is to the true output y, we calculate the loss $l(y, \hat{y})$ using a loss function, such as mean squared error or cross-entropy. It is important that the loss function is non-negative and differentiable everywhere, as we use gradient-based optimization algorithms to update the network parameters.

1.5.2 Optimization by Gradient Descent Based Optimization Algorithms

To train a neural network, we need to find the values for the model's parameters that minimize the loss function [48]. There are various optimization algorithms that can be used for this purpose, and many of these algorithms rely on the gradient of the loss function with respect to the parameters.

The gradient ∇l of the loss function with respect to the parameters θ, which includes the weight and biases, is calculated and used to update the parameters according to the gradient descent algorithm.

$$\nabla l = \left(\frac{\partial l}{\partial \theta}\right) \tag{1.17}$$

The gradient can be calculated using *back-propagation* algorithm [61], which involves applying the chain rule of differentiation repeatedly. Many deep learning frameworks, such as Theano [62], TensorFlow [63], PyTorch [64], can perform back-propagation automatically, so it is not necessary to derive the gradient formulas manually. One such algorithm is *gradient descent*, which involves iteratively computing the gradient of the loss function and updating the parameters by subtracting the gradient multiplied by a learning rate. The change in the loss l denotes as Δl can be calculated by multiplying it with the change $\Delta\theta$ in the parameters θ as:

$$\Delta l \approx \nabla l \cdot \Delta\theta \tag{1.18}$$

The objective of the gradient descent is to minimize l, by updating Δl based on $\Delta\theta$, which would make the change in loss Δl negative. We use the negative sign in the relation:

$$\Delta\theta = -\eta\Delta l \tag{1.19}$$

Where η is the learning rate and is always greater than zero, $\eta > 0$.Substituting 1.19 in 1.18 gives us:

$$\Delta l \approx -\eta||\nabla l||^2 \tag{1.20}$$

This entails that $\Delta l \leq 0$ and therefore l will decrease. The weights and biases will be updated as:

$$\theta \leftarrow \theta + \Delta\theta = \theta\eta\nabla l \tag{1.21}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}} \tag{1.22}$$

$$b \leftarrow b - \eta\frac{\partial l}{\partial b} \tag{1.23}$$

Stochastic Gradient Descent (SGD)

In order to update the weights and biases one option is to calculate the loss for each input after the forward pass and then calculate the average loss [48]. This method is known as *batch gradient descent*. In this method, all training input needs to be forward propagated before any parameter update. In contrast to full batch gradient descent, which uses the entire dataset to calculate loss and update the network's parameters at once, *stochastic gradient descent* uses a smaller, randomly selected subset of the data to compute the average loss and makes updates. This is typically done iteratively until all data has been used for training. This method is more common, especially during the early stages of training a neural network when the model is less accurate and requires more frequent updates to improve its performance.

In SGD, the training input data is converted into *mini-batches*. Updating the network's parameters based on the gradients calculated from each mini-batch helps prevent the model from getting stuck in sub-optimal solutions. The process of going through the entire training dataset is called an *epoch*. To reduce the risk of the model learning invalid patterns from the order of the mini-batches, it is common to shuffle them. The learning rate is typically adjusted after each epoch, and when the training data is very large, additional validation steps may be inserted, with one epoch containing multiple checkpoints.

In the SGD optimization method, the network is trained for a set number of epochs, typically ranging from 100 to 500 for sound event detection tasks. As the training progresses through each epoch, the updates to the network's parameters result in different loss values for the same input. When the updates become sufficiently small, the network's parameters converge and the training process is terminated.

Momentum

Due to the increasing complexity of deep neural networks (DNNs) and the use of large datasets, the efficiency and speed of training networks using SGD have become a concern. DNNs often have millions of parameters, making the process of calculating gradients and updating parameters time-consuming. The vanishing gradient problem can also cause the number of updates to be smaller for parameters in lower layers compared to those in higher layers, even with a fixed learning rate. In addition, large datasets, which are beneficial for training complex networks such as DNNs, can take a long time to process. To speed up the training process of DNNs another commonly used technique *momentum* is used.

In the momentum optimization method, the update to the network's parameters takes into account not only the gradient from the current mini-batch but also the total update from previous iterations. It does this by adding a fraction of the update from the current iteration to the update from the previous iteration. This can help the model escape from local optima and saddle points, and can also help the optimization algorithm to continue moving in the same direction when making progress. Let δ_{i+1}, be the difference between the parameter and after $(i+1)^{th}$ minibatch, then momentum with SGD can be described as:

$$\theta_{i+1} = \theta_i + \delta_{i+1} \tag{1.24}$$

$$\delta_{i+1} = \mu\delta_i - \eta\nabla l \tag{1.25}$$

The momentum coefficient, often denoted as $\mu \in (0, 1)$, determines the weight of the previous update in the current update. A high μ value means that the previous update will have a large influence on the current update, while a low μ value means that the previous update will have a small influence. Momentum helps optimization algorithms navigate narrow ravines in the loss function by adding a component of the gradient from previous iterations to the current iteration. This can allow the learning rate to be set to a higher value, leading to faster convergence. Without momentum, the learning rate must be set to a smaller value to avoid oscillation in the ravine, which slows progress.

Adaptive Moment Estimation (Adam)

Adjusting the learning rate η is one of the most crucial steps in fine-tuning the hyper-parameters for the neural networks. The learning of the neural network is directly affected by the learning rate. Fixed learning rate, as in SGD, means that the optimization is heavily influenced by the chosen learning rate. However, it is more effective to have larger updates at the beginning of training when the network is not yet familiar with the task and tends to make larger errors in its output estimates, and smaller updates as training progresses and the network only needs minor adjustments to the parameters before convergence. There are optimization algorithms available that adjust the learning rate during training to address the issue of having a fixed learning rate in SGD. One example is *Adam* [65], which takes into account both the first and second moments of the gradient and includes bias correction to reduce bias early in training. Adam has gained popularity among deep learning researchers due to its strong performance across a range of hyper-parameter values.

Activation Functions

In equation 1.4 we introduced the use of activation function a non-linear function denoted by σ. The output of h_k^l of the neuron k in the layer l is obtained by applying the activation function to the weighted sum of the neuron outputs for the layer $(l-1)$. Commonly used non-linear functions including element-wise functions such as the logistic sigmoid function (sigm), the hyperbolic tangent function (tanh), and the rectified linear unit function (ReLU). These functions are shown in Figure 1.15 and are defined as:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{1.26}$$

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{1.27}$$

$$Relu(z) = max(0, z) \tag{1.28}$$

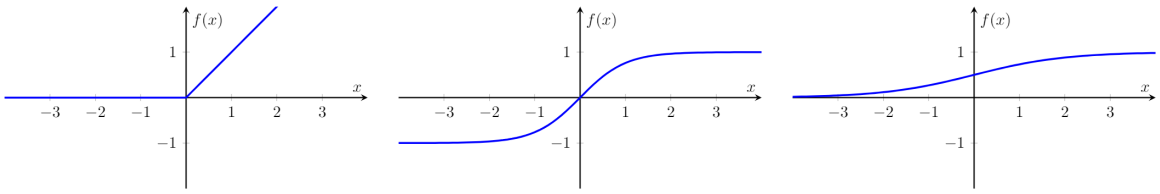
For all the layers, the choice of activation function is flexible except for the output layer. The activation function of the output layer must be chosen based on the task that the network is attempting to solve. If the task is regression, then the identity function should be used. If the task is binary classification, the logistic sigmoid function should be used to generate probabilities between 0 and 1. If the task is multi-class classification, a non-element-wise softmax function $\sigma(z_i)$ should be used to create a probability distribution. let $z_1, z_2, \ldots, z_k$ the elements of the vector input to the softmax function, then the $i-th$ component of the output will be calculated as :

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{1.29}$$

The $\sigma(z_i)$ gives a probability distribution at the output as it can be easily demonstrated that $\sigma(z_i) > 0$, $\forall i$ and $\sum_{i=1}^{K} \sigma(z_i) = 1$.

Hyper-parameter Fine Tuning

Hyper-parameter tuning involves adjusting specific parameters in a machine learning model to improve its performance on a given dataset. These hyper-parameters, which are set before training the model, are not determined by the data and can include options like the learning rate or the number of hidden units in a neural network. There



(a) Relu activation function (b) Tan hyperbolic activation function (c) Sigmoid activation

Figure 1.15 – Relu, Tan hyperbolic function, and sigmoid activation function response.

are different methods for tuning hyper-parameters, such as manually adjusting them and evaluating the model's performance, using a grid search to test all possible combinations of hyper-parameters, or sampling random combinations using random search. While hyper-parameter tuning is crucial for maximizing a model's performance, it can be time-consuming and resource-intensive.

1.5.3 Network Regularization

Regularization is a technique used in machine learning models, especially neural networks, to prevent over-fitting. Over-fitting occurs when the model is too complex and has too many parameters, fitting the noise or random variations in the training data rather than the underlying pattern. This can lead to poor generalization performance for new unseen data. We used two most commonly used regularization techniques: Dropout and batch normalization.

Dropout [66] is a technique used in neural networks to prevent over-fitting by randomly "disabling" or dropping out a subset of the neurons during training. During training, randomly a fraction of the activations of the hidden units to zero, typically done with a probability of around 0.1 to 0.5. The probability of dropping out of the activation of each hidden unit is randomly sampled for each epoch of training. This forces the network to learn more robust features that are useful in different contexts, reducing complexity and improving generalization. During test time, all neurons are used but the activations are scaled down to compensate for the dropout rate used during training. Dropout is commonly used in combination with other regularization methods and has been shown to be effective in deep neural networks.

Batch normalization [67] is a technique used to speed up the training of deep neural networks and improve their ability to generalize. It works by normalizing the activations

of neurons within layers for each mini-batch during training. Neuron activation values are typically distributed around a mean and vary in magnitude. This can cause problems in the optimization process. This is because the gradient of the weights can be very small or very large, depending on the range of activations. This can slow down training and makes the gradient disappear or explode. Batch normalization addresses this problem by normalizing neuronal activations so that each mini-batch has a mean and unit variance of zero. This results in a stable distribution of activations, more consistent gradients, faster training, and better generalization. Batch normalization is usually applied after linear transformations of activation and before nonlinear activation functions.

1.5.4 Evaluation of sound classification model

Evaluating the performance of a proposed method is essential to determine its effectiveness. This evaluation should use metrics that reflect the real-world application of the method. Using standardized performance metrics makes it easier to compare and measure the progress made by the proposed method.

1.5.5 Performance Metrics

To evaluate the performance of the ESC system, a performance metric is calculated using the binary detection outputs and the target outputs for a test or evaluation set. Commonly used performance metrics for ESC include *accuracy*, *precision*, *recall* (also called the true positive rate), *F1 score*, *error rate*. For polyphonic sound event detection, metrics such as accuracy are less effective to evaluate the performance, and error rate is generally used. Discussed in more detail in [68].

To evaluate the performance of the ESC system, a set of intermediate statistics, including true positives, true negatives, false positives, and false negatives, are calculated. These statistics can be used to calculate performance metrics at either the segment or event level.

A true positive is when both the reference and system output show that an event is happening during a specific time period. A false positive is when the reference says that an event is not happening, but the system output says it is. A false negative is when the reference says an event is happening, but the system output says it is not. True negatives are when both the reference and system output shows that an event is not happening. The total number of true positives, false positives, false negatives and true negatives are

represented as TP, FP, FN, and TN, respectively.

Accuracy

Accuracy is a measure of the classifier's performance, calculated as the percentage of correct predictions made by the classifier out of the total number of predictions. It indicates how often the classifier is able to correctly identify the class of an input sample.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.30}$$

Precision

Precision is a measure of the accuracy of the detection made by the system. It is calculated as the ratio of the number of correctly detected examples to the total number of detection made by the system.

$$P = \frac{TP}{TP + FP} \tag{1.31}$$

Recall

Recall, also known as sensitivity or true positive rate, is a measure of a classifier's performance that indicates the proportion of positive instances that were correctly identified by the classifier.

$$R = \frac{TP}{TP + FN} \tag{1.32}$$

F1 Score

The F1 score is a measure of a classifier's performance that combines precision and recall. It is calculated as the harmonic mean of precision and recall, with a higher score indicating better performance. The F1 score is defined as:

$$F1 = \frac{2.P.R}{P + R} \tag{1.33}$$

The F1 score is useful for evaluating the performance of a classifier because it can be applied to both single-label and multi-label classification tasks and considers both precision and recall equally. However, the F1 score does not take into account true negatives

in its calculation.

1.5.6 Cross Validation

Cross-validation is a statistical method used to evaluate the performance of a predictive model by partitioning the original sample into a training set to train the model, and a test set to evaluate it. Cross-validation is an important aspect of comparing and reproducing results in experiments. It is important to carefully set up the cross-validation procedure to ensure that all classes are represented in each fold, as missing classes can result in errors in the calculation of some performance metrics [69]. The model is trained on the training set and then tested on the test set. This process is repeated a number of times, with each partition serving as the test set once as depicted in Figure 1.16 [70]. The average performance across all iterations is used to assess the model's performance. It is important to treat the cross-validation folds as a single experiment and only calculate the final metrics after testing all the folds. Cross-validation helps to reduce the risk of over-fitting, as the model is trained and evaluated on different data each time. It is a widely used technique for model selection and evaluation.

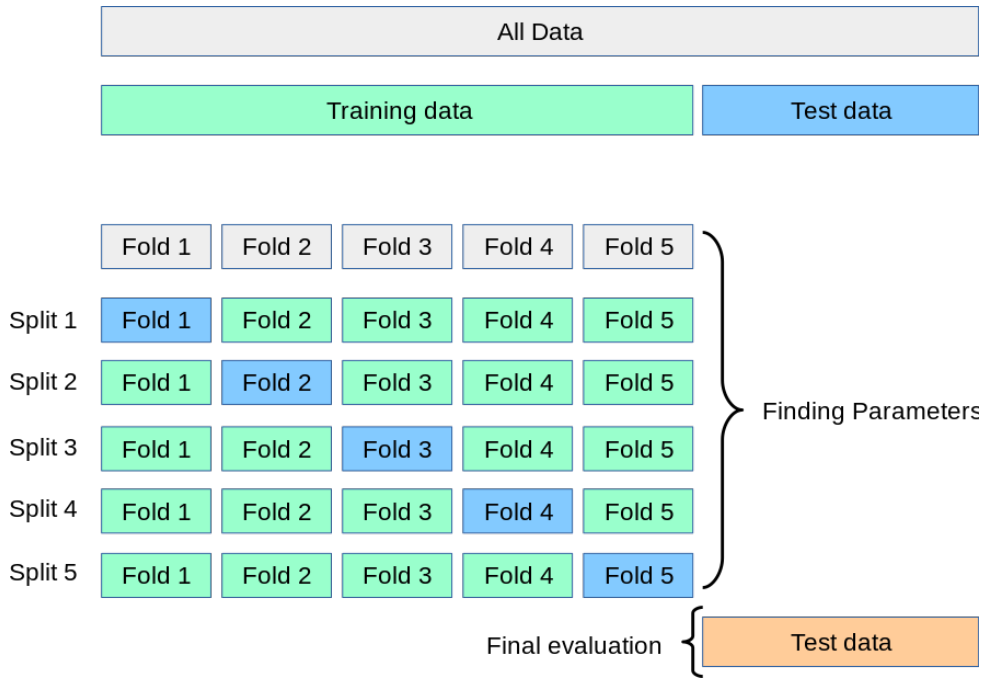


Figure 1.16 – Multi fold cross-validation setup.

1.6 Datasets

The dataset used to train and evaluate the ESC method should be diverse and representative of the range of ESC tasks it is intended to handle. This dataset can be created using sounds recorded in real-world environments and annotated manually, or by synthesizing sounds using individual sound events and mixing them to mimic real-world conditions. Synthesizing the dataset has the advantage of providing more accurate annotations, but it can be challenging to create sound event mixtures that accurately simulate a real-world recording.

The database consists of the audio recording and their associated labels. There are two types of labels that describe the granularity of the labels: strong labels and weak labels. In strong labelling, the desired sound and its time stamp are both present, for example, in an audio recording the time stamps of the start and end of the dog bark is presented along with the label. In weak labelling, only the general label is present without any information about the locality of the sound event present in the audio recording. Obtaining strong labels is a difficult and arduous task and is required for overlapping sound events. In this thesis, weak-labelled sound databases are used. The databases are described in the following section. The usage of these datasets for the development of proposed methods is discussed in the experimental section of the following chapters.

1.6.1 Acoustic Scene Classification Dataset

Detection of an acoustic scene has been considered a complex problem by the research community for several years, and various efforts have been dedicated to solving this issue. Acoustic scenes contain acoustic events in a particular environment such as metro stations, airports, train stations, etc. Classifying these categories becomes complex due to the similar nature of the sound events occurring in those environments. To solve this issue, the Detection and Classification of Acoustic Scenes and Events (DCASE) provided a dataset that contains audio recordings of 10 different categories: airports, indoor shopping malls, metro stations, pedestrian streets, public squares, streets with a medium level of traffic, traveling by train, traveling by bus, traveling by an underground metro, and urban park (TUT Urban Acoustic Scenes 2018 dataset) [71].

1.6.2 Low-Complexity Acoustic Scene Classification Dataset

This dataset is provided by the DCASE community [71] and contains three categories. The dataset comprises recordings from 12 European cities in 10 distinct acoustic scenes. The 10 different categories are then divided into three separate categories as follows:

- Indoor scenes: airport, indoor shopping mall, and metro station;
- Outdoor scenes: pedestrian street, public square, a street with a medium level of traffic, and urban park;
- Transportation-related scenes: traveling by bus, traveling by tram, traveling by underground metro.

The audio signal is recorded at 48kHz and in 24-bit in a binaural format using only one recording device. The dataset is divided into two categories: the development set and the evaluation set. Due to the unavailability of the labels of the evaluation set, the system was evaluated on the development set only. The development set contains 40 hours of audio recordings divided into a training set and a test set. Each audio file is 10 seconds long. The baseline system [72] is evaluated using the development set by log Mel filter bank energy features.

1.6.3 Urbansound8k

Urbansound8k is a dataset containing 10 different classes and 8732 short-duration (less than or equal to 4 seconds) files [17], [18]. The collection is composed of environmental sounds such as air conditioners, car horns, playing children, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music. Recordings are available in 10-fold cross-validation and recorded at 22.05KHz sampling frequency.

1.6.4 Custom Database

The audio recordings are collected from FreeSound [73] from several contributors. Each recording was registered by a different publisher and with different locations, lengths, equipment, and sampling rates. The recordings are gathered for four categories, i.e., rain, wind, car passing, and human walking. The recording's sampling rates were from 44100 Hz to 96000 Hz. The database was processed to obtain uniform characteristics. Ten-second audio files were extracted with a sampling rate of 441000 Hz, resulting in 750 files of 10-second length with a total duration of 125 minutes for each recording[74].

Part II

Feature Engineering

Chapter 2

Environmental Sound Classification Systems based on Empirical Features

2.1 Introduction

Neural networks, described in the previous chapter, have played a vital role in the growth of classification systems for environmental sounds. Convolutional neural networks (CNN) are at the forefront of this change, along with recurrent neural networks (RNN) and long short-term memory (LSTM), which are still used in many systems [6], [75]–[81]. As mentioned before the performance of these ANNs depends on the quality of the features used. In image classification, an image is used as a feature. In speech and sound classification, an image form of the sound is provided through the time-frequency-energy information of the signal, namely, a spectrogram. Mel spectrogram, the most widely used spectrogram, is extracted from Mel filter bank energies by applying Mel triangular filter banks to the spectrogram calculated from the application of fast Fourier transform (FFT) on a signal. This resulted in the general use of Fourier transforms, even implicitly, for spectrogram construction and feature extraction. However, there are some crucial restrictions to performing Fourier spectral analysis, which make Fourier transform valid under extremely general conditions [82], [83].

To perform Fourier spectral analysis on a signal provided by a system, the system must be linear and the signal must be ergodic and stationary; failing to meet these criteria will result in little physical sense. Sound is a time-varying signal whose frequency and energy change depending on the source generating the sound, which implies that the assumptions of stationarity and ergodism may not be satisfied. In addition, the Fourier spectrum establishes global uniform harmonic components, resulting in the necessity of additional components to simulate data that is non-stationary and globally non-uniform.

Consequently, it spreads the energy over a wide frequency range. To analyze data of a non-stationary nature in the time domain, numerous Fourier components are applied, causing energy dispersion to a much wider frequency scale. Furthermore, Fourier spectral analysis employs a priori-defined basis functions that require additional harmonic components to analyze deformed wave profiles. Features based on short-time fast Fourier transform (STFT), introduced by Cooley and Tukey in 1965 [2], are predominately used in extracting frequency domain features [3]–[7]. The wavelet transform, which is a windowed Fourier transform in the time domain, provides the solution to the limitations of STFT. Wavelets [44] overcome limitations because the window is scaled in both time and frequency [45]. Wavelet analysis provides a solution for analyzing non-stationary data. However, in wavelet transform, we still require a priori-defined basis in terms of a wavelet function, which makes wavelet analysis non-adaptive in nature. The most commonly used Morlet wavelet function is based on Fourier and suffers from the same shortcomings as Fourier analysis [83]–[85].

Due to the ubiquitous usage of Fourier spectral analysis, the notions of instantaneous frequency (IF) and instantaneous amplitude (IA) are relatively less accepted [83]. Traditionally, the frequency is defined with the sine and cosine functions as basis functions spanning the whole data length with constant amplitude. According to this approach, the instantaneous frequency also must be defined on either the cosine or sine basis function. As result, it would be compulsory to have one complete oscillation. This approach would make no sense for a non-stationary signal that changes from time to time. In real life, most systems are non-linear and operate or generate non-stationary data [86], [87]. To analyze non-stationary and non-linear time series data and processes, a novel method of decomposing temporal signals called empirical mode decomposition was introduced by Huang [83]. This decomposition is adaptive and highly efficient. This method decomposes the signal into a finite number of oscillatory units called intrinsic mode functions (IMFs). These modes are extracted based on characteristics of local time series data with zero mean with symmetric AM–FM components. The decomposition of the signal is highly adaptive and is based on the direct extraction of energy with local time scales. Using the Teager–Kaiser energy operator (TKEO)[88], we can extract instantaneous frequency and amplitude from the IMFs, thus, allowing us to locate any event on a time scale and a frequency scale. The IMFs serve as the basis in this case and are calculated for every signal rather than being defined a priori. The EMD combined with the TKEO method provides the estimation of instantaneous amplitude (IA) and instantaneous frequency (IF) for any

non-stationary signal without defining an a priori basis function; this method generates the basis function dynamically for each signal. In addition, this preliminary work adds another path for future development and applications of IA and IF in different domains where time-frequency analysis is required. The EMD method has been used in speech recognition systems [89]–[91] and human emotion recognition systems [55]. The EMD method has also been used to perform the classification of respiratory sound in conjunction with FFT to extract features [92]. The EMD extracts IMFs and later selects the best IMF based on the entropy parameters.

In the ESC system, we are interested in the feature extraction stage. The system relies heavily on the type of features to learn sound events. In this study, we introduce the use of empirical mode decomposition along with the Teager–Kaiser energy tracking operator to estimate instantaneous frequency and amplitude, which are used to construct features in terms of spectrogram for classification using neural networks. We apply the most commonly used Mel filter banks for the spectrogram. In this study, we introduced the novel Mel filter based on a spectrogram generated through IA and IF. The EMD method decomposes the signal into several mono-component IMFs; on each, the IMF TKEO and DESA methods are applied to obtain the IA and IF information of the signal. We call this Mel spectrogram obtained through EMD and TKEO the empirical mode decomposition Mel filter bank energies (EMD-MBE). We also introduce SMBE, in which we remove the signal trend from the signal using the EMD method. We compare our proposed features with fast Fourier-based Mel filter bank energies (FFT-MBE) on four ESC data sets. We propose an aggregation of all three features, which results in an improvement of accuracy over traditional FFT-based log Mel filter bank energies.

2.2 Proposed method

We use the empirical mode decomposition method to decompose the environmental sound signal into its intrinsic mode functions (IMFs), as described in the next section. The combination of EMD decomposition and TKEO is used to estimate instantaneous amplitude (IA) and instantaneous frequency (IF). Using the components of IA and IF we construct a spectrogram. Afterward, a Mel filter bank is applied to obtain Mel filter bank energies (MBE). These features are then used to train machine learning algorithms. The proposed method is depicted in Figure 2.1.

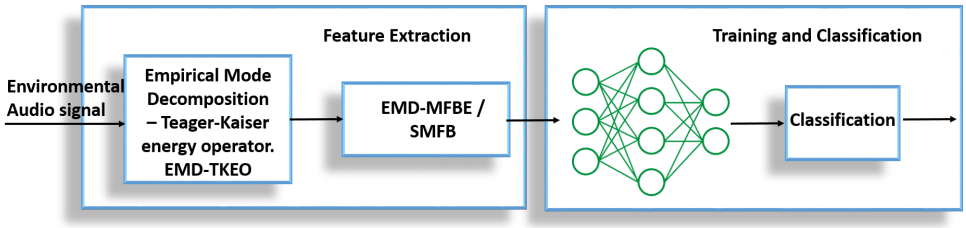


Figure 2.1 – Block diagram of proposed system—feature extraction using EMD-TKEO method and classification using neural networks.

2.2.1 Signal representation

In this study, we extend the work of P. Maragos [88], [93], [94] from the application of speech and underwater acoustic signals to extracting AM–FM modulation information from environment sound signals. In [93], the authors defined the real-valued signal with combined AM and FM structure as :

$$r_i(t) = Re(a_i(t) \times exp(j\phi_i(t))) \tag{2.1}$$

This expression can be used to formulate a signal as [95]:

$$x(t) = \sum_{i=1}^{N} r_i(t) + rest_N(t) \tag{2.2}$$

and

$$f_i(t) = \frac{1}{2\pi}\frac{d\phi_i(t)}{dt} \tag{2.3}$$

where, $rest(t)$ is the last component containing very low-frequency information, which could be neglected from the original signal. Re represents the real part, $\phi_i(t)$ is the phase, and $a_i(t)$ and $f_i(t)$ are instantaneous amplitudes and instantaneous frequencies respectively of the ith IMF.

2.2.2 Empirical Mode Decomposition

EMD is a method of decomposing a non-stationary signal into a collection of mono-component AM–FM signals. These mono-component signals are referred to as intrinsic mode functions (IMFs). The extraction of the IMFs follows an envelope subtraction process

and a linear combination of all the IMF-extracted results into the original signal. The signal is decomposed in the time domain, hence preserving the time-varying frequency and amplitude of the signal. As compared to Fourier transform, EMD does not require a priori-defined basis function for the computation of IMFs. Fourier transform uses harmonic components of the signal, whereas EMD is based on the oscillation present in the signal. The oscillatory decomposition is defined by the sifting process. The signal is examined for local maxima and minima. Using the information of local maxima and minima, the upper envelope and lower envelope are determined using via cubic spline. The mean envelope is generated using the upper and lower envelopes, which represents the trend of the signal. This mean envelope is subtracted from the original signal to create an IMF candidate. Before counting this candidate as an actual IMF, a test is conducted, i.e., if the number of zero crossings and the number of extrema differs by no more than one. If the candidate satisfies the criteria, it is counted as an IMF and the counter is incremented; otherwise, the counter is set to zero. Verification is conducted to check if the candidates meet the criteria of IMF for each generated IMF. If the criteria are not fulfilled, the sifting process is applied again until the conditions are matched. The obtained IMF is then stored and subtracted from the original signal to start a new sifting process for another IMF. The method is repeated until the signal is deconstructed to a level that it contains no more than two extrema [83].

2.2.3 Sifting Process for IMFs

The EMD method could be defined in simplest terms as a filter that sifts through the signal and breaks it down into a mono-component signal, defined above as IMFs. A function is defined as an intrinsic mode function when it satisfies the following criteria:

1 The number of extrema (maxima and minima) in a signal must be equal to the zero-crossing number or differs at most by one.

2 The mean of the envelopes obtained through local maxima and local minima must be equal to zero at all times.

The IMFs are obtained through a process known as the sifting process. which is described in Algorithm 1 [55]:

The number of IMFs extracted from a particular signal depends on two factors.

1 The process terminates when the $res(t)$; the last IMF, is either a monotonic function or function with only one extremum.

Algorithm 1 Sifting process for intrinsic mode functions

Input: a sound event signal
Output: a collection of IMFs

1 Compute all local extrema in the signal $x(t)$: local maxima and local minima;
2 Construct the upper envelope $e_{up}(t)$ and lower envelope $e_{low}(t)$ by joining the local maxima and local minima with a cubic spline on the given signal $x(t)$;
3 Calculate the mean of the envelopes $m(t) = (e_{up}(t) + e_{low}(t))$ / 2 ;
4 Subtract the mean from the original signal $x(t)$, then obtain a new data sequence $r(t)$ from which the low frequency is deleted $r(t) = x(t)$ - $m(t)$;
5 Repeat steps $1 - 4$ until $r(t)$ is an IMF (satisfying the two conditions above);
6 Subtract this IMF $r(t)$ from the original signal $x(t)$: $res(t) = x(t)$ - $r(t)$;
7 Repeat steps $1 - 6$ until the residual signal $res(t)$ is obtained that does not meet the above-mentioned conditions of an IMF, resulting in all IMFs $r_1(t)$, $r_2(t)$, ..., $r_N(t)$ of the signal $x(t)$.

2 The number of IMFs is subjected to stopping criteria, where the user terminates the sifting process after a particular number of IMFs have been created.

In the first case, the output of the EMD sifting process delivers N IMFs $r_1(t)$, $r_2(t)$, ..., $r_N(t)$ along with the residual signal $res(t)$ of the original signal $x(t)$. $x(t)$ can be presented as a linear combination of all the IMFs and $res(t)$.

$$x(t) = \sum_{i=1}^{N} r_i(t) + rest_N(t) \tag{2.4}$$

With this method, the signal $x(t)$ is decomposed empirically into a finite number of functions also depicted in Figure 2.2 [96]. The IMFs of an audio of car passing are shown in Figure 2.3. Each IMF can be used separately to obtain instantaneous Frequency (IF) and instantaneous amplitude (IA) for sound event detection systems, explained in the next section.

In the case of early stopping, the original signal cannot be reconstructed, as some information is discarded deliberately. However, in some cases, it could be used to remove low-frequency components from the parent signal. In [97], the authors used the first five IMFs, on the basis that those IMFs gave an ample amount of information about energy and pitch in their study.

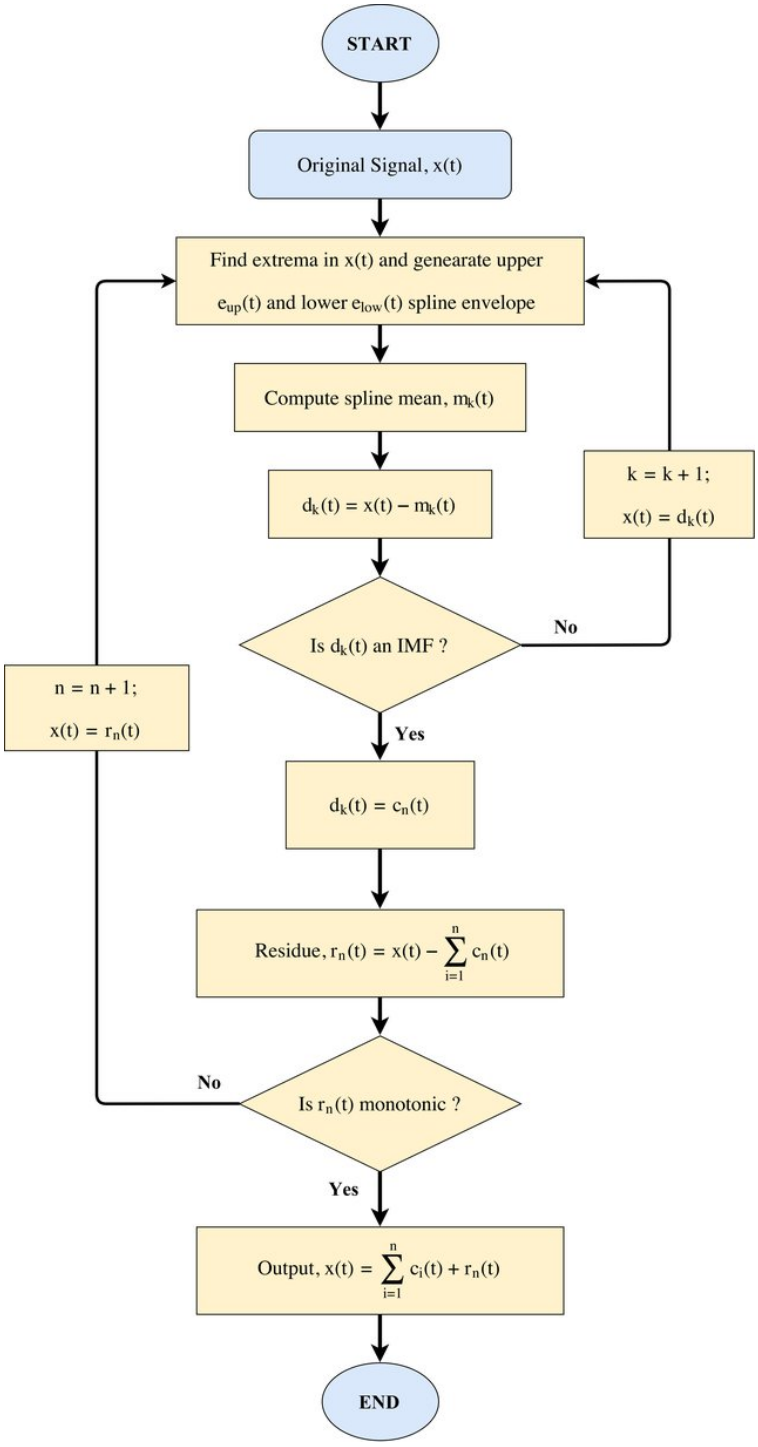


Figure 2.2 – Flow chart of sifting process.

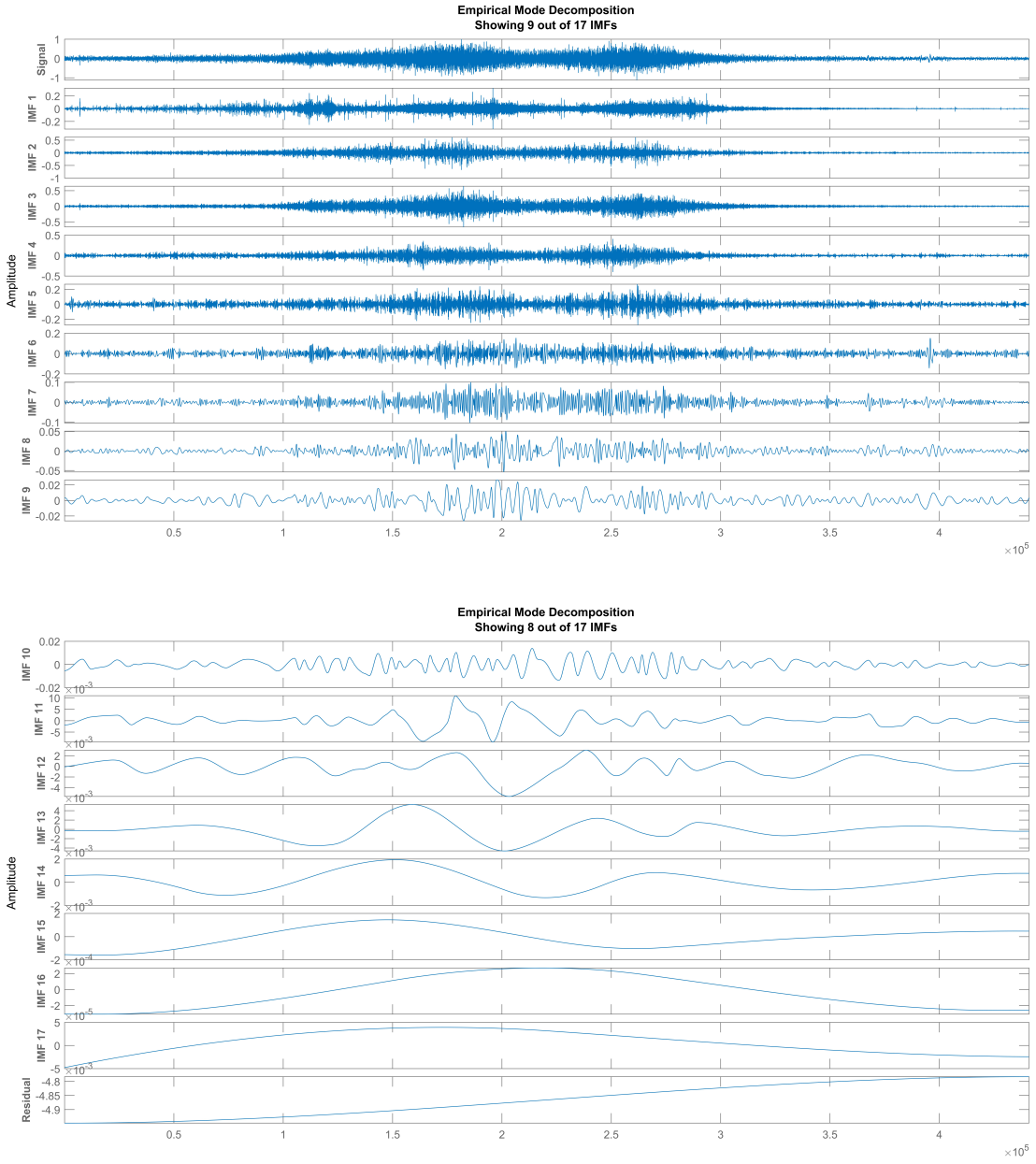


Figure 2.3 – Empirical mode decomposition and intrinsic mode function extraction.

2.2.4 Teager–Kaiser Energy Operator (TKEO)

The energy separation algorithm (ESA) is applied to extract the IA and IF information from the signal, as standalone IMFs do not provide meaningful information about IA and IF. The EMD method takes a multi-component signal and provides us with IMFs that are mono-component. Introduced by J.F. Kaiser [98], the TKEO, an energy tracking operator used with an energy separation algorithm, computes these IA and IF features without using integrals, as in the Hilbert transform and Fourier transform. Rather, it is completely comprised of differentiation. The property of differentiation gives the TKEO the advantage of good localization [88]. It becomes more natural to use the TKEO for local estimation of IA and IF functions. The TKEO is a non-linear operator that computes the energy of the signal as a product of the square of the amplitude and frequency of the signal [99]:

$$\Psi[r_i(t)] = [\dot{r}_i(t)]^2 - r_i(t)\ddot{r}_i(t) \tag{2.5}$$

where $\dot{r}_i(t)$ and $\ddot{r}_i(t)$ are the first and second order derivatives of $r_i(t)$. For a discrete time signal $r_i(n)$, Equation 2.5 can be written as [100]:

$$\Psi[r_i[n]] = r_i^2[n] - r_i[n+1]r_i[n-1] \tag{2.6}$$

The instantaneous features are extracted by applying ESA in a discrete form to the signals. The discrete energy separation algorithm (DESA) [99] provides us with IA and IF. We used DESA-1 in this study, given as :

$$y[n] = x[n] - x[n-1]$$

$$f[n] \approx arccos(1 - \frac{\Psi[y[n]] + \Psi[y[n+1]]}{4\Psi[y[n]]}) \tag{2.7}$$

$$|a[n]| \approx \frac{2\Psi[x[n]]}{\sqrt{\Psi[x[n+1] - x[n-1]]}} \tag{2.8}$$

Here, $x[n]$ is a mono-component signal. The DESA-1 algorithm should be applied to mono-component signals only. In [94] authors proposed the use of a low pass filter to smooth the output of the energy tracking operator. They found that a high-frequency error component was introduced by the energy operator. To eliminate this issue, a seven-

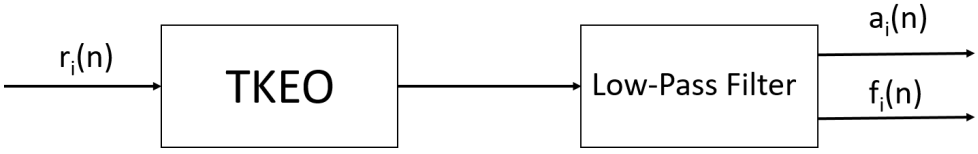


Figure 2.4 – Block diagram of smoothing the output of the energy operator.

point linear binomial low-pass smoothing filter with impulse response (1,6,15,20,15,6,1) is applied after TKEO as shown in Figure 2.4 [55], [94].

2.3 Feature Extraction

2.3.1 Mel Band Energies

The general Mel band energies (MBEs) are computed through discrete Fourier transform as follows. Let $x[n]$ be a discrete audio signal having sampling rate f_s. It is divided into P frames, each of length N samples with $N/2$ overlapping samples, such that $\{\vec{x}_1[n], \vec{x}_2[n], ..., \vec{x}_p[n], ..., \vec{x}_P[n]\}$, where $\vec{x}_p[n]$ represents the p^{th} frame of the signal $x[n]$ and is given as :

$$\vec{x}_p[n] = \left\{ x\left[p * \left(\frac{N}{2} - 1\right) + i\right] \right\}_{i=0}^{N-1} \tag{2.9}$$

The input signal $x[n]$ can be represented as a matrix of size $N \times P$ as $X = [\vec{x}_1, \vec{x}_2, ..., \vec{x}_p, ..., \vec{x}_P]$. When calculating DFT, the signal for each $\vec{x}_p$, one assumes that the signal is repeated infinitely, which introduces an issue of spectral leakage. To reduce spectral leakage, the Hanning window is applied.

$$w[n] = 0.5 * \left(1 + \cos\left[\frac{2\pi n}{N_p}\right]\right) \tag{2.10}$$

and the discrete Fourier transform of the signal is given as :

$$X_p[k] = \sum_{n=0}^{N_p-1} x_p[n] w[n] \exp^{-j\frac{2\pi k n}{N_p}} \tag{2.11}$$

Here, $k = 0, 1, 2, ..., N_p - 1$, where N_p represents the number of points used by FFT for a particular frame $\vec{x}_p$. Using the sampling rate f_s of the input signal, the corresponding frequency can be computed using the frequency bin as $l_f(k) = kf_s/N$ and the frequency

resolution can be computed as $f_{res} = l_f(k+1) - l_f(k)$. The DFT of the p^{th} frame x_p can be represented as $\vec{X_p} = [X_p(0), X_p(1), X_p(2), ..., X_p(N-1)]^T$; similarly, for the complete signal $x[n]$ we obtain $X = [\vec{X_1}, \vec{X_2}, ..., \vec{X_P}]$. Here, the X matrix has the dimensions $N \times P$ and is defined as a short-time Fourier transform. The magnitude spectrum of the signal is obtained by taking the modulus of X. The magnitude spectrum is warped according to the Mel scale to obtain human ear-like properties. The Mel frequency (ϕ_f) and the linear frequency l_f are defined by the relation $\phi_f = 2595 * log_{10}(1 + \frac{l_f}{700})$. Mel filter banks, which are comprised of overlapping triangular filters defined by their center frequencies $l_{f_c}(m)$, are used to segment the spectrum X depending on the band number m. The Mel filter bank is shown in Figure 1.3.

The three parameters that define Mel filter banks are:

· Number of Mel filters, F;

· Minimum frequency, $l_{f_{min}}$;

· Maximum frequency, $l_{f_{max}}$.

Using the minimum and maximum frequencies and the number of Mel filters, the constant frequency resolution is calculated using the relation $\delta\phi_f = (\phi_{f_{max}} - \phi_{f_{min}})/(F+1)$, where $\phi_{f_{max}}$ and $\phi_{f_{min}}$ are frequencies on the Mel scale defined by the corresponding linear frequencies $l_{f_{max}}$ and $l_{f_{min}}$, respectively. The centre frequencies on the Mel scale are obtained through $\phi_{f_c}(m) = m.\delta\phi$ and $m \in \{1, 2, 3, ..., F\}$. Similarly, we can inverse the relation to obtain center frequencies in the linear frequency in Hz as $l_{f_c}(m) = 700(10^{\phi_{f_c}(m)/2595} - 1)$. The resulting Mel filter bank matrix M (m,k) of size $F \times N$ is given by:

$$M(m,k) = \begin{cases} 0 & \text{for } l_f(k) < l_{f_c}(m-1) \\ \frac{l_f(k) - l_{f_c}(m-1)}{l_{f_c}(m) - l_{f_c}(m-1)} & \text{for } l_{f_c}(m-1) \leq l_f(k) < l_{f_c}(m) \\ \frac{l_f(k) - l_{f_c}(m+1)}{l_{f_c}(m) - l_{f_c}(m+1)} & \text{for } l_{f_c}(m) \leq l_f(k) < l_{f_c}(m+1) \\ 0 & \text{for } l_f(k) > l_{f_c}(m+1) \end{cases} \tag{2.12}$$

To obtain Mel filter bank energies, we multiply the DFT matrix $X_p(k)$ by the Mel filter bank matrix $M(m,k)$. A logarithm is applied to obtain log Mel band energies of the size $F \times P$, as given in the following equation:

$$S_p(m,k) = log\left(\sum_{k=0}^{N_p - 1} M(m,k) * |X_p[k]|\right) \tag{2.13}$$

2.3.2 EMD-Mel Band Energies

In this method, AM and FM components are used to construct the magnitude spectrum; later a Mel filter bank is applied to obtain Mel band energies. Figure 2.5 demonstrates the process of obtaining MBEs; in this case, we call these features empirical mode decomposition-based Mel filter bank energies (EMD-MBE). The first step is the decomposition of the signal into its components using EMD. These distinct, adaptive decomposed components are known as intrinsic mode functions (IMFs). For each distinct IMF, instantaneous amplitude $a(i,n)$ and instantaneous frequency $f(i,n)$ are obtained through the energy tracking operator and energy separation algorithm, where $i = 1, 2...N_{IMFs}$ and $n = 1, 2, 3, ..., N_b$, and N_b represents the number of samples in the input signal. In this study, we used the Teager–Kaiser energy tracking operator (TKEO) and the discrete energy separation algorithm (DESA-1). The TKEO and DESA estimates the IA and IF over the complete length of the IMF. The TKEO and DESA algorithms in [94], [101], [102] have shown estimations of IA and IF with an error less than 10^{-3} . The framing /windowing function is applied later to the IA and IF obtained from the TKEO. In the next stage, we apply the Hanning windowing function to obtain short overlapping frames of instantaneous frequency $f_p(i,n_p)$ and instantaneous amplitude $a_p(i,n_p)$. Afterward, to obtain the magnitude spectrum, we use the definition provided by [95], where the authors defined the Hilbert Huang Transform as a generalized Fourier Transform and defined the spectrum using the Hilbert spectrum, which is derived from the time-frequency distribution of the instantaneous energy envelope, which is the square magnitude of the amplitude envelope. In this study, we used the instantaneous energy envelope ($|a(i,n)|^2$) and summed them over a large number of frequency bands, as compared to the Hilbert spectrum [55], [95]. This enables us to distribute the energy over a large number of frequencies and to obtain higher frequency resolution. To derive Mel band energies, we summed the energies similarly to the number of frequency bands defined in the previous section. The relation is defined as (2.14) :

$$X_p(k) = \sum_{i=1}^{N_{IMFs}} \sum_{n_p=1}^{N_p} |a_p(i,n_p)|^2 \mathbb{1}_{\mathrm{B_k}}(f_p(i,n_p)) \tag{2.14}$$

where i is a single IMF, N_p represents the number of samples in the frame p, and B_k represents the particular sub-band defined as $B_k = [l_f(k), l_f(k+1)]$, where $k \in \{1, 2, 3, ...N-1\}$ and $l_f(k) = kf_s/N$. The indicator function of set Ω is given as:

$$\mathbb{1}_{\Omega}(a) = \begin{cases} 0 & \text{if } a \in \Omega \\ 1 & \text{if } a \notin \Omega \end{cases}$$

After using (2.14), we obtain a matrix of size $N \times P$. This matrix is multiplied, similarly to equation (2.13), by the Mel filter bank matrix $M(m, k)$ of shape $F \times N$. The resulting matrix will have the shape $F \times P$. By taking the *log*, we obtain log Mel band energies.

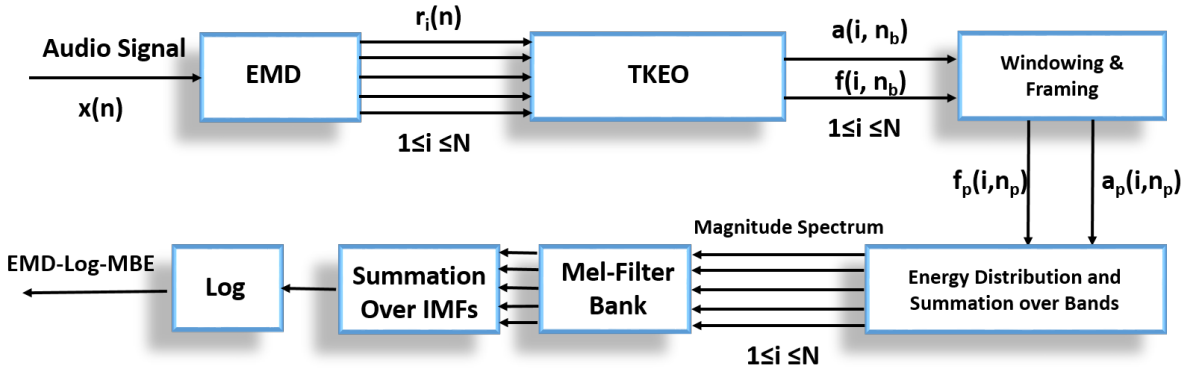


Figure 2.5 – Empirical mode decomposition-based Mel filter bank energies extraction block diagram.

2.3.3 S-MBE

Sounds produced in any environment are composed of complex and random changes, and the presence of a signal trend causes a negative effect in frequency domain power spectral analysis or time-domain correlational analysis, which could result in the loss of information in the low-frequency spectrum. To counter such problems, we apply a method of extracting Mel filter bank energies (MBEs) by removing the signal trend, calling these S-MBEs. In the literature, researchers have used a version of SMFCC for emotion recognition [103]; we present a different version that extracts the feature before applying discrete cosine transform (DCT) to the MBEs. The complete process of extracting S-MBEs is presented in Figure 2.6. In this method, EMD performs the decomposition on the signal and extracts the signal trend information from the IMFs. The signal trend is denoted by $T[n]$ and is computed by Equation (2.15) [103]:

$$T(n) = \sum_{i=i_d}^{N_{IMFs}} r_i[n] \tag{2.15}$$

where,

$$i_d = min\left[i \in [2, n], \frac{R_{ri}}{R_{r1}} < 0.01\right] \tag{2.16}$$

The signal trend is removed from the input signal by applying the zero-crossing rate (ZCR) detection method. $T[n]$ is defined as the sum of all IMFs that satisfy the following condition equation (2.17):

$$\frac{R_{ri}}{R_{r1}} < 0.01 \quad (i = 2, ...n) \tag{2.17}$$

where R represents the zero-crossing rate. Afterward, the reconstructed signal $S[n]$ is obtained by removing the signal trend $T[n]$ from the input signal.

$$S[n] = x[n] - T[n] \tag{2.18}$$

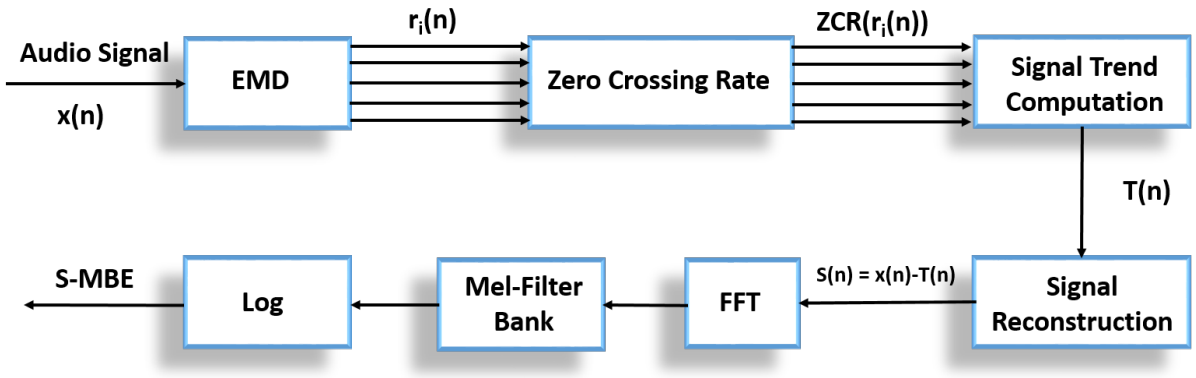


Figure 2.6 – SMBE feature extraction block diagram.

Finally, the MBEs are calculated using FFT and Mel filter banks on the reconstructed signal, as shown in Figure 2.6.

2.3.4 Features

The Mel spectrograms from the aforementioned techniques are extracted for a 10-sec recording of car passing and are depicted in Figure 2.7, 2.8, and 2.9. The FFT-based Log

Mel spectrogram is shown in Figure 2.7 along with Log S-MBE in Figure 2.8. The EMD of the input signal is shown in Figure 2.3 and the resulting Log EMD based MBE is depicted in Figure 2.9. Figure 2.10 shows a signal recording of a person walking and the resulting Log EMD-MBEs is shown in Figure 2.11. The IMFs obtained from EMD and the Log EMD-MBE estimation along with the Log EMD-MBE of the combination of the previously calculated IMFs are presented in Figures 2.12, 2.13, 2.14, and 2.15.

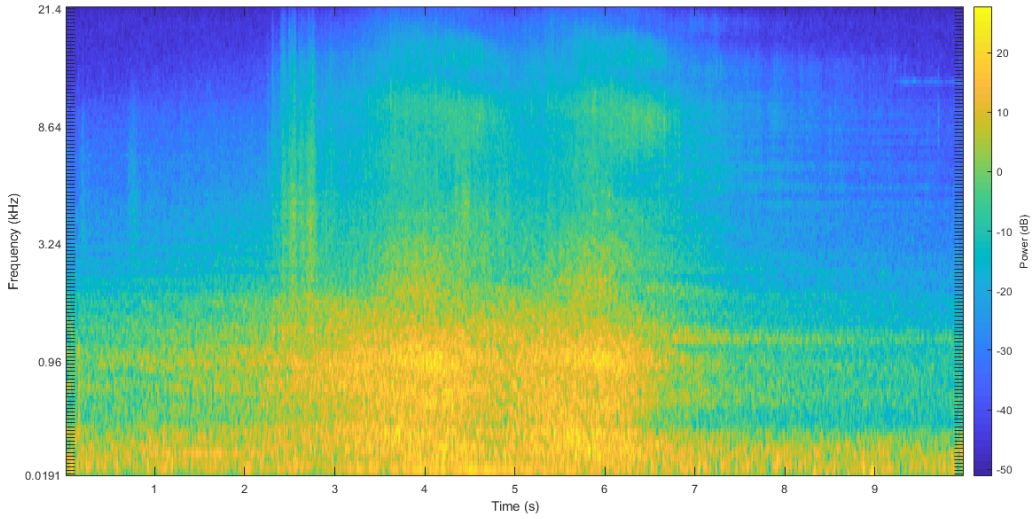


Figure 2.7 – FFT-MBE spectrogram extracted from a 10-sec audio file of a car passing.

2.4 Experimental Setup

2.4.1 Datasets

The development of an ESC system relies heavily on the dataset. Sound classification is a vast topic and contains many categories such as acoustic scene classification, sound event classification, environmental sound classification, and many more. We used datasets comprised of acoustic scenes, sound events, and environmental sounds for classification as the datasets are mentioned in the section 1.6, namely:

- Acoustic scene classification dataset
- Low-complexity acoustic scene classification dataset

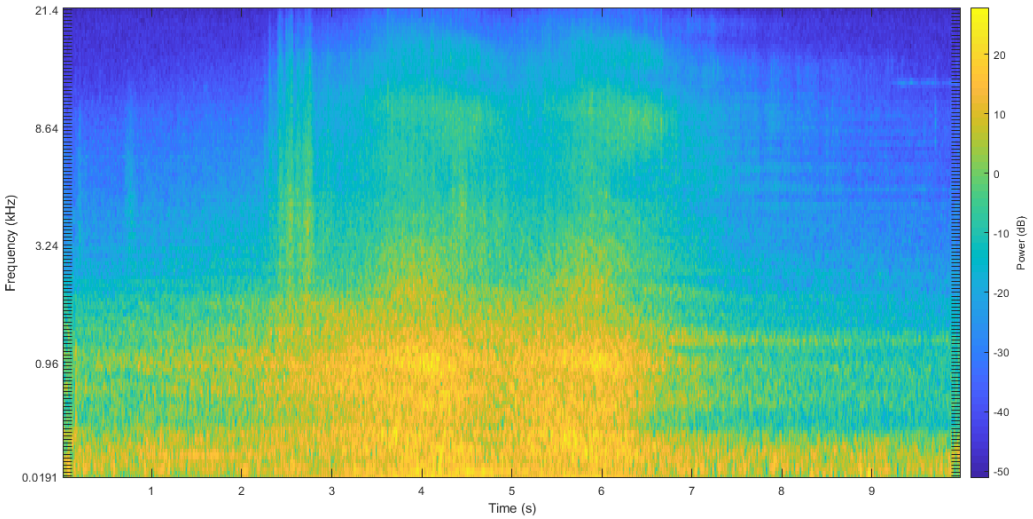


Figure 2.8 – Signal trend removed (S-MBE) spectrogram of a 10-sec audio file of a car passing.

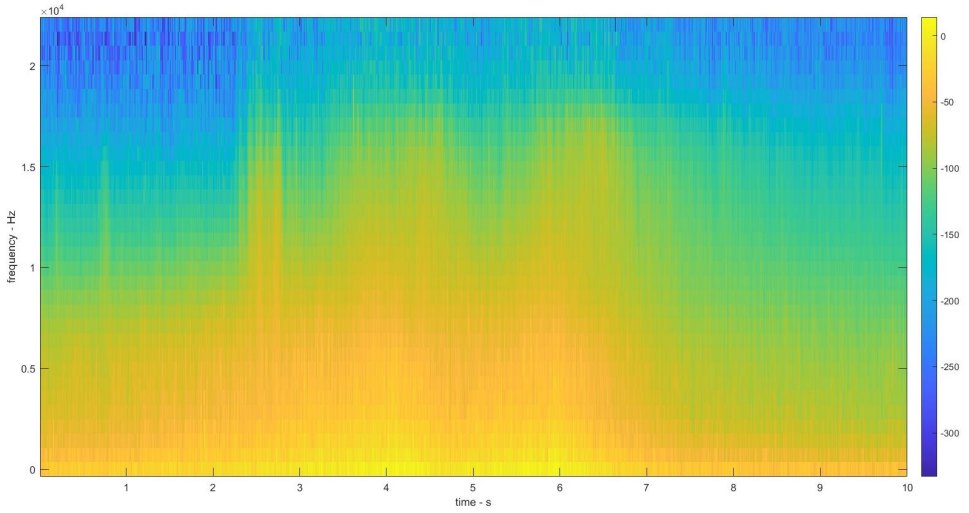


Figure 2.9 – Empirical mode decomposition- Mel band energies (EMD-MBE) of a 10-sec audio file of a car passing.

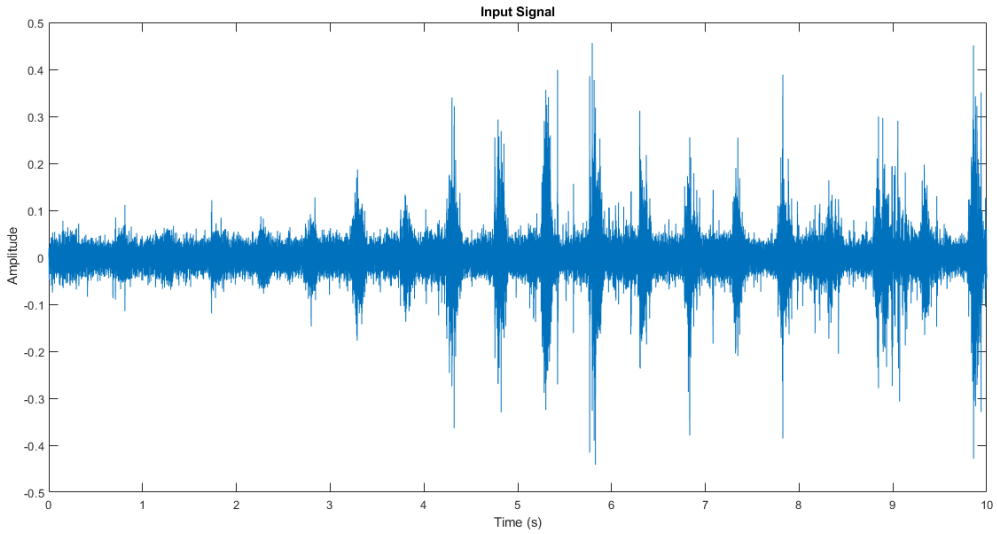


Figure 2.10 – Signal representation of a person walking

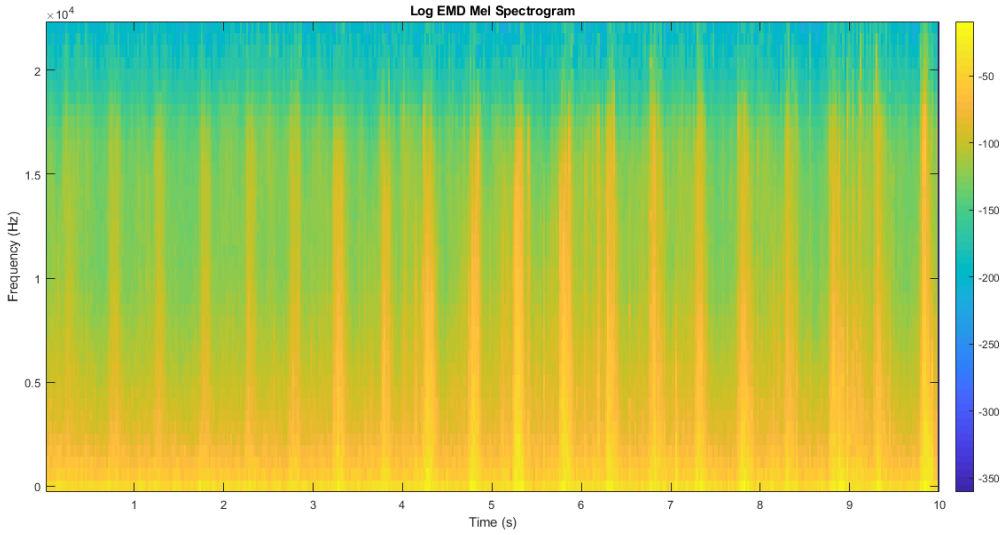
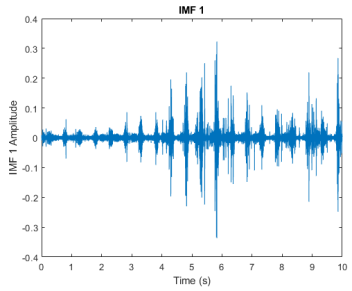
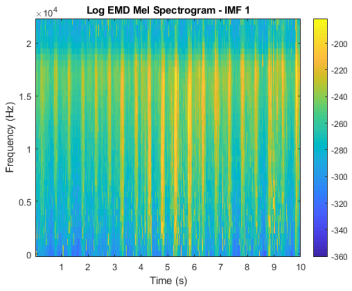


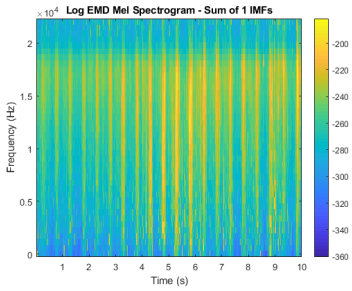
Figure 2.11 – Empirical mode decomposition- Mel band energies (EMD-MBE) of a 10-sec audio file of a person walking.



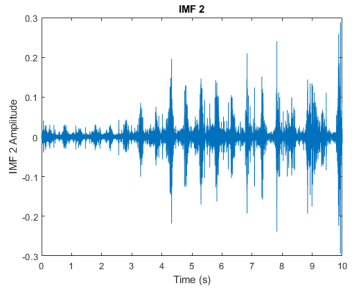
(a) IMF 1



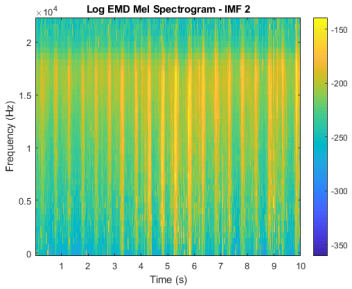
(b) IMF 1 Log EMD MBE



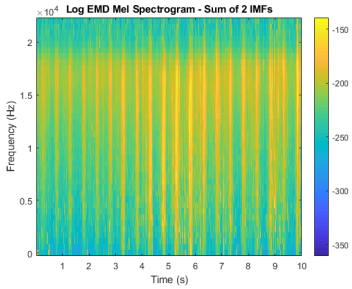
(c) Sum of 1 IMF(s) Log EMD MBE



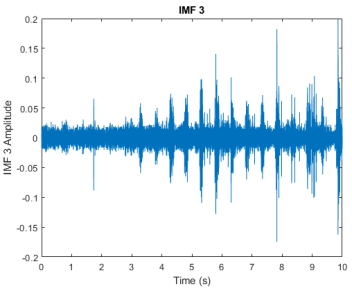
(d) IMF 2



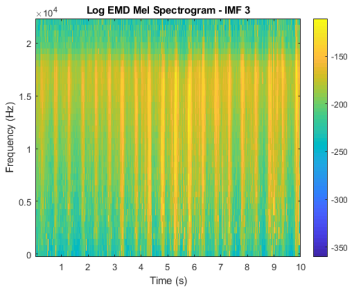
(e) IMF 2 Log EMD MBE



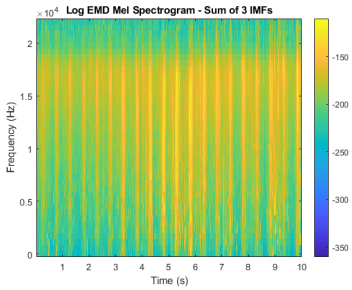
(f) Sum of 2 IMF(s) Log EMD MBE



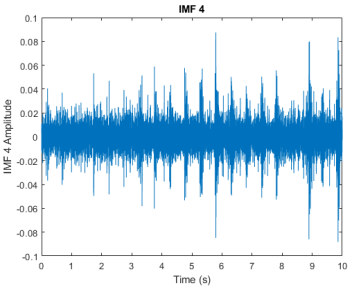
(g) IMF 3



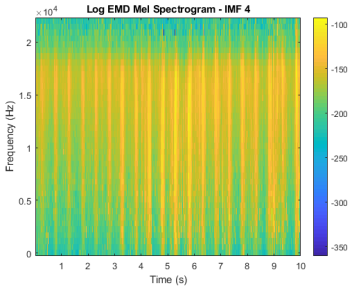
(h) IMF 3 Log EMD MBE



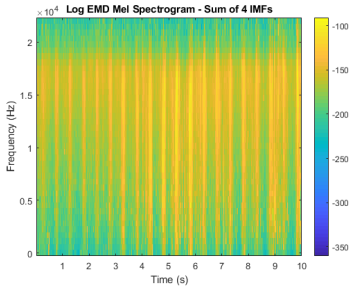
(i) Sum of 3 IMF(s) Log EMD MBE



(j) IMF 4

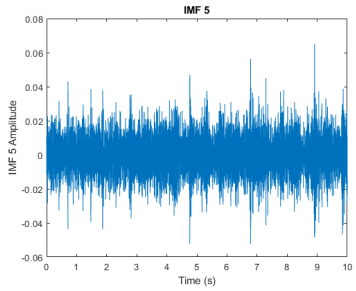


(k) IMF 4 Log EMD MBE

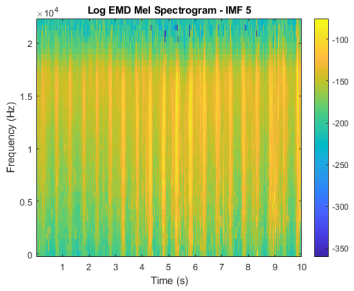


(l) Sum of 4 IMF(s) Log EMD MBE

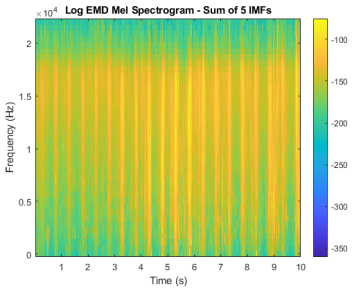
Figure 2.12 – IMF(s) 1 - 4 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s).



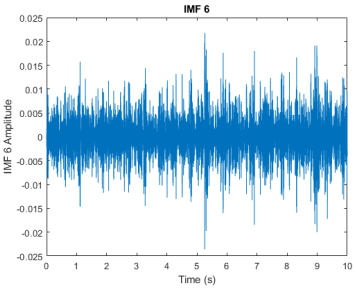
(a) IMF 5



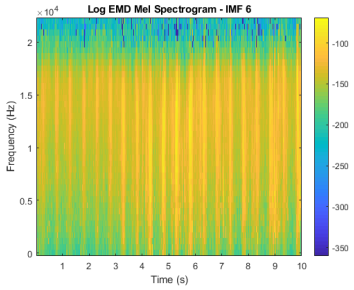
(b) IMF 5 Log EMD MBE



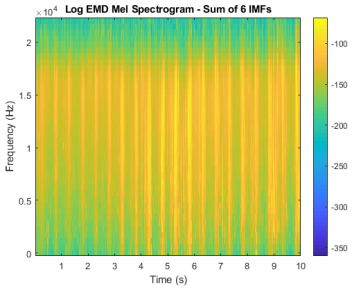
(c) Sum of 5 IMF(s) Log EMD MBE



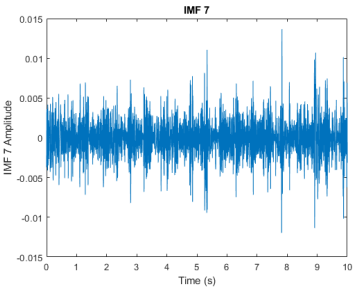
(d) IMF 6



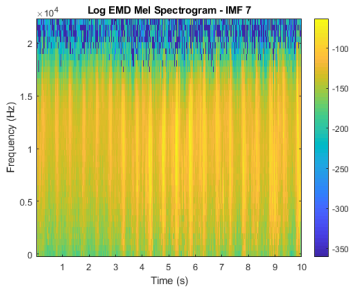
(e) IMF 6 Log EMD MBE



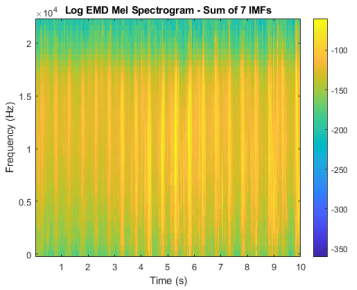
(f) Sum of 6 IMF(s) Log EMD MBE



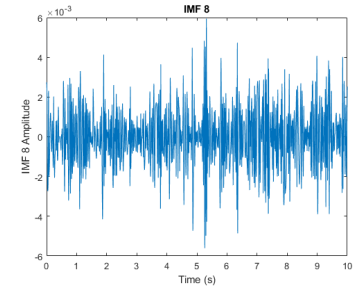
(g) IMF 7



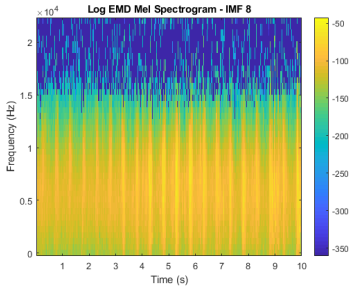
(h) IMF 7 Log EMD MBE



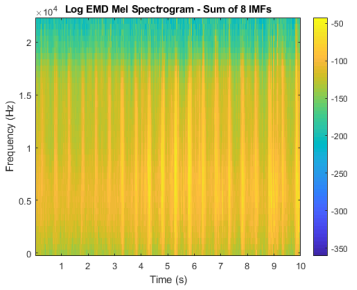
(i) Sum of 7 IMF(s) Log EMD MBE



(j) IMF 8

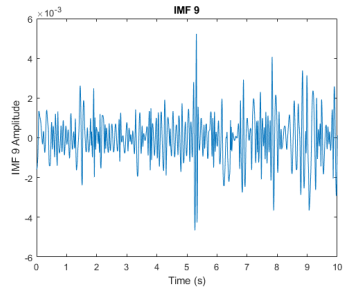


(k) IMF 8 Log EMD MBE

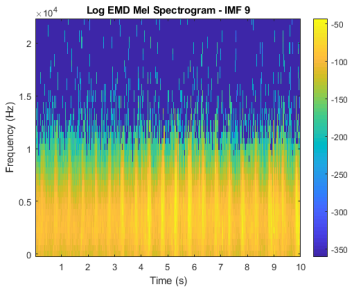


(l) Sum of 8 IMF(s) Log EMD MBE

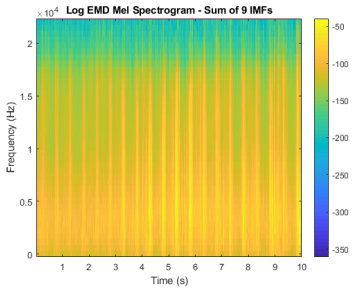
Figure 2.13 – IMF(s) 5 - 8 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s).



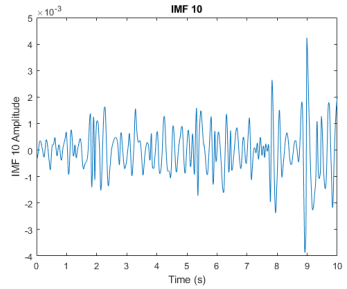
(a) IMF 9



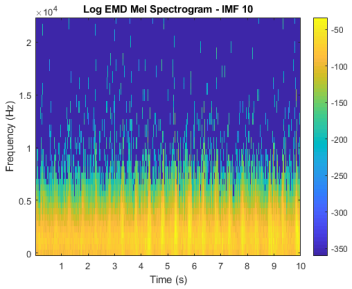
(b) IMF 9 Log EMD MBE



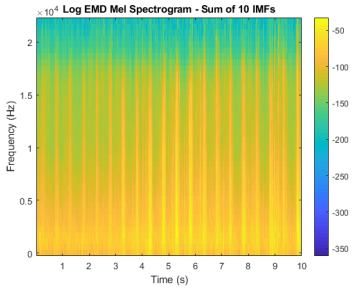
(c) Sum of 9 IMF(s) Log EMD MBE



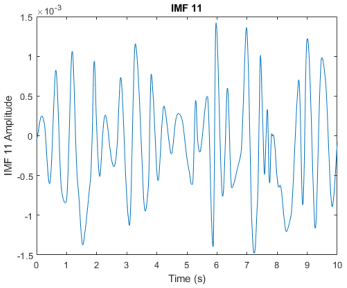
(d) IMF 10



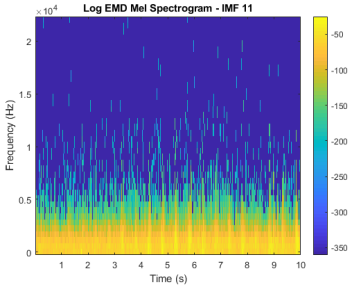
(e) IMF 10 Log EMD MBE



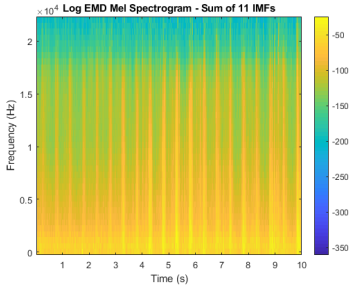
(f) Sum of 10 IMF(s) Log EMD MBE



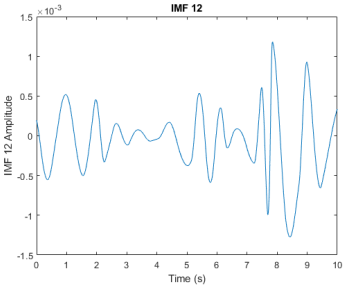
(g) IMF 11



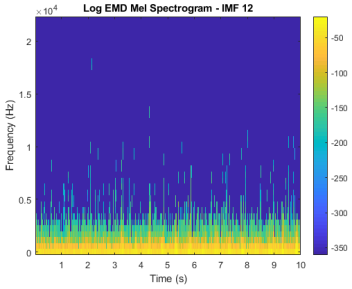
(h) IMF 11 Log EMD MBE



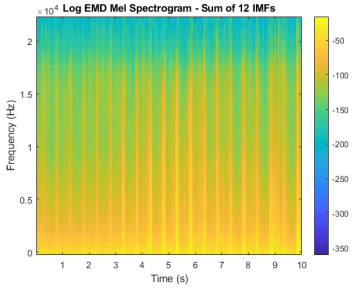
(i) Sum of 11 IMF(s) Log EMD MBE



(j) IMF 12

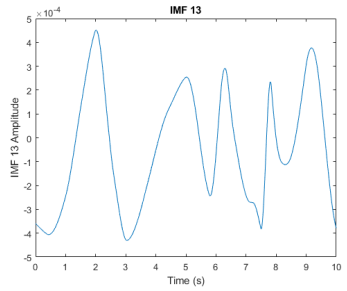


(k) IMF 12 Log EMD MBE

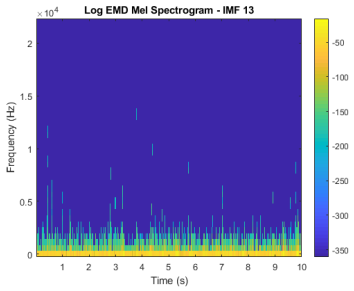


(l) Sum of 12 IMF(s) Log EMD MBE

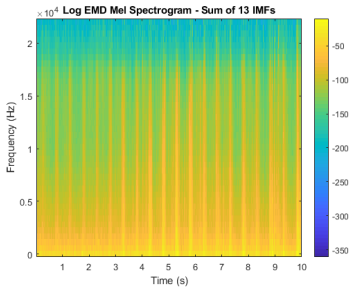
Figure 2.14 – IMF(s) 9 - 12 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s).



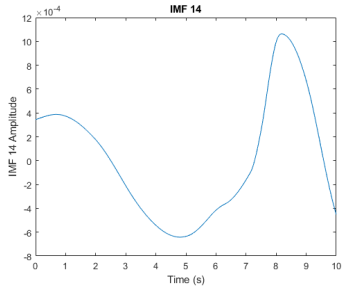
(a) IMF 13



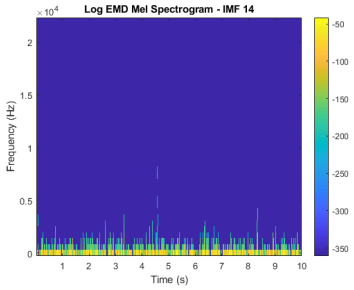
(b) IMF 13 Log EMD MBE



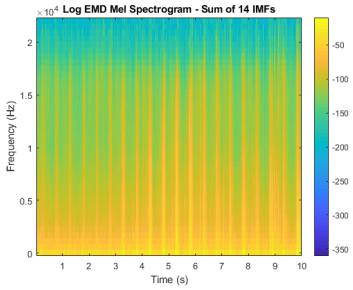
(c) Sum of 13 IMF(s) Log EMD MBE



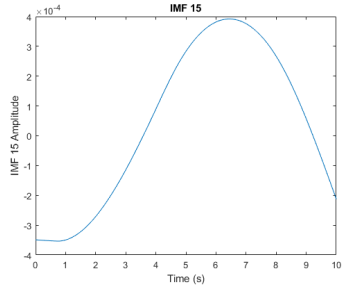
(d) IMF 14



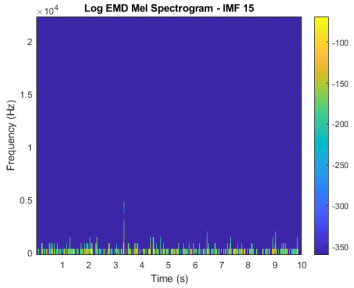
(e) IMF 14 Log EMD MBE



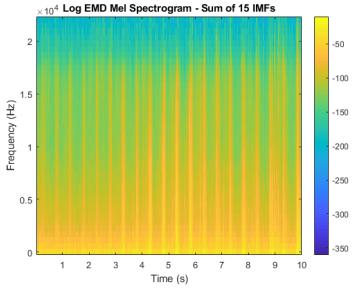
(f) Sum of 14 IMF(s) Log EMD MBE



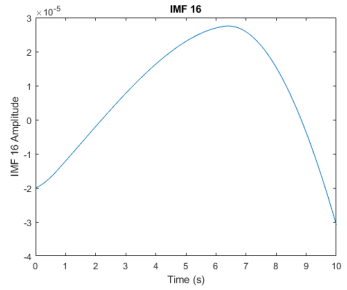
(g) IMF 15



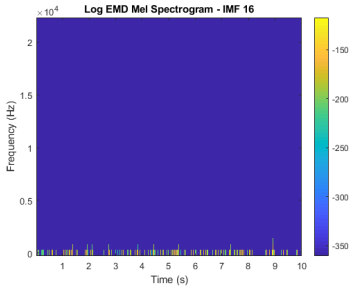
(h) IMF 15 Log EMD MBE



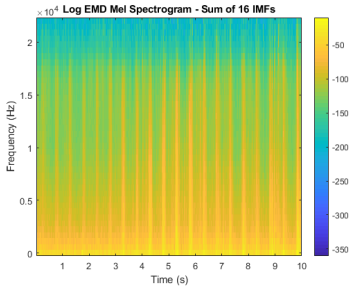
(i) Sum of 15 IMF(s) Log EMD MBE



(j) IMF 16



(k) IMF 16 Log EMD MBE



(l) Sum of 16 IMF(s) Log EMD MBE

Figure 2.15 – IMF(s) 13 - 16 and the resulting EMD-MBE per IMF and combination of EMD-MBE of IMF(s).

· UrbanSound8k dataset

· Custom dataset

2.4.2 Custom dataset creation description

In this study a custom dataset was created, briefly explained in chapter 1.6. The dataset is created in the context of developing an ESC system, with applications in security and automation in smart homes. The dataset consists of 4 categories, two categories belong to naturally occurring event and two belongs to human activities: rain, wind, human walking, and car passing respectively. The audio recordings are collected from the open-source repository Freesound [73]. The sound files were stored with associated labels, assigned to each audio recording. Recordings were sampled in the range of 44100Hz to 96000Hz and recorded with different recording devices and some in different formats and at non-similar bit rates. Initial processing of the dataset was applied to modify the recordings into a more standard form. The audio files were processed and set to a sampling rate of 44100Hz, 16-bit ".wav" files. In the later stage, 10 seconds audio chunks were extracted and stored as separate audio files. The total duration of the recordings was 125 minutes long for each category.

The advantage of using an open repository is that it provides heterogeneous recorded data. Audio recordings were made by different contributors, in different locations, with different environmental factors affecting the recordings, such as noise. Another advantage is the heterogeneity of recordings for each class, for example, audio recording in different shades during rain provides the different intensities of sound produced by the hitting of raindrops on the shade. The disadvantage is that it is a tedious task to listen to every recording and label each class.

The final taxonomy of the total number of recording minutes added to the dataset is presented in Table 2.1. The table also presents the division of dataset in train and test data, which is a standard procedure in developing a machine learning model. The test set contains examples that the classifier has not seen during the process of training the model. The training set is further divided into 80-20 percent for training and validation.

Table 2.1 – Dataset Recordings

Category	Train Set (mm:ss)	Test Set (mm:ss)
Rain	98:05	24:07
Wind	102:24	25:37
Car passing	90:26	22:57
Human walk	93:28	23:48
Total	384:23	96:29

2.4.3 Classification Model

We used convolutional neural networks (CNNs) in this study. CNNs have been widely used with Mel band energies for the classification of environmental sounds. For the Acoustic Scene Classification Dataset and the Low-Complexity Acoustic Scene Classification Dataset, we used baseline [71] CNN1 model. All the parameters were selected according to the baseline model mentioned by the authors in order to evaluate the same system using different feature inputs. For Urbansound8K and the custom dataset, we used the CNN2 model [74] given in Table 2.2.

In this study, we made a comparison between two feature extraction techniques. We have proposed an EMD-based feature extraction technique compared to the FFT-based feature extraction technique. In order to compare the performance of both methods, we employed baseline systems. We used baseline systems since they are built on the simple extraction of Mel band energies features, and no additional pre- or post-processing is applied during the training and testing of the systems. The systems that reached the highest score of accuracy on the evaluation of acoustic scene classification datasets [104] use an ensemble of features based on adaptive temporal division and classify using a VGGish-based neural network. In addition, the score on the development dataset is not published. The leading system on the development dataset [105] uses MFCC features with I-vector back-end processing with a fusion of CNNs and I-vectors to make predictions. Similarly, for the low-complexity acoustic scene classification dataset, the leading system uses Resnet with a receptive field. For Urbansound8k, different systems are proposed [106], [107]. These systems use feature pre-processing and post-processing, transfer learning, and other methods to enhance the accuracy of the system. Compared to the state-of-the-art methods, which employ different systems and employ different pre and post-processing methods, we followed the path of baseline systems. To compare with FFT-based log Mel band energies, we proposed EMD-based log Mel band energies. This allows the evaluation

of the performance of both features on the same system and with the same parameters without any pre- or post-processing of the feature. The specifications of the systems used are described below.

— Log-scaled Mel band energies were extracted for every dataset. For the Acoustic Scene Classification Dataset and Low-Complexity Acoustic Scene Classification Dataset, we extracted 40 Mel bands using an analysis frame of 40 ms with a 50 % overlap. Similarly, the EMD-Log Mel band energies and log-scaled S-MBEs were calculated for both datasets with similar characteristics, resulting in a similar shape. The input shape is 40×500, trained for 200 epochs with a mini-batch size of 16 and data shuffling between epochs. An Adam optimizer [65] is used for optimization, with a learning rate of 0.001. Model performance is checked after each epoch on the validation set, and the best performing is chosen.

— For Urbansound8k, the log Mel band energies, EMD-log Mel band energies, and log scaled S-MBEs were extracted with 60 Mel bands; a window size of 1024 samples with a hop length of 512 samples is used. The input size for the CNN was 60×41 and silent segments were discarded. The Urbansound dataset was trained using 10-fold cross-validation. The network was trained for 300 epochs with the Adagrad optimizer [108].

— For the custom dataset, the log Mel band energies, EMD-log Mel band energies, and log scaled S-MBEs were extracted with 128 Mel bands with 50 % overlap. The custom dataset was trained using seven-fold cross-validation. The system was trained for 200 epochs with an Adagrad optimizer, with an initial learning rate of 0.001.

— To evaluate the experimental results, this study uses classification accuracy as a metric:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TN and TP are defined as the number of negative and positive examples that are classified successfully, respectively. FN and FP are the numbers of misclassified positive and negative examples, respectively. The evaluation metric is chosen according to the baseline system [71] to perform comparisons between the feature extraction methods under the same evaluation metrics.

Table 2.2 – Convolutional neural network model specifications

Layers	**Model specifications**	
	CNN1	***CNN2***
Layer 1	Conv2D (32, (7,7)) Relu, strides = 1 MaxPool2D (5, 5) Dropout (0.3) Batch normalisation	Conv2D (64,(4,4)) tanh, strides = 1 MaxPool2D (2, 2), stride=2
Layer 2	Conv2D (64, (7,7)) Relu, strides = 1 MaxPool2D (4, 100) Dropout (0.3) Batch normalisation	Conv2D (32,(4,4)) tanh, strides = 1 MaxPool2D (2, 2), stride=2 Dropout (0.2)
Layer 3	-	Conv2D(16,(4,4)) tanh, strides = 1 MaxPool2D (2, 2), stride=2 Dropout (0.2)
Layer 4	Dense (100) Activation = relu Dropout = 0.3	2 X Dense (400) Activation = tanh
Layer 5	-	Dense (300) Activation = tanh Dropout = 0.2
Layer 6	Dense (classes,softmax)	

2.5 Results and Discussion

We trained the convolutional neural networks with the parameters given in the previous section. The model was evaluated using a test set and average classification accuracy is computed. First, we trained the models using only one feature at a time; for each feature, the system was trained and evaluated. Afterward, we combined the time-frequency analysis techniques. In the first case, we combined FFT-MBEs and EMD-MBEs. The model was trained and evaluated by aggregating these two features. Later, we combined the two proposed features extracted using EMD-MBEs and SMBEs with traditional FFT-MBEs. Similarly, we trained and evaluated the models by aggregating all the features.

The average classification accuracy of each dataset with different features are presented in Table 2.3. The class-wise average classification accuracies for the Acoustic Scene Classification Dataset with FFT-based log MBE, EMD-based log MB, log SMBE, and a combination of these features are presented in Table 2.4. Similarly, for the Low-Complexity Dataset, Urbansound8k, and the custom dataset, class-wise mean classification accuracies are presented in Table 2.5, Table 2.6, and Table 2.7, respectively.

It is evident that the FFT-based log MBEs performed better than EMD-based log MBEs and log S-MBEs for every dataset. However, the combination of all the features improves the performance of the system with respect to single FFT-based log MBE-based features in some cases. The EMD-MBE method outperforms FFT-MBE in some categories, as shown in the Tables 2.3, 2.4, 2.5, 2.6 and 2.7. The CNNs were able to perform better inferences for some categories than others. It is, however, uncertain what led the CNNs to better learn features of one method over another, and of one category over another. There are no evaluation methods available in the research domain to understand what leads to better performance of CNNs. In EMD-MBE, for each sample of monotonic IMF in the time domain, we obtained an equivalent instantaneous amplitude IA and instantaneous frequency IF component. To obtain a similar shape as that for FFT-MBE, we performed framing on the training sample, and later we performed summation over frequency bands. The frame consists of IA and IF samples. The total number of frequency components in a single frame of EMD-MBE for the original signal is defined by the number of samples in the window summed over frequency bands and the number of IMFs extracted. Contrary to this, in FFT, the windowing function is applied directly to the input signal then frequency and energy components are computed over this window. This method may introduce spurious higher harmonics in the result, as mentioned earlier. The FFT-MBE

Table 2.3 – System performance comparison.

Features	Accuracy per dataset			
	ASC Dataset	Low Complexity ASC	Urbansound8k	Custom
FFT	54.77 %	84.18 %	63.36 %	75.61%
EMD	52.5 %	79.74 %	54.64 %	75.05%
SMB	48.25 %	79.55 %	52.41 %	71.93%
FFT+EMD	56.08 %	84.49 %	63.70 %	78.87 %
FFT+EMD+SMB	57.78 %	84.83 %	62.31 %	79.25 %

is smoother and contains more components per frame, whereas EMD-MBE is limited by the number of IMFs and the number of samples in the window.

Furthermore, empirical mode decomposition suffers from mode mixing. Mode mixing of EMD is mainly caused by intermittence and noise. Sudden changes in the signal are one of the main causes of mode mixing, such as noise interference or a high-frequency wave discrete distribution in the original signal, which results in the signal being a local high-frequency signal, thus producing a local extreme value. The envelope generated by this local extreme value point jump phenomena results in the IMF not agreeing with the time scale and continues to the different frequency components in the original signals, which cannot be effectively separated according to the characteristics of time scale [109]. Mode mixing will affect the subsequent decomposition components; afterward, the time-frequency distribution of the following IMFs will be ambiguous, and, eventually, the EMD decomposition process loses physical meaning [110], [111]. Many researchers have studied this issue and there are several solutions have been given [112]–[114].

The EMD-MBE method requires more computational resources and time to extract features as compared FFT-based MBE. One evident reason is the calculation of IMFs during the decomposition of the signal. Secondly, the programming methods for the calculation of FFT have been highly optimized, which resulted in the current algorithm of Cooley and Tukey [2]. Similar efforts could be made in future for the EMD method in order to reduce the computational overheads.

2.6 Conclusion

The main objective of this study was to introduce an adaptive time-frequency analysis method for an audio signal and perform a comparative analysis with the traditionally used time-frequency analysis method. These methods were evaluated on their performance as

Table 2.4 – Classification accuracy of each class for the Acoustic Scene Classification Dataset.

Classes	Features				
	FFT-MBE	EMD-MBE	SMBE	FFT + EMD	SMBE + FFT + EMD
Airport	53.90%	40 %	37.36 %	40 %	55.094 %
Bus	52.89%	82.23%	40.9 %	63.22 %	76.033 %
Metro	60.53%	32.18%	34.1 %	51.34 %	57.85 %
Metro Station	57.14%	42.85 %	52.5 %	54.44 %	54.44 %
Park	70.24%	63.22 %	66.11 %	74.38 %	65.29 %
Public Square	42.12%	44.9 %	33.33%	47.22 %	49.07 %
Shopping Mall	56.27%	64.87 %	55.91 %	59.50 %	62.72 %
Street Pedestrian	32.79%	28.34 %	37.25 %	39.27 %	36.03 %
Street Traffic	75.61%	72.76 %	72.76 %	76.83 %	80.08 %
Tram	46.74%	54.4 %	50.96%	55.17 %	41.37 %

Table 2.5 – Classification accuracy of each class for the Low Complexity ASC Dataset.

Classes	Features				
	FFT-MBE	EMD-MBE	SMBE	FFT + EMD	SMBE + FFT + EMD
Indoor	78.72 %	77.87 %	73.01%	81.34 %	78.72 %
Outdoor	81.67 %	76.80 %	84.54 %	80.54 %	82.29 %
Transportation	92.83 %	85.28 %	79.90 %	92.60 %	94.15 %

Table 2.6 – Classification accuracy of each class for the Urbansound8k dataset.

Classes	Features				
	FFT-MBE	EMD-MBE	SMBE	FFT + EMD	SMBE + FFT + EMD
air_conditioner	39.2 %	41.5 %	30.6 %	44.9 %	43.1 %
car_horn	70.92 %	32.38 %	20.9 %	74.90 %	71.52 %
children_playing	70.4 %	50.5 %	55.6 %	66.5 %	61.4 %
dog_bark	71.3 %	63.6 %	66.1 %	69.4 %	64.2 %
drilling	60.3 %	60.7 %	54.6 %	65.6 %	64.1 %
engine_idle	51.49 %	51.54 %	40.3 %	52.26 %	58.61 %
gun_shot	84.60 %	52.80 %	24.2 %	76.54 %	70.82 %
jack_hammer	61.58 %	51.73 %	38.9 %	60.00 %	56.35 %
siren	69.18 %	73.35 %	66 %	77.14 %	72.80 %
street_music	78.3 %	57.7 %	62.3 %	72 %	73.7 %

Table 2.7 – Classification accuracy of each class for the custom dataset.

Classes	Features				
	FFT-MBE	EMD-MBE	SMBE	FFT+EMD	SMBE + FFT + EMD
Car Passing	74.24%	53.63 %	65.67 %	81.14 %	81.48 %
Rain	85.28 %	81.42 %	79.09%	89.33 %	88.71 %
Walking	61.57 %	71.09 %	80.52 %	66.29 %	71.81 %
Wind	83.28 %	70.57 %	62.43 %	78.71 %	75 %

features in an environmental sound classification system. The traditionally used method, Fourier transform, is valid under some general conditions and relies on a priori-defined basis. An adaptive method for signal decomposition into multiple components introduced by Huang et al., empirical mode decomposition (EMD), is applied to obtain intrinsic mode function (IMFs) as components. A discrete energy separation algorithm, the Teager–Kaiser energy operator (TKEO), is applied to each IMF individually to obtain instantaneous amplitude (IA) and instantaneous frequency (IF) on a local time scale. Afterward, a windowing function is applied to generate spectrograms, which are summed together. Later, a Mel filter bank is applied to generate log Mel band energies. We also proposed S-MBEs in this study, which use EMD to compute the signal trend, which is subsequently removed from the original signal; later, log Mel band energies are computed using Fourier transform (FFT). The features extracted from the proposed method estimated the change of frequencies with respect to time similar to the traditional method with different intensities. This could be attributed to the fact that the EMD-TKEO method is estimating the IA and IF, which were summed together with a fixed window size to match the dimensions of Mel filter banks. We compared the performance of features extracted with the proposed methods with features extracted from fast Fourier transform-based log Mel band energies. Two different CNN systems were employed in this study to evaluate the feature performance on four different datasets. The results demonstrate that the EMD-based Mel band energies (EMD-MBEs)'s performance lagged behind FFT-based Mel band energies (FFT-MBEs). S-MBE performed the worst among the three features under evaluation for every dataset. The aggregation of all three features resulted in an improvement in accuracy over FFT-MBEs. The improvement reflects the fact that EMD-MBEs performed better for some classes than FFT-MBEs, and the combination of these methods improves the overall result. The analysis of the low performance of the proposed method reveals that in the estimation of time-frequency representation, the resolution is limited by the number of IMFs and window size. Furthermore, during the process of decomposition of

the signal into IMFs, the EMD method suffers from the mode mixing problem, which degrades the quality of the extracted features. In the future, different EMD methods will be under consideration to obtain a better estimation of features with similar performance compared to FFT-based features.

Chapter 3

Environmental Sound classification System Optimization

3.1 Introduction

In the previous chapter, a novel feature extraction method is discussed which uses the empirical mode decomposition method along with the Teager-Kaiser energy operator and discrete energy separation algorithm (DESA) to generate a time-frequency image. Traditionally, ESC systems are trained using spectrograms as input features. Speech recognition methods have employed Mel frequency cepstral coefficients (MFCCs). MFCCs are obtained by applying discrete cosine transform (DCT). The DCT decorrelates the correlation introduced by the overlapping triangular filter banks. Generally, only the first 13 MFCCs are used for classification, as the remaining coefficients provide less valuable information. In this way, a feature size reduction is also obtained. For a classification sound system operating on a small processing footprint, it is essential to reduce the number of input feature size. The larger the input, the more processing time it requires for inference. The goal here is to use fewer parameters while achieving the same results.

For machine learning algorithms, especially neural networks, the accuracy of recognition is dependent upon the scale of the dataset used to train the network. Neural networks perform better when an ample amount of samples are available to train the network. Competitions such as DCASE [6], [22], [23] provides a good source of audio data to use or build custom datasets as the audio files are tagged and provided in a standard form for the competition. Freesound [115], an open-source library for audio files becomes an excellent platform to build over datasets by processing and tagging each audio file. Audioset [116] from Google provides sounds collected from Youtube videos. In this study, we built a custom dataset using the recordings from Freesound[115] of four categories: wind, rain, car passing, and human walk. Though the span of ESC applications overlaps many disciplines, this study focuses on the application of activity detection outside and around a

smart home. The activities such as the passing of a vehicle, rain shower, wind flow, and human gait detection. These categories manifest their usage in the domain of security, development of smart sensors, and automation of smart homes.

In neural networks, apart from the scale of the dataset, the classification accuracy relies on the input of the model. The features are the input of the neural network in audio classification. These features are extracted from spectrograms such as the Mel frequency band (MFB), or from cepstral analysis such as Mel frequency cepstral coefficients (MFCC). MfCC has demonstrated its usefulness successfully in the field of acoustic speech recognition and MFB has been used in research for sound event classification. These features occupy a large amount of memory and cost a lot of computing power in the training of neural networks. The increment of layers of neural networks increases the number of training parameters. Training huge neural networks also cost a significant amount of computational power, time, and energy resources. The computational power could be conserved through the reduction of dimensionality of the input features and also avoid over-fitting of the data on the model. In order to achieve the objective of using fewer parameters while maintaining the same level of accuracy, it is essential to optimize the features used in the classification system. One approach to feature optimization is to extract statistical parameters from the existing features. For example, the mean, standard deviation, skewness, and kurtosis of the extracted features can be calculated and used as input features. This can provide additional information about the distribution of the signal and potentially improve classification accuracy. Another approach is to perform feature selection. Feature selection is a technique that aims to identify the most relevant features that contribute the most to the classification task while discarding redundant or irrelevant ones. This can be done by using various algorithms, such as the sequential feature selection algorithm, which iteratively selects the best subset of features that maximizes the classification accuracy.

Feature selection is utilized to extract useful information from the features. The feature selection methods are categorized into filter method and wrapper method. Filter-based methods select features by measuring the relevancy of features and feature importance by correlation with dependent variables and use statistical methods for the evaluation of features. The wrapper method includes methods whose foundations are derived from nature such as genetic algorithms and ant bee-colony [117], [118]. Wrapper based method, contrary to the filter-based method, uses a classifier to validate the selection of the subset of features. In this study, we used sequential feature selection from the family of wrapper-based methods for handpicking the features. It is an agile search algorithm and has been

applied in various studies. Sequential feature selection [119] is further categorized into four methods: sequential forward Selection (SFS), sequential backward selection (SBS), sequential forward floating selection (SFFS), and sequential backward floating section (SBFS). These methods differ in defining the subset of characteristics, mainly inclusion and exclusion in a top-bottom or bottom-up search, hence reducing the dimension of the feature set to a desired number. The system is trained with an actively sought feature set and tested with a well-performing feature set. The process is illustrated in Figure 3.2.

The rest of the study is organized as follows. In the first section, the background is discussed, followed by a description of the dataset and feature extraction. Later, the proposed system is described, afterward, the experimental set is explained and in the last section results and conclusion are discussed.

3.2 Background

Dimensionality reduction and feature selection are two techniques used in machine learning and data science to reduce the number of variables or features in a dataset. Dimensionality reduction involves transforming the data into a lower-dimensional space while retaining as much information as possible. The most common method of dimensionality reduction is Principal Component Analysis (PCA)[48], which identifies a new set of uncorrelated variables capturing the most significant variation in the original data. Other techniques for dimensionality reduction include Linear Discriminant Analysis (LDA) [120], which identifies a set of variables that best separate the classes, and Non-Negative Matrix Factorization (NMF) [121]. All of these methods require additional processing steps to transform the data into a lower-dimensional space, which can be computationally expensive and may lead to loss of information. Feature selection is a technique used in machine learning and data science to reduce the number of features in a dataset while retaining the most important ones. By selecting the most relevant subset of features, the size of the feature set can be reduced, which can be crucial for classification or prediction tasks that operate on a small processing footprint. The process of feature selection involves ranking the features based on their importance, which is usually determined by a scoring metric such as mutual information or correlation coefficients. Alternatively, machine learning algorithms can be used to automatically select the most relevant features. These algorithms use different criteria to evaluate the importance of features, such as SVM.

The result of feature selection is a reduced set of features that still contains the most

important information for the prediction task. This can lead to faster training and inference times and more interpretable models. Furthermore, reducing the number of features can also help to prevent overfitting, which is a common problem in machine learning where a model becomes too complex and starts to fit noise in the data rather than the underlying patterns.

In this study, we use the feature selection method, sequential feature selection. Sequential feature selection is a type of feature selection method that works by iteratively adding or removing features from a dataset until an optimal subset of features is selected. It is a wrapper method that uses a machine learning model to evaluate the performance of each subset of features. The algorithm starts with an empty set or complete set of features depending on the type of method and adds or eliminates one feature at a time, evaluating the performance of the model after each addition or removal. The feature with the highest score is selected and added to the subset, and the process is repeated until a stopping criterion is met.

Sequential feature selection, contrary to dimensionality reduction, works directly with the original data and does not require any additional processing steps. This can be an advantage in situations where the computational resources are limited or when the original features are important for the analysis. Additionally, sequential feature selection can be used with any machine learning model, making it a flexible and widely applicable method for feature selection. Sequential feature selection and its types are discussed in the next section.

3.2.1 Feature Selection Method

Feature selection is performed to select a subset of features more applicable to the task from all the available feature sets. Thus enhancing computational efficiency and removing unnecessary features such as noise. There are two methods of feature selection known as the filter method and the wrapper method. Here we used the sequential feature selection method which belongs to the wrapper method family. The wrapper method is used to calculate feature weight by using the classification model to analyze the performance of the feature. A classifier is carried out in this technique to examine the subsets by their prediction accuracy after cross-validation. The wrapper technique provides better classification accuracy as compared to the filter technique as the former uses a classifier to fine-tune the particular interactions between the classifier and data set[122]. Also, with the cross-validation it is able to avoid over-fitting of data [122], [123].

Sequential Feature selection is a branch of the greedy search algorithm family which are quite handy in feature dimensional reduction. Converting a p-dimensional feature space into a reduced q-dimensional feature space, where q < p. As feature dimension vectors could be in immense order depending upon the task, but all features do not play a role when classification is performed. The selection of relevant features becomes indispensable to improving the computational efficiency of the system. SFS will add or remove one feature at a time until a q-dimensional subset of features limited by input is reached. The addition or removal of features is performed on the basis of the classification performance of a classifier. Sequential feature selection is further categorized into four methodologies differentiated by underlying algorithms.

Sequential Forward Selection

Sequential forward selection (SFS) is a straightforward greedy search algorithm depiction. SFS begins with an empty subset of features. In each iteration, it adds a new feature from the given feature set as described in the algorithm 2 [124]. The performance of each added feature is analyzed through a classifier using cross-validation and only the best-performing features are kept in the subset. Afterward, a new iteration commences with a tuned selection. The process is repeated until it matches the desired number of features. Hence, only the features which perform better are selected and added to the subset by the SFS algorithm.

Sequential Backward Selection

Sequential backward selection (SBS) functions in a manner contrary to SFS. SBS begins with the complete set of features and, in each iteration, it discards a feature from the remaining set of features. In each iteration, the performance of features is calculated through a classifier using cross-validation, for the removal of a feature. The feature resulting in the least performance is removed from the subset. Afterward, a new iteration begins with a tuned selection. The process is repeated until it matches the desired number of features described in algorithm 3. This removal method has two-fold advantages: firstly, various features can be eliminated and secondly, it permits the option of backtracking, if the removal of feature results in worse performance as compared to the previous subset of the feature, some features removed before can be added back to the new subset in the next iteration for re-evaluation.

Sequential Forward Floating Selection

Sequential forward floating feature selections (SFFS) is an extension to SFS and it rectifies the issue with SFS. In SFS the features added in the subset can not be removed. This issue is addressed in SFFS and a feature selected at any point of iteration can be eliminated. SFFS starts with the selected feature set by SFS and compares the significance of the features discarded before with the selected set and includes or excludes the feature with less or high significance [125]. Therefore, feature selected in the previous stage can be removed one by one, and a feature selected is evaluated with the subset to find out whether the previously shortlisted features contains the most significant set of features. This method permits the removal of undesired and least significant features selected from the initial set, as shown in Figure 3.1. In this way, an optimal feature is added to the subset previously chosen, by elimination of the sub-optimal feature. Thus, resulting in an optimal selection of features for the desired subset. There is no guarantee that the SFS may include sub-optimal features. In that case, SFFS would evaluate each feature and the resulting feature set could be the same as the set of features picked by SFS.

Sequential Backward Floating Selection

Sequential backward floating feature selection (SBFS) is an extension to SBS, just like SFFS is an extension of SFS, and aims to overcome the shortcoming of SBS. The SBFS works like SBS, except this could add the features that were previously eliminated. After adding the feature to the feature set it examines the performance of the classification and measures whether the previously removed feature is an optimal feature to the rest of the eliminated or added features to the subset and only eliminates the features with sub-optimal performance. Similar in goal to SFFS, this process results in an optimal selection of features for the desired subset.

Algorithm 2 Sequential forward selection

1: Start with empty feature set $F_0 = \emptyset$ and $k = 0$
2: Select the next best feature $f^* = \arg\max_{f \in F - F_k} J\left(F_k \cup \{f\}\right)$
3: Update $F_{k+1} = F_k \cup \{f^*\}$; $k = k + 1$
4: Go to 2 till $k == q$.

Algorithm 3 Sequential backward selection

1: Start with empty feature set $F_0 = F$ and $k = p$
2: Select the next best feature $f^* = \arg\max_{f \in F_k} J\left(F_k \setminus \{f\}\right)$
3: Update $F_{k-1} = F_k \setminus \{f^*\}$; $k = k - 1$
4: Go to 2 till $k == q$.

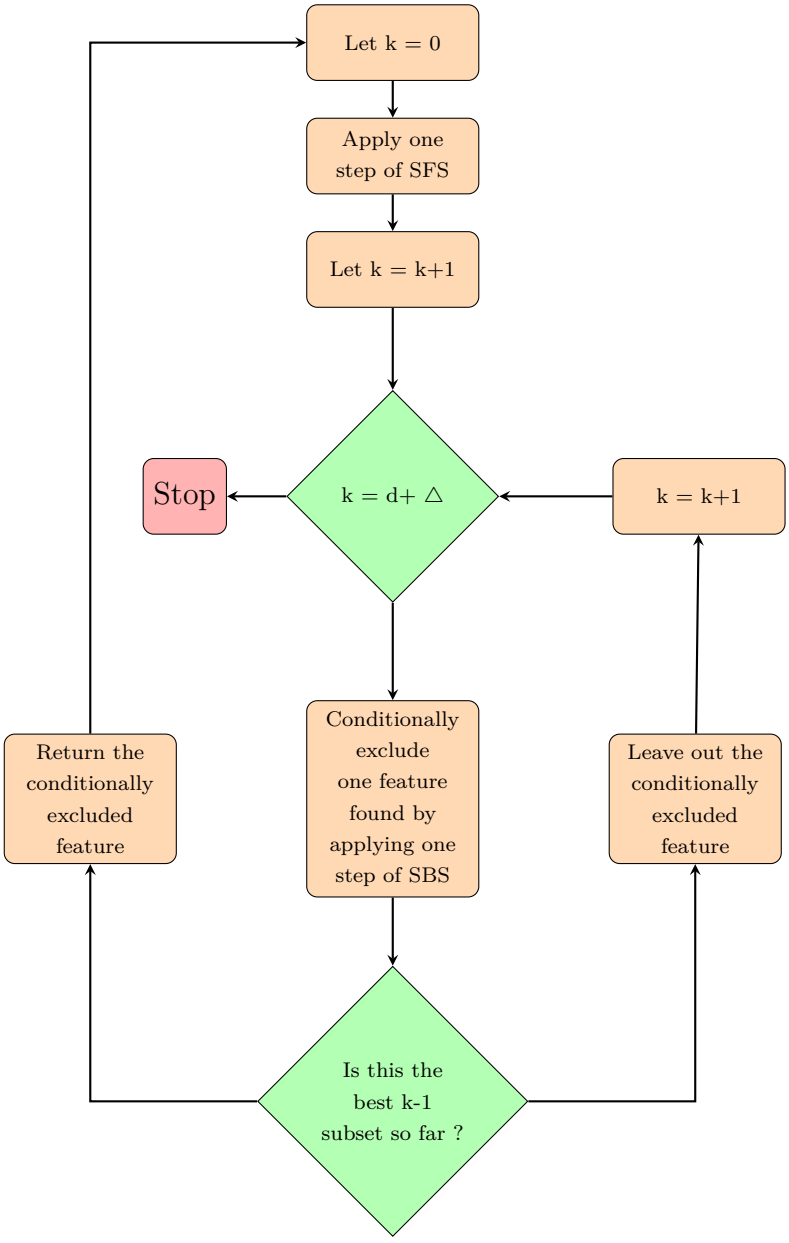


Figure 3.1 – Sequential Forward Floating Selection (SFFS) Algorithm.

3.3 Method description and Experimental Setup

The training of neural networks comprises updating the weights and biases of the neuron in the fully connected layer. System complexity increases with the addition of layers of neurons which would make the system take more time to update the biases, weights, and backtracking. Costing valuable time and energy. An increment of layers of neurons does not guarantee higher performance. Nor, in the case of CNN and RNN, it is guaranteed the increment of convolutional layers or recurrent layers might result in optimal results. Researchers have developed other techniques to better use features rather by increasing the depth of the system [126], [127]. Our focus is to have a good classification rate by using fewer parameters to train the network using a dataset developed for smart homes. Another way of reducing the number of parameters is by using reducing the dimensionality of features. We propose a system to reduce feature dimensions by using a feature selection technique. Afterward, the obtained features are presented to the Long-short term memory neural network. The proposed system is shown in Figure 3.2.

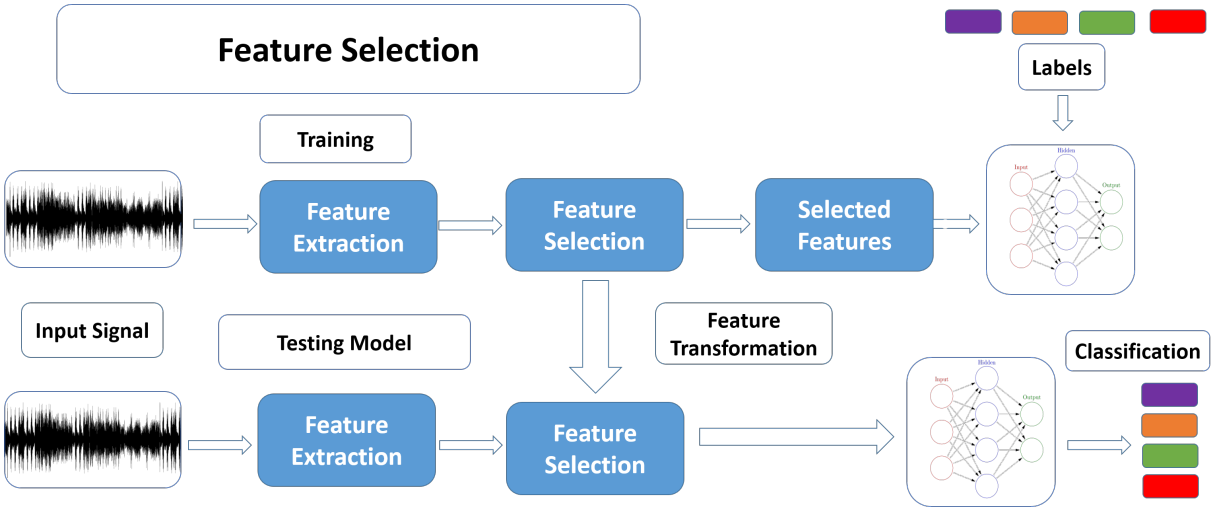


Figure 3.2 – Block diagram of feature selection training and testing process.

3.3.1 Acoustic Data

The acoustic data is a collection of recordings collected from an open-source audio library. A similar dataset was created for the DCASE competition [128], using recordings from the open-source data repository. Each audio file was verified before including in the dataset. The acoustic data contains four categories of sounds present in the environment

i.e. rain, human walk, wind, and car passing. These recordings were gathered through various contributors on FreeSound [115]. Recordings are registered by different users in different environments, and of different audio lengths. They were present in the range of 44100Hz to 96000Hz sampling rate, recorded with various kinds of recording devices, and some in different formats and at a non-similar bit rate. Initial processing of the dataset was applied to modify the recordings into a more standard form. The audio files were processed and set to a sampling rate of 44100Hz, 16-bit .Wav files. In the later stage, 10 seconds audio chunks were extracted and stored as separate audio files. The chunks shorter than 10 seconds were discarded resulting in 750 files. The total duration of the recordings were 125 minutes long for each category. Given in Table 3.1.

Table 3.1 – Sound event categories

Category	***Duration in minutes***
Rain	125
Wind	125
Car passing	125
Human walk	125

3.3.2 Feature Extraction

To obtain a set of features from the audio samples, Mel Frequency Cepstral Coefficients (MFCC) based on the perceptual Mel scale were used. The 10-second file is divided into blocks of 46 ms with a sliding Hamming window of 2048 samples to upscale the frequency resolution of Discrete Fourier Transform (DFT). With an overlap of 50% of the sliding window. These blocks are converted into the frequency domain using DFT to analyze the contribution of every band of the spectrum. A filter bank is used commonly known as a Mel filter bank which consists of a 48 Mel scale filter. Finally, Discrete Cosine Transform is applied to reduce the correlation and the first 13 coefficients are used. Consequently, we obtain a vector of 13 coefficients that characterizes the 46 ms frame of the audio signal. Using Hamming window, 13 MFCCs are obtained for a 10-sec file and are concatenated into one tensor. For each coefficient, four values: mean, variance, kurtosis, and skewness are calculated. These four values for every coefficient are calculated and concatenated into a sequence of 52 features. For MBE features we used 128 Mel bands with the rest of the configuration similar to MFCC. In this study, we denoted MFCCs as FFT-MFCCs and MBE as FFT-MBE, due to the underlying algorithm used to extract these features. In

the previous chapter, two new features were introduced i.e. EMD-MBE and S-MBE. In this section, we used those features to extract EMD-MFCC and S-MFCC. Similar to the process described above for extracting FFT-MFCC, 13 cepstral coefficients for each type of feature were extracted.

3.3.3 Experimental setup description

Features extracted are subjected to the feature selection stage. The aforementioned feature selection techniques are applied with a support vector machine (SVM) classifier, cross-validation is set to 5, and simulated with different subset sizes. The selected feature set is trained on recurrent neural networks, using long short-term memory layers described in Table 3.2. The dataset is divided into the training set and test set, the former containing 600 audio files and the latter containing 100 audio files. From the test set 10% is reserved for evaluation during the training of the neural network. 7-fold cross-validation is applied by dividing the whole dataset into 7 segments and using 6 as the training data set and 1 as the testing data set. As depicted in Figure 3.2, the training of LSTM given in Table 3.2 is performed. In the case of training with feature selection techniques, we used relu units, and tanh in the case of training LSTM directly with MFCC features and not using feature selection techniques. Optimizer for the network used in both cases Adadelta optimizer and metric is set to accuracy. Each neural network was trained for 100 epochs. In the testing phase, the input features are extracted and then transformed into a feature set corresponding to features selected by the corresponding selection method. In the last stage, the transformed feature set is sent to the trained model of the neural network, where classification is performed on the input features.

3.4 Results

In the feature selection stage, each set of features is cross-validated with the SVM classifier. Figures 3.3, 3.5, and 3.6 represent the performance of all 52 features selected with the SFS algorithm. Starting from an empty set, the algorithms start by including a feature in a different feature set. After the inclusion of a new feature, the set is subject to cross-validation using an SVM classifier to analyze the performance of the new feature set created. In the beginning, the performance depicts an exponential rising trend. After including a certain amount of features the performance does not increase and remains

Table 3.2 – Neural network construction

Layers	**Model specification**	
	RNN	***CNN***
Layer 1	LSTM(52,Relu/tanh))	Conv2D(10, Relu) (4,4),stride = 1 MaxPool2D((2, 2), stride=2
Layer 2	LSTM(300,Relu/tanh))	Conv2D(4, Relu) (4,4),stride = 1 MaxPool2D((2, 2), stride=2
FCNN #1-3	Two hidden unit (400, Relu) each	
	Hidden unit(300,Relu))	
	Hidden unit(4,softmax)))	

constant. After adding all we observed a sudden fall in performance as we reached feature 51. If all the features are selected irrelevant features which depreciate the performance might also be included, as evident from the Figure3.3, 3.5, and 3.6. Further investigation into the performance of different subsets to obtain reveals an optimal subset as shown in

Figure 3.4(a) and Figure 3.4(b). Similarly, subsets for EMD-MFCC and S-MFCC are presented in Figures 3.7, 3.9, and Figures 3.8, and 3.10. Different subsets were selected to find the optimal feature combination and were trained on LSTM neural network. The four feature selection algorithms are tested with 7-fold cross-validation and a different size of subsets as shown in Table 3.3.

Table 3.3 displays the classification accuracy of different feature selection methods and input feature types. The feature selection methods used in this study include Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), Sequential Floating Forward Selection (SFFS), and Sequential Floating Backward Selection (SBFS). The input feature types used in this study include Fast Fourier Transform-Mel Frequency Cepstral Coefficients (FFT-MFCC), Ensemble Empirical Mode Decomposition-Mel Frequency Cepstral Coefficients (EMD-MFCC), and S-Mel Frequency Cepstral Coefficients (S-MFCC). The classification accuracy is measured and the results are presented for different values of k (k=10, 20, 30, 35, 40, and 52). The accuracy is reported as a percentage value.

From the table, it can be observed that the classification accuracy varies with different feature selection methods and input feature types. For example, in the case of the

FFT-MFCC input feature type, the SBS feature selection method shows the highest classification accuracy (80.40%) for k=35, whereas the SFS feature selection method shows the highest accuracy (81.42%) for k=40. In contrast, for the EMD-MFCC input feature type, the SFFS feature selection method shows the highest classification accuracy (68.16%) for k=40 and S-MFCC shows the highest classification accuracy (61.07%) for k=35 with SFS. The trend in the feature performance is also evident in Figures 3.11, 3.12, 3.13, and 3.14.

It is also important to note that the classification accuracy is generally lower for the features extracted from the S-MFCC input feature type compared to the other input feature types. This suggests that the S-MFCC feature extraction method may not be as effective in distinguishing between the different types of environmental sound. Also, features extracted from EMD-MFCC were far behind FFT-MFCC, revealing the ineffectiveness of this type of feature in this case.

All the sequential feature selection methods were trained with RNN (SFS-RNN) and performed well in achieving classification accuracy above 75%. At k= 40 and 35, we obtained slightly higher classification among all the feature selection techniques. CNN trained with MBE feature and RNN trained with MFCCs were trained and compared to feature selection techniques, given in.

Table 3.5 presents the results of various feature-classifier combinations in terms of the number of features used, the number of trained parameters, and the classification accuracy. The first row of the table, FFT-MBE-CNN, shows the highest number of trained parameters with 5,812,718 but the lowest classification accuracy of 77.077%. This may be due to the overfitting of the model, this can lead to the model memorizing the training data rather than generalizing well to new data. The second row, FFT-MFCC-SFFS-RNN with k=10 features used, has a lower number of trained parameters and a slightly higher classification accuracy of 78.5%. This indicates that selecting the most relevant features through Sequential Forward Floating Selection (SFFS) can lead to better performance with fewer parameters. As the number of selected features increases to k=35 and k=40, the classification accuracy also increases to 81.35% and 81.152%, respectively. This shows that there is an optimal number of features that can lead to the best performance of the model. However, when the number of selected features increases to k=52, the classification accuracy drops to 79.61%. This suggests that adding too many features can also include sub-optimal features which reduces the model's performance.

Moving on to EMD-MFCC-SFFS-RNN, we see that this feature extraction method does not perform as well as FFT-MFCC-SFFS-RNN. Even with the same number of

selected features, the classification accuracy is lower, indicating that EMD may not be as effective in capturing the relevant information in the audio data. Lastly, S-MFCC-SFFS-RNN also performs poorly compared to FFT-MFCC-SFFS-RNN, with the lowest classification accuracy of 44.71% when k=35 features are used

It is clear that for the proposed system with less amount of features, we obtained a relatively higher classification using only fewer parameters as compared to CNN. RNN trained with MFCC features uses more parameters as compared to feature selection techniques due to the size of the input and fewer parameters than CNN. The accuracy achieved is very low. Table 3.4, further demonstrates the recognition rate per category. Overall, the results suggest that FFT-MFCC-SFFS-RNN with a moderate number of selected features (k=35 or k=40) leads to the best performance in terms of classification accuracy while also having a relatively small number of trained parameters. This indicates that the combination of FFT for feature extraction, SFFS for feature selection, and RNN for classification is effective in recognizing environmental sounds. The results provide useful insights into the effectiveness of different feature selection methods and input feature types using statistical features derived from MFCCs in classifying environmental sounds. The results can be used to inform the development of more accurate and reliable environmental sound classification systems using fewer parameters.

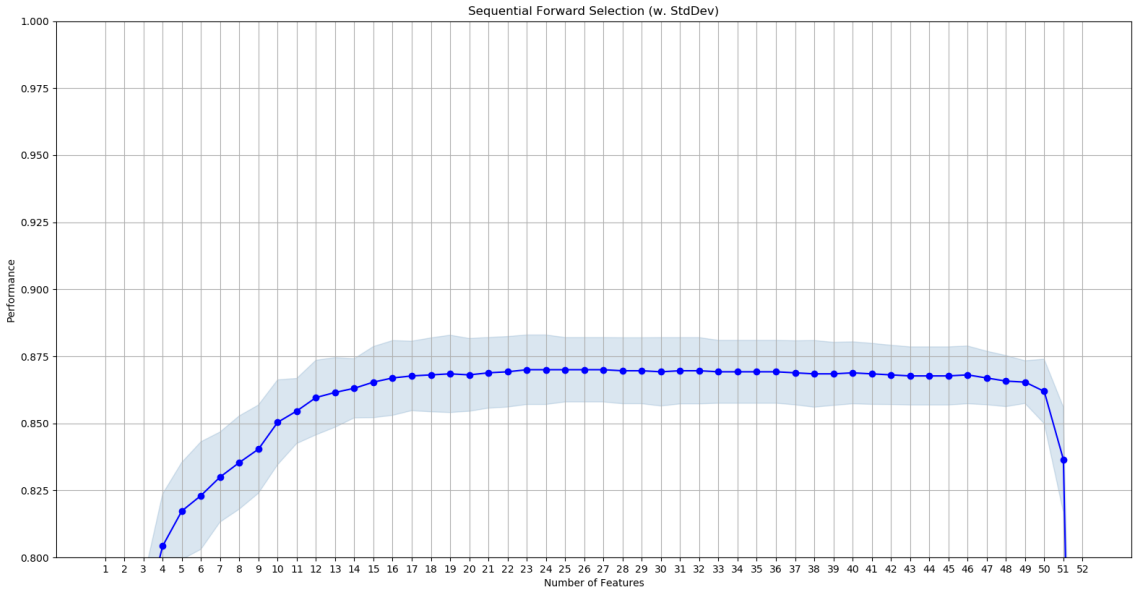
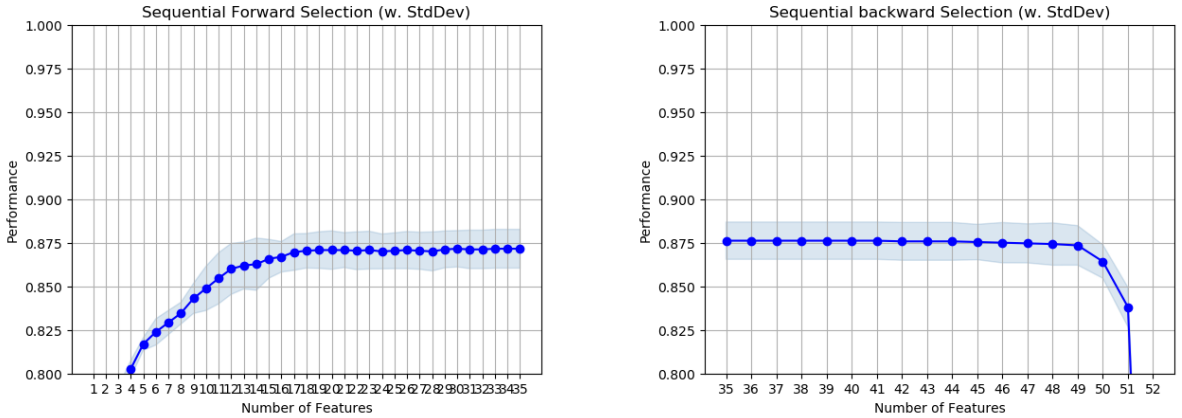


Figure 3.3 – Plots of All feature selected. (k = 52) for FFT-MFCC-SFS.



(a) Plots of SFS feature selected. (k = 35)

(b) Plots of SBS feature selected. (k = 35)

Figure 3.4 – Plots of SFS feature selected. (k = 35) and Plots of SBS feature selected. (k = 35)

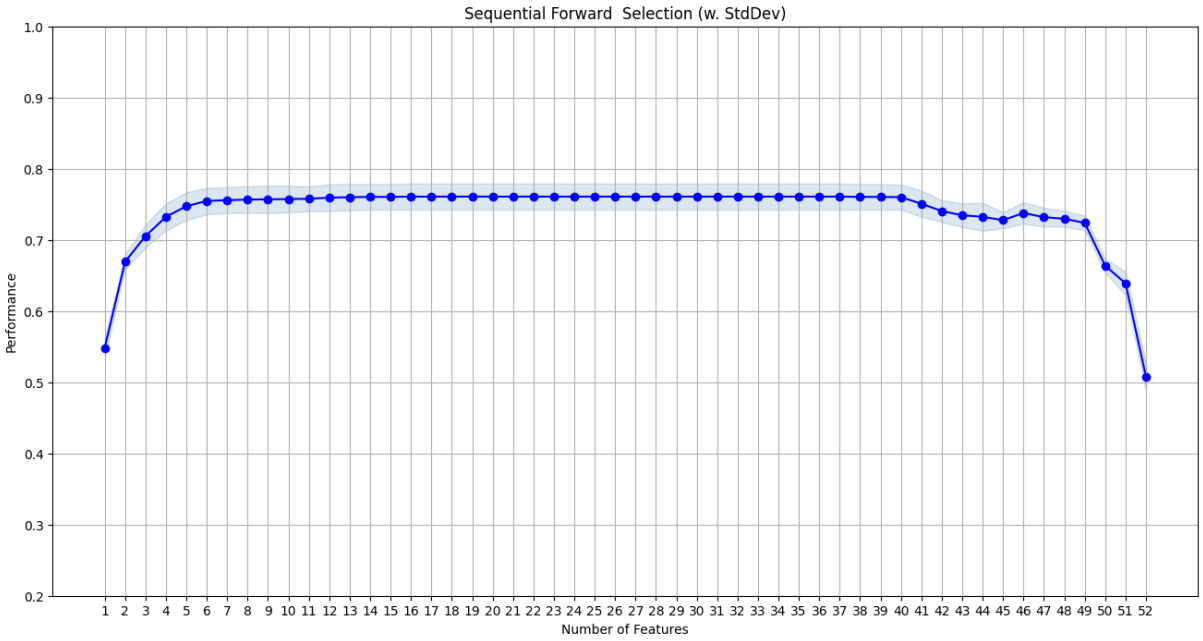


Figure 3.5 – Plots of All feature selected. (k = 52) for EMD-MFCC-SFS.

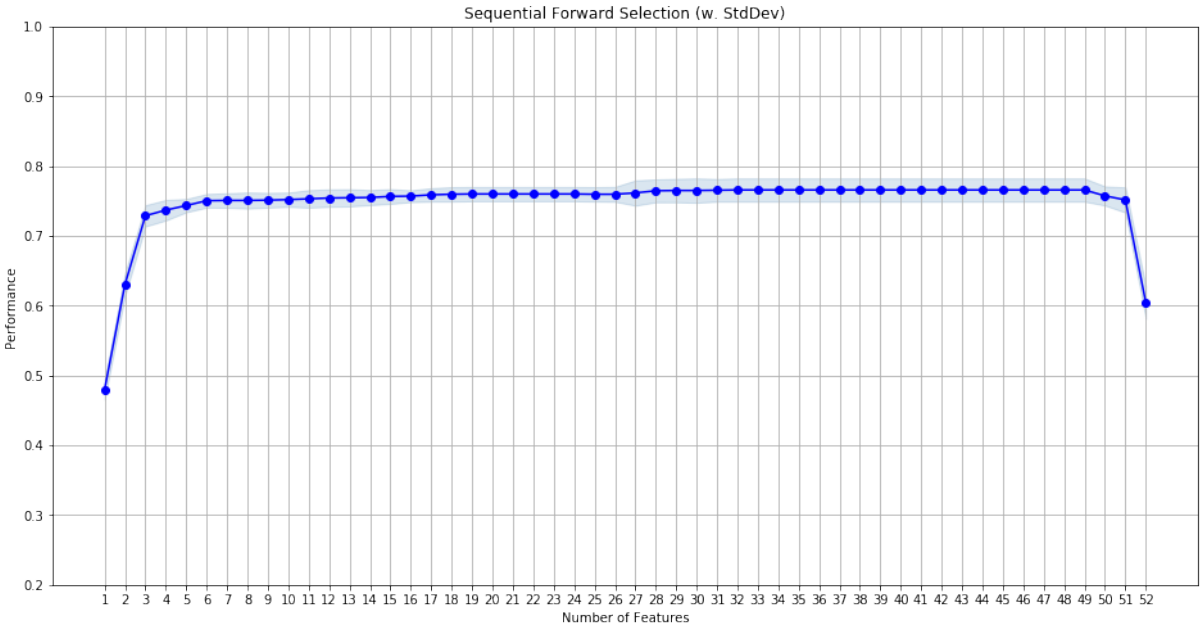


Figure 3.6 – Plot of All feature selected. (k = 52) for FFT-MFCC-SFS.

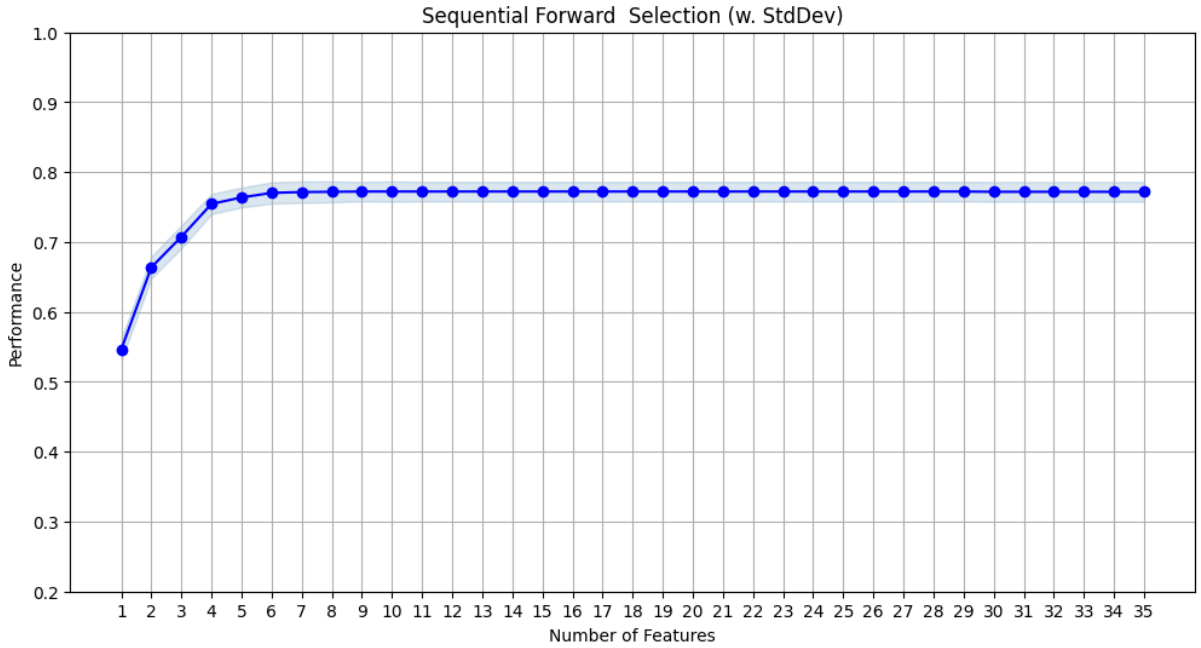


Figure 3.7 – Plot of SFS for EMD-MFCC feature selected (k = 35).

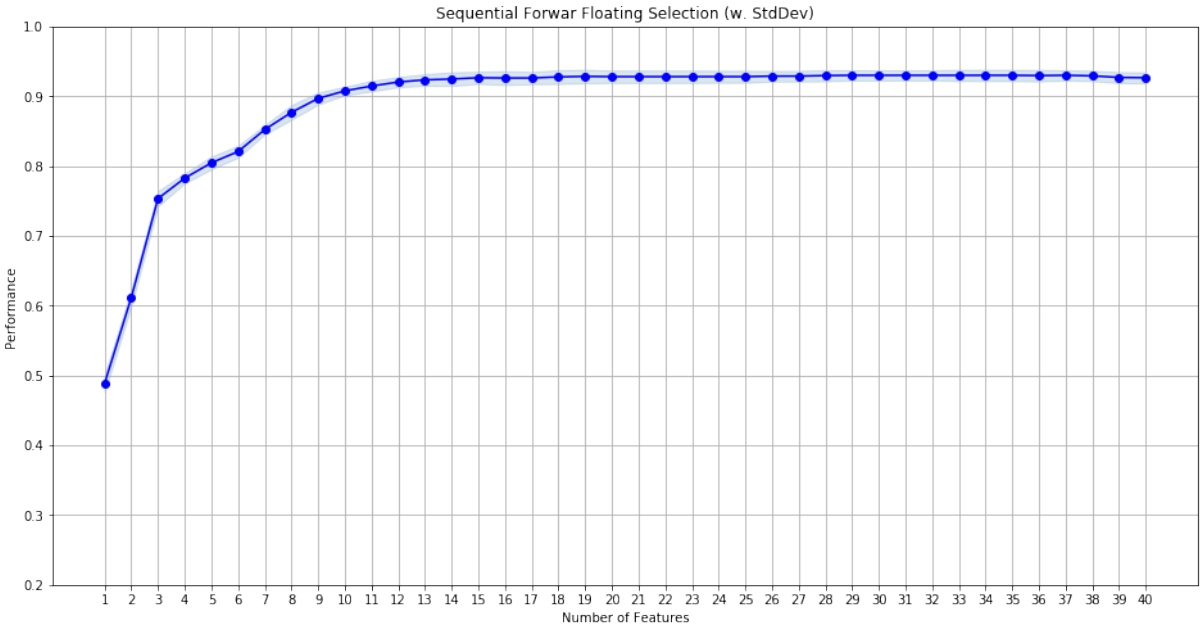


Figure 3.8 – Plot of SFS for S-MFCC feature selected (k = 35).

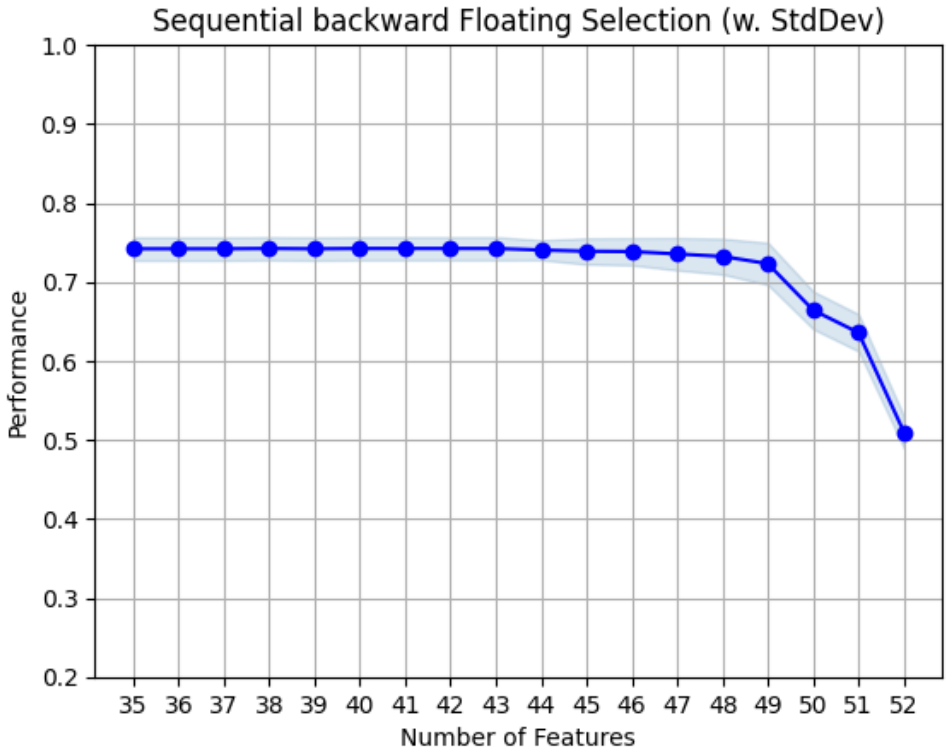


Figure 3.9 – Plot of SBS for EMD-MFCC feature selected (k = 35).

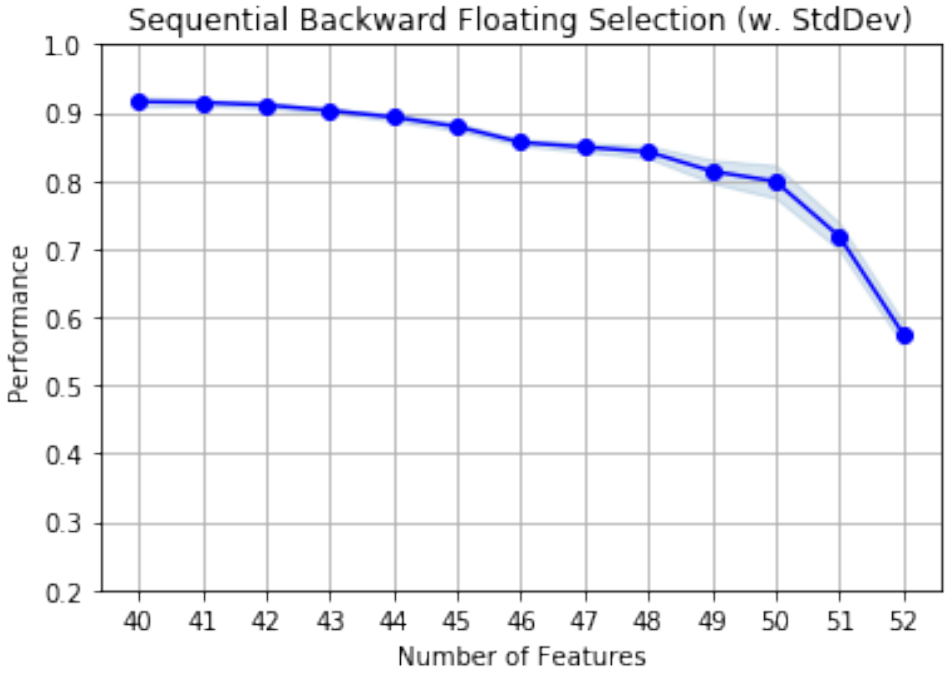


Figure 3.10 – Plots of SBS for S-MFCC feature selected. (k = 35)

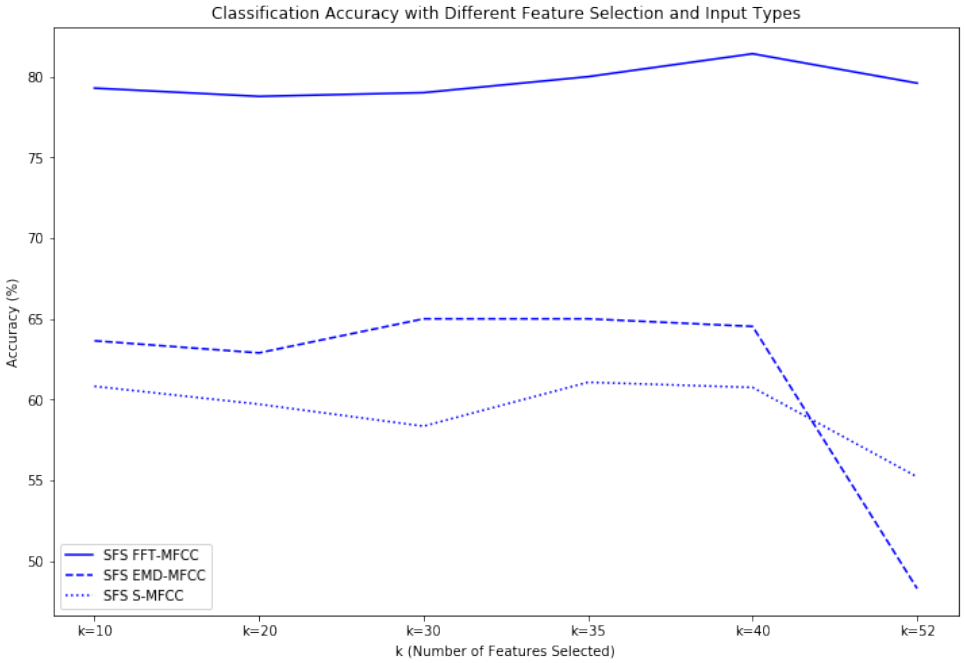


Figure 3.11 – Feature performance of SFS-FFT-MFCC, SFS-EMD-MFCC, and SFS-S-MFCC

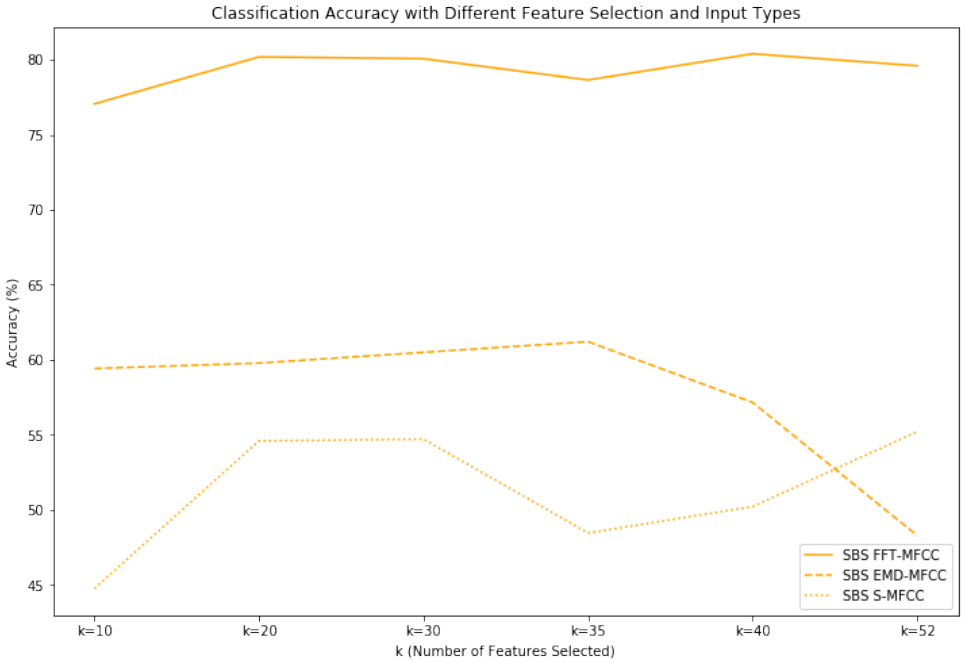


Figure 3.12 – Feature performance of SBS-FFT-MFCC, SBS-EMD-MFCC, and SBS-S-MFCC

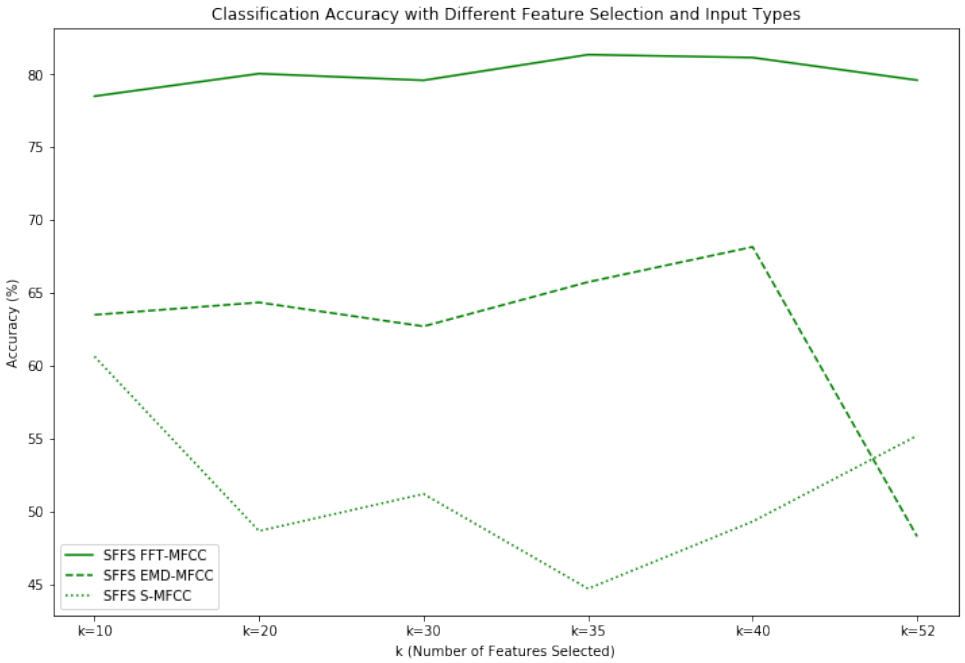


Figure 3.13 – Feature performance of SFFS-FFT-MFCC, SFFS-EMD-MFCC, and SFFS-S-MFCC

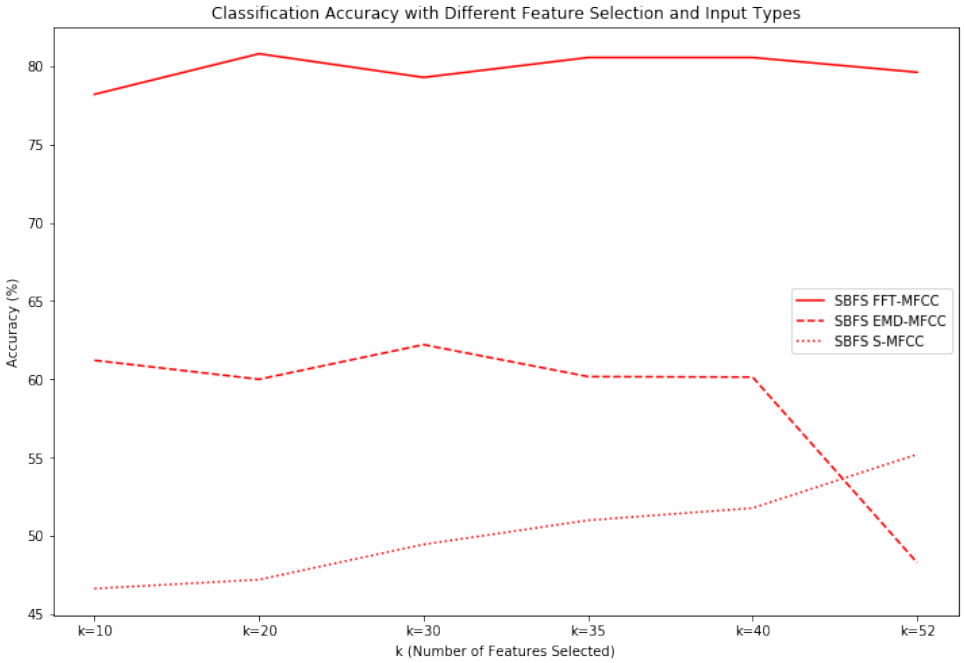


Figure 3.14 – Feature performance of SBFS-FFT-MFCC, SBFS-EMD-MFCC, and SBFS-S-MFCC

Table 3.3 – Classification accuracy with different subsets

Feature Selection	Input Feature Type	Classification accuracy					
		k = 10	k = 20	k = 30	k = 35	k = 40	k = 52
SFS	FFT-MFCC	79.29%	78.78%	79.01%	80.00%	81.42 %	79.6%
SBS	FFT-MFCC	77.06%	80.19%	80.07%	78.652%	80.40 %	79.6%
SFFS	FFT-MFCC	78.5%	80.048%	79.59%	81.35%	81.15%	79.6%
SBFS	FFT-MFCC	78.19%	80.78%	79.27%	80.54%	80.54 %	79.6%
SFS	EMD-MFCC	63.64 %	62.89 %	65 %	65 %	64.53 %	48.29 %
SBS	EMD-MFCC	59.42 %	59.78 %	60.5 %	61.21%	57.16 %	48.29 %
SFFS	EMD-MFCC	63.5 %	64.35 %	62.71 %	65.75%	68.16 %	48.29 %
SBFS	EMD-MFCC	61.21 %	60 %	62.21 %	60.17%	60.14 %	48.29 %
SFS	S-MFCC	60.82 %	59.71 %	58.35 %	61.07 %	60.75 %	55.21 %
SBS	S-MFCC	44.75 %	54.6 %	54.71 %	48.46 %	50.21 %	55.21 %
SFFS	S-MFCC	60.64 %	48.67 %	51.21 %	44.71 %	49.32 %	55.21 %
SBFS	S-MFCC	46.64 %	47.21 %	49.46 %	51 %	51.78 %	55.21 %

k is the number of features selected

Table 3.4 – Classification accuracy per category at k = 35

Feature Selection Method	Input Feature Type	Classification Categories			
		Car passing	Rain	Human Walkig	Wind
MFCC-RNN	FFT-MFCC	61.95%	60.14%	40.42%	58.47%
MBE-CNN	FFT-MBE	71.8 %	74.21 %	83.86%	**77.92** %
SFS-RNN	FFT-MFCC	67%	82%	85%	73 %
SBS-RNN	FFT-MFCC	75.29 %	**84.86**%	85.29%	69.71%
SFFS-RNN	FFT-MFCC	**81.71**%	82.85 %	**87.43** %	73.86 %
SBFS-RNN	FFT-MFCC	74.28 %	82.71%	87 %	73.14%
SFS-RNN	EMD-MFCC	60.28 %	71.57 %	78.85 %	59.28 %
SBS-RNN	EMD-MFCC	53.71 %	62.85 %	73.85 %	54.42 %
SFFS-RNN	EMD-MFCC	61.85 %	63.42 %	71.85 %	65.85 %
SBFS-RNN	EMD-MFCC	51.28 %	64.42 %	71.42 %	54.42 %
SFS-RNN	S-MFCC	55 %	75 %	66.42 %	47.85 %
SBS-RNN	S-MFCC	26.28 %	67.71 %	55.86 %	46 %
SFFS-RNN	S-MFCC	9.85 %	78 %	48.14 %	42.85 %
SBFS-RNN	S-MFCC	13.85 %	79.42 %	66.42 %	38.14 %

k is the number of features selected

Table 3.5 – Parameters trained versus accuracy

Type of Feaure-Classifier combination	Number of features used	Number of Parameters trained	Classification accuracy
FFT-MBE- CNN	-	5,812,718	77.077%
FFT-MFCC-SFFS-RNN	k =10	839,008	78.5%
FFT-MFCC-SFFS-RNN	k =35	844,208	**81.35**%
FFT-MFCC-SFFS-RNN	k =40	845,248	81.15%
FFT-MFCC-SFFS-RNN	k =52	847,744	79.61%
EMD-MFCC-SFFS-RNN	k =10	839,008	63.5 %
EMD-MFCC-SFFS-RNN	k =35	844,208	65.75%
EMD-MFCC-SFFS-RNN	k =40	845,248	68.16 %
EMD-MFCC-SFFS-RNN	k =52	847,744	48.29 %
S-MFCC-SFFS-RNN	k =10	839,008	60.64 %
S-MFCC-SFFS-RNN	k =35	844,208	44.71 %
S-MFCC-SFFS-RNN	k =40	845,248	49.32 %
S-MFCC-SFFS-RNN	k =52	847,744	55.21 %
FFT-MFCC-RNN	-	926,576	55.27%

3.5 Conclusion

The chapter presents a study on the effectiveness of different feature selection methods and input feature types in classifying environmental sounds using neural networks. The database comprised of four categories: rain, wind, passing of car, and human gait. LSTM was trained with the subset of features obtained from the feature selection algorithm. The optimal amount of features also contributed to the less utilization of resources. The results suggest that the combination of FFT for feature extraction, SFFS for feature selection, and RNN for classification is effective in recognizing environmental sounds, particularly with a moderate number of selected features (k=35 or k=40), leading to the best performance in terms of classification accuracy while having a relatively small number of trained parameters. In contrast, the EMD-MFCC-SFFS-RNN and S-MFCC-SFS-RNN combinations show lower accuracy rates compared to FFT-MFCC-SFFS-RNN. Additionally, the results show that the FFT-MBE-CNN combination has a higher number of trained parameters and lower accuracy than the FFT-MFCC-SFFS-RNN combination. The optimal amount of features also contributed to the less utilization of resources. For future work, implement different feature selection techniques on this custom dataset and with different feature extraction techniques. These results provide useful insights into the effectiveness of different feature-classifier combinations in classifying environmental sounds and can inform the development of more accurate and reliable environmental sound classification systems using fewer parameters.

Futur work involves implementing different feature selection techniques on this custom database and with different feature extraction techniques. Also, applying features selection techniques on the different available datasets. Further investigation is required for the development of the proposed method for a shorter duration. In this study, 10-sec long files were used. The results may differ depending on the length of the signal. For example, the passing of a car is completely represented in a 10-sec file, however, in a shorter duration such as 1 sec, the stats may change and may lead to different results. In the future, further optimization in terms of the reduction of the size of the number of layers could be done for the implementation on a low processing power edge device.

Part III

Hardware Implementation

Chapter 4

Hardware Implementation of Environmental sound classification system

4.1 Introduction

Environmental sound classification is a crucial technology that finds its applications in various real-world applications, including home automation, security systems, and acoustic event detection. In the previous chapter, the optimization of the Environmental Sound Classification (ESC) system was discussed using sequential feature selection. This optimization approach involved reducing the number of features used in the system to enhance its efficiency. In this chapter, the implementation of ESC on an edge device is presented. This implementation of ESC on the edge device is critical as it enables the system to process sound data in real-time and hence provide timely responses in various sound-based applications including ESC.

Traditional approaches to sound classification involve processing audio signals on powerful computers or cloud servers, which can be costly and not practical for resource-constrained devices With the proliferation of low-power microcontrollers, there is a growing interest in implementing sound classification algorithms on these devices. Machine learning techniques, such as Convolutional neural networks (CNNs), have improved the classification of environmental sounds[129]. Recent developments in this domain lead to numerous public competitions and several data sets have been published for the research community [11], [18], [130]. These datasets consist of recordings for environment classification and acoustic event detection. Few publicly available data sets contain audio recordings in a particular environment for detection of sound events [130], [131]. Other datasets have been gathered with a focus on the detection of acoustic scenes [11], [132]. The systems that are developed using these databases in the competition mostly focus on

achieving high accuracy as compared to the baseline systems based on the evaluation metrics proposed by the organizers [22]. These systems are tailored to achieve high accuracy scores regardless of the complexity and computational costs involved. Recently, organizers of such competitions have encouraged the researchers to tackle the issue of complexity and propose solutions to cater accuracy vs complexity problem [68]. The researchers are bound to focus on the complexity of the models trained and disregard the complexity of the feature extraction stage and processing time of the inference of the model, which requires more attention for the implementation stage on the microcontroller.

The emergence of the Internet of Things (IoT) has led to a significant shift towards performing data processing on edge nodes rather than transmitting the data to a central hub. This approach enhances the ability of nodes to process data at the edge, reducing the need for power-intensive data transfer. Since edge nodes are often battery-operated, there is a growing need for energy-efficient applications with a small footprint. However, traditional approaches to sound classification involve processing audio signals on powerful computers or cloud servers, which may not be practical for resource-constrained devices. With the rise of machine learning on microcontrollers, there has been an increased interest in implementing sound classification algorithms on these small, low-power devices. This chapter will explore the feasibility of using machine learning models, specifically Convolutional Neural Networks (CNN) and Depthwise Separable Convolutional Neural Networks (DS-CNN), for sound classification on microcontrollers, including an evaluation of their accuracy and processing time. The focus will be on developing a less processing time application, with a trade-off between complexity and accuracy, for resource-constrained IoT devices. We will then discuss the implementation of these models on microcontrollers, including memory and processing time. The STM32L4-based SensorTile and CMSIS-NN Library will be used as the platform for this study. The results of this study will be useful for researchers and developers interested in deploying sound classification applications on resource-constrained IoT devices.

In the experimental section, we will evaluate the performance of CNN and DS-CNN models on a microcontroller platform, specifically the SensorTile. We will use the custom dataset to train and test the models and evaluate their accuracy, memory usage, processing time, and complexity. We will compare the performance of the two models and analyze their trade-offs in terms of accuracy, complexity, and inference duration.

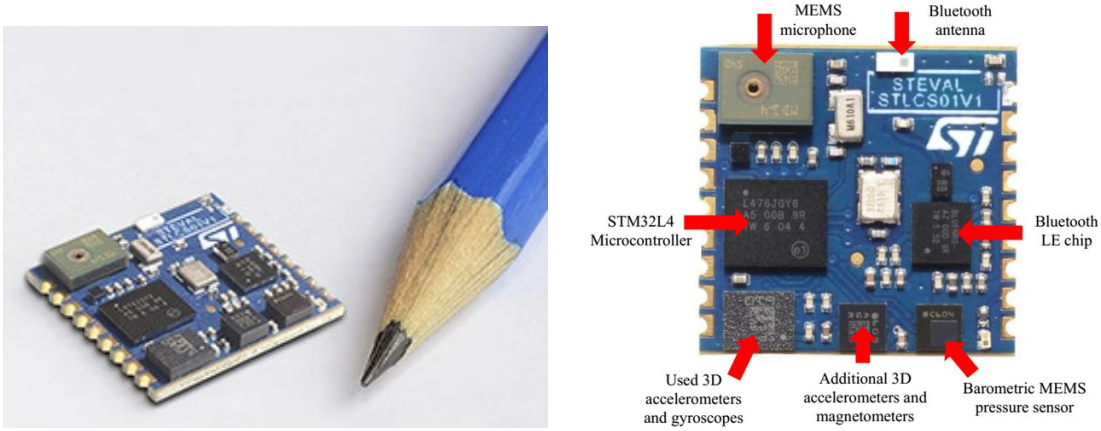
4.2 Background

The development of a sound classification system consists of three major steps i.e. database creation, feature extraction, and machine learning mode training and testing. Among hundreds of event classes present in the environment, datasets are usually constructed with a limited number of classes. In this work, four classes are selected and the selection of these classes was made in the context of smart homes, where these edge devices can be installed to detect the sounds in the environment and pass this information to a central hub. A classification model based on a convolutional neural network is generated by training the model with the subset of the database and testing it with the rest of the recordings. A similar model using a depth-wise separable convolutional neural network is trained to perform the comparison between the models. In order to ensure that the model can be deployed on microcontrollers, it is compressed to match the available storage capacity and RAM. The model is then implemented on a microcontroller, and its processing time is measured to evaluate its performance.

4.2.1 Hardware Description

The environmental sound classification system is operated on the SensorTile development kit shown in 4.1(a) [133]. The STM32 SensorTile is a compact, low-power, and high-performance sensor development kit designed by STMicroelectronics. The kit includes a small printed circuit board (PCB) with a wide range of sensors and a powerful microcontroller as shown in Figure 4.1(b) [133]. The STM32 SensorTile is ideal for developing and prototyping a wide range of Internet of Things (IoT) applications, including wearable devices, smart homes, and industrial automation. The STM32 SensorTile is designed for low power consumption, making it an ideal choice for battery-powered applications. The STM32 SensorTile is powered by a powerful STM32L4 microcontroller, which is based on the ARM Cortex-M4 core. The microcontroller has a maximum clock speed of 80 MHz and comes with 1 MB of Flash memory and 128 KB of SRAM. The STM32 SensorTile also includes a built-in digital MEMS microphone. This allows for capturing and analyzing sound data in IoT applications. It is connected via inter ic sound (I2S) protocol and is connected to the direct memory access (DMA) controller and provides a digital filter for pulse density modulation to convert pulse density modulation (PDM) bitstream into pulse code modulation (PCM) in hardware depicted in Figure 4.3. The STM32 SensorTile also includes an SD card slot, which allows for expandable storage of sensor data or other

information and also accessing stored files using a microSD card as shown in Figures 4.2, [133].



(a) SensorTile board size comparison.

(b) SensorTile onboard components.

Figure 4.1 – SensorTile board size and microprocessor and sensors equipped onboard.

4.3 Dataset and Pre-processing

4.3.1 Custom Dataset

The recordings of four categories i.e. rain, wind, human walk, and car passing are collected through an open-source repository Freesound [134]. Each audio recording was verified before adding to the database. Recordings were made available by different users with different specifications and were open for public access. The dataset had different lengths, and sampling rates and was recorded with different devices. A few recordings were made by the authors using the commercially available multi-sensor board, named SensorTile [133]. The SensorTile comprises multiple sensors, which allow collecting, and storage on the board using µSD mass storage or transmitting to the computer using a USB port the data coming from different sensors on the board. In view of our application, audio signals were obtained from a digital MEMS microphone on the board implemented through audio pre-processing pdm2pcm conversion and a high bass pass band tuned to acoustic bandwidth. The audio recordings were first converted to 16-KHz sampling rate and 16-bit Wav files. The recordings from SensorTile were recorded in 16-KHz format. Afterward, the total dataset was distributed into two categories i.e. Training set and

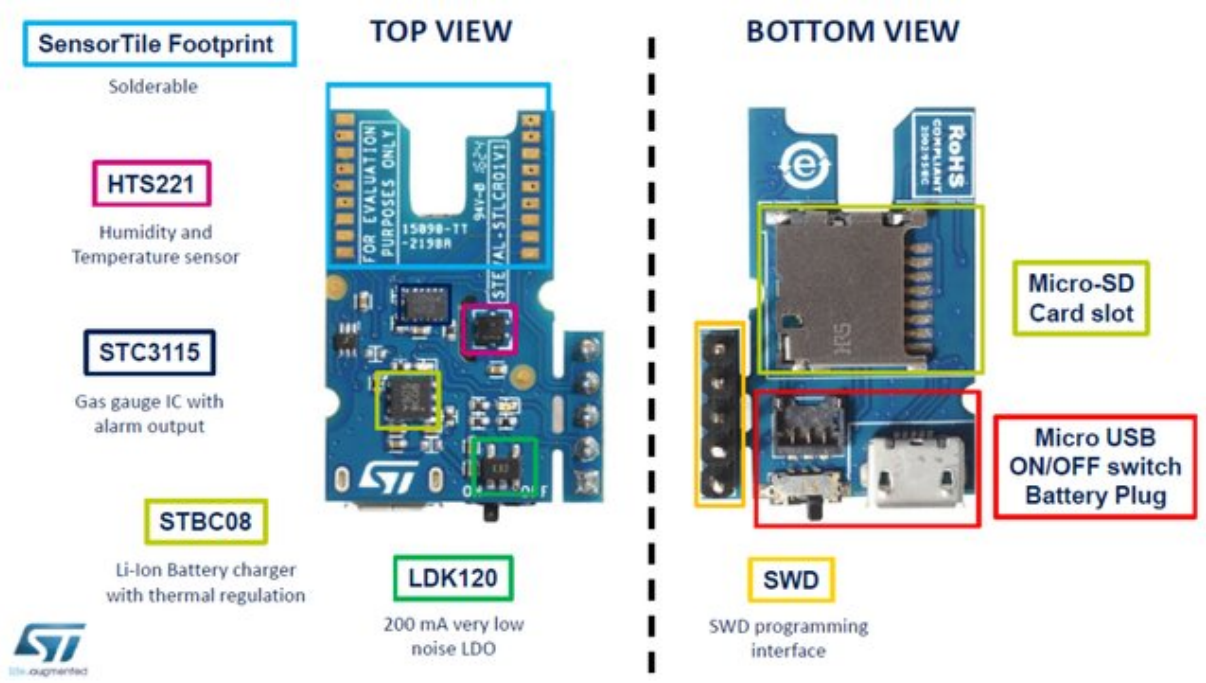


Figure 4.2 – SensorTile Cradle and onboard components description.

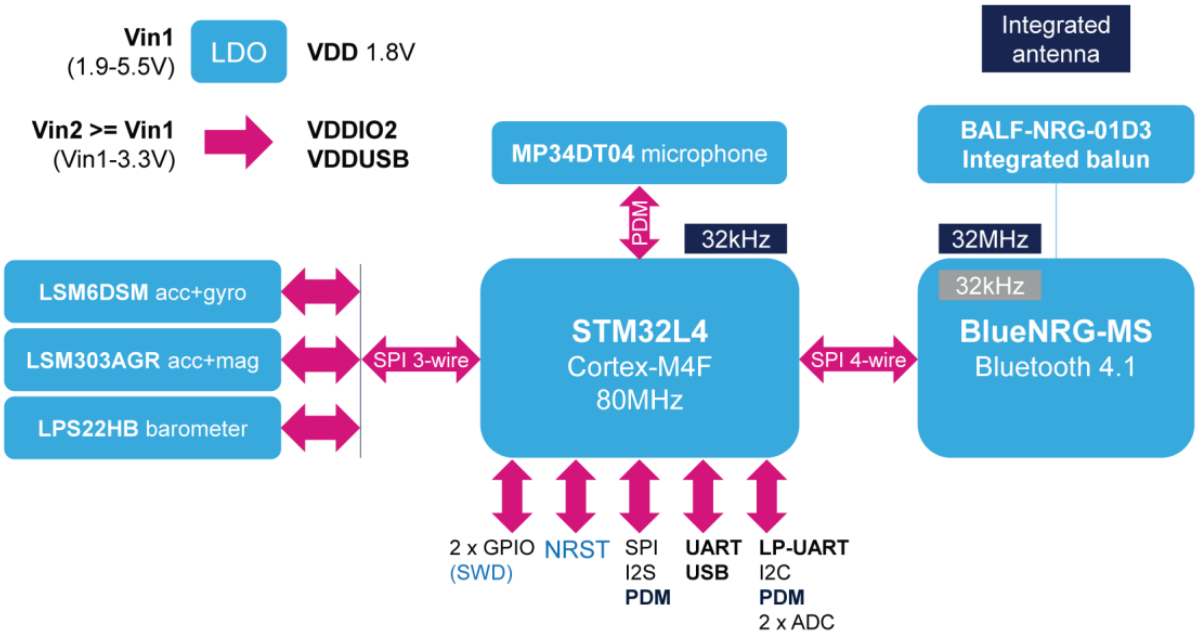


Figure 4.3 – SensorTile board components and connectivity with STM32L4 processor block diagram [133].

Testing set with 80-20 distribution. The recordings made with SensorTile were added to the test set only. The length of recordings present in each set with respect to each category is presented in Table 2.1. The train set is further divided into train and validation sets, where 20 percent of recordings were used as validation data during the training of the neural network.

4.3.2 Feature Extraction

In machine learning, pre-processing raw audio data is an essential step before passing it to a neural network for classification. The pre-processing step involves transforming raw audio signals into a format that can be easily analyzed by a machine learning algorithm. This process may include several steps such as sampling, filtering, and feature extraction. Sampling involves converting continuous audio signals into discrete signals at a specific frequency, known as the sampling rate. This process allows the audio signal to be digitized and processed by a digital processor. In this chapter, the data is provided in the form of a PCM audio stream. Filtering is used to remove unwanted noise or interference from the audio signal. This can be achieved using various filtering techniques such as high-pass, low-pass, or band-pass filters. Feature extraction is the process of selecting and extracting relevant information from the audio signal, such as pitch, frequency, and amplitude. The extracted features are then used as inputs to a neural network to train a model for classification or analysis.

Another step that is included in the feature extraction process, is the scaling of the feature extracted. The feature is scaled to ensure the features in a data set are measured on a consistent scale. In this study, we used a z-score scaling to measure the impact of the scaling on the input data and the classification score. z-score scaling is more commonly applied to the Mel spectrogram as compared to other scaling methods such as min-max scaling. Min-max scaling (also known as normalization) rescales the data to a fixed range between 0 and 1. In spectrograms, the amplitude of the signal can vary widely depending on the characteristics of the sound being analyzed. Min-max scaling would not be ideal for spectrograms since it may distort the original amplitude range of the signal, resulting in loss of information. On the other hand, Z-scaling standardizes the amplitude of the signal by dividing each data point by the standard deviation, resulting in values that are centered around zero with a standard deviation of one. This can be helpful when working with spectrograms since it helps ensure that the features are not influenced by the signal amplitude and are comparable across different spectrograms.

Pre-processing audio data is a crucial step in machine learning, as it enables the neural network to better understand the characteristics of the audio signal and make accurate predictions or classifications. Therefore, careful consideration should be given to pre-processing techniques to ensure that the audio data is appropriately transformed for optimal performance of the machine learning algorithm.

4.3.3 Z-Score Scaling

Feature scaling is an aspect of Machine Learning that ensures the features in a data set are measured on a consistent scale. This concept originated from statistics and involves placing various variables on the same scale, particularly when dealing with data sets that have varying scales. In some cases, the range of features in a dataset may vary significantly, and standardization is employed to bring all features onto the same scale. Therefore, feature scaling is a commonly used technique to address differences in feature ranges and ensure consistency in data analysis.

Z-score scaling, also known as standardization, is a common technique used in feature scaling for Machine Learning. It involves transforming the features in a data set to have a mean of zero and a standard deviation of one. In [135], authors applied z-score standardization on the Log Mel spectrogram to standardize the features.

To apply z-score scaling, one can subtract the mean of the feature from each value and then divide the result by the standard deviation. The resulting transformed values, known as z-scores, provide a relative measure of how far each data point is from the mean in terms of the number of standard deviations. This technique ensures that all features have the same scale and allows for easier comparison of features. The equation of z-score, also known as the standard score, can be written as:

$$z = \frac{x - \mu}{\sigma}$$

where x is the raw score, μ is the mean of the population, and σ is the standard deviation of the population. The z-score measures how many standard deviations away from the mean of a particular raw score is, and it is useful for comparing values from different populations or distributions. A z-score of 0 indicates that the raw score is equal to the population average. In contrast, positive and negative z-scores indicate that the raw score is above or below the mean, respectively, in terms of standard deviations.

4.3.4 Audio signal accusation and feature extraction

The feature extraction method involves using a 2D time-frequency representation of the audio signal. To achieve this, log-Mel energies are computed from the audio PCM samples, the commonly used method in ESC [136]. The PCM samples are divided into non-overlapping frames of 1024 samples (64ms), and a feature matrix is computed every 32ms with 50% overlapping. Each frame contains 512 new samples at a sampling rate of 16 KHz. An asymmetric Hanning window is applied to avoid spectral leakage. Next, 1024 samples are subjected to an FFT calculation using floating-point 32 to obtain a power spectrum. A Mel filter bank lookup table is then used to calculate Mel energies, and a logarithm is applied to obtain log Mel band energies. A total of 30 Mel filter banks are used for each frame, resulting in a 30x32 matrix.

Finally, this matrix is used as the input to a convolutional neural network, which is used to classify the audio based on its features. By using a pre-processed feature matrix, rather than raw audio data, we can improve the performance and efficiency of the neural network, making it easier to train and deploy on a device. The description of audio accusation, feature extraction, and inference is depicted in Figure 4.5. In the case of scaling the feature, the z-scaling parameters i.e. means and variances are computed for the training set. The z-scaling is applied to each input log Mel spectrogram before the inference step.

4.4 Experimental Setup

In this chapter, we present the evaluation of five deep neural network models for environmental sound classification tasks. The models were trained and evaluated on a custom dataset consisting of four categories: walk, rain, wind, and car passing. The input features for the models are log Mel spectrograms and z-scaled log Mel spectrograms described in the feature extraction section. The dataset is divided into train set and test set 2.1. Five-fold cross-validation is applied on the dataset as defined in section 1.5.6. The classification model are presented in Tables 4.1, 4.2 and 4.3. The classification model in Table 4.1 is named CNN-0. Table 4.2 contains two CNN classification models, CNN-1 and CNN-2. Table 4.3 also consists of two DSCNN models, DSCNN-1 and DSCNN2. The models are trained using TensorFlow libraries and later converted into TFLite model [137].

Afterward, int8-bit quantization is applied to reduce the size and complexity of the model. The inputs and outputs are also changed to 8bit from float32 using scaling and zero point information extracted from the quantized model. The quantized model is validated

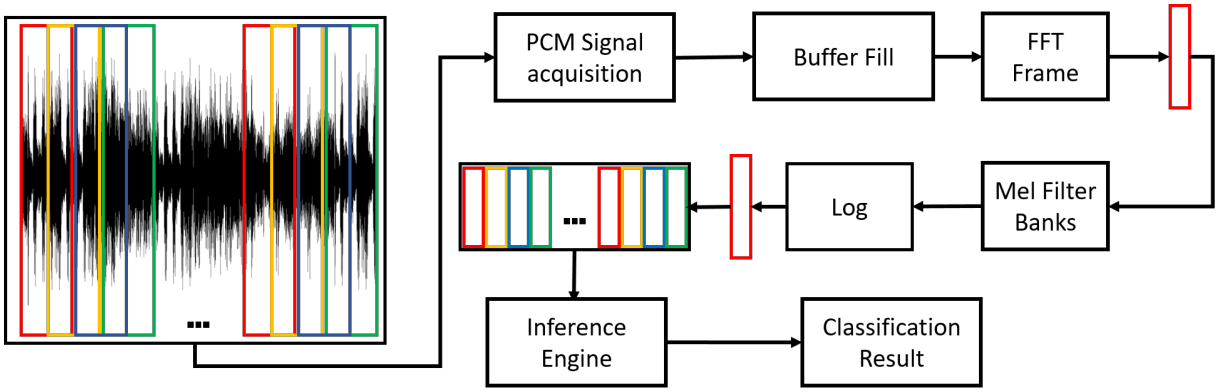


Figure 4.4 – Block diagram of classification process on a microcontroller.

with the quantized inputs using the train set. In the next stage, the quantized models are implemented on STM32 SensorTile using CMSIS and STM32 AI libraries. The model is then tested on the micro-controller by running the model completely on the microprocessor and the audio files for the test set are stored on the SD card. The audio is read through the memory by the DMA controller in a 16-bit PCM buffer of 1024 samples with 50% overlap. The feature matrix is calculated every 32ms seconds. The chain of feature extraction is shown in Figure 4.5. These features are fed to the quantized model to perform inference. The classification models are run on the devices and the inference time of each model is measured. The block diagram 4.4 depicts the process starting from signal acquisition to the inference and classification results.

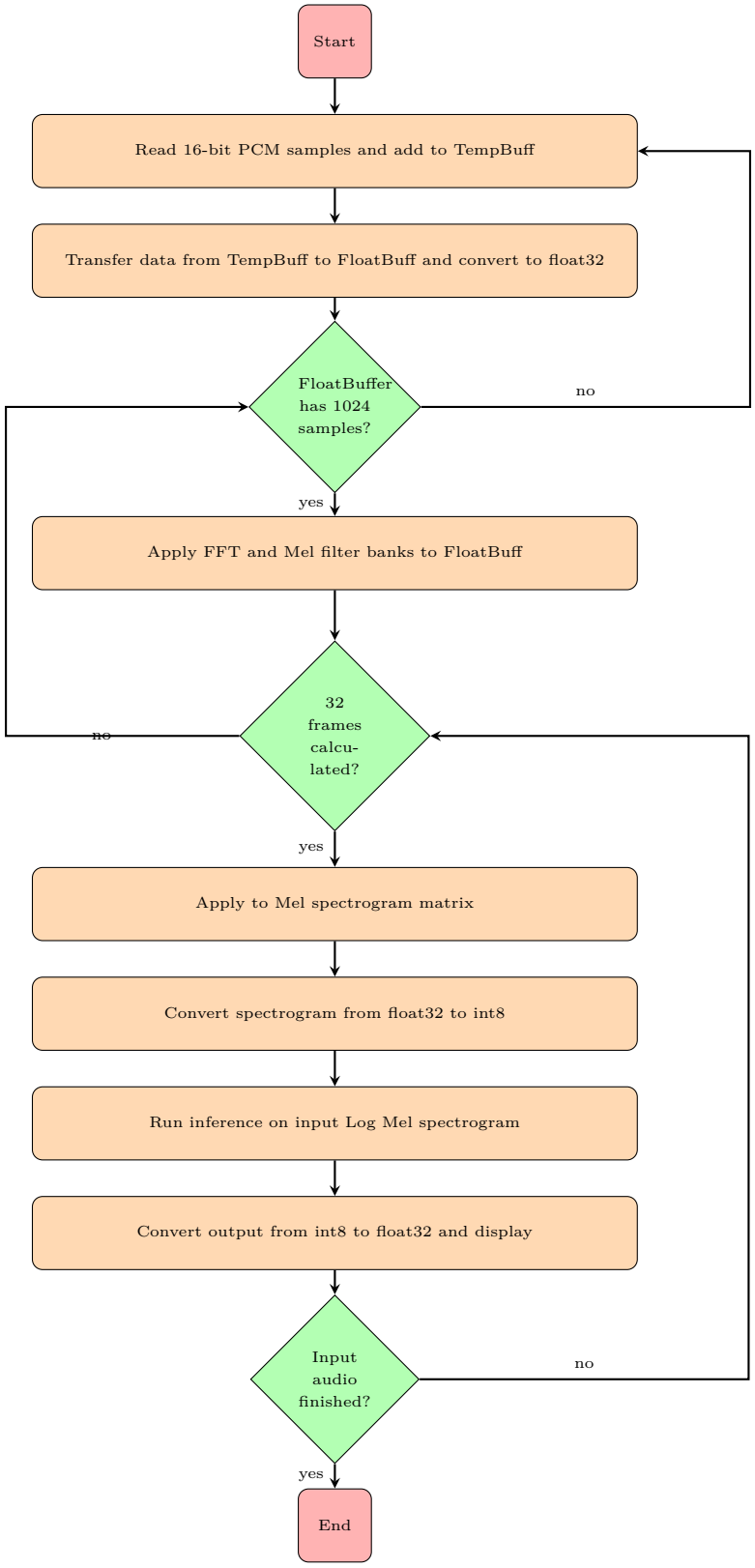


Figure 4.5 – Processing chain for inference on STM32 SensorTile.

4.5 Results and Discussion

Figure 4.6 provides the overview of the accuracies of nine different results of five neural network models on five different folds of the dataset. The neural network models CNN-0, CNN-1, CNN-2, DSCNN-1, and DSCNN-2 are described before. The models CNN-1-zScaled, CNN-2-zScaled, DSCNN-1-zScaled, and DSCNN-2-zScaled uses the same model as aforementioned but with z-score scaling applied to the Mel spectrograms. For each model, five-fold cross-validation is applied. The CNN-0 model has an average accuracy of 82.03% and it is the first model trained to perform classification. The parameters used by this model are 311,584, which is the highest number of parameters among all the models presented. Further optimization is done, in terms of reducing the number of parameters by using a different combination of CNN. The models CNN-1, CNN-2, DSCNN-1, and DSCNN-2 are trained with the goal of reducing the number of parameters and also producing similar results. CNN-1 has an average accuracy of 79.37%, with a range of 74.38% to 83.04%. This model has 50,212 parameters and uses 1,044,336 Multiply-Accumulate Operations (MACCs) as shown in Table 4.4. CNN-2 has an average accuracy of 80.08%, with a range of 77.47% to 82.43%. It has 55,575 parameters and uses 1,221,875 MACCs, which is relatively higher than CNN-1 because of the difference in the number of filters used in CNN and the number of neurons in the first layer of deep layers.

Depth wise separable convolutional neural networks are also used, as they use less number of parameters and consequently less number of MACCs. DSCNN-1 has an average accuracy of 80.72%, with a range of 77.60% to 86.60%. It is a depthwise separable convolutional neural network. DSCNN-2 has an average accuracy of 79.10%, with a range of 76.14% to 82.71%. The results of the z-scaled are as follows: CNN-1-zScaled has an average accuracy of 78.97%, with a range of 76.01% to 79.01%. NN-2-zScaled has an average accuracy of 79.06%, with a range of 78.04% to 80.17%. DSCNN-1-zScaled has an average accuracy of 77.80%, with a range of 71.68% to 81.10% and DSCNN-2-zScaled has an average accuracy of 76.08%, with a range of 68.83% to 77.56%.

Overall, we can see that the best-performing models are DSCNN-1 and CNN-2, with average accuracies of 80.72% and 80.08%, respectively. These models have a range of accuracies that is not too wide, indicating that they perform consistently well across the five folds. On the other hand, the worst-performing models are DSCNN-2-zScaled and DSCNN-1-zScaled, with average accuracies of 76.08% and 77.80%, respectively. These models have a wider range of accuracies, indicating that they are less consistent in their

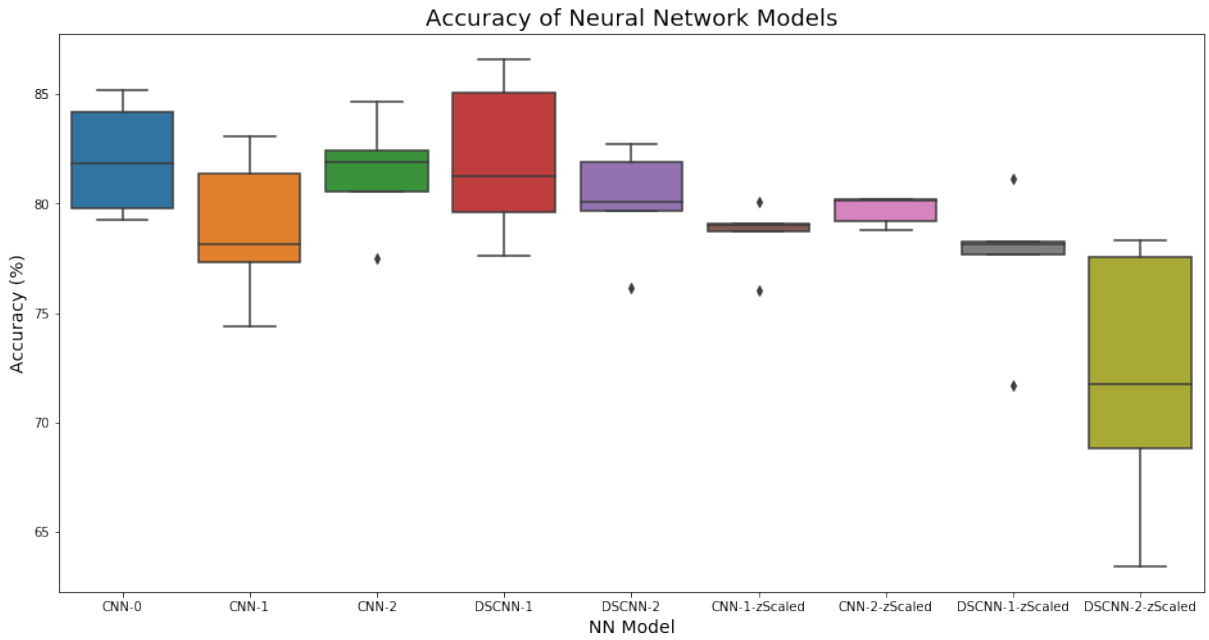


Figure 4.6 – Model accuracy over five-fold cross-validation.

performance across the five folds.

Table 4.6 shows the accuracy metric measures how often the neural network correctly predicted the class of the input data. It is calculated by dividing the number of correct predictions by the total number of predictions. In this table, we can see that the DSCNN-1 model achieved the highest accuracy of 0.8659, while CNN-2 achieved an accuracy of 0.8467. The table also demonstrates the inference duration of each neural network, measured in milliseconds. Inference refers to the process of using a trained neural network to make predictions on new, unseen data. Inference duration is an important performance metric, especially for real-time applications that require quick predictions. In this table, we can see that the DSCNN-1 model has the shortest inference duration of 52 ms, while CNN-2 takes 68 ms to make a prediction. The last column shows the number of multiply-accumulate operations (MACCs) required for each neural network's inference. MACCs are a measure of the total computational workload of the neural network, and they are calculated by multiplying the number of weight parameters in the network by the input size. In this table, we can see that DSCNN-1 requires the fewest MACCs, with only 249,997, while CNN-2 requires the most, with 1,221,875. The inference rate vs number of MACCs is also depicted in Figure 4.7.

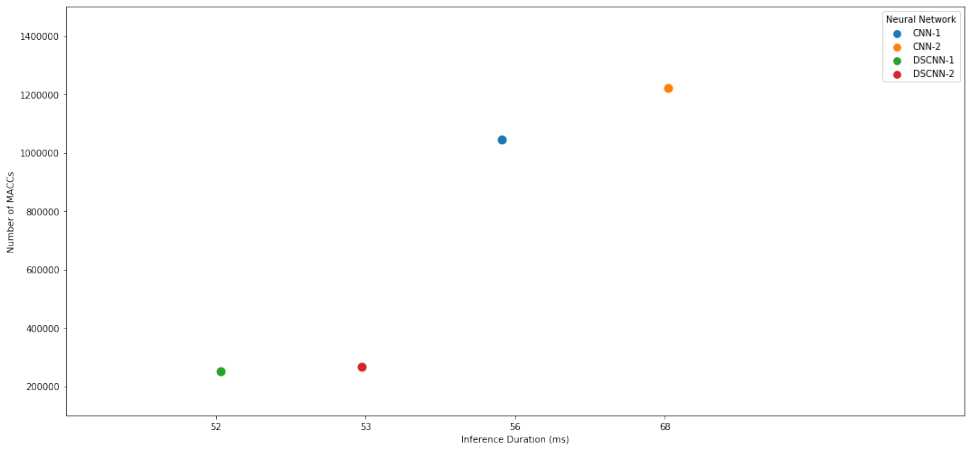


Figure 4.7 – Inference rate vs MACCs

Table 4.1 – Neural Network Mode Description.

Layer	(type)	Output shape	Param#
Input Layer	Input shape	(, 30, 32, 1)	0
conv2d	(Conv2D)	(, 30, 32, 20)	340
max_pooling2d (Pool)	(Max-Pool 2D)	(, 15, 16, 20)	0
conv2d_1	(Conv2D)	(, 15, 16, 8)	2568
max_pooling2d_1	(Max-Pool 2D)	(, 8, 8, 8)	0
dropout_1	(Dropout)	(, 8, 8, 8)	0
conv2d_2	(Conv2D)	(, 8, 8, 4)	516
max_pooling2d_2	(Max-Pool 2D)	(, 4, 4, 4)	0
dropout_2	(Dropout)	(, 4, 4, 4)	0
flatten_1	(Flatten)	(, 64)	0
batch_normalization	(BatchNorm)	(, 64)	256
dense	(Dense)	(,400)	26000
dense_1	(Dense)	(,400)	160,400
dense_2	(Dense)	(,300)	120,300
dropout_3	(Dropout)	(,300)	0
dense_3	(Dense)	(,4)	1,204
Trainable parameters			311,456
Non-trainable parameters			128
Total parameters			311,584

Table 4.2 – Convolutional neural networks model description

CNN Layers and Type		CNN-1		CNN-2	
Layer	(type)	Output shape	Param#	Output shape	Param#
Input Layer	Input shape	(, 30, 32, 1)	0	(, 30, 32, 1)	0
conv2d	(Conv2D)	(, 30, 32, 20)	340	(, 30, 32, 20)	340
max_pooling2d (Pool)	(Max-Pool 2D)	(, 15, 16, 20)	0	(, 15, 16, 20)	0
conv2d_1	(Conv2D)	(, 15, 16, 8)	2568	(, 15, 16, 10)	3210
max_pooling2d_1	(Max-Pool 2D)	(, 8, 8, 8)	0	(, 8, 8, 10)	0
dropout_1	(Dropout)	(, 8, 8, 8)	0	(, 8, 8, 10)	0
conv2d_2	(Conv2D)	(, 8, 8, 4)	516	(, 8, 8, 5)	805
max_pooling2d_2	(Max-Pool 2D)	(, 4, 4, 4)	0	(, 4, 4, 5)	0
dropout_2	(Dropout)	(, 4, 4, 4)	0	(, 4, 4, 5)	0
flatten_1	(Flatten)	(, 64)	0	(, 80)	0
batch_normalization	(BatchNorm)	(, 64)	256	(, 80)	320
dense	(Dense)	(,64)	4160	(, 80)	6480
dense_1	(Dense)	(,128)	8320	(,128)	10368
dense_2	(Dense)	(,256)	33024	(,256)	33024
dropout_3	(Dropout)	(,256)	0	(,256)	0
dense_3	(Dense)	(,4)	1028	(,4)	1028
Trainable parameters			50,084		55,415
Non-trainable parameters			128		160
Total parameters			50,212		55,575

Table 4.3 – Depth-wise separable convolutional neural networks model description

DCSNN Layers and Types		DSCNN-1		DSCNN-2	
Layer	(type)	Output shape	Param#	Output shape	Param#
Input Layer	Input shape	(, 30, 32, 1)	0	(, 30, 32, 1)	0
conv2d	(Separable_Conv2D)	(, 30, 32, 20)	56	(, 30, 32, 20)	56
max_pooling2d (Pool)	(Max-Pool 2D)	(, 15, 16, 20)	0	(, 15, 16, 20)	0
conv2d_1	(Separable_Conv2D)	(, 15, 16, 8)	488	(, 15, 16, 10)	530
max_pooling2d_1	(Max-Pool 2D)	(, 8, 8, 8)	0	(, 8, 8, 10)	0
dropout_1	(Dropout)	(, 8, 8, 8)	0	(, 8, 8, 10)	0
conv2d_2	(Separable_Conv2D)	(, 8, 8, 4)	164	(, 8, 8, 5)	215
max_pooling2d_2	(Max-Pool 2D)	(, 4, 4, 4)	0	(, 4, 4, 5)	0
dropout_2	(Dropout)	(, 4, 4, 4)	0	(, 4, 4, 5)	0
flatten_1	(Flatten)	(, 64)	0	(, 80)	0
batch_normalization	(BatchNorm)	(, 64)	256	(, 80)	320
dense	(Dense)	(,64)	4160	(, 80)	6480
dense_1	(Dense)	(,128)	8320	(,128)	10368
dense_2	(Dense)	(,256)	33024	(,256)	33024
dropout_3	(Dropout)	(,256)	0	(,256)	0
dense_3	(Dense)	(,4)	1028	(,4)	1028
Trainable parameters			47,368		51,861
Non-trainable parameters			128		160
Total parameters			47,496		52,021

Table 4.4 – CNN and DSCNN computational complexity

Neural Netwrok	Number of MACCs	model/c-model: macc	Change	RAM
CNN-1	1,044,336	1,240,736/1,044,336	-196,400(-15.8%)	12,676 B = 11,712 + 960 + 4
CNN-2	1,221,875	1,423,315/1,221,875	-201,440(-14.2%)	12,804 B = 11,840 + 960 + 4
DSCNN-1	249,997	446,397/249,997	446,397/249,997	8,208 B = 7,244 + 960 + 4
DSCNN-2	268,306	469,746/268,306	-201,440(-42.9%)	8,208 B = 7,244 + 960 + 4

Table 4.5 – CNN and DSCNN parameters, activations and weights

Neural Network	Number of Parameters	Activations	Weights
CNN-1	49,956 items (50.20 KiB)	11,712 B (11.44 KiB)	51,408 B (50.20 KiB)
CNN-2	55,255 items (55.43 KiB) 56764 bytes	11,840 B (11.56 KiB)	56,764 B (55.43 KiB)
DSCNN-1	47,269 items (47.66 KiB) 48808 bytes	7,244 B (7.07 KiB)	48,808 B (47.66 KiB)
DSCNN-2	51,732 items (52.08 KiB) 53336 bytes	7,244 B (7.07 KiB)	53,336 B (52.09 KiB)

Table 4.6 – Model Accuracy inference Time and MACCs

Neural Netwrok	accuracy test set	Infernce Duration	Number of MACCs
CNN-1	0.8304331729	56 ms	1,044,336
CNN-2	0.8467234358	68 ms	1,221,875
DSCNN-1	0.8659755646	52 ms	249,997
DSCNN-2	0.8271010737	53 ms	268,306

4.6 Conclusion

Environment sound classification on edge nodes is a complex task and requires attention on various aspects for the classification of environmental sounds such as constraints of resources, and power on board. The machine-learning algorithms are usually trained to operate on high computational power devices. The SensorTile is used to validate the concept of performing sound classification using an artificial intelligence-based model running on a limited computational power micro-controller device. In this chapter, a feasibility study is provided which evaluates different neural network models on the edge device and provides insight into the resources required and also the computational duration. This chapter presents the evaluation of five different neural network models, CNN-0, CNN-1, CNN-2, DSCNN-1, and DSCNN-2, for the classification of environmental sounds. The models were evaluated using five-fold cross-validation, and the results show that DSCNN-1 and CNN-2 are the best-performing models, with average accuracies of 80.72% and 80.08%, respectively. These models have a relatively narrow range of accuracies, indicating consistent performance across the five folds. On the other hand, DSCNN-2-zScaled and DSCNN-1-zScaled are the worst-performing models, with wider ranges of accuracies, indicating less consistent performance. The evaluation also includes the inference time and the number of multiply-accumulate operations (MACCs) required for each neural network, with DSCNN-1 achieving the highest accuracy with the shortest inference duration and lowest number of MACCs. These results can guide the selection of the most suitable model for environmental sound classification, depending on the trade-offs between accuracy, consistency, and calculation efficiency. We showed that the low-powered devices could be employed to perform environmental sound classification both on recorded audio and through audio acquisition in real-time. The low memory usage and real-time performance of the system enable us to address future developments in more sound categories or multi-inference model applications.

Part IV

Conclusion

General Conclusion

The objective of the thesis, presented in this manuscript, was to study the feasibility and to propose a prototype, of a low-complexity, low-power, small memory footprint, and low-cost system for environmental sound automatic recognition (or classification, ESC). A smart system embedded on an edge device with a low processing power microcontroller was realized and demonstrates the feasibility of such a system as a practical and cost-effective solution for monitoring environmental sounds in smart homes and other applications.

The first contribution of the thesis is the introduction of a novel approach to feature extraction based on the Instantaneous Amplitude (IA) and Frequency (IF) spectrogram, which is constructed using Empirical Mode Decomposition (EMD) coupled with the Teager-Kaiser Energy Operator (TKEO). This method provides estimated time-frequency characteristics of the signal compared to traditional approaches, such as FFT-based Log Mel spectrograms. The IA-IF spectrogram has several advantages over the FFT-based spectrogram, including improved temporal and spectral resolution and better handling of non-stationary signals. The study compares the IA-IF spectrogram with traditional FFT-based spectrograms using log Mel spectrograms and shows that the IA-IF approach provides promising results and has overcome the limitations of traditional approaches.

Secondly, a feature size reduction method is discussed that involves extracting statistical parameters from Mel Cepstral Coefficients (MFCCs) in the cepstral domain. These parameters are then optimized using the Sequential Feature Selection (SFS) method. The study provides a detailed analysis of how the SFS algorithm can be used to select a subset of the most relevant features from a large set of features. This method can help reduce the computational burden and improve the performance of environmental sound classification (ESC) systems. Furthermore, the study discusses the impact of feature size reduction on the overall accuracy and efficiency of ESC systems.

Finally, the thesis explores the use of machine learning models for the efficient classification of environmental sounds. Specifically, it delves into two types of models: convolutional neural networks (CNN) and depthwise separable convolutional neural networks (DSCNN), both of which are optimized in terms of the number of parameters to im-

prove efficiency. These models have proven effective in accurately classifying environmental sounds, and the thesis evaluates their performance using a low-processing power microcontroller. The study measures the processing time and complexity of the models, providing insights into their efficiency and scalability. Overall, the study provides valuable information for developing more efficient and accurate environmental sound recognition systems.

The new method EMD-TKEO of extracting IA and IF and generation of spectrograms provides a huge potential in analysis of non-stationary signals. This method could be extended to other fields such as biomedical signals, vibration signal analysis, seismic signal analysis, underwater acoustic signals, etc. The EMD-TKEO-based log MBEs showed good results for some datasets and in others for a few classes, it surpassed traditionally used features. In the case of feature selection, it has been shown that handpicking features and removing irrelevant ones can improve the system performance by a significant amount. Finally, the implementation of the ESC system on the microcontroller and analysis of the performance of different neural networks on board provided significant insight into the complexity vs performance problem. The analysis of operations with respect to time and available resources serves as the basis for future development in this domain. Overall, the proposed systems have demonstrated the potential of using ESC for the automatic recognition of environmental sounds in smart homes and other applications. The thesis provides a contribution toward the advancement of the field of automatic environmental sound classification and provides a foundation for future research in this area.

Prespectives

The thesis provides a foundation for future work in the field of environmental sound recognition.

— The proposed IA-IF spectrogram feature extraction method has shown promising results in environmental sound classification. However, there is always room for improvement, and further research can be conducted to enhance this approach's accuracy and efficiency. One potential area of improvement is the use of more advanced methods such as Multivariate Empirical Mode Decomposition (MEMD), which can overcome some of the limitations posed by EMD. Future studies can explore the performance of MEMD in constructing IA-IF spectrograms and compare them with the traditional EMD-based approach. Additionally, as the proposed system is de-

signed to operate on high-processing power computers, the development of EMD libraries for low-processing power devices, such as microcontrollers, can improve the applicability of the system. These libraries can enable the efficient implementation of the feature extraction process on microcontrollers with limited computational resources.

— The thesis has focused on the classification of four environmental sounds, but in reality, there are many more environmental sounds that need to be classified for effective monitoring in smart homes and other applications. Future research could focus on the expansion of the database of environmental sounds and the development of more advanced machine-learning models for ESC. Moreover, the proposed system can be extended to other applications beyond smart homes, such as automatic activity recognition and surveillance systems. Additionally, the system's low power consumption and small memory footprint make it suitable for deployment in resource-constrained environments, such as in remote or low-power locations. Overall, the proposed system has demonstrated the potential of using ESC for the automatic recognition of environmental sounds in smart homes and other applications. The thesis has contributed to the advancement of the field of automatic environmental sound classification and provides a foundation for future research in this area.

— Future work can explore the use of transfer learning techniques to further improve the performance of the models, as well as the evaluation of other types of deep neural network models and input features for environmental sound classification tasks. Also, in the future, systems can be designed for multi-label sound classification and sound classification in noisy environments. Multi-label sound classification is a more complex problem in the development of sound classification systems. Similarly, designing a sound classification for noisy environments requires careful attention.

Résumé

Inroduction générale

Problématique générale

Les sons environnementaux proviennent généralement de sources diverses et variées, telles que l'activité humaine, les objets et la nature. La classification automatique de ces sons *(ESC pour environnement sound classification)* suscite récemment un grand intérêt grâce à son grand potentiel d'application dans divers domaines, comme l'interaction home-machine, l'habitat intelligent, l'audition robotique, la reconnaissance automatique d'activités, les systèmes de surveillance automatique, etc. Cette classification, qui consiste à identifier automatiquement le son présent dans un enregistrement en lui attribuant une étiquette, est composée essentiellement de plusieurs étapes dont les plus essentielles sont l'extraction des caractéristiques les plus pertinentes pour différentier les sons et l'élaboration d'un modèle d'apprentissage automatique performant. La recherche des caractéristiques les plus pertinentes, qui relève de l'ingénierie des paramètres, est un aspect important de l'apprentissage automatique, puisque la qualité de celles-ci impacte significativement la capacité du modèle à faire des prédictions précises. Les techniques utilisées dans cette ingénierie incluent la définition de nouvelles caractéristiques, la modification de celles existantes et la sélection des caractéristiques les plus pertinentes. Dans le contexte de la classification de sons environnementaux, les caractéristiques les plus couramment utilisées sont les énergies dans des bandes fréquentielles réparties selon l'échelle de Mel (MBE for Mel Bands Energy) et les coefficients cepstraux (MFCC). L'extraction de ces caractéristiques est basée sur une analyse temps-fréquence par transformée de Fourier. Cette analyse est adaptée pour des signaux vérifiant l'hypothèse de stationnarité et d'ergodisme. Cependant, dans les sons environnementaux, ne sont pas des signaux stationnaires mais peuvent être considérés comme stationnaire à court-terme. Pour ce type de signaux la transformée de Fourier à court terme (STFT) est utilisée dans l'extraction des caractéristiques. Cette analyse à court-terme augmente la dimension de l'espace des caractéristiques et engendre ainsi une augmentation en termes de mémoire et de puissance de calcul pour

l'extraction des caractéristiques et pour l'entrainement du modèle. Pour rendre le modèle plus efficace, il est possible de réduire la dimensionnalité des caractéristiques afin de réduire le coût de calcul. La mise en œuvre de modèles d'apprentissage automatique pour l'ESC dans le monde réel pose plusieurs défis en termes de capacité du modèle, d'échelle et de ressources. Les modèles actuels sont conçus pour atteindre une précision et une généralisation élevées, mais ils ne sont pas optimaux en termes de complexité, d'empreinte mémoire, de consommation énergétique et de coût. Pour résoudre ce problème, la plupart des travaux de recherche sont orientés vers la réduction de la complexité des modèles tout en maintenant leur précision. Cependant, ils négligent souvent la complexité de l'étape d'extraction des caractéristiques, qui peut être plus complexe que le modèle formé lui-même. Ces défis sont plus accentués lorsque le système de classification de son environnementaux est conçu pour être implémenté sur des nœuds périphériques. En effet, avec la croissance de l'Internet des objets, l'accent est de plus en plus mis sur le traitement des données aux nœuds périphériques, plutôt que sur leur transmission à un concentrateur. Cela réduit la consommation d'énergie en évitant le transfert de données et permet le développement d'applications avec un équilibre entre précision et complexité, en tenant compte des ressources limitées des nœuds périphériques.

Objectifs et motivation de la thèse

L'objectif de cette thèse est d'étudier les méthodes d'extraction de caractéristiques utilisées pour la classification des sons environnementaux et d'étudier l'implémentation de ces méthodes de classification sur des nœuds de périphérie dans le contexte d'habitat intelligent. Les techniques modernes d'apprentissage automatique, telles que les réseaux de neurones artificiels, les réseaux de neurones profonds et les réseaux de neurones convolutifs, sont utilisées pour la détection et la classification de sons environnementaux. Dans cette thèse, nous avons abordé :

— L'impact de différentes formes de représentation temps-fréquence sur les performances des modèles d'apprentissage automatique pour l'ESC.

— L'impact de la réduction des caractéristiques d'entrée.

— L'étude de la faisabilité et l'implémentabilité de systèmes optimaux, pour la classification de sons environnementaux, sur des nœuds de périphérie à base de microcontrôleurs à faible puissance de traitement.

Contribution

— Une nouvelle méthode d'extraction de caractéristiques pour construire une image temps-fréquence est proposée. La décomposition en mode empirique (EMD) est utilisée pour analyser des signaux issus de systèmes non linéaires et non stationnaires. L'EMD est hautement adaptative et basée sur l'extraction directe d'énergie avec des échelles de temps locales. L'opérateur d'énergie de Teager – Kaiser (TKEO) est utilisé pour extraire la fréquence et l'amplitude instantanées.

— La sélection séquentielle des caractéristiques est adoptée et adaptée. Le système proposé est composé de LSTM *(pour « long short terme memory » en anglais)* et de paramètres extraits, par ajout d'un niveau d'abstraction, à partir des coefficients cepstraux, et il est comparé à un autre système basé sur un réseau CNN formé avec des MBE.

— Le système de classification de son environnementaux développé est implémenté sur un nœud à base d'un microcontrôleur à faible puissance de calcul, à faible empreinte mémoire et à basse consommation. Le système a été validé et testé en temps réel, en prouvant ainsi le concept et la faisabilité d'un ESC sur un nœuds de périphérie en temps réel.

Conception d'un système de calssaification de sons environnementaux

Décomposition en modes empiriques - Opérateur d'énergie Teager-Kaiser et représentation temps-fréquence

La classification d'images repose sur l'utilisation de l'image elle-même comme caractéristique, tandis que la classification de la parole et des sons nécessite l'utilisation d'un spectrogramme, qui est une représentation en image des informations temporelles, fréquentielles et énergétiques du son. Le spectrogramme le plus couramment utilisé est le Mel-spectrogramme, qui est créé en appliquant des bancs de filtres en échelle de Mel à un spectrogramme généré par la Fourier rapide à court terme (STFT). Dans cette étude, nous introduisons l'utilisation de la décomposition en modes empiriques (EMD) avec l'opérateur d'énergie de Teager-Kaiser (TKEO) pour estimer la fréquence et l'amplitude instantanées, qui sont utilisées pour construire un nouveau type de spectrogramme pour

la classification des sons environnementaux. Cette méthode, adaptative et efficace, décompose le signal en un nombre fini d'unités oscillatoires appelées fonctions de mode intrinsèque (IMF). Ces modes, extraits en fonction des propriétés locales du signal supposé multi-composantes et modulé en amplitude et en fréquence, sont ses composantes AM-FM symétriques et avec une moyenne nulle. En utilisant l'opérateur d'énergie de Teager-Kaiser (TKEO), nous pouvons extraire la fréquence et l'amplitude instantanées des IMFs, permettant ainsi de localiser tout événement sur une échelle de temps et de fréquence. A partir de ces amplitudes et fréquences instantanées, nous construisons le spectrogramme, en utilisant des bancs de filtres en échelle de Mel. En supprimant la tendance du signal à l'aide de la décomposition EMD, nous avons également introduit une autre catégorie de caractéristiques appelées SMBE. Nous avons comparé les performances de ces caractéristiques, extraites par les méthodes proposées, avec celles extraites à partir des énergies en bandes fréquentielles réparties selon l'échelle de Mel et calculées par la transformée de Fourier à court terme.

Optimisation du système par sélection de caractéristiques

La réduction du nombre de caractéristiques dans un système de classification permet à la fois d'améliorer la précision, en supprimant la redondance, et de diminuer la complexité du modèle. Cette réduction peut être faite par sélection des meilleures caractéristiques ou par projection de celles-ci dans un espace de dimension réduite. Dans cette étude, nous utilisons la méthode de sélection séquentielle de caractéristiques. La sélection séquentielle de caractéristiques consiste à ajouter ou à éliminer de manière itérative des caractéristiques jusqu'à ce qu'un sous-ensemble optimal de caractéristiques soit construit. Elle utilise un modèle d'apprentissage automatique pour évaluer les performances de chaque sous-ensemble de caractéristiques. L'algorithme commence avec un ensemble vide ou complet de caractéristiques en fonction du type de méthode et ajoute ou élimine une caractéristique à la fois, tout en évaluant la performance du modèle après chaque ajout ou élimination. La caractéristique ayant le score le plus élevé est sélectionnée et ajoutée au sous-ensemble, et le processus est répété jusqu'à ce qu'un critère d'arrêt soit atteint. Dans notre étude, différentes méthodes de sélection sont adaptées et comparées.

Évaluation des méthodes proposées

Bases de données

Dans cette étude nous avons utilisé plusieurs bases de données pour évaluer les performances de notre système. Trois de ces bases sont issues d'autres travaux de recherche relatives à la problématique de la classification de sons environnementaux : *acoustic scene classification, low complexity acoustic scene classification et urbansound8k*. Ces trois bases ont été utilisé pour valider la première version de notre système et comparer ses performances avec les autres travaux de recherche.

Pour avoir une base de données dédiées au contexte d'habitat intelligent , nous avons créé une quatrième base de données. Cette base de données contient des enregistrements pour quatre scènes : pluie, vent, passage de voiture et bruit de pas humain. Le système optimisé, ainsi que le système implémenté sur un microcontrôleur, ont été évalués par rapport à cette base de données.

Résultats

Pour l'évaluation des méthodes d'extraction de caractéristiques proposées, nous les avons comparées aux caractéristiques extraites par des méthodes traditionnelles basées sur la transformée de Fourier. Deux systèmes, basées sur des réseaux de neurones convolutifs ont été utilisés dans cette étude pour évaluer les performances des caractéristiques pour quatre bases de données. Les résultats montrent que les méthodes basées sur la décomposition EMD (EMD-MBE) ont des performances inférieures à celles basées sur la transformée de Fourier (FFT-MBE). Les caractéristiques S-MBE sont les moins performantes des trois catégories de caractéristiques évaluées pour chaque base de données. Par contre la combinaison de ces trois catégories de caractéristiques permet d'obtenir une amélioration en termes de précision de classification. Cependant nous constatons que les EMD-MBEs ont de meilleurs résultats pour certaines classes que les STFT-MBEs. L'analyse de la faible performance de la méthode proposée révèle que dans l'estimation de la représentation temps-fréquence, la résolution est limitée par le nombre d'IMFs et la taille de la fenêtre. De plus, lors du processus de décomposition du signal en IMFs, la méthode EMD souffre du problème de mélange de modes, ce qui dégrade la qualité des caractéristiques extraites.

Pour l'évaluation des méthodes basées sur la sélection de caractéristiques, nous avons adopté et adapté un système de sélection séquentielle basé sur un réseau de neurones

récurent LSTM *(pour « long short terme memory » en anglais)*. Les résultats affirment que la combinaison, des méthodes d'extraction de caractéristiques traditionnelles avec la sélection des celle-ci par la méthode SFFS *(pour «sequential floating forward selection» en anglais)* et un réseau de neurones récurrent RNN pour la classification, est efficace pour la reconnaissance automatique de sons environnementaux. Cette combinaison permet d'atteindre des un taux de calssification moyenne de (81.35% et 81.15%), avec un nombre faible de caractéristiques sélectionnées (k=35 et k=40), et un nombre relativement faible de paramètres entraînés (844,208 et 845,248).

Implémentation sur une carte à base d'un microcontrôleur à faible puissance de calcul

Les approches traditionnelles d'ESC nécessitent le traitement des signaux sur des ordinateurs puissants ou sur des serveurs cloud. Cependant de nombreuses applications d'ESC, comme dans le contexte d'habitat intelligent, nécessitent des systèmes à basse consommation énergétique et a faible empreinte mémoire. Ces contraintes justifient l'intérêt accru pour la réalisation des systèmes d'ESC sur des microcontrôleurs. Avec l'avènement des microcontrôleurs à faible consommation d'énergie et de l'Internet des objets (IoT), il existe un intérêt croissant pour la mise en œuvre d'algorithmes d'ESC sur ces nœuds de périphérie. Cette approche réduit le besoin de transfert intensif de données et améliore la capacité des nœuds à traiter les données localement.

Dans cette partie nous avons étudié la faisabilité de l'utilisation de modèles d'apprentissage automatique, en particulier les réseaux de neurones convolutifs (CNN) et les réseaux de neurones convolutifs à séparation en profondeur (DS-CNN), pour la classification de sons environnementaux sur des microcontrôleurs. L'accent a été mise sur le développement d'une application à faible temps de traitement, en assurant un compromis entre complexité et précision. La carte SensorTile basée sur STM32L4 et la bibliothèque CMSIS-NN sont utilisées pour valider le concept et évaluer les performances du système d'ESC proposé. Nous avons évalué cinq modèles de réseaux de neurones différents, CNN-0, CNN-1, CNN-2, DSCNN-1 et DSCNN-2, pour la classification de sons environnementaux. Les modèles ont été évalués en utilisant une validation croisée à cinq volets, et les résultats montrent que DSCNN-1 et CNN-2 sont les modèles les plus performants, avec des précisions moyennes de 80,72% et 80,08%, respectivement. Ces modèles ont une plage de précisions relativement étroite, indiquant une performance cohérente sur les cinq volets.

Le temps d'inférence et le nombre d'opérations de multiplication-accumulation (MACC) requis pour chaque réseau de neurones ont été aussi évalué. Le réseau DSCNN-1 permet d'atteindre la précision la plus élevée avec la durée d'inférence la plus courte et le nombre le plus faible de MACC. Ces résultats peuvent guider la sélection du modèle le plus approprié pour la classification de son environnementaux, en fonction des compromis entre précision et complexité.

BIBLIOGRAPHY

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, « Imagenet: a large-scale hierarchical image database », *in 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[2] J. W. Cooley and J. W. Tukey, « An algorithm for the machine calculation of complex fourier series », *Mathematics of computation*, vol. 19, *90*, pp. 297–301, 1965.

[3] N. Ono, N. Harada, Y. Kawaguchi, *et al.*, Eds., *Proceedings of the Fifth Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2020)*, English. 2020. DOI: 10.5281/zenodo.4061781.

[4] A. Mesaros, T. Heittola, and T. Virtanen, « Acoustic scene classification: an overview of dcase 2017 challenge entries », *in 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, IEEE, 2018, pp. 411–415.

[5] A. Mesaros, T. Heittola, and T. Virtanen, « Assessment of human and machine performance in acoustic scene classification: dcase 2016 case study », *in 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 319–323. DOI: 10.1109/WASPAA.2017.8170047.

[6] A. Mesaros, T. Heittola, E. Benetos, *et al.*, « Detection and classification of acoustic scenes and events: outcome of the dcase 2016 challenge », *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, *2*, pp. 379–393, 2017.

[7] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, « Detection and classification of acoustic scenes and events », *IEEE Transactions on Multimedia*, vol. 17, *10*, pp. 1733–1746, 2015.

[8] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, « Sound event detection: a tutorial », *IEEE Signal Processing Magazine*, vol. 38, *5*, pp. 67–83, 2021.

[9] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, « Context-dependent sound event detection », *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, *1*, pp. 1–13, 2013.

[10] J. Abeßer, « A review of deep learning based methods for acoustic scene classification », *Applied Sciences*, vol. 10, *6*, p. 2020, 2020.

[11] A. Mesaros, T. Heittola, and T. Virtanen, « Tut database for acoustic scene classification and sound event detection », *in 2016 24th European Signal Processing Conference (EUSIPCO)*, IEEE, 2016, pp. 1128–1132.

[12] S. Kahl, T. Wilhelm-Stein, H. Hussein, *et al.*, « Large-scale bird sound classification using convolutional neural networks. », *CLEF (working notes)*, vol. 1866, 2017.

[13] S. Adavanne, K. Drossos, E. Çakir, and T. Virtanen, « Stacked convolutional and recurrent neural networks for bird audio detection », *in 2017 25th European signal processing conference (EUSIPCO)*, IEEE, 2017, pp. 1729–1733.

[14] N. Turpault, S. Wisdom, H. Erdogan, *et al.*, « Improving sound event detection in domestic environments using sound separation », *arXiv preprint arXiv:2007.03932*, 2020.

[15] Y. Lavner, R. Cohen, D. Ruinskiy, and H. IJzerman, « Baby cry detection in domestic environment using deep learning », *in 2016 IEEE international conference on the science of electrical engineering (ICSEE)*, IEEE, 2016, pp. 1–5.

[16] Y. R. Pandeya, D. Kim, and J. Lee, « Domestic cat sound classification using learned features from deep neural nets », *Applied Sciences*, vol. 8, *10*, p. 1949, 2018.

[17] J. Salamon and J. P. Bello, « Unsupervised feature learning for urban sound classification », *in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 171–175.

[18] K. J. Piczak, « Environmental sound classification with convolutional neural networks », *in 2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*, IEEE, 2015, pp. 1–6.

[19] I. Choi, K. Kwon, S. H. Bae, and N. S. Kim, « Dnn-based sound event detection with exemplar-based approach for noise reduction », *in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016, pp. 16–19.

[20] L. Perez and J. Wang, « The effectiveness of data augmentation in image classification using deep learning », *arXiv preprint arXiv:1712.04621*, 2017.

[21] L. Deng, G. Hinton, and B. Kingsbury, « New types of deep neural network learning for speech recognition and related applications: an overview », *in 2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 8599–8603.

[22] A. Mesaros, T. Heittola, A. Diment, *et al.*, « Dcase 2017 challenge setup: tasks, datasets and baseline system », *in DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[23] A. Mesaros, A. Diment, B. Elizalde, *et al.*, « Sound event detection in the dcase 2017 challenge », *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, *6*, pp. 992–1006, 2019.

[24] C. S. Bojer and J. P. Meldgaard, « Kaggle forecasting competitions: an overlooked learning opportunity », *International Journal of Forecasting*, 2020, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2020.07.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207020301114.

[25] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin, « Bird detection in audio: a survey and a challenge », *in 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2016, pp. 1–6.

[26] H. Lim, J. Park, and Y. Han, « Rare sound event detection using 1d convolutional recurrent neural networks », *in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017, pp. 80–84.

[27] S. M. Adnan, A. Irtaza, S. Aziz, M. O. Ullah, A. Javed, and M. T. Mahmood, « Fall detection through acoustic local ternary patterns », *Applied Acoustics*, vol. 140, pp. 296–300, 2018.

[28] N. Ono, N. Harada, Y. Kawaguchi, *et al.*, *Proceedings of the Fifth Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2020)*. Tokyo, Japan, Nov. 2020.

[29] M. Mandel, J. Salamon, and D. P. W. Ellis, *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. NY, USA: New York University, Oct. 2019.

[30] *DCASE 2022 challenge results published*, https://dcase.community/articles/dcase2022-challenge-results-published, Accessed: 2023-02-12.

[31] J. B. Allen, « Cochlear modeling—1980 », *in Mathematical Modeling of the Hearing Process*, Springer, 1981, pp. 1–8.

[32] S. S. Stevens, J. Volkmann, and E. B. Newman, « A scale for the measurement of the psychological magnitude pitch », *The journal of the acoustical society of america*, vol. 8, *3*, pp. 185–190, 1937.

[33] J. L. Flanagan, *Speech analysis synthesis and perception*. Springer Science & Business Media, 2013, vol. 3.

[34] T. P. Mathieu Lagrange Annamaria Mesaros, G. Richard, R. Serizel, and D. Stowell, *Proceedings of the 7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2022)*. Nancy, France: Tampere University, Nov. 2022, ISBN: 978-952-03-2677-7.

[35] D. Community, *Detection and classification of acoustic scenes and events (dcase) challenge 2021*, [Online; accessed 13-February-2023], 2021. [Online]. Available: `https://dcase.community/challenge2021/`.

[36] R. D. Patterson, K. Robinson, J. Holdsworth, D. McKeown, C. Zhang, and M. Allerhand, « Complex sounds and auditory images », *in Auditory physiology and perception*, Elsevier, 1992, pp. 429–446.

[37] M. Slaney *et al.*, « An efficient implementation of the patterson-holdsworth auditory filter bank », *Apple Computer, Perception Group, Tech. Rep*, vol. 35, *8*, 1993.

[38] A. Plinge, R. Grzeszick, and G. A. Fink, « A bag-of-features approach to acoustic event detection », *in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 3704–3708.

[39] W. Yang and S. Krishnan, « Sound event detection in real-life audio using joint spectral and temporal features », *Signal, Image and Video Processing*, vol. 12, *7*, pp. 1345–1352, 2018.

[40] H. Phan, M. Krawczyk-Becker, T. Gerkmann, and A. Mertins, « Dnn and cnn with weighted and multi-task loss functions for audio event detection », *arXiv preprint arXiv:1708.03211*, 2017.

[41] J. Dennis, H. D. Tran, and H. Li, « Spectrogram image feature for sound event classification in mismatched conditions », *IEEE Signal Processing Letters*, vol. 18, *2*, pp. 130–133, 2011. DOI: `10.1109/LSP.2010.2100380`.

[42] J. Dennis, H. D. Tran, and E. S. Chng, « Image feature representation of the subband power distribution for robust sound event classification », *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, *2*, pp. 367–377, 2013. DOI: 10.1109/TASL.2012.2226160.

[43] H. Lim, M. J. Kim, and H. Kim, « Robust sound event classification using lbp-hog based bag-of-audio-words feature representation », *in Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[44] N. Ricker, « The form and nature of seismic waves and the structure of seismograms », *Geophysics*, vol. 5, *4*, pp. 348–366, 1940.

[45] K. Wirsing, « Time frequency analysis of wavelet and fourier transform », *in Wavelet Theory*, IntechOpen London, UK, 2020.

[46] T. M. Mitchell and T. M. Mitchell, *Machine learning*. McGraw-hill New York, 1997, vol. 1.

[47] A. Karpathy, *Neural networks part 1: setting up the architecture*, Notes for CS231n convolutional neural networks for visual recognition. Stanford University, 2015.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[49] D. Datta, P. E. David, D. Mittal, and A. Jain, « Neural machine translation using recurrent neural network », *International Journal of Engineering and Advanced Technology*, vol. 9, *4*, pp. 1395–1400, 2020.

[50] H. Sak, A. Senior, and F. Beaufays, « Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition », *arXiv preprint arXiv:1402.1128*, 2014.

[51] T. Makino, H. Liao, Y. Assael, *et al.*, « Recurrent neural network transducer for audio-visual speech recognition », *in 2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, IEEE, 2019, pp. 905–912.

[52] V. Chernykh and P. Prikhodko, « Emotion recognition from speech with recurrent neural networks », *arXiv preprint arXiv:1701.08071*, 2017.

[53] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, « Attention based convolutional recurrent neural network for environmental sound classification », *Neurocomputing*, vol. 453, pp. 896–903, 2021.

[54] G. Parascandolo, H. Huttunen, and T. Virtanen, « Recurrent neural networks for polyphonic sound event detection in real life recordings », *in 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2016, pp. 6440–6444.

[55] L. Kerkeni, Y. Serrestou, K. Raoof, M. Mbarki, M. A. Mahjoub, and C. Cleder, « Automatic speech emotion recognition using an optimal combination of features based on emd-tkeo », *Speech Communication*, vol. 114, pp. 22–35, 2019.

[56] L. Kerkeni, Y. Serrestou, K. Raoof, M. Mbarki, M. A. Mahjoub, and C. Cleder, « Automatic speech emotion recognition using an optimal combination of features based on emd-tkeo », *Speech Communication*, vol. 114, pp. 22–35, 2019.

[57] Colah, *Understanding lstms*, Aug. 2015. [Online]. Available: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[58] M. Schuster and K. K. Paliwal, « Bidirectional recurrent neural networks », *IEEE transactions on Signal Processing*, vol. 45, *11*, pp. 2673–2681, 1997.

[59] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, 2001.

[60] F. Chollet, « Xception: deep learning with depthwise separable convolutions », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[61] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, « Learning representations by back-propagating errors », *nature*, vol. 323, *6088*, pp. 533–536, 1986.

[62] Theano Development Team, « Theano: A Python framework for fast computation of mathematical expressions », *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: `http://arxiv.org/abs/1605.02688`.

[63] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[64] A. Paszke, S. Gross, F. Massa, *et al.*, « Pytorch: an imperative style, high-performance deep learning library », *in Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[65] D. P. Kingma and J. Ba, « Adam: a method for stochastic optimization », *arXiv preprint arXiv:1412.6980*, 2014.

[66] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, « Dropout: a simple way to prevent neural networks from overfitting », *The journal of machine learning research*, vol. 15, *1*, pp. 1929–1958, 2014.

[67] S. Ioffe and C. Szegedy, « Batch normalization: accelerating deep network training by reducing internal covariate shift », *in International conference on machine learning*, PMLR, 2015, pp. 448–456.

[68] A. Mesaros, T. Heittola, and T. Virtanen, « Metrics for polyphonic sound event detection », *Applied Sciences*, vol. 6, *6*, p. 162, 2016.

[69] K. Sechidis, G. Tsoumakas, and I. Vlahavas, « On the stratification of multi-label data », *in Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2011, pp. 145–158.

[70] *scikit-learn: cross-validation*, `https://scikit-learn.org/stable/modules/cross_validation.html`, Accessed March 10, 2023, n.d.

[71] A. Mesaros, T. Heittola, and T. Virtanen, « A multi-device dataset for urban acoustic scene classification », *in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Nov. 2018, pp. 9–13. [Online]. Available: `https://arxiv.org/abs/1807.09840`.

[72] S. Kumari, D. Roy, M. Cartwright, J. P. Bello, and A. Arora, « Edgelˆ 3: compressing lˆ 3-net for mote scale urban noise monitoring », *in 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2019, pp. 877–884.

[73] F. Font, G. Roma, and X. Serra, « Freesound technical demo », *in Proceedings of the 21st ACM International Conference on Multimedia*, ser. MM '13, Barcelona, Spain: Association for Computing Machinery, 2013, pp. 411–412, ISBN: 9781450324045. DOI: `10.1145/2502081.2502245`. [Online]. Available: `https://doi.org/10.1145/2502081.2502245`.

[74] A. Ahmed, Y. Serrestou, K. Raoof, and J.-F. Diouris, « Sound event classification using neural networks and feature selection based methods », *in 2021 IEEE International Conference on Electro Information Technology (EIT)*, IEEE, 2021, pp. 1–6.

[75] M. D. Plumbley, C. Kroos, J. P. Bello, G. Richard, D. P. Ellis, A. Mesaros, *et al.*, *Proceedings of the detection and classification of acoustic scenes and events 2018 workshop (dcase2018)*, 2018.

[76] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, « Convolutional recurrent neural networks for polyphonic sound event detection », *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, *6*, pp. 1291–1303, 2017. DOI: 10.1109/TASLP.2017.2690575.

[77] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, « Polyphonic sound event detection using multi label deep neural networks », *in 2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–7. DOI: 10.1109/IJCNN.2015.7280624.

[78] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, « Squeezenet: alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size », *arXiv preprint arXiv:1602.07360*, 2016.

[79] S. Hershey, S. Chaudhuri, D. P. Ellis, *et al.*, « Cnn architectures for large-scale audio classification », *in 2017 ieee international conference on acoustics, speech and signal processing (icassp)*, IEEE, 2017, pp. 131–135.

[80] P. Zinemanas, P. Cancela, and M. Rocamora, « End-to-end convolutional neural networks for sound event detection in urban environments », *in 2019 24th Conference of Open Innovations Association (FRUCT)*, IEEE, 2019, pp. 533–539.

[81] S. Adavanne, G. Parascandolo, P. Pertilä, T. Heittola, and T. Virtanen, « Sound event detection in multichannel audio using spatial and harmonic features », *arXiv preprint arXiv:1706.02293*, 2017.

[82] E. C. Titchmarsh *et al.*, *Introduction to the theory of Fourier integrals*. Clarendon Press Oxford, 1948, vol. 2.

[83] N. E. Huang, Z. Shen, S. R. Long, *et al.*, « The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis », *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, vol. 454, *1971*, pp. 903–995, 1998.

[84] P. Kumar and E. Foufoula-Georgiou, « Wavelet analysis for geophysical applications », *Reviews of geophysics*, vol. 35, *4*, pp. 385–412, 1997.

[85] J. Morlet, « Sampling theory and wave propagation », *in Issues in acoustic Signal—image processing and recognition*, Springer, 1983, pp. 233–261.

[86] V. Djordjević, V. Stojanović, D. Pršić, L. Dubonjić, and M. M. Morato, « Observer-based fault estimation in steer-by-wire vehicle », *Engineering Today*, vol. 1, *01*, pp. 7–17, 2022.

[87] Z. Xu, X. Li, and V. Stojanovic, « Exponential stability of nonlinear state-dependent delayed impulsive systems with applications », *Nonlinear Analysis: Hybrid Systems*, vol. 42, p. 101 088, 2021.

[88] A.-O. Boudraa and F. Salzenstein, « Teager–kaiser energy methods for signal and image analysis: a review », *Digital Signal Processing*, vol. 78, pp. 338–375, 2018.

[89] K. Khaldi, A.-O. Boudraa, and A. Komaty, « Speech enhancement using empirical mode decomposition and the teager–kaiser energy operator », *The Journal of the Acoustical Society of America*, vol. 135, *1*, pp. 451–459, 2014.

[90] P. T. Krishnan, A. N. Joseph Raj, and V. Rajangam, « Emotion classification from speech signal based on empirical mode decomposition and non-linear features », *Complex & Intelligent Systems*, vol. 7, *4*, pp. 1919–1934, 2021.

[91] C. De La Cruz and B. Santhanam, « A joint emd and teager-kaiser energy approach towards normal and nasal speech analysis », *in 2016 50th Asilomar Conference on Signals, Systems and Computers*, IEEE, 2016, pp. 429–433.

[92] S. Jayalakshmy and G. F. Sudha, « Gtcc-based bilstm deep-learning framework for respiratory sound classification using empirical mode decomposition », *Neural Computing and Applications*, vol. 33, *24*, pp. 17 029–17 040, 2021.

[93] P. Maragos, J. Kaiser, and T. Quatieri, « Energy separation in signal modulations with application to speech analysis », *IEEE Transactions on Signal Processing*, vol. 41, *10*, pp. 3024–3051, 1993. DOI: 10.1109/78.277799.

[94] A. Potamianos and P. Maragos, « A comparison of the energy operator and the hilbert transform approach to signal and speech demodulation », *Signal processing*, vol. 37, *1*, pp. 95–120, 1994.

[95] R. Sharma, L. Vignolo, G. Schlotthauer, M. A. Colominas, H. L. Rufiner, and S. Prasanna, « Empirical mode decomposition for adaptive am-fm analysis of speech: a review », *Speech Communication*, vol. 88, pp. 39–64, 2017.

[96] A. Bin Queyam, S. Kumar Pahuja, and D. Singh, « Quantification of feto-maternal heart rate from abdominal ecg signal using empirical mode decomposition for heart rate variability analysis », *Technologies*, vol. 5, *4*, p. 68, 2017.

[97] V. Sethu, E. Ambikairajah, and J. Epps, « Empirical mode decomposition based weighted frequency feature for speech-based emotion classification », *in 2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2008, pp. 5017–5020.

[98] J. Kaiser, « On a simple algorithm to calculate the 'energy' of a signal », *in International Conference on Acoustics, Speech, and Signal Processing*, 1990, 381–384 vol.1. DOI: 10.1109/ICASSP.1990.115702.

[99] P. Maragos, J. F. Kaiser, and T. F. Quatieri, « On amplitude and frequency demodulation using energy operators », *IEEE Transactions on signal processing*, vol. 41, *4*, pp. 1532–1550, 1993.

[100] J. F. Kaiser, « Some useful properties of teager's energy operators », *in 1993 IEEE international conference on acoustics, speech, and signal processing*, IEEE, vol. 3, 1993, pp. 149–152.

[101] A. Bouchikhi, « Am-fm signal analysis by teager huang transform: application to underwater acoustics », Ph.D. dissertation, Université Rennes 1, 2010.

[102] P. Maragos, J. F. Kaiser, and T. F. Quatieri, « On separating amplitude from frequency modulations using energy operators », *in Proc. IEEE Proc. ICASSP*, vol. 92, 1992.

[103] X. Li, X. Li, X. Zheng, and D. Zhang, « Emd-teo based speech emotion recognition », *in Life System Modeling and Intelligent Computing*, Springer, 2010, pp. 180–189.

[104] Y. Sakashita and M. Aono, « Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions », *Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge*, 2018.

[105] M. Dorfer, B. Lehner, H. Eghbal-zadeh, H. Christop, P. Fabian, and W. Gerhard, « Acoustic scene classification with fully convolutional neural networks and i-vectors », *Proceedings of the Detection and Classification of Acoustic Scenes and Events*, 2018.

[106] E. Tsalera, A. Papadakis, and M. Samarakou, « Comparison of pre-trained cnns for audio classification using transfer learning », *Journal of Sensor and Actuator Networks*, vol. 10, *4*, p. 72, 2021.

[107] J. Guo, C. Li, Z. Sun, J. Li, and P. Wang, « A deep attention model for environmental sound classification from multi-feature data », *Applied Sciences*, vol. 12, *12*, p. 5988, 2022.

[108] J. Duchi, E. Hazan, and Y. Singer, « Adaptive subgradient methods for online learning and stochastic optimization. », *Journal of machine learning research*, vol. 12, *7*, 2011.

[109] J. Zheng, J. Cheng, and Y. Yang, « Partly ensemble empirical mode decomposition: an improved noise-assisted method for eliminating mode mixing », *Signal Processing*, vol. 96, pp. 362–374, 2014.

[110] G. Xu, Z. Yang, and S. Wang, « Study on mode mixing problem of empirical mode decomposition », *in Joint International Information Technology, Mechanical and Electronic Engineering Conference*, vol. 1, 2016, pp. 389–394.

[111] Y. Gao, G. Ge, Z. Sheng, and E. Sang, « Analysis and solution to the mode mixing phenomenon in emd », *in 2008 Congress on Image and Signal Processing*, IEEE, vol. 5, 2008, pp. 223–227.

[112] Z. Wu and N. E. Huang, « Ensemble empirical mode decomposition: a noise-assisted data analysis method », *Advances in adaptive data analysis*, vol. 1, *01*, pp. 1–41, 2009.

[113] W.-C. Shen, Y.-H. Chen, and A.-Y. A. Wu, « Low-complexity sinusoidal-assisted emd (saemd) algorithms for solving mode-mixing problems in hht », *Digital Signal Processing*, vol. 24, pp. 170–186, 2014.

[114] B. Tang, S. Dong, and T. Song, « Method for eliminating mode mixing of empirical mode decomposition based on the revised blind source separation », *Signal Processing*, vol. 92, *1*, pp. 248–258, 2012.

[115] F. Font, G. Roma, and X. Serra, « Freesound technical demo », *in ACM International Conference on Multimedia (MM'13)*, ACM, Barcelona, Spain: ACM, 21/10/2013 2013, pp. 411–412, ISBN: 978-1-4503-2404-5. DOI: 10.1145/2502081.2502245.

[116] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, *et al.*, « Audio set: an ontology and human-labeled dataset for audio events », *in Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[117] B. Yuce, M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase, « Honey bees inspired optimization method: the bees algorithm », *Insects*, vol. 4, *4*, pp. 646–662, 2013.

[118] H. Peng, C. Ying, S. Tan, B. Hu, and Z. Sun, « An improved feature selection algorithm based on ant colony optimization », *Ieee Access*, vol. 6, pp. 69 203–69 209, 2018.

[119] P. Somol, J. Novovicová, and P. Pudil, « Efficient feature subset selection and subset size optimization », *Pattern recognition recent advances*, vol. 56, 2010.

[120] S. Balakrishnama and A. Ganapathiraju, « Linear discriminant analysis-a brief tutorial », *Institute for Signal and information Processing*, vol. 18, *1998*, pp. 1–8, 1998.

[121] D. Lee and H. S. Seung, « Algorithms for non-negative matrix factorization », *Advances in neural information processing systems*, vol. 13, 2000.

[122] G. Chandrashekar and F. Sahin, « A survey on feature selection methods », *Computers & Electrical Engineering*, vol. 40, *1*, pp. 16–28, 2014.

[123] S. Nakariyakul and D. P. Casasent, « An improvement on floating search algorithms for feature subset selection », *Pattern Recognition*, vol. 42, *9*, pp. 1932–1940, 2009.

[124] K. De and V. Masilamani, « No-reference image quality measure for images with multiple distortions using random forests for multi method fusion », *Image Analysis and Stereology*, vol. 37, May 2018. DOI: 10.5566/ias.1534.

[125] P. Pudil, J. Novovičová, and J. Kittler, « Floating search methods in feature selection », *Pattern recognition letters*, vol. 15, *11*, pp. 1119–1125, 1994.

[126] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, « Convolutional recurrent neural networks for polyphonic sound event detection », *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, *6*, pp. 1291–1303, 2017.

[127] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, « Generative adversarial networks », *arXiv preprint arXiv:1406.2661*, 2014.

[128] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, *et al.*, « Audio set: an ontology and human-labeled dataset for audio events », *in Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[129] A. Mesaros, T. Heittola, and T. Virtanen, « Acoustic scene classification in dcase 2019 challenge: closed and open set classification and data mismatch setups », 2019.

[130] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, « Fsd50k: an open dataset of human-labeled sound events », *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 829–852, 2021.

[131] J. Salamon, C. Jacoby, and J. P. Bello, « A dataset and taxonomy for urban sound research », *in Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044.

[132] A. Mesaros, T. Heittola, and T. Virtanen, « A multi-device dataset for urban acoustic scene classification », *arXiv preprint arXiv:1807.09840*, 2018.

[133] STMicroelectronics, *STEVAL-STLKT01V1*, `https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html`, accessed 2023.

[134] F. Font, G. Roma, and X. Serra, « Freesound technical demo », *in Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 411–412.

[135] F. Naccari, I. Guarneri, S. Curti, and A. A. Savi, « Embedded acoustic scene classification for low power microcontroller devices. », *in DCASE*, 2020, pp. 105–109.

[136] T. Heittola, A. Mesaros, and T. Virtanen, « Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions », *arXiv preprint arXiv:2005.14623*, 2020.

[137] R. David, J. Duke, A. Jain, *et al.*, « Tensorflow lite micro: embedded machine learning for tinyml systems », *Proceedings of Machine Learning and Systems*, vol. 3, pp. 800–811, 2021.

[138] J. Allen, « Short term spectral analysis, synthesis, and modification by discrete fourier transform », *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, *3*, pp. 235–238, 1977.

[139] E. Sejdić, I. Djurović, and J. Jiang, « Time–frequency feature representation using energy concentration: an overview of recent advances », *Digital signal processing*, vol. 19, *1*, pp. 153–183, 2009.

[140] D. Griffin and J. Lim, « Signal estimation from modified short-time fourier transform », *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, *2*, pp. 236–243, 1984.

[141] M. Portnoff, « Time-frequency representation of digital signals and systems based on short-time fourier analysis », *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, *1*, pp. 55–69, 1980.

[142] J. Shor, A. Jansen, R. Maor, *et al.*, « Towards learning a universal non-semantic representation of speech », *arXiv preprint arXiv:2002.12764*, 2020.

[143] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, and T. Virtanen, « Sound event detection with depthwise separable and dilated convolutions », *in 2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–7.

[144] C. Szegedy, W. Liu, Y. Jia, *et al.*, « Going deeper with convolutions », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[145] X. Zhang, X. Zhou, M. Lin, and J. Sun, « Shufflenet: an extremely efficient convolutional neural network for mobile devices », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[146] R. Serizel, N. Turpault, A. Shah, and J. Salamon, « Sound event detection in synthetic domestic environments », *in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 86–90.

[147] Y. Huang, X. Wang, L. Lin, H. Liu, and Y. Qian, « Multi-branch learning for weakly-labeled sound event detection », *in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 641–645.

[148] G. Demirezen and A. S. Fung, « Application of artificial neural network in the prediction of ambient temperature for a cloud-based smart dual fuel switching system », *Energy Procedia*, vol. 158, pp. 3070–3075, 2019.

[149] A. K. Uysal, « On two-stage feature selection methods for text classification », *IEEE Access*, vol. 6, pp. 43 233–43 251, 2018.

[150] S. Abdoli, P. Cardinal, and A. L. Koerich, « End-to-end environmental sound classification using a 1d convolutional neural network », *Expert Systems with Applications*, vol. 136, pp. 252–263, 2019.

[151] *STM SensorTile Sensor Nodes*, `https://www.mouser.fr/new/stmicroelectronics/stm-sensortile-sensor-nodes/`, Accessed: March 9, 2023.

Titre : Capteur polyvalent acoustique pour l'habitat intelligent

Mot clés : Classification Automatique des Sons Environnementaux (ECS), amplitude instantanée, fréquence instantanée, décomposition en modes empiriques (EMD), Teager-Kaiser energy operator (TKEO) , réseau de neurones récurrents (RNN), réseau de neurones convolutif (CNN), réseau de neurones convolutif profond (DSCNN).

Résumé : Les sons environnementaux proviennent généralement de sources diverses et variées, telles que l'activité humaine, les objets et la nature. La classification automatique de ces sons suscite récemment un grand intérêt grâce à son grand potentiel d'application dans divers domaines, comme l'interaction home-machine, l'habitat intelligent, l'audition robotique, la reconnaissance automatique d'activités, les systèmes de surveillance automatique, etc. Dans le cas d'un habitat intelligent, L'hétérogénéité des événements à surveiller conduit à l'usage d'un grand nombre de capteurs, de différentes natures, ce qui impacte le coût, la consommation énergétique, la complexité d'installation et de gestion ainsi que l'encombrement et le volume de données à traiter. L'objectif de cette thèse est de démontrer que l'utilisation de la classification automatique des sons environnementaux (ECS pour Environmental Sound Classification en anglais), apporte une solution à cette problématique de réduction du nombre et de la diversité des capteurs en remplaçant tout ou une partie de ces capteurs par des capteurs acoustiques. La faisabilité de cette solution a été validé par le développement d'un système, implémenté sur une carte composée d'un microcontrôleur (Cortex M4) à faible empreinte mémoire et à basse consommation et d'un microphone, pour la reconnaissance automatique en temps réel de quatre sons environnementaux (pluie, vent, pas humain et passage de voiture). Pour une meilleure adéquation algorithme-architecture, différentes méthodes d'extraction et de sélection de caractéristiques et différents modèles d'apprentissage automatique ont été étudiés et comparés. Une nouvelle approche d'extraction de caractéristiques à partir du spectrogramme d'amplitude et de fréquence instantanées (SAFI), a été proposée et comparée à l'approche traditionnelle utilisant le spectrogramme de Mel. Pour s'affranchir des limitations de l'approche traditionnelle, liées au fait que les sons environnementaux sont des signaux multi-composantes, non-stationnaire et issues de systèmes non-linéaires, la construction du spectrogramme d'amplitude et de fréquence instantanées est effectuée par décomposition en modes empiriques (EMD) couplée à l'opérateur d'énergie de Teager-Kaiser (TKEO). Pour générer le modèle d'apprentissage, une base de données de sons environnementaux choisis a été construite. Les modèles d'apprentissage retenus sont les réseaux de neurones convolutifs (CNN) et convolutifs profonds (DSCNN), optimisés en termes du nombre de paramètres et de caractéristiques. Cette optimisation est effectuée grâce à une méthode de sélection de caractéristiques par réseau de neurones récurrent (RNN). Ce système de reconnaissance automatique développé a été évalué avec de nouveaux signaux pour une classification en temps réel.

Title: Capteur polyvalent acoustique pour l'habitat intelligent

Keywords: : Environmental sound classification (ESC), empirical mode decomposition (EMD), Teager-Kaiser energy operator (TKEO), instantaneous amplitude (IA), instantaneous frequency (IF), convolutional neural networks (CNN), depth-wise separable convolutional networks (DSCNN), recurrent neural networks (RNN).

Abstract: Environmental sounds emanate from a variety of sources, such as human and non-human activities, traffic sounds, birds, rain, and sounds produced by human activity in houses, offices, cafes, and numerous others. The automatic classification of these sounds has recently attracted a lot of interest due to its great potential for application in various domains, such as human-machine interaction, smart homes, robotic hearing, automatic activity recognition, automatic surveillance systems, etc. In the case of a smart home, the heterogeneity of the events to be monitored leads to the use of a large number of sensors of different types, which impacts the cost, energy consumption, and complexity of installation and management, as well as on the volume of data to be processed. The objective of this thesis is to demonstrate that the use of ESC (Environmental Sound Classification) provides a solution to the problem of reducing the number and diversity of sensors by replacing all or part of these sensors with acoustic sensors. The feasibility of this solution has been validated by the development of a system, implemented on an edge device composed of a microcontroller (Cortex M4) with a small memory footprint and low power consumption and a microphone, for the automatic recognition in real-time of four environmental sounds (rain, wind, human footsteps, and passing cars). For better algorithm-architecture matching, different feature extraction and selection methods and different machine-learning models are studied and compared. A new feature extraction approach based on the Instantaneous Amplitude (IA) and Frequency (IF) spectrogram has been proposed and compared to the traditional approach using the Mel spectrogram. To overcome the limitations of the traditional approach, related to the fact that environmental sounds signals are multi-component, non-stationary signals from non-linear systems, the construction of the instantaneous amplitude and frequency spectrogram is carried out by Empirical Mode Decomposition (EMD) coupled with the Teager-Kaiser Energy Operator (TKEO). To generate a model for ESC, a database of selected environmental sounds was constructed. The selected learning models are convolutional neural networks (CNN) and depth-wise separable convolutional neural networks (DSCNN), optimized in terms of the number of parameters and features. The feature size reduction and optimization are carried out using a recurrent neural network (RNN) feature selection method. This developed automatic recognition system was evaluated with new signals for real-time classification.