ROBUST DEEP LEARNING FRAMEWORKS FOR ACOUSTIC SCENE AND RESPIRATORY SOUND CLASSIFICATION

A THESIS SUBMITTED TO

THE UNIVERSITY OF KENT

IN THE SUBJECT OF COMPUTER SCIENCE

FOR THE DEGREE

OF PHD.

By

Lam Dang Pham

January 2021

Acknowledgements

Firstly, I acknowledge with thanks my supervisors, Professor Ian McLoughlin and Professor Palaniappan Ramaswamy for their invaluable advice and guidance in my research. I am very much thankful to Professor Ian McLoughlin for spending time on discussing ideas and revising papers, and I am specially grateful for his kind care and help in my personal life.

Secondly, I would like to thank Doctor Huy Phan, School of Electronic Engineering and Computer Science, Queen Mary University of London, and Professor Alfred Mertins, Institute for Signal Processing, University of Luebeck for their invaluable advice and supporting devices used to conduct experiments.

Finally, I am also thankful for my colleagues who have made the School and Computing in Medway a friendly and comfortable working environment.

Abstract

Although research on Acoustic Scene Classification (ASC) is very close to, or even overshadowed by different popular research areas known as Automatic Speech Recognition (ASR), Speaker Recognition (SR) or Image Processing (IP), this field potentially opens up several distinct and meaningful application areas based on environment context detection. The challenges of ASC mainly come from different noise resources, various sounds in real-world environments, occurring as single sounds, continuous sounds or overlapping sounds. In comparison to speech, sound scenes are more challenging mainly due to their being unstructured in form and closely similar to noise in certain contexts. Although a wide range of publications have focused on ASC recently, they show task-specific ways that either explore certain aspects of an ASC system or are evaluated on limited acoustic scene datasets.

Therefore, the aim of this thesis is to contribute to the development of a robust framework to be applied for ASC, evaluated on various recently published datasets, and to achieve competitive performance compared to the state-of-the-art systems. To do this, a baseline model is firstly introduced. Next, extensive experiments on the baseline are conducted to identify key factors affecting final classification accuracy. From the comprehensive analysis, a robust deep learning framework, namely the *Encoder-Decoder* structure, is proposed to address three main factors that directly affect an ASC system. These factors comprise low-level input features, high-level feature extraction methodologies, and architectures for final classification. Within the proposed framework, three spectrogram transformations, namely Constant Q Transform (CQT), gammatone filter (Gamma), and log-mel, are used to convert recorded audio signals into spectrogram representations that resemble two-dimensional images. These three spectrograms used are referred to as low-level input features. To extract high-level features from spectrograms, a novel *Encoder* architecture, based on Convolutional Neural Networks, is proposed. In terms of the *Decoder*, also referred as to the final classifier, various models such as Random Forest Classifier, Deep Neural Network and Mixture of Experts, are evaluated and structured to obtain the best performance.

To further improve an ASC system's performance, a scheme of two-level hierarchical classification, replacing the role of *Decoder* classification recently mentioned, is proposed. This scheme is useful to transform an ASC task over all categories into multiple ASC sub-tasks, each spanning fewer categories, in a divide-and-conquer strategy. At the highest level of the proposed scheme, meta-categories of acoustic scene sounds showing similar characteristics are classified. Next, categories within each meta-category are classified at the second level. Furthermore, an analysis of loss functions applied to different classifiers is conducted. This analysis indicates that a combination of entropy loss and triplet loss is useful to enhance performance, especially with tasks that comprise fewer categories.

Further exploring ASC in terms of potential application to the health services, this thesis also explores the 2017 Internal Conference on Biomedical Health Informatics (ICBHI) benchmark dataset of lung sounds. A deep-learning framework, based on our novel ASC approaches, is proposed to classify anomaly cycles and predict respiratory diseases. The results obtained from these experiments show exceptional performance. This highlights the potential applications of using advanced ASC frameworks for early detection of auditory signals. In this case, signs of respiratory diseases, which could potentially be highly useful in future in directing treatment and preventing their spread.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and State-of-the-art Approaches

1.1.1 Motivation

The role of audio in communication applications has become essential to modern human life, comparable to the role played by images or text. A historical development of audio processing systems began with the invention of the phonograph in the 1870s, and ran to current high-tech systems, as summarised by Ian McLoughlin in his textbook [2]. This highlights the important, but often hidden, role of audio-related research over the years. In terms of audio signal detection, there are four major fields of research including Automatic Speech Recognition (ASR), Speaker Recognition (SR) and non-speech research into Acoustic Event Detection (AED) and Acoustic Scene Classification (ASC). As regards the first two research areas, human languages are the focus, and impressive achievements have been demonstrated from these research areas. For example, the Siri tool from Apple and the Alexa tool from Amazon are two famous applications of ASR, while SR has become popular in security systems that consider speech biometrics as one of the necessary security layers, especially in modern telephone banking applications.

Regarding the two remaining research areas, these are considered to be newly (or recently) emerging fields over the past few years. One of the pioneers, Lyon [3], termed this "Machine Hearing", described in terms of how accurately a machine could listen to and understand sounds in a real-world environment compared to humans.

In terms of acoustic scenes in ASC, they refer to environmental sound occurrences such as the sounds heard *in an office, on a train, in a car* or *in a forest.* Detecting surrounding environments could be described by the terms "scene detection", "context detection", or "sound scene classification". By detecting the current location, devices could obtain useful information to enable them to respond appropriately or adjust certain functions, opening up a wide range of distinct applications. For instance, [4, 5] show the contribution of scene detection for enhancing the listening experience of users. Moreover, scene classifiers can support sound event detection when these sound events are mixed in real-world environments [6]. There are also significant applications in robotics, where the function of scene detection integrated into a robotics system was an early proposal made by Clarkson *et al.* [7], followed by El-Maleh *et al.* with mobile applications [8]. Recent success in developing a distributed sensor-server system for acoustic scene classification was described by Jakob Abeber *et al.* [9], which promisingly opens up the likelihood of further practical applications in the near future.

Despite the great potential for enabling a variety of applications, compared to the mature fields of automatic speech recognition (ASR) or speaker recognition (SR), ASC-based applications are still in their infancy due to the presence of several challenges. In particular, analyses of audio recordings readily reveals that sound events and sound scenes always exit simultaneously in real-world environments. For example, *bird song* is usually heard in a park, a *car horn* is usually outdoors; but a *barking dog* could be outside or inside a house. If the background and foreground are referred to as noise and signal respectively, it is a fact that the signal-to-noise ratio exhibits extremely high variability due to the diverse range of environments and recording conditions. To complicate matters further, a lengthy sound event could be considered background in certain contexts and foreground in others. For instance, a *pedestrian street* recording may have a generally quiet background, but with short vehicle engine foreground events, as traffic passes. However, a lengthy engine sound in an recording *on a bus* would be considered a background sound rather than a foreground event. Furthermore, both background and foreground contain true noise – continuous, periodic or aperiodic acoustic signals that interferes with the understanding of the scene. Besides, other challenges may come from the available datasets for studying this area. In particular, some datasets lack sufficient recorded data, contain unbalanced data (i.e. a large number of recordings of some classes with other classes having very little data), or even high-cross correlation among sound categories. Recently, the issue of mismatched recording devices, something which often occurs in practical applications has been raised as a new challenge for the ASC task (i.e. different devices record data in different classes, or data from some devices are used for training, but different devices are used in practice).

The variabilities and difficulties mentioned make acoustic scene classification (ASC) particularly challenging. To deal with such challenges, recent ASC publications have tended to focus on two main aspects of machine hearing, which are discussed in the following subsections.

1.1.2 State-of-the-art Approaches

The first aspect of machine hearing addressed by most state-of-the-art systems aims to solve the lack of discriminative information by exploiting various methods of low-level feature extraction. In particular, it is notable that early publications used Mel Frequency Cepstral Coefficients (MFCC) parameters [2] or combined MFCCs with temporal characteristics of audio sound such as loudness, average short-time energy, zero-crossing rate, spectral flux, or spectral centroid [10, 11, 12, 13]. More recently, ASC research has exploited spectrogram representations, and has made efforts to explore information from different recording channels [14, 15], different kinds of spectrogram [16, 17, 18, 19], and different time resolutions of spectrogram input [15]. By using multiple input features, recent ASC systems apply various ensemble methods such as majority voting, sum or product fusion [20, 21] to fuse results obtained from discrete models. Although using multi-input features, combined with ensemble models, helps to achieve high performance, none of the publications has provided a comprehensive analysis of how to select optimum input features for different tasks. Furthermore, ensemble models exhibit a high cost of computation, which may include significant redundancy. Due to the computation cost, almost all state-of-the-art systems were evaluated on limited size datasets.

The second research trend found in state-of-the-art publications focuses on constructing and training powerful learning models, with the aim of obtaining high-performing high-level features. For example, Lidy and Schindler [22] proposed two parallel CNN-based models with different kernel sizes to learn from a CQT spectrogram input, capturing different regions of the spectrogram. Focusing on pooling layers, where high-level features are condensed, Zhao *et al.* [23, 24] proposed an attention pooling layer that showed an effective improvement compared to conventional max or mean pooling layers. With the inspiration that different frequency bands in a spectrogram contain distinct features, Phaye *et al.* [25] proposed a *SubSpectralNet* network which is able to extract discriminative information from 30 sub-spectrograms. These examples of ASC systems recently mentioned all involve an end-to-end training process. In such systems, values of the second to last layer are referred to as high-level features (or sometimes as embeddings), while the final layer, normally implemented as a softmax, performs the final classification. Another approach applies two different models which are trained separately. While the first model is used to extract high-level features, the second model takes the role of the final classifier.

It can be seen that recent works have made efforts to improve the overall performance of ASC systems with more and more complex approaches. However, they mainly focus on specific aspects of an ASC system, and there are various issues that have not been deeply considered or directly addressed. Firstly, although multiple low-level features have been shown to be effective at enhancing the performance of systems, no recent publication has addressed which low-level factors have the greatest influence on overall performance. Furthermore, ensembles of multiple input features face a direct trade-off between system performance and extensive computation cost, essentially throwing computing power at the problem. This unfortunately makes the approaches incompatible with many real-time applications or platforms which are constrained in terms of computing power or applications which are constrained in terms of latency. A question arises as to whether there is an effective way to combine the most important low-level features, thus solve both the lack of refined input information and simultaneously avoid high computation costs. Secondly, although systems using complicated deep-learning networks show effectiveness in extracting good high-level features, none of the publications investigates the role of the final classifier in exploring those high-level features. Most simply present a complete architecture without further experimentation or analysis. It can be seen that these issues mentioned above may be grouped into three main topics of low-level feature input, high-level feature extraction, and output classification – each of which affects ASC system performance in different ways. Almost all state-of-the-art ASC systems are chosen or optimised in a task- specific way (i.e. they perform well for one task, but are not evaluated for others), and no consensus has emerged regarding an optimum choice for any of the three factors. Finally, while the ability to perform early detection (i.e. low latency classification) is very useful for real-world applications such as human-robot interaction, noise reduction during calls or in acoustic security systems, very few publications explore or even mention this [26, 27, 28]. These several factors motivate this research to develop an ASC system that targets the three most important issues mentioned above while providing a comprehensive analysis on the ability of early detection and exploration of the main factors involved in ensuring good performance.

1.2 Contribution

During my PhD research in this field, I have made contributions in the five following areas;

A comprehensive analysis of low-level features in ASC. I have explored a variety of low-level features to tackle the lack of input information in ASC research. To understand how low-level features affect the classification result in an ASC system, this thesis firstly proposes a baseline system which is used to provide a comprehensive analysis. The baseline uses spectrogram representations as low-level feature input and employs a C-DNN-based architecture (defined later in Chapter 3) for classification. By using the baseline described, various low-level feature settings such as channel information, spectrogram type, time resolution, and data augmentation are evaluated. This analysis helps to identify the most important low-level features and how they affect the final classification accuracy. Part of this contribution was published in the Audio Engineering Society (AES) 2019 Conference [21] and the 20th INTERSPEECH 2019 Conference [29].

A novel *Encoder-Decoder* framework for Acoustic Scene Classification. It is a fact that condensed and discriminative high-level features directly affect the final classification accuracy in an ASC system. Therefore, successfully developing a high-performing extractor is one of the most important aspects of building an effective ASC system. This thesis contributes to ASC research by introducing a novel and robust architecture which we denote the *Decoder-Encoder* framework. This enables a system to learn multiple low-level input features, from which the framework generates high-performing high-level features (specifically, this is the role of the *Encoder*). Furthermore, it then provides an analysis of various final classifiers models (the *Decoder* function). This contribution was published in Journal of Digital Signal Processing [30].

The ability of early detecting recording environments. Although early detection of recording environments promisingly opens a wide range of applications as introduced in Section 1.1.1, a few of research [28, 27] mentioned and not many experiments have been conducted. This contribution shows an analysis of early detecting recording environments by using the novel *Decoder-Encoder* framework recently mentioned. This contribution was published in Journal of Digital Signal Processing [30].

Two-level hierarchical classification scheme as an effective encoder in an ASC system. Further investigation to improve the final classifiers, referred to as *decoders* mentioned under the description of the *Decoder-Encoder* framework. This research contribution is to present a novel scheme of two-level hierarchical classification. By exploiting cross-relation among environmental categories and using the triplet loss function for training, the scheme is able to enhance system performance. This contribution was published in the International Joint Conference on Neural Networks (IJCNN) 2020 [31].

Application of the above for early prediction of respiratory disease. This thesis has already claimed that an effective ASC can enable important future applications, and in this work one such important application is explored. Up to this point, all evaluations have been done using public databases of audio clips that are presented as part of worldwide ASC challenges, particularly as part of Detection and Classification of Acoustic Scenes and Events (DCASE). These have real-world aspects, but are essentially artificially constructed datasets collected under controlled conditions. This contribution now considers an important real-time application – specifically the detection of respiratory diseases from lung sounds. This follows the 2017 Internal Conference on Biomedical Health Informatics (ICBHI) lung sound dataset and challenge. The results obtained show the potential to apply the deep-learning frameworks developed in this research (and described in the early chapters of this thesis) to create advanced computational techniques for early detection of respiratory diseases. Furthermore, for these frameworks to be compatible with real time portable or wearable computational devices. This contribution is published in the 42nd Annual International Conferences of the IEEE Engineering in Medicine and Biology Society [32] and being considered for publication in IEEE Journal of Biomedical and Health Informatics [33], the 43th Annual International Conferences of the IEEE Engineering in Medicine and Biology Society [34]

1.3 Published and Preprint Papers

This section shows published and preprint papers that are relevant to and contribute into the thesis.

-First-author papers:

1. **L. Pham**, I. McLoughlin, H. Phan, R. Palaniappan, and Y. Lang, "Bag-of-features models based on C-DNN network for acoustic scene classification", in Audio Engineering Society Conference: 2019 AES International Conference on Audio Forensic (AES), 2019 [21].

2. **L. Pham**, I. McLoughlin, H. Phan, and R. Palaniappan, "A robust framework for acoustic scene classification", in Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 3634–3638, 2019 [29].

3. **L. Pham**, H. Phan, T. Nguyen, R. Palaniappan, A. Mertins, and I. McLoughlin, “Robust acoustic scene classification using a multi-spectrogram encoder-decoder framework”, Digital Signal Processing, vol. 110, 2021 [30].

4. **L. Pham**, I. McLoughlin, H. Phan, R. Palaniappan, and A. Mertins, “Deep feature embed- ding and hierarchical classification for audio scene classification”, in Proc. International Joint Conference on Neural Networks (IJCNN), pp. 1-7, 2020 [31].

5. **L. Pham**, I. McLoughlin, H. Phan, M. Tran, T. Nguyen, and R. Palaniappan, “Robust deep learning framework for predicting respiratory anomalies and diseases”, in Proc. 42nd Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 164-167, 2020 [32].

6. **L. Pham**, H. Phan, A. Schindler, R. King, A. Mertins, and I. McLoughlin, “Inception- based network and multi-spectrogram ensemble applied for predicting res- piratory anomalies and lung diseases”, in Proc. 43nd Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC), 2021 [34].

7. **L. Pham**, H. Phan, R. Palaniappan, A. Mertins, and I. McLoughlin, “Cnn-moe based framework for classification of respiratory anomalies and lung disease detection”, IEEE Journal of Biomedical and Health Informatics, 2021 [33].

8. **L. Pham**, A. Schindler, M. Schütz, J. Lampert, S. Schlarb, R. King “Deep Learning Frameworks Applied For Audio-Visual Scene Classification”, The 4th International Data Science Conference, 2021 [35].

9. **L. Pham**, H. Tang, A. Jalali, A. Schindler, R. King “A Low-Compexity

Deep Learning Framework For Acoustic Scene Classification", The 4th International Data Science Conference, 2021 [36].

10. **L. Pham**, C. Baume, Q. Kong, T. Hussain, W. Wang, M. Plumbley "An Audio-Based Deep Learning Framework For BBC Television Programme Classification", in Proc. European Signal Processing Conference (EUSIPCO), 2021 [37].

-Co-author papers related to the thesis:

1. I. McLoughlin, Y. Song, **L. Pham**, H. Pham, P. Ramaswamy, and L. Yue, "Early detection of continuous and partial audio events using CNN," in Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 3314–3318, 2018 [28].

2. H. Phan, O. Y. Chen, P. Koch, **L. Pham**, I. Mcloughlin, A. Mertins, and M. D. Vos, "Unifying isolated and overlapping audio event detection with multi-label multi-task convolutional recurrent neural networks," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 51-55, 2019 [20].

3. H. Phan, O. Y. Chen, P. Koch, **L. Pham**, I. McLoughlin, A. Mertins, and M. De Vos, "Beyond equal-length snippets: How long is sufficient to recognize an audio scene?," in Audio Engineering Society Conference: 2019 AES International Conference on Audio Forensic (AES), Jun 2019 [20].

4. H. Phan, O. Y. Chen, **L. Pham**, P. Koch, M. De Vos, I. Mcloughlin, and A. Mertins, "Spatio-temporal attention pooling for audio scene classification," in Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 3845–3849, 2019 [38].

5. D. Ngo, H. Hoang, A. Nguyen, T. Ly, and **L. Pham**, “Sound context classification basing on join learning model and multi-spectrogram features” arXiv preprint arXiv:2005.12779, 2020 [39].

6. H. Phan, **L. Pham**, P. Koch, N. Duong, I. Mcloughlin, and A. Mertins, “On multitask loss function for audio event detection and localization,” in Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop, pp. 160-164, 2020 [40].

7. H. Phan, HL. Nguyen, OY. Chen, **L. Pham**, P. Koch, I. Mcloughlin, and A. Mertins, “Multi-view Audio and Music Classification”, in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 611-615, 2021 [41].

1.4 Organisation of This Thesis

These remaining chapters of this thesis are organised as follows.

Chapter 2 carries out a comprehensive literature review of ASC research, which covers state-of-the-art systems and recently published datasets. From the detailed analyses, open issues related to the ASC task are raised and discussed.

Chapter 3 presents a baseline system applied for ASC. Using the baseline, the effect of different low-level features (and their settings) on classification accuracy is analysed, thus identifying low-level features which are able to perform well for various scenarios. When the most influencing low-level features are indicated, mixup data augmentation applied on these features is also evaluated (The first-author paper 1 mentioned in Section 1.3 mainly contributes into this chapter).

Based on the comprehensive analysis of low-level features provided in Chapter 3, **Chapter 4** develops a novel *Encoder-Decoder* framework applied for ASC. In particular, this chapter presents a novel *Encoder* architecture that helps to learn multiple spectrograms simultaneously, thus extracts and combines high-performing high-level features. Furthermore, these high-level features extracted are explored by various *Decoder* models, reports the final classification accuracy. The results obtained prove the proposed *Encoder-Decoder* framework to be robust and general for ASC (The first-author papers 2 and 3 mentioned in Section 1.3 mainly contribute into this chapter).

Chapter 5 proposes a scheme of two-level hierarchical classification. The scheme is used to train and explore high-level features extracted from an *Encoder* architecture mentioned in Chapter 3. The results obtained in this chapter indicate that the combination of the proposed scheme and a triplet loss function during training are useful to exploit the cross-correlation between environmental categories, which helps to improve accuracy (The first-author paper 4 mentioned in Section 1.3 mainly contributes into this chapter).

Chapter 6 further explores ASC, but this time in the context of investigating a specific application of respiratory diseases detection. The extremely good results obtained from this system indicate the great potential for applying such deep-learning frameworks to not only the early detection of lung-related diseases, but also to similar application areas (The first-author papers 5, 6 and 7 mentioned in Section 1.3 mainly contribute into this chapter)..

Chapter 7 presents conclusion and future works.

Appendix where computation of spectrograms and network layers are described in detail.

Chapter 2

Literature Review

This chapter first defines the ASC task, then introduces some acoustic scene datasets which are popular in recent research literature. Next, it analyses acoustic scene representations and classification algorithms. From this analysis, this chapter then continues to identify issues with current ASC research – which in turn for the main motivation behind this thesis.

2.1 ASC Definition

The ASC task aims to classify a recording into one or more predefined categories that characterise the environment in which it was recorded. For example, a recording is classified into *in caffe, on bus, in office* or *on train*, as shown in Figure 1. A general system structure for performing ASC is described as Figure 2, showing a waveform analysed in two main steps; front-end feature extraction and back-end classification, respectively. The purpose of the first step, front-end feature extraction, is to transform a segment of recorded audio into another form that contains compact information and is suitable for the subsequent stage of classification. A high-performed transformation generates well-presented features that benefit the back-end classifier. By using features extracted during the front-end

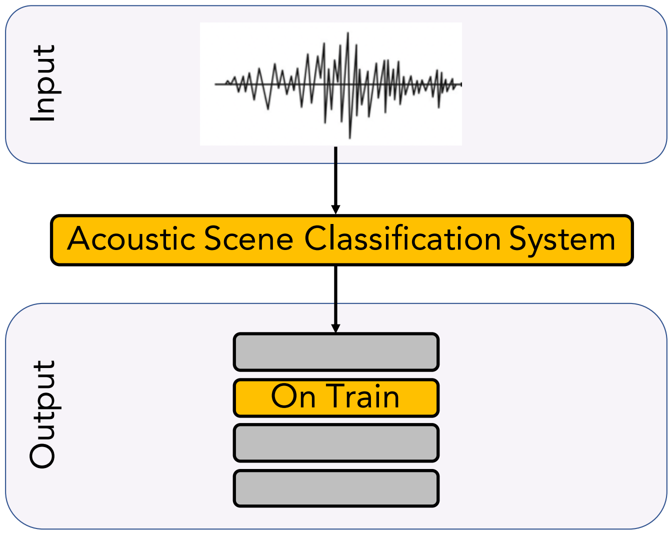

Figure 1: Task definition of Acoustic Scene Classification.

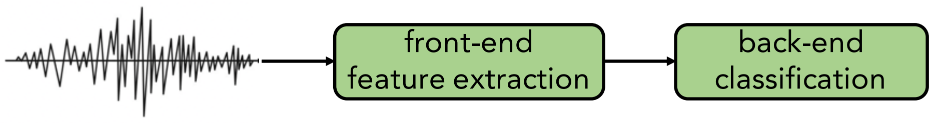

Figure 2: A general system applied for Acoustic Scene Classification.

extraction step, the back-end classifier aims to classify an environmental recording into certain predefined categories.

Techniques, which are generally applied to ASC systems, are diverse and have often been borrowed from different related research fields as shown in Figure 3. They focus on developing effective feature extraction techniques and robust models for the classification.

2.2 ASC Datasets

Table 1 lists the most prominent acoustic scene datasets which have been published as part of international challenges. These datasets were recorded in real environments and released alongside the necessary meta information in Wave format (i.e. .wav files). The Litis-Rouen dataset [47] shows the highest number of

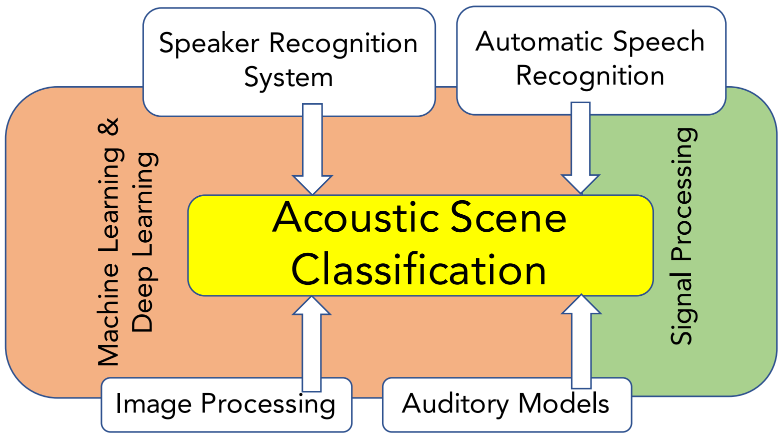
Speaker Recognition System
Automatic Speech Recognition
Machine Learning & Deep Learning
Acoustic Scene Classification
Signal Processing
Image Processing
Auditory Models

Figure 3: Acoustic Scene Classification and relationship to overlapping research areas.

Table 1: The main Acoustic Scene Classification challenge datasets.

ASC Dataset (Name)	**Published In (Year)**	**Classes (No.)**	**Time Recorded (Hours)**	**Segment Lenght (Second)**
DCASE 2019 Task 1A [42]	2019	10	40.00	10
DCASE 2019 Task 1B [42]	2019	10	46.00	10
DCASE 2019 Task 1C [42]	2019	10	44.00	10
DCASE 2018 Task 1A [43]	2018	10	24.00	10
DCASE 2018 Task 1B [43]	2018	10	28.00	10
DCASE 2017 Task 1 [44]	2017	15	17.50	10
DCASE 2016 Task 1 [45]	2016	15	13.00	30
AucoDer07 [46]	2015	4	4.20	not fixed
Litis-Rouen [47]	2014	19	25.51	30
DCASE 2013 [48]	2013	10	0.83	30
CASA 2010 [49]	2010	13	8.88	4
CASA 2009 [49]	2009	10	18.88	4
UEA-Series2 [50]	2006	10	2.92	not fixed
UEA-Series1 [50]	2006	10	0.66	not fixed

separate classes at 19, followed by DCASE 2016 [45] and DCASE 2017 [44] with 15 classes and CASA 2010 [49] with 13 classes. The remaining dataset challenges have ten different environments, with the exception of AucoDer07 [46] which only comprises 4 separate classes. Over time, as this research field has progressed, the recorded duration has increased from 0.66 hours for UAE-Series1 [50] to 46 hours for the recordings in DCASE 2019 [42]. Recently, the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) has provided a diverse set of acoustic scene datasets, motivates a lot of publications that have been evaluated with these datasets. In this thesis, the analysed systems are mainly evaluated over Litis-Rouen [47] and DCASE 2016 Task 1 [45], DCASE 2017 Task 1 [44], DCASE 2018 Task 1A & 1B [43] and DCASE 2019 [42] Task 1A & 1B datasets. These datasets are independently evaluated (i.e. each dataset is separated into Train. and Eva. subsets for training and evaluating, respectively).

2.3 Evaluation Metric

As this thesis evaluates ASC datasets of Litis Rouen and EEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) in years of 2016, 2017, 2018 and 2019, the evaluation metric of accuracy used in this thesis follows these challenges. In particular, if C is considered as the number of audio segments which are correctly predicted, and the total number of audio segments is T, the classification accuracy (Acc.%) mentioned in these challenges shares the similar computation as (note that the segment length evaluated depends on specific datasets),

$$Acc.(\%) = 100\frac{C}{T}. \tag{1}$$

As these datasets are slightly unbalanced and experimental results across categories are shown, other metrics such as Recall, Precision, or F1 score are not presented for ASC tasks in this thesis.

2.4 Acoustic Scene Representation

According to the basic stages in a typical ASC system presented in Section 2.1, the state-of-the-art systems applied to ASC use two main approaches for front-end feature extraction, namely one-dimensional frame-base measures or two-dimensional spectrogram representations respectively. As there are various transformation used for generating spectrograms, mathematical definitions of transofrmation methods are described in detail in Section A separately. The outputs from the front-end feature extraction are referred to as low-level features, and these are analysed in the following sections.

2.4.1 Frame-based Representations

Frame-based representations often utilise MFCC [2], and provide powerful feature extraction capabilities which are borrowed from the ASR community [45]. To improve discrimination between MFCC frames, MFCCs are often combined with a wide range of temporal features such as loudness, probability of voicing, average short-time energy, sub-band energy, zero-crossing rate, spectral flux, or spectral centroid [10, 11, 12, 13] (note that an MFCC frame is represented as a real valued vector, with the temporal features concatenated to that vector, effectively increasing the feature dimension for each frame).

Some systems first transform audio signals into MFCC spectrograms or log-mel spectrograms, then attempt to learn different aspects of those spectrograms to extract frame-based features. For instance, Nico *et al.* [13] applied an Amplitude Modulation Filter Bank (AMFB) method to analyse and extract features from MFCC spectrogram before concatenating other temporal features such as flux, centroid spectral entropy. Meanwhile, MultiScale-Kernel Fisher Discriminant Analysis (MSKFDA), coming from the emotion recognition field, was used by Erik [10] to provide multi-scale analysis over acoustic scene factors (combinations of MFCCs and temporal features). From log-mel spectrogram, Alain *et al.* [47] applied Non-negative Matrix Factorisation (NMF) techniques to extract condensed features. I-vector extraction, a powerful technique widely used in the SR research community [51], has also recently been applied for ASC. Recent publications include various methods to extract i-vectors from MFCC spectrograms by using a Universal Background Model (UBM) model [52, 53, 54] or Gaussian Mixture Model (GMM) [55]. In an ASC system proposed by Abidin *et al.* [56], frame-based features were extracted after many steps, via a complicated extractor.

Firstly, auditory signals were transformed into a CQT spectrogram. Then, authors used Local Binary Pattern (LBP) techniques, borrowed from image texture extraction research, to extract a Time-Frequency Representation (TFR) from the CQT spectrogram. The TFR was continuously solved by two different image process techniques. The first method, using Histogram of Oriented Gradients (HOG), extracted HOG features from the TFR. The second method, based on Local Binary Patterns (LBP), extracted histogram features located at linear zones of the TFR. Eventually, two frame-based features, HOG and LBP, were concatenated before being fed into a Support Vector machine (SVM) model for classification.

Operating directly with audio signals, Song *et al.* [57] applied the auditory statistics of a cochlear filter model to extract discriminative features, operating without any spectrogram transformation step in their proposed system.

2.4.2 Spectrogram Representations

Spectrogram images have higher resolution and contain richer information, in terms of both temporal and frequency dimensions, than general frame-based approaches. This thesis therefore explores a variety of spectrograms including short term Fourier transform (STFT), log-mel [16, 24, 25], MFCC [44], CQT [22], Gamma [19, 58] and scalogram [59].

Further exploring spectrograms, publications applied various filters or image processing techniques for improving spectrogram quality. For instance, Truc *et al.* [16] applied a Nearest Neighbour Filter (NNF) on a log-mel spectrogram to generate a new NFF spectrogram. By using a median-filtering harmonic percussive source separation over a log-mel spectrograms, Octave Mariotti *et al.* [14] and Yuma *et al.* [15] generated two spectrograms each of which focuses on either the time or the frequency resolution. Yang *et al.* [60] used the Kullback-Leibler

(KL) divergence scale to develop a KL filter bank. Next they applied these filters on a log-mel spectrogram, generating a KL spectrogram that experimentally outperformed log-mel and CQT. Waldekar and Saha [61] firstly generated a log-mel spectrogram. Then they applied a Haar wavelet on the log-mel spectrogram to generate new features named Mel-Frequency Discrete Wavelet Coefficients (MFDWC).

Combining image texture techniques known as Difference of Gaussians (DoG) and the Sobel edge detection operator [62], Wu *et al.* [63] applied these techniques to log-mel spectrograms in order to enhance sound textures. Similarly, Park *et al.* [64] extracted temporal energy density and energy variations for each frequency bin of a Gamma spectrogram by using a covariance matrix (COV) and double Fast Fourier Transform (FFT) image.

2.4.3 Multiple Low-level Input Features and Data Augmentation to Address ASC Challenges

To deal with the ASC challenges mentioned in Section 1.1.1, publications adopt two main approaches in terms of exploring low-level features. The first approach considers that each low-level feature may capture distinct features of an audio signal. Therefore, if multiple input features are used, it is effective at improving system performance. Meanwhile, the second approach considers the use of data augmentation to tackle the issues related to datasets, such as lack of, or unbalanced nature of the data.

Multiple low-level input features: For frame-base representations, MFCCs are often combined with temporal features as mentioned in Section 2.4.1, or even with a variety of features such as perceptual linear prediction (PLP) coefficients, power nomalised cepstral coefficients (PNCC), robust compressive gamma-chirp filter-bank cepstral coefficients (RCGCC) or subspace projection cepstral coefficients (SPPCC) [65] that helps to achieve top-three system proposed in DCASE

2016 challenge. For spectrogram representations, published papers show a diverse combination of log-mel and different types of spectrogram such as Mel-based Nearest Neighbour Filter (NNF) spectrogram [16, 17], CQT [18], or MFCC and Gamma [19]. Testing a wavelet-transform derived spectrogram representation, Ren *et al.* [59] compared results from STFT spectrograms and both *Bump* and *Morse* scalograms. They indicated that combination of STFT spectrogram and *Bump* scalogram is useful to enhance the proposed ASC system.

By exploiting channel information, Yuma *et al.* [15] generated multi-spectrogram input from two channels, the average and difference of two channels, and explored separated harmonic and percussive spectrograms from each channel. By fusing results from channel information, the authors achieved the top-one score in DCASE 2018 challenge. Some papers proposed combining both frame-based and spectrogram features such as i-vectors with an MFCC spectrogram in [52, 53, 66].

Data augmentation: To deal with the challenges causing by unbalanced classes within a dataset or the lack of representative data, some publications have proposed a variety of data augmentation methods. These can improve the robustness and enhance the learning ability of deep network models. Early data augmentation methods combined the signals with multiple lengths of recorded audio [67]. This idea was improved by Salamon and Bello [68] who provided an analysis of various data augmentation methods, including pitch shifting, time stretching, and the addition of background noise. The research indicated that pitch shifting is useful for all types of experimental sound and a combination of all augmentation methods helps to improve the ASC system proposed. Interestingly, Zang *et al.* [69] proposed a sequence augmentation method. Firstly, an audio signal was transformed into a STFT spectrogram. Next, a certain number of continuous STFT frames, referred to as the segment length L, were grouped as segments. These segments thus were shuffled, re-arranged at different positions, and eventually were concatenated and generate a new sequence of STFF frames.

By this way, the authors improved their proposed ASC system by 2% and show that the proposed ASC system achieves the best performances with the segment length set to $L = 64$.

Recently, mixing input data (mixup) [70, 71, 72] and the application of Generative Adversarial Network (GAN) for data augmentation [73, 74, 60] have become popular, and are shown to be effective. The mixup method is easy to implement, which makes it popular in ASC and other systems. Indeed, the top-eight highest performance systems for DCASE 2018 Task 1A challenge used this method, it was also used by almost all submitted systems for the DCASE 2019 Task 1A.

Using GAN to generate more fake data has been similarly shown to be effective in helping to improve system performance. This is proved by systems in [73, 74, 60], achieving the highest scores in DCASE 2017 and 2019 challenges. However, systems using GAN for data augmentation [73, 74, 60] show competitive performance but are very complicated. In particular, these systems need to configure and train a GAN network to be used to generate new data. After training the GAN generator, they require a classified model to be trained as a filter to be able to select generated data which shows an appropriate distribution (SVM can be used). Both old data and generated data need to then be shuffled, or interspersed, before being fed into a final classifier.

2.5 Classification Algorithms

It can be seen that the front-end feature extraction methods tend to fall under one of two main approaches, either frame-based or spectrogram representations. Meanwhile the back-end classification methods are also divided into two main groups, analysed in some depth below.

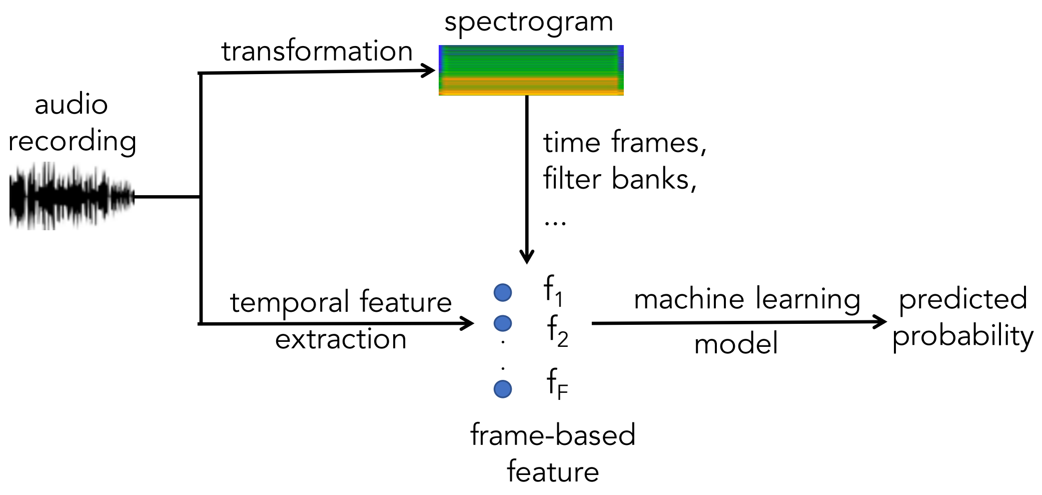

Figure 4: ASC framework using frame-based feature and machine learning model

2.5.1 Machine Learning Models

The frame-based feature approaches are normally combined with traditional machine learning models as shown in Figure 4 and Table 2. For example, baseline of the DCASE 2016 challenge [45] introduced MFCC feature extraction and Gaussian Mixture Model (GMM), which showed a very similar architecture to systems used in ASR research. Similarly, Park *et al.* [65] applied GMM models to evaluate various frame-based features such as MFCC, PLP, PNCC, RCGCC and SPCC. Meanwhile, Support Vector Machine (SVM) was widely used with diverse types of frame-based input features such as MFCC [55, 75], auditory-summary-statistics features [57], HOG and LBP features in [56], MFDWC features in [61]. Linear-based models have also been used, with Bisot *et al.* [76] proposing a modified version of supervised dictionary model (TDL) to classify NMF features. Meanwhile, Hamidn *et al.* [52] used both Linear Discriminant Analysis (LDA) [77] and Within-Class Covariance Normalization (WCCN) [78] to train i-vector features. Recently, Multilayer-Perceptron-based (MLP-based) networks have been very widely used to train frame-base features. For instance, MLP-based networks

Table 2: The state-of-the-art frame-based ASC frameworks

Author	**Front-end Feature Extraction**	**Back-end Classification**
Mesaros *et al.* [45]	MFCC	GMM
Park *et al.* [65]	MFCC, PLP, PNCC, RCGCC, SPCC	GMM
Elizalde *et al.* [55]	MFCC	SVM
Mafra *et al.* [75]	MFCC	SVM
Abidin *et al.* [56]	HOG, LBP	SVM
Waldekar *et al.* [61]	MFDWC	SVM
Bisot *et al.* [76]	NMF	TDL
Eghbal-Zadeh *et al.* [52]	i-Vector	LDA,WCCN
Mafra *et al.* [75]	MFCC	MLP
Mika *et al.* [79]	MFCC	MLP
Kong *et al.* [80]	MFCC	MLP
Jee-Weon *et al.* [54]	MFCC	MLP
Moritz *et al.* [13]	AMFB	TDNN

in [75, 79], [80], [54] were used to learn MFCC, log-mel features, and a combination of MFCC and i-vector, respectively. A variant of MLP-based networks which introduces a dependency in time between frames, namely the Time Delay Neural Network (TDNN), was used in [13] to train AMFB features.

2.5.2 Deep Learning Models

Regarding the second approach of using spectrogram representations, publications show a similar wide variety of back-end classification methods. In this case, most are using deep learning networks as shown in Figure 5 and Table 3. Spectrogram features resemble two-dimensional images, and so to feed these into deep-learning models (note that the entire variable-length spectrogram for variable-length sound input is normally split into small overlapping or non-overlapping patches of equal size [29, 69, 21, 81, 16]). Some systems obtain short spectrograms by adjusting hop size, and then feeding the entire spectrogram into back-end classifiers [43, 82]. Deep learning models applied to ASC can themselves be separated into three main categories: Multilayer Perceptron (MPL), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) based architectures. The deep learning

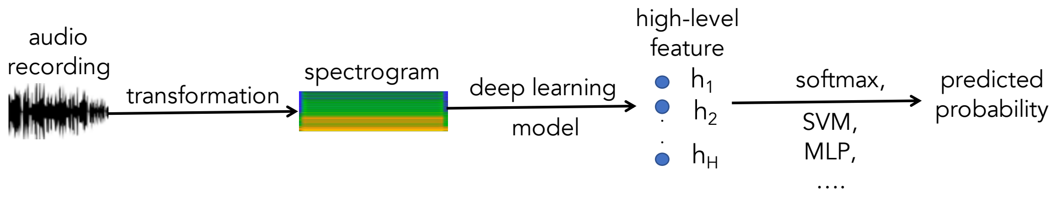

Figure 5: ASC framework using spectrogram and deep learning model

framework used as the baseline of the DCASE 2017 challenge is an example of the first category of using MPL-based networks. The input features of the MPL architecture systems need to be vectors, so the two-dimensional patches split from the much larger full spectrograms are flattened into vectors in these systems before being fed into the network [80]. Although a wide range of deep learning network are applied to ASC, CNN-based architectures are now the most popular approach. Indeed, there are a variety of ASC systems using CNN-based network such as Lenet [83], VGG [84], Resnet [85], Capsule [86], etc. which have published recently. Analysing the system characteristics submitted to the DCASE challenges over time, while half of DCASE 2016 submissions used traditional machine learning models and the remaining system applied CNN-based networks, CNN-based architectures were used in almost all DCASE 2017 systems. In DCASE 2018 and DCASE 2019 challenges, all submitted systems that achieved higher performance than the two challenge baselines either completely applied CNN-based networks or partly used them in their systems. To further analyse the CNN-based networks that have been published, publications tend to make efforts to exploit certain aspects of the CNN networks. For examples, Yang *et al.* [87] proposed a complicated CNN-based architecture called the *xception* network. This is inspired by the fact that a deep learning network trained by a wide range of feature scales and over separated channels can result in a very powerful model. Focusing on attention mechanisms, an attention-based pooling layer proposed by Zhao Ren *et al.* [24, 23] helped to improve the quality of pooling layers compared with traditional pooling layers. Exploring different frequency bands in a spectrogram, Phaye *et al.* [25]

Table 3: The state-of-the-art spectrogram based ASC frameworks

Author	**Front-end Feature Extraction**	**Back-end Classification**
Kong *et al.* [80]	MFCC	MLP
Lecun *et al.* [83]	log-Mel	Lenet (CNN)
Simonyan *et al.* [84]	log-Mel	VGGish (CNN)
Ren *et al.* [24]	log-Mel	VGGish (CNN)
Zhao *et al.* [23]	log-Mel	VGGish (CNN)
Phaye *et al.* [25]	log-Mel	VGGish (CNN)
He *et al.* [85]	log-Mel	Resnet (CNN)
Patrick *et al.* [86]	log-Mel	Capsule (CNN)
Yang *et al.* [87]	log-Mel	x-Ception (CNN)
Zhang *et al.* [69]	log-Mel	LSTM (RNN)
Zhang *et al.* [89]	log-Mel	LSTM, attention (RNN)
Zhang *et al.* [81]	log-Mel	LSTM,temporal transformer (RNN)
Phan *et al.* [38]	log-Mel	GRU (RNN)
Phan *et al.* [58]	log-Mel	GRU, attention (RNN)
Phan *et al.* [20]	log-Mel	GRU, CNN (CNN & RNN)

proposed a SubSpectralNet network which was able to extract discriminative information from 30 sub-spectrograms. Recently, Song et al. [88] proposed a new way to handle distinct features in a sound scene recording; a deep learning model extracts a bag of features from a log-mel spectrogram, including both similar and distinct ones, from which a back-end network is exploited to enhance accuracy.

RNN-based networks are very powerful methods able to learn sequences across time series in addition to spectral relationships. Zang et al. [81, 69, 89] provided a deep analysis of the application of Long Short-term Memory (LSTM), a kind of RNN network, for ASC. In particular, the authors not only evaluated a single LSTM [69] but also conducted extensive experiments on combinations of the LSTM with other techniques, such as an attention scheme [89] or a temporal transformer layer [81]. Another example showed to be effective in exploiting RNN-based network was published by Huy *et al.* [20, 58, 38]. Instead of using LSTM-based RNN, Huy *et al.* proposed using a Gate Recurrent Unit (GRU) based RNN [38]. Then they further improved the model by applying an attention scheme [58] or combining this with CNN-based architectures [20]. Although

RNN-based networks prove to be very powerful approaches in Acoustic Event Detection (AED) due to their effectiveness at capturing time sequence, they show poor performance when applied to ASC compared with CNN approaches. Indeed, not a single RNN-based model was submitted to the recent DCASE 2019 ASC challenge, and furthermore, when RNN-based architectures are used in different contexts, they are normally combined with a CNN network to improve system performance [20].

2.5.3 High-level Features

It can be seen that deep-learning-based systems using spectrogram representations have complicated architectures [87, 25, 24, 20, 58]. A deeper analysis of the kind of deep learning networks used in these systems, shows that they belong to two main groups divided by the number of training processes used. Systems which only use one training process are called *end-to-end* learning systems. In these *end-to-end* systems, the network architecture is separated into two main parts. While the first part helps to transfer low-level features (spectrogram representation) to high-level features, which contain condensed and discriminative information, the second part takes the role of classification from those condensed features. In particular, high-level features are normally referred to as the values of the next-to-last layer, and the final layer (normally using Softmax) is referred to as the classification part [87, 25, 24]. In order to gain high-performed high-level features, a variety of complicated architectures have been proposed. For examples, Truc *et al.* [16] applied two parallel CNNs to learn from two type of spectrograms, then concatenated outputs of the CNNs to generate high-level features. Similarly, Lidy *et al.* [22] used two parallel CNNs, each of which used different kernel sizes to capture different regions of a CQT spectrogram. Meanwhile, Soo *et al.* [90] used both CNN and RNN to capture spatial and time sequence features. Normally, high-level features extracted by CNN or RNN based structures are concatenated before feeding into fully-connected layers (i.e. Multilayer perceptron), referred to as the final classifier.

Inspired by the idea that if results of a first training process used to model low-level features are transferred into a second model to aggregate those features, it can improve classification accuracy without requiring an unduly complex single network architecture, the second group of ASC systems use two, or even more, different learning models. While the first model is again used to extract high-level features (note that these high-level features are also called embedded features or embeddings in some papers), the second model aims to explore high-level features, reporting final classification accuracy. An early system from this trend was described in [91]. In that system, the authors applied Random Forests (RF) to train from low-level Gamma spectrograms, converting the output into another form of features called labelled tree embedded (LTE) features. These LTE features were then classified by a SVM model. To further explore LTE features, authors conducted various experiments on the second model by using both CNN and SVM [19] or RNN [58]. Other examples were shown in [38, 20]. In these systems, deep-learning frameworks of either parallel [20] or continuous [38] combinations of CNN and RNN were used to extract high-level features. Then, a SVM model was used as the final classifier. This trend includes a variety of high-level features such as x-vectors extracted from CNNs [18], feature maps from C-NN networks [92, 93], deep-scalogram representations from CNN [59]. Recently, transfer learning technique [94], a variant of this basic approach, has been widely applied.

2.5.4 Ensemble Models

As mentioned in Section 2.4.3, recent publications reporting high performance have tended to explore multiple low-level input features to deal with the lack of sufficient training information. Systems using multi-input features often use separated learning models, then fuse the models' results to obtain a final classification accuracy. In general, these fusion methods are separated into three main groups, namely *max-fusion, mean-fusion* and *multiplication-fusion* methods, each of which is discussed in [20]. Although these systems prove to have a high cost of computation as well as a high volume of training parameters, they are able to achieve competitive results. Indeed, the top-three performing systems in the DCASE challenges [95, 52, 65, 73, 96, 97, 98, 15] and on the Litis-Rouen dataset [38, 81], use a variety of fusion methods. In particular, these high-performing systems use a spectrogram representation for low-level input features.

2.6 Open Issues

Several of the existing works mentioned above have described the open issues that this thesis is going to analyse and investigate. Firstly, while multiple low-level input features such as spectrograms, channels, frequency and time resolution, etc. have been explored for use on ASC challenges, there has been much less comprehensive analysis to identify the most effective low-level feature. Furthermore, using multiple input features usually combined with ensemble models, has a very high computational cost – effectively throwing computational power at the problem. It would clearly be better to more precisely identify optimum features (and their characteristic settings) rather than blindly combining a large set of multiple features. In terms of back-end classification, although a wide range of deep learning frameworks have been proposed, they mainly exploit specific features (i.e. are highly feature-specific) and are almost always evaluated over a very limited set of datasets. The danger there is of building locally-optimum systems which work well on one challenge, but perform poorly on others. As ASC challenges mainly come from various sound events and scenes inside environmental recording datasets, focusing on specific aspects of an ASC system easily causes overfitting issues, especially troublesome when a model is evaluated over different datasets to those it is trained for. Further analysis of back-end classification models shows that, although some high-performance high-level features have been recently defined, no current publications adequately explore the relationship between high-level features which have been extracted from different low-level features. Moreover, very few published papers explore the nature and the effect of those high-level feature characteristics. Finally, although environmental sounds have high cross-correlation due to the presence of similar types and degrees of background noise, few publications provide an analysis of this aspect of systems.

Chapter 3

Low-Level Feature Analysis

This chapter aims to provide a comprehensive analysis of low-level features in an ASC system and identify how these features affect the final classification accuracy. To this end, a wide range of low-level features such as channel information, spectrogram types and various image patch sizes, etc., are evaluated. These experiments are conducted over a C-DNN-based model, referred to as the proposed baseline, and evaluated using the DCASE 2018 Task 1A, 1B dataset. After indicating the most influencing low-level features, C-DNN baseline's performance is evaluated again with mixup data augmentation.

3.1 High-level Architecture

Starting with a spectrogram representation as the low-level feature, a general system architecture for ASC is presented in Figure 6. It can be seen that the entire ASC system is separated into two main processes. The first process (top half) has the role of transforming the selected audio channels into one or more types of spectrogram, and then splitting the full spectrogram into smaller patches of different sizes to form a bag-of-features. In this case, the patches are non-overlapping and of predefined size.

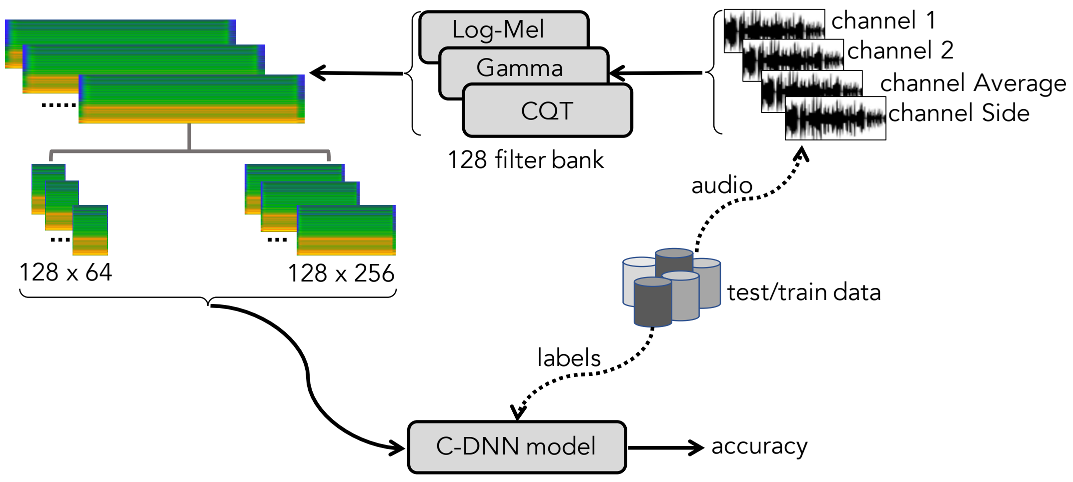

Figure 6: The high-level architecture of proposed baseline.

The second process (bottom half) is a machine learning model which trains the input patches extracted by the first process and performs classification. In this case, the model is a CNN combined with a DNN (abbreviated to C-DNN). The output of the machine learning model is the reported classification accuracy.

3.2 Low-level Feature Analysis

Due to ASC challenges discussed previously in Section 2.4.3, it is known that an ensemble of multiple low-level features or models can promisingly enhance the classification accuracy in current state-of-the-art systems. This motivates the analysis and exploration of the effects on classification accuracy of different bags-of-features. Specifically, it drives to explore into three groups as shown in Table 4.

The first exploration focuses on the effect of different channels (since the data sets include two-channel recordings namely Left and Right). The possibilities are using the first channel alone (Left), the second channel alone (Right), and the space information of two channels obtained by Average and Side, where Average

Table 4: Bag-of-feature analysis settings for ASC.

Channels	Channel 1, Channel 2, Average, and Side of two channels
Patch sizes	0.37 s (64 bins), 0.74 s (128 bins), 1.11 s (192 bins), 1.48 s (256 bins)
Spectrograms	log-mel, Gamma, and CQT (each with 128 filters)

means (Left+Right)/2 and Side means (Left-Right).

The second exploration is for different patch sizes, in terms of the time duration that they cover. The number of frequency bins used for the analysis is fixed at 128, but a different numbers of time bins are evaluated, specifically 64, 128, 192, and 256 – giving rise to the four different patch durations shown in Table 4.

The third exploration considers the use of three alternative spectrogram transformations, namely log-mel, gammatone (Gamma) and Constant Q Transform (CQT). Since these spectrograms are derived from different auditory models, it is plausible that they can each contribute distinct features for classification. By ensuring that parameters such as window size, hop size and patch number are fixed, the number of data items fed into the back-end learning model will be equal for different types of spectrogram, hence any difference in classification accuracy is due to the characteristics of each spectrogram, not due to different learning requirements. In this thesis, while log-mel and CQT are generated by using a popular toolbox namely Librosa [99], gammatone-like spectrogram toolbox [100] is used to generate Gamma. The methods of transforming a selection of audio signal into a spectrogram are presented in detail in Chapter A.

Furthermore, this chapter also analyses how mixup data augmentation, a simple-implemented and popular method of data augmentation mentioned in Chapter 2, affects to the classification accuracy. Description and settings for mixup data augmentation are presented in detail in next Section 3.5.6.

3.3 Propose a Baseline

Table 5: The primary operating parameters of the proposed baseline for DCASE 2018 challenge, alongside the parameters from the challenge baseline.

System Setting	**Proposed C-DNN baseline**	**DCASE 2018 baseline**
Window size	0.044s	0.04s
Hop size	12.5%	50%
Spectrogram method	log-mel	log-mel
Filter number	128	40
Spectrogram size	128×1728	40×500
Like-image features	13 patches (128×128)	entire spectrogram
Deep learning model	C-DNN (Lenet-7)	C-DNN (The best model form DCASE 2016)

To analyse different bags-of-features, a baseline architecture is first established, configured using the parameters set out on the left hand side in Table 5. Additionally, this chapter also compares overall performance against the standard DCASE 2018 baseline system of [43]. The proposed C-DNN parameters are listed in Table 6.

Given by Table 5, it can be seen that the proposed baseline architecture uses a window size of 0.044 s (albeit extracted from a higher dimension spectrogram) and a smaller hop size of 12.5% compared with a window size at 0.04s and a hop size of 50% in the DCASE 2018 baseline, since the baseline's purpose aims to extract additional useful information from the input spectrogram. Regarding the spectrogram, the proposed architecture uses a log-mel filter with 128 bands which is much larger than the 40 Mel filter bands of the DCASE 2018 baseline. Based on these parameters, the proposed spectrogram, which is generated from 10-second segments with sample rate of 48,000 Hz, has a bigger size in both time and frequency bins of 128×1728 compared with 40×500 in the DCASE 2018 baseline. Then, the entire spectrogram of 128×1728 is split into 13 non-overlapping patches of 128 × 128 before feeding into a back-end classification.

Table 6: The proposed C-DNN network configuration (the upper part is for CNN and the lower part is for DNN).

Layers	**Output shape**	**Kernel/Drop Ratio**
Input	128×128×1	-
Convolutional 1	128×128×32	[3×3] @ 32
ReLU 1	128×128×32	-
Batch normalization 1	128×128×32	-
Average pooling 1	64×64×32	[2×2]
Dropout 1	64×64×32	10%
Convolutional 2	64×64×64	[3×3] @ 64
ReLU 2	64×64×64	-
Batch normalization 2	64×64×64	-
Average pooling 2	32×32×64	[2×2]
Dropout 2	32×32×64	15%
Convolutional 3	32×32×128	[3×3] @ 128
ReLU 3	32×32×128	-
Batch normalization 3	32×32×128	-
Average pooling 3	16×16×128	[2×2]
Dropout 3	16×16×128	20%
Convolutional 4	16×16×256	[3×3] @ 256
ReLU 4	16×16×256	-
Batch normalization 4	16×16×256	-
Global average pooling 4	256	-
Dropout 4	256	25%
Fully connected 5	512	-
ReLU 5	512	-
Drop out 5	512	30%
Fully connected 6	1024	-
ReLU 6	1024	-
Dropout 6	1024	35%
Fully connected 7	10	-
Softmax 7	10	-

As regards learning models, the DCASE 2018 baseline reuses the architecture of the top ranked submission of DCASE 2016 [101]. However the DCASE 2016 system only had two convolution blocks (convolutional, batch normalization, rectify linear unit, and dropout layers) followed by one dense layer and an output layer with Softmax classification.

The new baseline is much more complicated, with four convolutional blocks based on Lenet-7 [83]. In particular, the C-DNN proposed presents four convolutional blocks, as configured in the upper part of Table 6, each of which includes a convolutional layer followed by a rectify linear unit (ReLU), a batch normalization, an average pooling, and a dropout layer. At the final convolution block, instead of using a average pooling layer, a global average pooling layer is applied to enhance the accuracy based on the idea of considering the contribution of all channels output to the final convolution block as a bag-of-features and reducing noise.

The classification operation is shown in the lower part of Table 6 (note that while blocks are separated by single line, the role of classification in the lower parts is separated by a double line), handled by fully connected, ReLU and dropout layers. At the final layer, a Softmax function is used for classifying into different ten scene contexts.

3.4 Experimental Setting

3.4.1 Dataset

Experiments in this chapter are conducted using the development set (Dev. set) of DCASE 2018 Task 1A and 1B [43]. The audio files in these datasets are all wave file format recordings with a sample rate of 48000 Hz and have a 10-second duration.

As DCASE 2018 Task 1A dataset, all of the audio files were recorded by the same device, denoted ‘device A’ (Soundman OKM II Klassik/studio A3 electret microphone and a Zoom F8 audio recorder), and are grouped into ten different categories, with one category label per recording. The data is unbalanced so that the number of recordings per category is slightly uneven. This can be seen in the left hand side of Table 7 which identifies the categories and the number of 10-second audio files that each categories contains. In total, the task includes 8640 audio files. Using the DCASE 2018 suggested test/train split [43], recordings of the development set are separated into a Training subset (6122 audio files) and a

Table 7: Development set of DCASE 2018 Task 1A and 1B datasets are split into Test/Training subsets

Categories	**Training subset (1A)**	**Test subset (1A)**	**Training subset (1B)**	**Test subset (1B)**
Airport	599	265	707	301
Bus	622	242	730	278
Metro	603	261	711	297
Metro Station	605	259	713	295
Park	622	242	730	278
Public Station	648	216	756	252
Shopping Mall	585	279	693	315
Pedestrian Street	617	247	725	283
Traffic Street	618	246	726	282
Tram	603	261	711	297
Total files	**6122**	**2518**	**7202**	**2878**

Test subset (2518 audio files).

DCASE 2018 Task 1B reuses all audio files from Task 1A, but extends that with additional recordings obtained from two other recording devices named B and C (e.g. recorded from a variety of smart phones and cameras). However, it should be noted that the number of recordings made by devices B and C is much smaller than that of device A; in total just 4 hours for devices B and C compared to 24 hours for device A.

While performance evaluation for subtask 1A is based on classification accuracy for device A recordings, scoring for subtask 1B is only based on the classification accuracy assessed for devices B and C. It thus tests how well a system, trained mainly with recordings from one device, performs on recordings made on other devices (which is a common real world scenario).

Like DCASE 2018 Task 1A, the subtask 1B dataset is split into test/train portions [43]. Recordings within the development set are separated into a Training subset (7202 audio files) and a Test subset (2878 audio files), shown in two right hand columns of Table 7.

3.4.2 Setting Hyperparametes and Training Process

To train the baseline, cross entropy, defined in Equation (2), is minimised to tune the parameters, denoted as Θ.

$$LOSS_{EN}(\Theta) = -\sum_{c=1}^{C} y_c \log\{\hat{y}_c(\Theta)\} + \frac{\lambda}{2}||\Theta||_2^2, \tag{2}$$

where $LOSS_{EN}(\Theta)$ is the entropy loss function for all parameters Θ of the C-DNN model, λ denotes the ℓ_2-norm regularization coefficient set to 0.001 (The setting of λ value is similar to [58, 20]), C is number of sound scene categories classified, y_c and $\hat{y}_c$ are ground truth and predicted result for class c respectively, in one-hot format.

The C-DNN baseline is built in the Tensorflow framework, set with epoch number, batch size and initial learning rate of 100, 100 and 0.0001 respectively, and using the Adam method for learning rate optimisation [102]. Trainable parameters are initialised randomly with a normal distribution, having mean and variance set to 0 and 0.1, respectively.

3.4.3 Ensemble Method

As the back-end classification models returns the predicted probability of single patch, predicted probability of entire spectrogram is computed by taking the average of all patches' probabilities (this is similar to the *mean-fusion* method mentioned in [20]). If $\mathbf{p^n} = (p_1^n, p_2^n, ..., p_C^n)$, with C being the category number and the n^{th} out of N patches fed into learning model, are considered as the probability of a test sound scene instance, then the average classification probability is denoted as $\bar{\mathbf{p}} = (\bar{p}_1, \bar{p}_2, ..., \bar{p}_C)$ where,

$$\bar{p}_c = \frac{1}{N}\sum_{n=1}^{N} p_c^n \tag{3}$$

and the predicted label from the C-DNN is determined using,

$$\hat{y} = \underset{c \in \{1,2,...,C\}}{\text{argmax}} \; \bar{p}_c \tag{4}$$

As regards an ensemble of channels, spectrograms or patch sizes, this *mean-fusion* method can also be applied in the same way. If $\mathbf{p^{m.n}} = (p_1^{m.n}, p_2^{m.n}, ..., p_C^{m.n})$, with C being the class number, the n^{th} out of N patches fed into learning model, and the m^{th} out of M channels, spectrograms, or patch sizes, are considered as the probability of a test sound scene instance. The mean classification probability is then denoted as $\bar{\mathbf{p}} = (\bar{p}_1, \bar{p}_2, ..., \bar{p}_C)$ where,

$$\bar{p}_c = \frac{1}{M.N} \sum_{m=1}^{M} \sum_{n=1}^{N} p_c^{m.n} \tag{5}$$

and similarly the predicted label is determined as in Equation (4). sddsss

3.5 Experimental Results

To evaluate how low-level features such as channel, patch size, type of spectrogram, and ensembles of these features affect ASC performance, the baseline with using channel 1, patch size of 128×128, and log-Mel spectrogram proposed in Section 3.3 is firstly compared with DCASE baseline. This comparison is described in the next Section 3.5.1. Then, ensembles of channels, sizes of patches, and types of spectrograms are conducted and compared to individual models (i.e. an individual model receives only one channel input, and use one type of patch size and spectrogram), which are shown in Section 3.5.2, 3.5.3, and 3.5.4, respectively. The comparison shows how ensembles help to improve performance, compared with individual model and DCASE baseline. Next, these ensembles are compared together, indicating the most effective low-level feature in Section 3.5.5. Finally, the effect of data augmenation is evaluated in Section 3.5.6.

3.5.1 Baseline Comparison

Table 8: Performance comparison (percentage category classification accuracy - %) between DCASE 2018 Task 1A baseline and the proposed C-DNN baseline.

Categories	**DCASE 2018 baseline (%)**	**Proposed C-DNN baseline (%)**
Airport	72.9	56.2
Bus	62.9	66.1
Metro	51.2	39.1
Metro Station	55.4	67.6
Park	79.1	80.6
Public Square	40.4	64.8
Shopping Mall	49.6	87.8
Street Pedestrian	50.0	46.6
Street Traffic	80.5	79.3
Tram	55.1	72.8
Average	**59.7**	**66.2**

The accuracy of every category reported by the DCASE 2018 Task 1A baseline and by the proposed C-DNN network is displayed in Table 8. In general, the C-DNN baseline improves average accuracy by 6.5%, compared to 59.7% of DCASE 2018, but not every category improves.

In terms of each category performance, *Tram, Shopping Mall, Public Square* and *Metro Station* show significant improvements, increasing by 17.7%, 38.2%, 24.4%, and 12.2%, respectively. Performances on *Airport, Metro*, by contrast, decrease to 56.2% and 39.1%, compared to 72.9% and 51.2% of DCASE 2018. The accuracy of the remaining categories is similar. To further analyse the C-DNN baseline performance, 10 categories are divided into three meta categories, and re-compute the performance on 3 meta categories rather than on 10 categories. Specifically, grouping the categories is as follows - *vehicle (Bus, Metro, Tram), indoor (Airport, Metro Station, Shopping Mall)* and *outdoor (Park, Public Square, Street Pedestrian, Street Traffic).* The classification accuracy over each meta category is 85.8%, 81.5% and 91.5% for *indoor, vehicle* and *outdoor*, respectively, with an averaging of 86.3% over three meta categories. The relatively high performance indicates that environmental sounds show high-cross correlation, with a majority

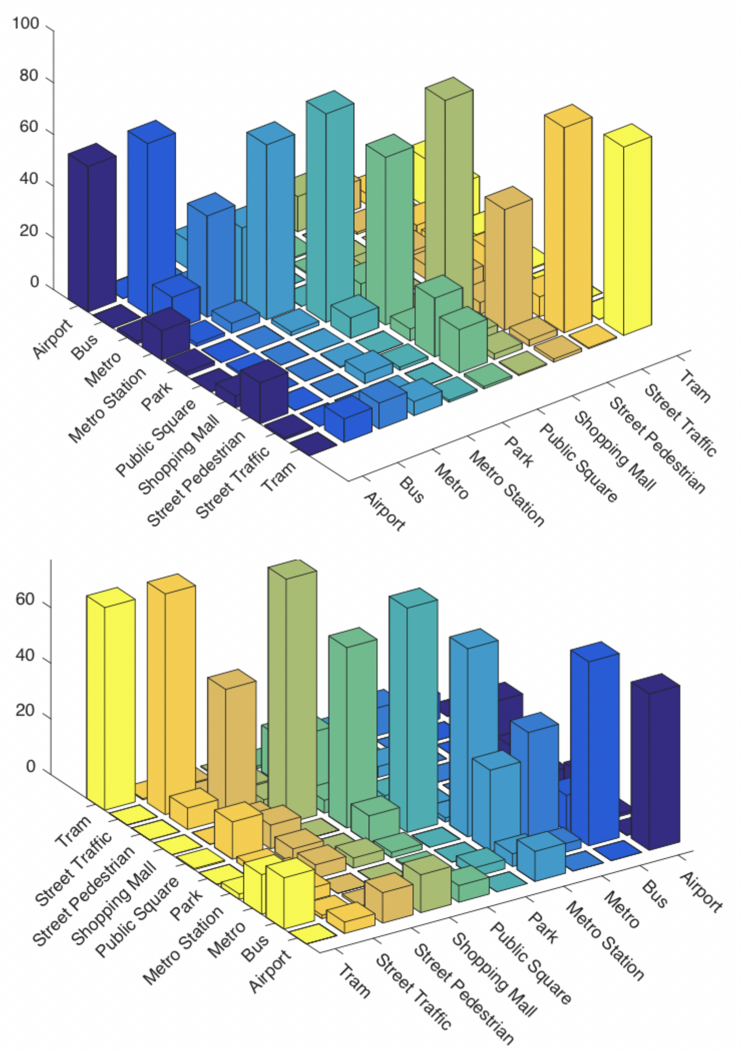

Figure 7: Confusion matrix plot (displayed from two sides) of C-DNN baseline performance per category.

of incorrectly recognized samples dropping into the same meta categories. Indeed, the confusion matrix result of the C-DNN baseline, shown in Figure 7 and Table 9, indicates that incorrect cases in the same meta categories are larger than those across different meta categories. These are in fact 20.7% and 13.1% on average, respectively. This interesting discovery motivates a further comprehensive analysis on cross correlation among both individual and meta categories that will be presented in Chapter 5.

Table 9: Accuracy (Acc. %) and Inaccuracy (Inacc. %) across individual and meta categories.

Categories	**Acc. (%)**	**Inacc. inside meta category (%)**	**Inacc. outside meta category (%)**
Airport	56.2	24.4	19.4
Bus	66.1	21.5	12.4
Metro	39.1	27.0	33.9
Metro Station	67.6	14.9	17.5
Park	80.6	15.0	4.4
Public Square	64.8	29.8	5.4
Shopping Mall	87.8	5.0	7.2
Street Pedestrian	46.6	31.6	21.8
Street Traffic	79.3	19.1	1.6
Tram	72.8	19.1	8.1
Average	**66.2**	**20.7**	**13.1**

3.5.2 Bag-of-channel Ensembles

Working from the proposed C-DNN baseline, the effect on classification accuracy was analysed when using the four different channel arrangements (Note that computing the ensemble of different channels is mentioned in Section 3.4.3). Results are presented in Figure 8 and reveal that Channel 1 (Left), Channel 2 (Right), their Side, and Average differ more widely at the output of the C-DNN baseline with the highest score of 69.3% obtained from the average of the two channels. Ensemble models exploiting different channels help to improve accuracy by 11.9% better than the 59.7% achieved by the DCASE 2018 baseline.

3.5.3 Bag-of-feature-size Ensembles

This section presents an experiment to determine whether ensembles of different patch sizes could improve accuracy as shown in Figure 9 (Note that computing the ensemble of different patch sizes is mentioned in Section 3.4.3). In general, the C-DNN performance results in different sizes are similar and are not significantly improved. Ensemble result over all patch sizes slightly improve – by 2.1% and 8.6% compared to the C-DNN baseline and DCASE 2018, respectively, which shows poorer performance than the channel ensemble.

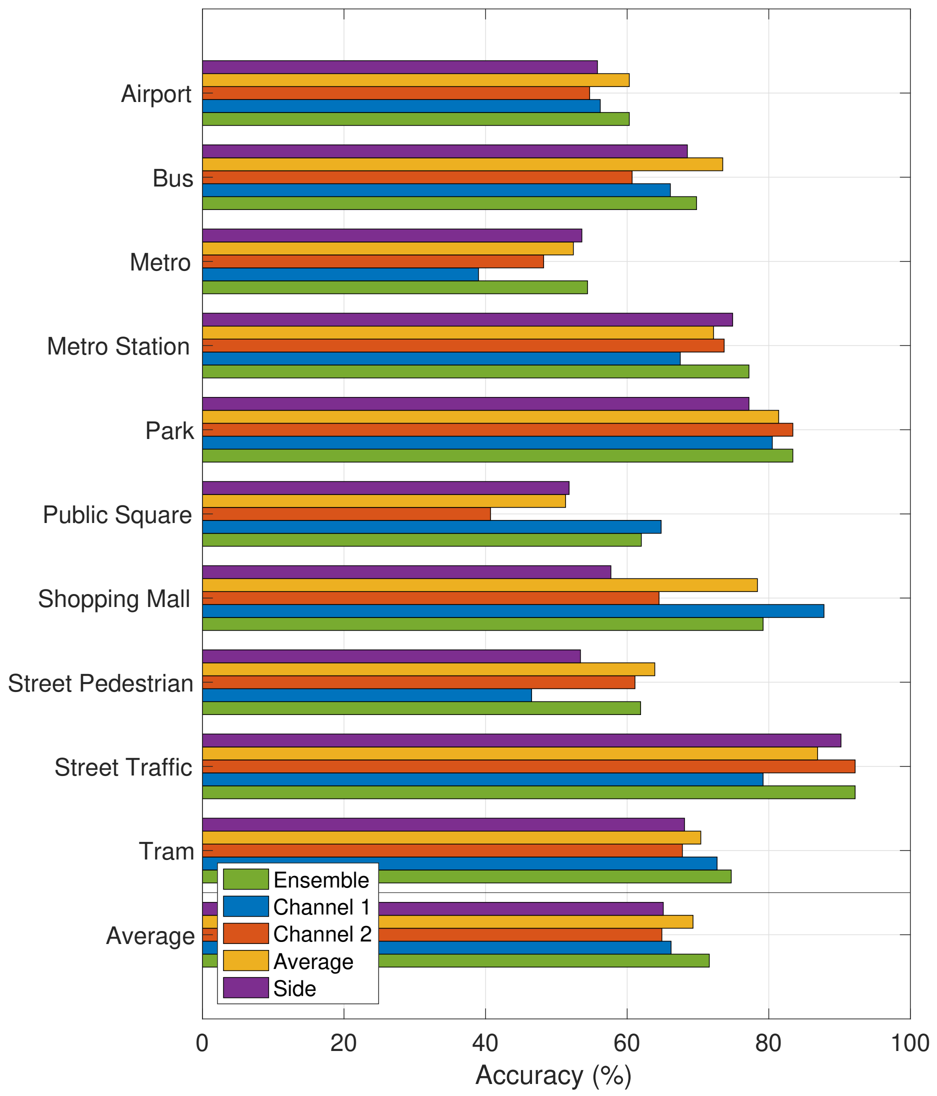

Figure 8: Class-wise performance with channel effect.

3.5.4 Bag-of-spectrogram Ensembles

Next, the effect of using the three spectrogram transformation types listed in Table 4 for the classification of patches sized 128×128 (in the proposed baseline), is evaluated (Note that computing the ensemble of different types of spectrogram is mentioned in Section 3.4.3). The results, shown in Figure 10, indicate that the best C-DNN result among the three spectrogram types is 67.3%, for the Gamma spectrogram. While log-mel results are competitive to Gamma, CQT performance

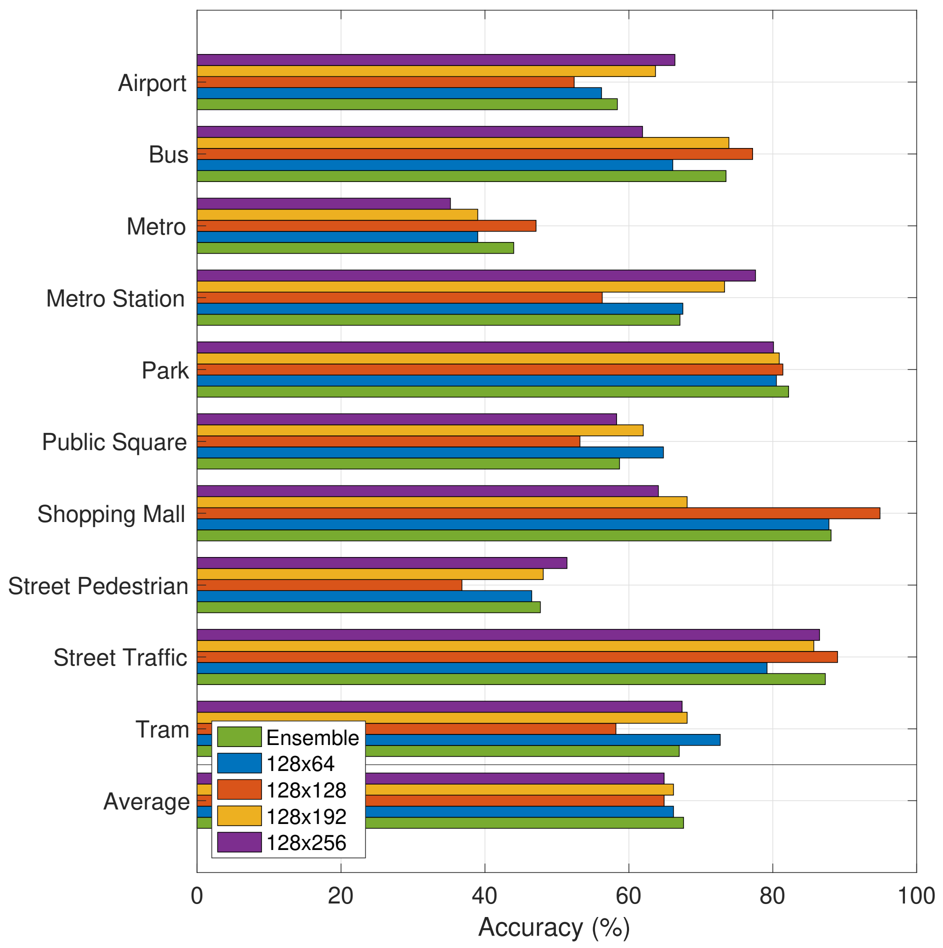

Figure 9: Class-wise performance with patch size effect.

tends to be significantly poorer than the others. However, CQT performs well with *vehicle*-related categories. Noticeably, an ensemble among spectrograms helps to improve the overall accuracy 13.4% more than that of DCASE 2018 baseline, and on average 7.0% better than the highest single-spectrogram performance.

3.5.5 Comparison of bag-of-feature Ensembles

Comparing the performance of bag-of-feature ensembles with DCASE 2018 baseline for Task 1A, as shown in Table 10, the spectrogram ensemble achieves the highest scores, followed by the channel ensemble and then the patch size ensemble.

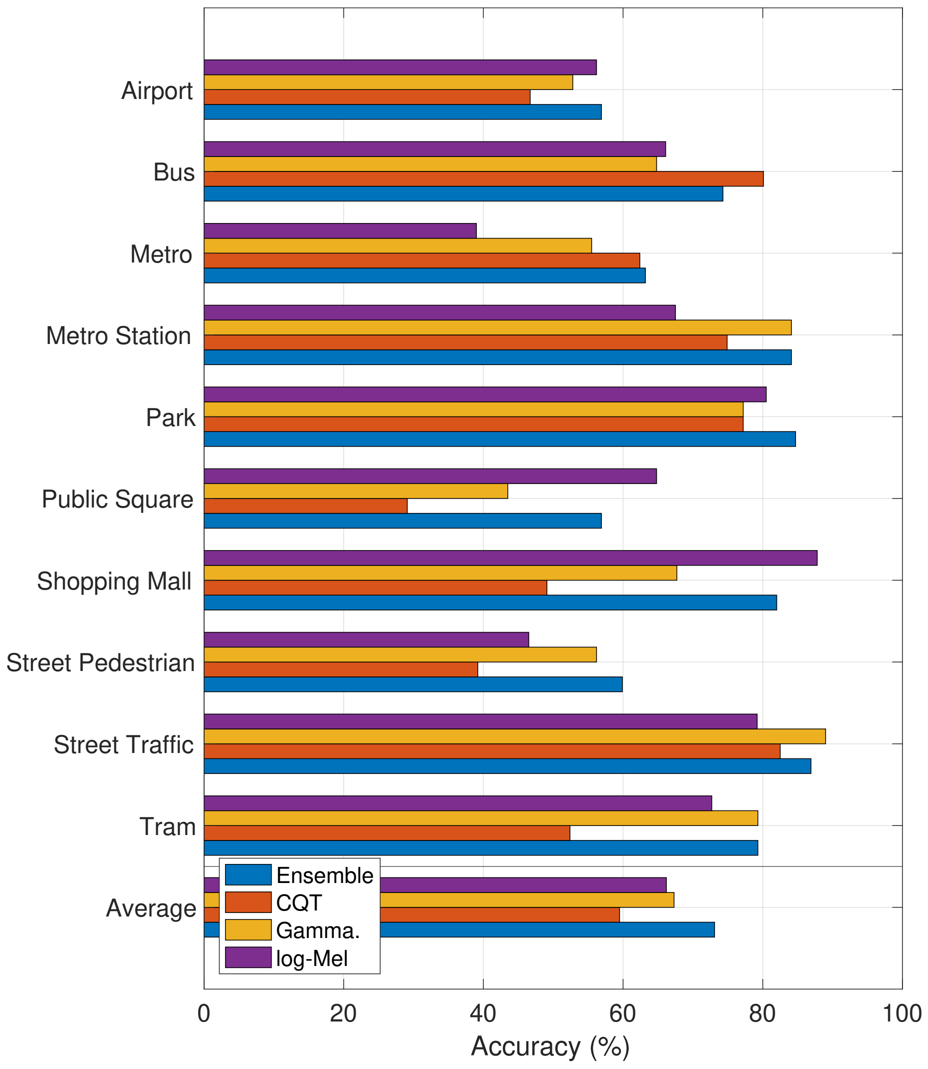

Figure 10: Class-wise performance with spectrogram effect.

Noticeably, ensemble methods are clearly able to improve on the baseline performance for almost all categories, with the exception of *Airport*. Specially, *Tram, Shopping Mall* and *Metro Stations* categories show significant improvements when applying ensembles. For both DCASE 2018 baseline and ensemble methods, *Public Square* achieves the lowest scores. By contrast, *Street Traffic* achieves the highest scores for all of the models evaluated.

Moving on to the DCASE 2018 Task 1B which addresses the issue of mismatched recording devices, it is noted that only one channel is provided and thus

Table 10: Performance comparison (percentage accuracy - %) between DCASE 2018 Task 1A baseline and bag-of-feature ensemble of C-DNN proposed.

Categories	**DCASE 2018 baseline (%)**	**Spectrogram Ensemble (%)**	**Patch Size Ensemble (%)**	**Channel Ensemble (%)**
Airport	72.9	57.0	64.9	60.3
Bus	62.9	74.4	69.8	69.8
Metro	51.2	63.2	42.9	54.4
Metro Station	55.4	84.2	73.3	77.2
Park	79.1	84.7	80.5	83.4
Public Square	40.4	56.9	60.1	62.0
Shopping Mall	49.6	82.1	82.4	79.2
Street Pedestrian	50.0	59.9	51.4	61.9
Street Traffic	80.5	87.0	86.5	92.2
Tram	55.1	79.3	70.1	74.7
Average	**59.7**	**73.1**	**68.3**	**71.6**

Table 11: Performance comparison (percentage accuracy - %) between DCASE 2018 Task 1B baseline with the use of spectrogram ensemble of the C-DNN.

Categories	**DCASE 2018 Dev. A (%)**	**Ens. of Spec. Dev. A (%)**	**DCASE 2018 Dev. B&C (%)**	**Ens. of Spec. Dev. B&C (%)**
Airport	73.4	68.7	72.5	52.8
Bus	56.7	70.7	78.3	88.9
Metro	46.6	70.9	20.6	27.8
Metro Station	52.9	83.4	32.8	61.1
Park	80.8	82.2	59.2	83.3
Public Square	37.9	52.8	24.7	55.6
Shopping Mall	46.4	67.4	61.1	80.6
Street Pedestrian	55.5	64.0	20.8	52.8
Street Traffic	82.5	90.2	66.4	83.3
Tram	56.5	78.9	19.7	27.8
Average	**58.9**	**72.9**	**45.6**	**61.4**

channel ensemble is not possible (even though it performed well for Task 1A). The performance of the spectrogram ensemble method is now explored in DCASE 2018 Task 1B, achieving the results shown in Figure 11 and Table 11.

From results shown in Figure 11, the classification performances on devices B and C are poorer than A due to unbalanced data and mismatch recorded devices, reporting an average of 72.9%, 61.7% and 61.1% for devices A, B, and C, respectively. Compared to the DCASE 2018 baseline, results in Table 11 show a significant improvement, increasing accuracy of B&C by 15.8% (note that DCASE 2018 Task 1B challenge only evaluates accuracy on device B&C).

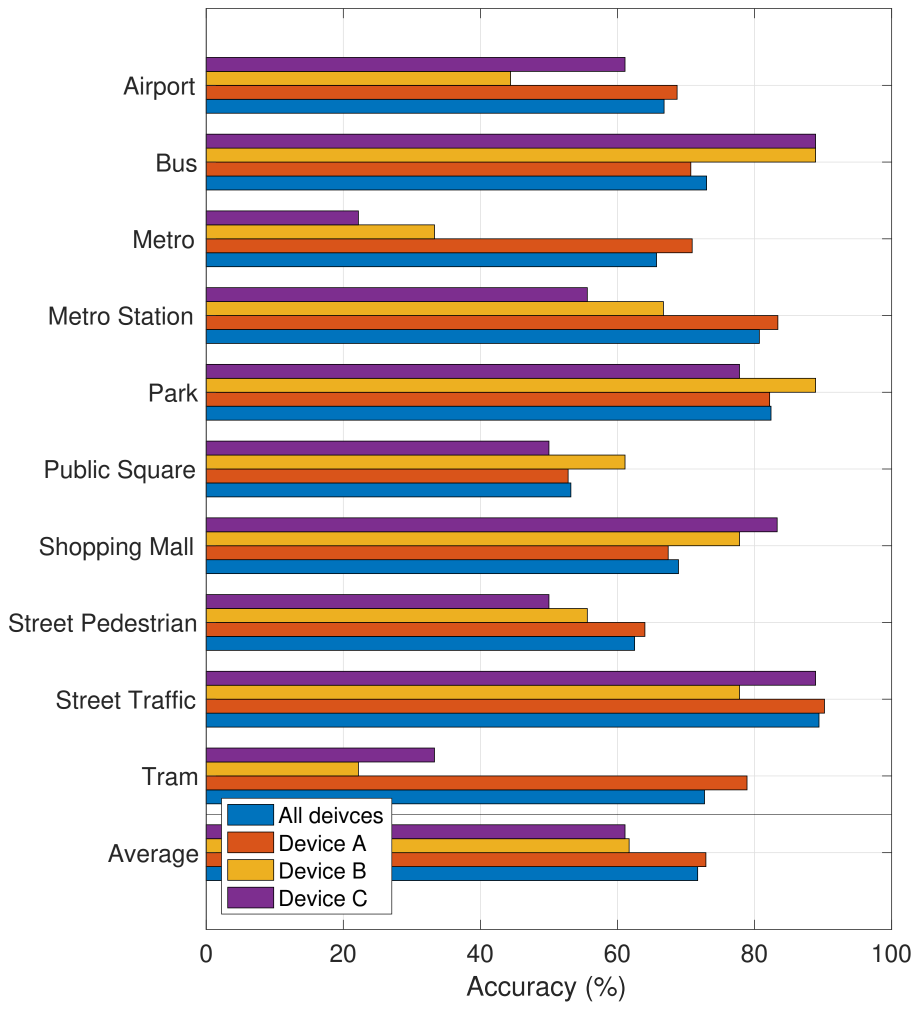

Figure 11: Performance comparison (Accuracy %) among Devices over DCASE 2018 Task 1B with spectrogram ensemble.

3.5.6 Effect of Mixup Data Augmentation

As mentioned in Section 2.4.3, applying multiple input features and data augmentation are two main approaches to deal with ASC challenges in terms of low-level features. The recently comprehensive analysis of bag-of-features has shown that ensemble of spectrograms is effective to improve an ASC system's performance, outperform channel and time resolution features. In this section, the effect of augmentation affects on classification accuracy is evaluated. In particular, Figure

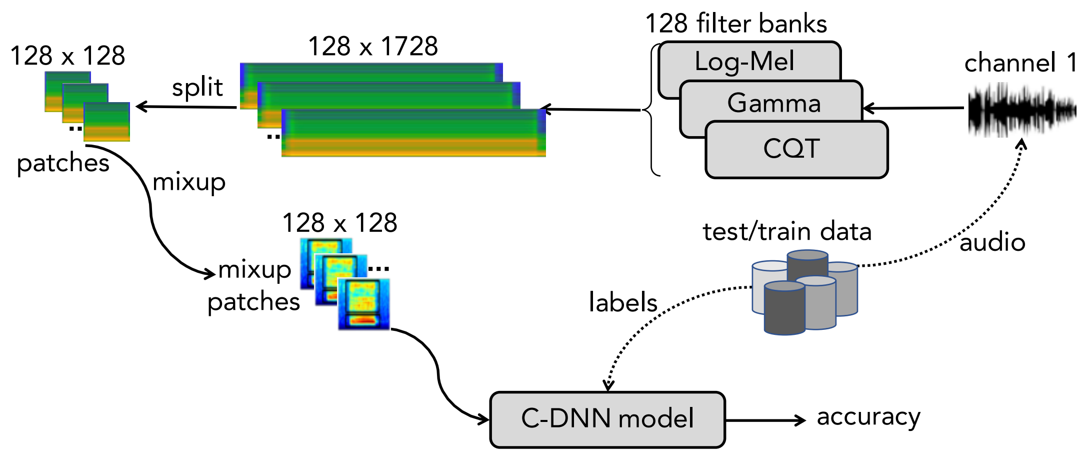

Figure 12: The baseline system architecture with mixup data augmentation.

12 describes how to apply mixup data augmentation in the baseline system proposed. Firstly, only channel 1 is used to transform into three types of spectrogram (log-mel, Gamma, and CQT). The entire spectrograms are thus split into non-overlapping image patches of 128×128. These two steps with setting parameters such as the filter number, window size or hop size, etc. are the same as those used in experiments of bag-of-features mentioned in Table 5. Next, mixup data augmentation [71, 72] is applied on the image patches. Let consider $\mathbf{X}_1$ and $\mathbf{X}_2$ as two image patches randomly selected from the set of original image patches with their labels $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively, mixup data augmentation helps to generate new image patches as Equations below,

$$\mathbf{X}_{\mathrm{mp1}} = \alpha\mathbf{X}_1 + (1-\alpha)\mathbf{X}_2, \tag{6}$$

$$\mathbf{X}_{\mathrm{mp2}} = (1-\alpha)\mathbf{X}_1 + \alpha\mathbf{X}_2, \tag{7}$$

$$\mathbf{y}_{\mathrm{mp1}} = \alpha\mathbf{y}_1 + (1-\alpha)\mathbf{y}_2, \tag{8}$$

$$\mathbf{y}_{\mathrm{mp2}} = (1-\alpha)\mathbf{y}_1 + \alpha\mathbf{y}_2. \tag{9}$$

where α is drawn from both Uniform or Beta Distribution, $\mathbf{X}_{\mathrm{mp1}}$ and $\mathbf{X}_{\mathrm{mp2}}$ are two new image patches resulted by mixing $\mathbf{X}_1$ and $\mathbf{X}_2$ with a random mixing coefficient

Table 12: Effect of mixup data augmentation on individual and ensemble spectrograms, evaluated on C-DNN baseline and DCASE 2018 Task 1A and 1B dataset (only devices B & C in DCASE 2018 Task 1B dataset is reported).

C-DNN	**Task 1A (%)** (w/o mixup)	**Task 1A (%)** (w/ mixup)	**Task 1B (%)** (w/o mixup)	**Task 1B (%)** (w/ mixup)
Gamma	67.3	68.3	55.5	58.9
log-mel	66.2	67.8	56.4	59.4
CQT	59.5	60.2	51.7	51.4
Ensemble	**73.1**	**74.0**	**61.4**	**66.9**

Table 13: Performance comparison of the proposed system (multiple-spectrogram low-level features, mixup data augmentation, C-DNN ensemble) to top-ten DCASE 2018 challenge.

DCASAE 2018 1A	**Acc. (%)**	**DCASE 2018 1B**	**Acc. (%)**
Li [103]	72.9	Baseline [104]	45.6
Jung [66]	73.5	Li [105]	51.7
Hao [106]	73.6	Tchorz [107]	53.9
Christian [108]	74.7	Kong [109]	57.5
Zhang [110]	75.3	Wang [111]	57.5
Li [112]	76.6	Waldekar [113]	57.8
Dang [114]	76.7	Zhao [23]	58.3
Octave [14]	78.4	Truc [16]	63.6
Yang [87]	79.8		
Golubkov [115]	**80.1**		
Proposed system	74.0	Proposed system	**66.9**

α. Similarly, $\mathbf{y}_{\text{mp1}}$ and $\mathbf{y}_{\text{mp2}}$ are two new labels resulted by mixing $\mathbf{y}_1$ and $\mathbf{y}_2$. After mixup, old data and generated data from mixup data augmentation are shuffled and fed into C-DNN baseline proposed, double batch size and consider learning time. Because of using multiple-spectrogram input features (log-mel, Gamma, and CQT), effect of mixup data augmentation on individual and ensemble spectrogram is evaluated.

By applying mixup data augmentation technique, the new labels $\mathbf{y}_{\text{mp1}}$ and $\mathbf{y}_{\text{mp2}}$ of the two mixup patches are no longer one-hot labels, Kullback-Leibler (KL) divergence loss [116] rather than the standard cross-entropy loss is used as shown in Equation below,

$$LOSS_{KL}(\Theta) = \sum_{c=1}^{C} y_c \log\left(\frac{y_c}{\hat{y}_c(\Theta)}\right) + \frac{\lambda}{2}||\Theta||_2^2, \tag{10}$$

where $LOSS_{KL}(\Theta)$ is KL loss function, Θ denotes the trainable network parameters and λ denotes the ℓ_2-norm regularization coefficient, set to 0.001. y_c and $\hat{y}_c$ denote the ground truth and the network output at category c, respectively. Other hyper parameters are same as those mentioned in Section 3.4. Obtained experimental results conducted on DCASE 2018 Task 1A and 1B datasets are shown in Table 12. It can be seen that performance of all experimental systems is improved on both datasets. In particular, mixup data augmentation helps to improve by 0.9% with ensemble of spectrograms on DCASE 2018 Task 1A. Noticeably, this technique is very effective for DCASE 2018 Task 1B when it shows an improvement of 5.5%.

Compare the best results obtained (74.0% and 66.9% for DCASE 2018 Task 1A and 1B, respectively) to the top-ten DCASE 2018 challenge as shown in Table 13, it can be seen that C-DNN baseline model with multi-spectrogram input and using mixup data augmentation stands on the top-eight position as regards DCASE 2018 Task 1A, and outperforms DCASE 2018 Task 1B challenge.

3.6 Conclusion

From these experimental results obtained from the DCASE 2018 Task 1A and 1B datasets, there is a clear indication that using different spectrograms, coming from different auditory models, is effectively to improve classification accuracy. This improvement can also be achieved on different ASC tasks (assessed by comparing the classification performance on matched and mismatched recording devices). Furthermore, applying mixup data augmentation on image patches is effective to enforce learning ability of the back-end learning model, thus improve the accuracy.

Chapter 4

A novel Encoder-Decoder Framework

Comprehensive analysis provided in Chapter 3 indicates that the combination of using three spectrograms of log-mel, gammatonegram (Gamma) and CQT as low-level features, along with mixup data augmentation, is effective at improving the performance of an ASC system. However, the results from deep learning models used in Chapter 3 show some issues of concern. Firstly, although an ensemble of multi-spectrogram input is useful to enhance ASC system performance, this method fuses the predicted probability of single models learned from individual spectrograms, but does not explore the interrelation between those spectrograms. Secondly, although both single models using individual spectrograms and multi-spectrogram ensemble models were proposed in Chapter 3, and these outperform DCASE 2018 baselines, their performance is not competitive with the most recent, state-of-the-art systems. Furthermore, because the ASC systems proposed in Chapter 3 were only evaluated on two datasets (DCASE 2018 Task 1A and 1B), there is still not enough evidence to conclude whether the proposed learning models perform well or not in general, or whether they only work on a restricted task.

Motivated by these issues, this chapter aims to improve the back-end classification process, proposes a novel deep learning model called the *Encoder-Decoder* framework, then evaluates it over a wide variety of published datasets including Litis Rouen [47], DCASE 2016 Task 1 [45], DCASE 2017 Task 1 [44], DCASE 2018 Task 1A [43], 1B, and DCASE 2019 Task 1A, 1B [42] – all of which will be summarised below.

Conducted experiments below obtain very good results; competitive with the best single-task systems, and far better than any previously published multi-task methods.

4.1 High-level Architecture

An overall ASC system using the proposed *Encoder-Decoder* framework is illustrated as Figure 13. As the comprehensive analysis of various low-level features in Chapter 3 identified the best settings to deal with ASC challenges, the *Encoder-Decoder* framework only uses one channel 1 and one patch size based on that, but applies it to three spectrograms (log-mel, Gamma, and CQT) for low-level feature extraction. Firstly, the recorded audio signal from one channel (Channel 1 - Left) is transformed into three types of spectrogram (log-mel, Gamma, and CQT) with 128 filters each. Next, the entire spectrograms are sliced into non-overlapping patches of 128×128 before applying mixup data augmentation [71, 72] as mentioned in Section 3.5.6. Patches after mixup are then fed into the *Encoder* model to start the first training process. This training process helps to map low-level features to high-level features which are vectors containing discriminative and condensed multi-dimensional information. In other words, the role of the *Encoder* is to be a high-level feature extractor. Next, the high-level features are extracted patch-by-patch, mixup data augmentation is again applied, and the resulting augmented dataset is used to train the *Decoder* model. The *Decoder* model

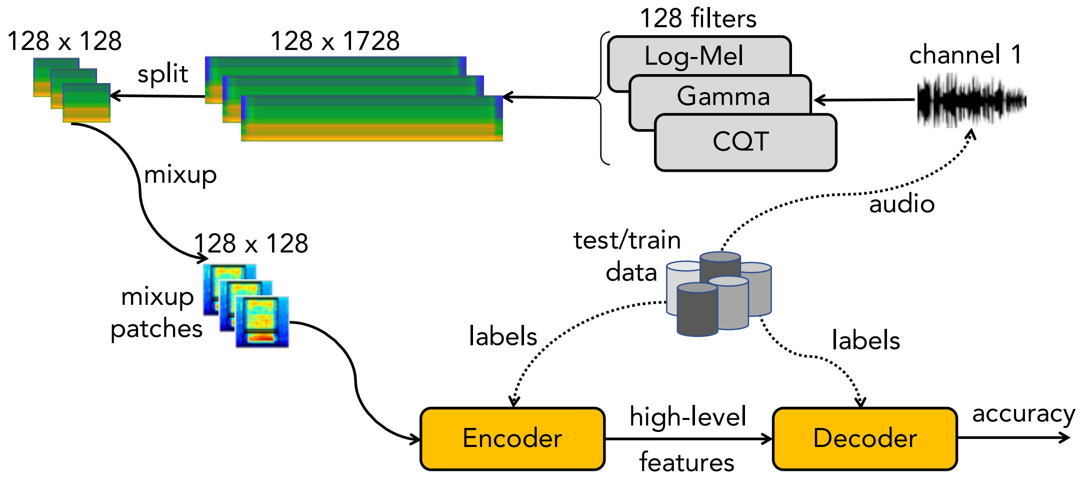

Figure 13: High-level architecture of an ASC system using the proposed Encoder-Decoder framework.

has the responsibility to perform final classification, and reports the classification accuracy at its output.

4.2 *Encoder-Decoder* Network Configuration

4.2.1 *Encoder* as High-level Feature Extractor

The architecture of the *Encoder* network, performing high-level feature extraction, is shown in Figure 14. Three types of image patches of size 128×128 pixels (i.e. three types of image patches from CQT, Gamma, and log-Mel repectively), after mixup, are fed into the three parallel networks each of which comprises a CNN and a DNN-01 block, like the VGG-7 architecture [84]. Subscripts LM, GA, and CQ are used to denote the three paths, as shown in Figure 14, referring to the kind of spectrogram of log-Mel, Gamma, and CQT, respectively (e.g. CNN_{LM} and $\mathrm{DNN}-01_{LM}$ blogs in Figure 14 are used for learning log-Mel input only).

The architecture of the $\mathrm{CNN}_{LM/GA/CQ}$ and $\mathrm{DNN}-01_{LM/GA/CQ}$ blocks are described in the upper and middle sections of Table 14, resepective. As regards the

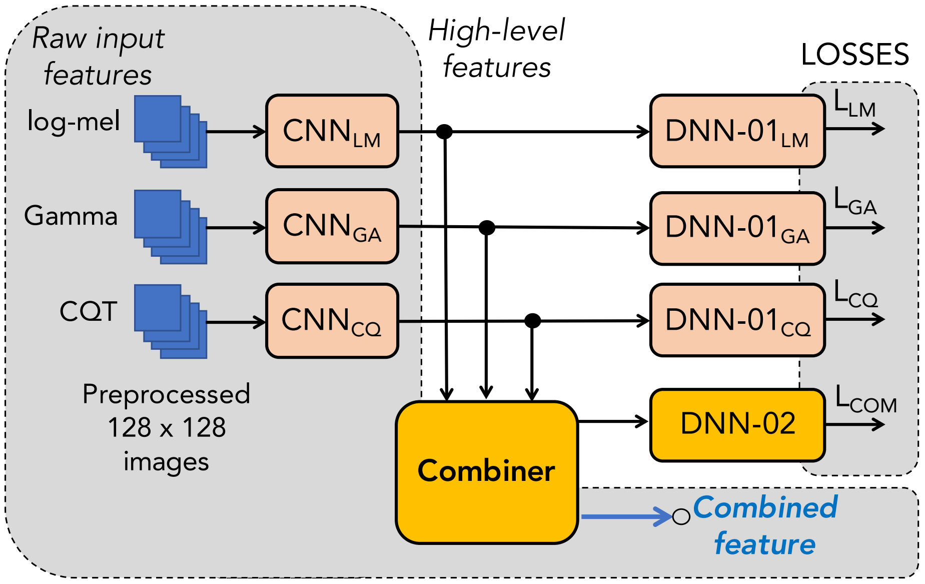

Figure 14: High-level feature extraction from the Encoder network.

$\mathrm{CNN}_{LM/GA/CQ}$, they share the same architecture, which comprises six layers employing sub-blocks of batch normalization (BN), convolutional (Cv [kenel size] @ kernel number), rectified linear units (ReLU), average pooling (AP [kernel size]), global average pooling (GAP), dropout (Dr(%)). The $\mathrm{DNN}-01_{LM/GA/CQ}$ blocks also share the same architecture, which performs fully connected (FC), and Softmax layers, with dimensions given in Table 14. The number of categories within the given dataset is denoted by "C"; this depends on the particular evaluation task.

The three parallel networks, each of which is configured to contain a $\mathrm{CNN}_{LM/GA/CQ}$ and $\mathrm{DNN}-01_{LM/GA/CQ}$, are used to learn and extract high-level features from one type of spectrogram for each. While the structures of these three $\mathrm{CNN}_{LM/GA/CQ}$ as well as the three $\mathrm{DNN}-01_{LM/GA/CQ}$ blocks are identical, they will contain very different weights (trainable parameters) after training due to their different spectrogram input.

The output of each of the $\mathrm{CNN}_{LM/GA/CQ}$ block shown in the upper part of Table 14 is a 256-dimensional vector. The vector extracted from each individual

Table 14: Encoder network structures of the CNN (top), DNN-01 (middle) and DNN-02 (bottom).

Encoder network architecture	**Output**
CNN$_{LM/GA/CQ}$ shares the same architecture	
Input layer (image patch)	128×128
BN - Cv [3×3] @32 - ReLU - BN - AP [2×2] - Dr (10%)	64×64×32
BN - Cv [3×3] @64 - ReLU - BN - AP [2×2] - Dr (15%)	32×32×64
BN - Cv [3×3] @128 - ReLU - BN - Dr (20%)	32×32×128
BN - Cv [3×3] @128 - ReLU - BN - AP [2×2] - Dr (20%)	16×16×128
BN - Cv [3×3] @256 - ReLU - BN - Dr (25%)	16×16×256
BN - Cv [3×3] @256 - ReLU - BN - GAP - Dr (25%)	256
DNN−01$_{LM/GA/CQ}$ shares the same architecture	
Input layer (vector)	256
FC - Softmax	C
DNN-02	
Input layer (vector)	256
FC - ReLU - Dr (30%)	512
FC - ReLU - Dr (30%)	1024
FC - Softmax	C

spectrogram is referred as to a high-level feature. To combine the three high-level features, which are 256-dimensional vectors independently extracted from three parralell network streams, into a single combined feature, the "Combiner" block, as shown in Figure 14, is proposed. There are three methods to combine the high-level features, which are evaluated. The vector outputs of the CNN blocks are denoted as $\mathbf{x_{LM/GA/CQ}}\ [x_1, x_2, ..., x_{256}]$. The first combination method, called "sum-comb", is the unweighted sum of the three vectors. i.e. the individual vectors contribute equally to the combined high-level feature,

$$\mathbf{x_{sum-comb}} = \mathbf{x_{LM}} + \mathbf{x_{GA}} + \mathbf{x_{CQ}} \tag{11}$$

The second method, which is called "max-comb", obtains $\mathbf{x_{max-comb}}[x_1, x_2, ..., x_{256}]$ by selecting the element-wise maximum of the three vectors across the dimensions as in Equation (12). The motivation is to pick the most important (highest magnitude) feature from among the three high level feature vectors,

$$\mathbf{x_{max-comb}}[x_i] = max(\mathbf{x_{LM}}[x_i], \mathbf{x_{GA}}[x_i], \mathbf{x_{CQ}}[x_i]) \quad for \quad 1 \leq i \leq 256 \tag{12}$$

For the final method, it is assumed that elements of three vectors have a linear relationship across dimensions. Then, a simple data-driven combination method called "lin-comb" is proposed by employing a fully connected layer trained to weight and combine the three high level features, as in

$$\mathbf{x_{lin-comb}} = Relu\left\{\mathbf{x_{LM}}\mathbf{w_{LM}} + \mathbf{x_{GA}}\mathbf{w_{GA}} + \mathbf{x_{CQ}}\mathbf{w_{CQ}} + \mathbf{w_{bias}}\right\} \tag{13}$$

where $\mathbf{w_{LM/GA/CQ/bias}}[w_1, w_2, ..., w_{256}]$ are the trained parameters. The combined high level feature vector from the output of the "Combiner" block is then fed into DNN-02, with the structure shown in the lower part of Table 14. Note that the combined high level feature vectors, like the individual high level vectors, have a dimension of 256 – meaning that the higher layer classifier of the decoder can be set for evaluation with either individual or combined feature input, without changing its structure or complexity.

Regarding training loss, four loss functions to train the encoder network are defined; three to optimize individual spectrograms, and the final one for their combination. Eventually, the overall loss function $LOSSES$ is computed as

$$LOSSES = \alpha(L_{LM} + L_{GA} + L_{CQ}) + \beta L_{com} \tag{14}$$

and L_{LM}, L_{GA}, L_{CQ} and L_{com} are individual losses from the log-mel, Gamma and CQT spectrograms, and their combinations. These are depicted from Figure 14 and will be defined in Section 3.2. The balancing parameters α and β focus on learning particular features or combinations and are set to 1/3 and 1.0 here, making the contributions from each spectrogram equal.

After training the *Encoder* network, the combined feature (i.e. the combined feature is also the input of DNN-02 block in Figure 14) is extracted. Then, it is fed into the *Decoder* described below for classification.

4.2.2 *Decoders* for Back-end Classification

Table 15: MLP-based architecture of Decoder network.

Network architecture	**Output**
Input layer (vector)	256
FC - ReLU - Dr (30%)	512
FC - ReLU - Dr (30%)	1024
FC - ReLU - Dr (30%)	1024
FC - Softmax	C

As regards the baseline architecture proposed in Chapter 3, fully connected layers with the final Softmax layer takes the role of classification. In the *Encoder-Decoder* framework proposed in this chapter, the *Decoder* is responsible for this role and receives combined high-level feature vectors extracted from *Encoder* as its input (note that mixup data augmentation is applied on these feature before feeding into *Decoder* network during training). There are three types of *Decoder* evaluated: A Random Forest Classification (RFC) with classifier, a Multilayer Perceptron (MLP), and a Mixture of Experts (MoE), described below,

a) Random Forest Classification (RFC *Decoder*): A regression forest [117] is a type of ensemble model, comprising multiple regression trees. The role of each tree is to map the complex input space defined by the high level features from the encoder network, into a continuous class-dimension output space. Its nonlinear mapping is achieved by dividing the large original input space into smaller sub-distributions. Individual trees are trained using a subset randomly drawn from the original training set. By using many trees (e.g. 100), the structure is effective at tackling overfitting issues that can occur with single trees. Additionally, the regressor structure benefits from the continuous mixed-class training labels provided by employing mixup. Eventually, the decoded output spaces are classified as in our previous work [91] by average pooling the output over all trees.

b) Multilayer Perceptron Network (MLP *Decoder*): The proposed MLP *Decoder* comprises four fully connected dense blocks as shown in Table 15. Comparing the MLP *Decoder* architecture to DNN-02 block used in *Encoder*,

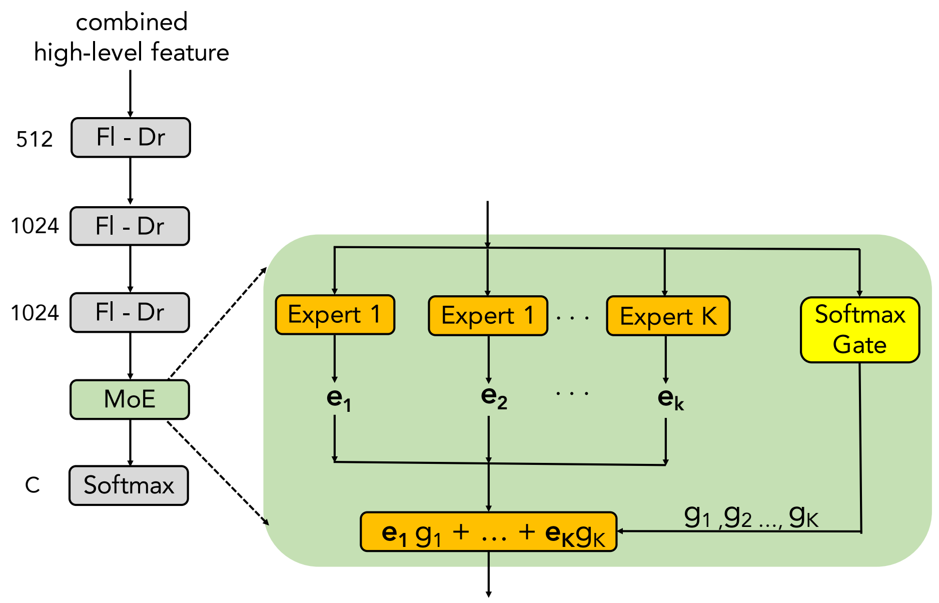

Figure 15: Proposed mixture of experts (MoE) as back-end decoder network.

one more fully connected layer with 1024 nodes is added – this is to handle the additional complexity of the input information.

c) Mixture of Experts (MoE *Decoder*): MoE is a machine learning technique that divides the problem spaces into homogeneous regions by using an array of different trained (but in this case identical structure) models, referred to as experts [118]. A conventional MoE architecture comprises many experts and incorporates a gate network to decide which expert is applied in which input region. The MoE technique is used to classify the combined high-level features, as shown in Figure 15. Specifically, the 256-dimensional input vector goes through three dense layers with dropout, having 512, 1024, and 1024 hidden nodes, respectively, matching MLP *Decoder* in the number of hidden units. The output enters the MoE layer, which is explained in Figure 15. The combined result from the experts is gated before passing through a Softmax layer to determine the final C class scores. Each MoE expert comprises a dense block with a Relu activation function. Its input dimension is 1024 and its output size is C. The gate network is implemented as a Softmax gate – an additional fully connected layer with Softmax activation function and a gating dimension equal to the number of experts.

If $\mathbf{e_1}, \mathbf{e_2}, \ldots \mathbf{e_K} \in \mathbb{R}^{\mathbb{C}}$ is considered as the output vectors of the K experts, and $g_1, g_2, \ldots, g_K$ as the outputs of the gate network where $g_k \in [0,1], \sum_{k=1}^{K} g_k = 1$ The predicted output is then defined as,

$$\hat{\mathbf{y}} = softmax\left\{\sum_{k=1}^{K} \mathbf{e_k} g_k\right\}. \tag{15}$$

4.3 Experiment Setup

4.3.1 Dataset

To clearly demonstrate the general performance of the proposed systems, five different ASC tasks are used for the evaluation. While it is relatively easy to perform well in one challenge, it is considerably more difficult to do so for all – this helps to explore one of the hypothesised strengths of this proposed combined-spectrogram approach, that it can be more generic. Four of the datasets used are derived from annual DCASE challenges (DCASE 2016 Task 1, DCASE 2017 Task 1, DCASE 2018 Task 1A and 1B, DCASE 2019 Task 1A and 1B), whereas the fifth is the extensive LITIS Rouen dataset. Each is described below.

DCASE 2016 Task 1A and DCASE 2017 Task 1A: Firstly, DCASE 2016 Task 1 dataset [45] as shown in Table 16 were recorded at a sample frequency at 44100 Hz with a 30-second recording duration for every audio file. The data is subdivided into two sets; a development set (Dev. Set) and an evaluation set (Eva. Set), one for training and another for evaluating, with 15 categories as described in detailed in Table 16. In total, the development and evaluation sets comprise 13 hours of data. As regards DCASE 2017 Task 1 dataset as shown in Table 16 [44], it reuses all DCASE 2016 dataset. In particular, each 30-second segment from the DCASE 2016 dataset was split into three 10-second segments used in DCASE 2017 dataset. Besides, more 10-second audio segments were recorded and

Table 16: Development and Evaluation Sets of DCASE 2016 Task 1 and DCASE 2017 Task 1 Datasets.

Categories	**DCASE 2016 Dev. Set**	**DCASE 2016 Eva. Set**	**DCASE 2017 Dev. Set**	**DCASE 2017 Eva. Set**
Beach	78	26	312	108
Bus	78	26	312	108
Cafe/Restaurant	78	26	312	108
Car	78	26	312	108
City center	78	26	312	108
Forest Path	78	26	312	108
Grocery Store	78	26	312	108
Home	78	26	312	108
Library	78	26	312	108
Metro station	78	26	312	108
Office	78	26	312	108
Park	78	26	312	108
Residential area	78	26	312	108
Train	78	26	312	108
Tram	78	26	312	108

included, which create a total of 17.5 hours for both development and evaluation sets. Similar to DCASE 2016 settings, while the development set (Dev. Set) is used to train the model, the evaluation set (Eva. Set) is for evaluating. Both DCASE 2016 and DCASE 2017 contain balanced data, and each challenge has 15 categories.

DCASE 2018 Task 1A, 1B and DCASE 2019 Task 1A, 1B: As DCASE 2018 Task 1A and 1B challenges [43] have not released their evaluation sets, only the development sets are explored in this Chapter. Description and setting evaluation for DCASE 2018 Task 1A and 1B development set are similar and were introduced in previously Chapter 3. Regarding DCASE 2019 Task 1A and 1B datasets [42], these reuses DCASE 2018 Task 1A and Task 1B data, but incorporates additional audio segments. Therefore, the recording files in DCASE 2018 and DCASE 2019 challenges have similar formats as well as the same number of categories, as shown in Table 17. The proposed *Encoder-Decoder* framework described above was submitted to compete in the DCASE 2019 challenge, so this

Table 17: Development and Evaluation set of DCASE 2019 Task 1A and 1B datasets.

Categories	**Dev. Set (1A)**	**Eva. Set (1A)**	**Dev. Set (1B)**	**Eva. Set (1B)**
Airport	911	421	1019	529
Bus	928	415	1036	523
Metro	902	433	1010	541
Metro Station	897	435	1005	543
Park	946	386	1054	494
Public Station	945	387	1053	495
Shopping Mall	896	441	1004	549
Pedestrian Street	924	429	1032	537
Traffic Street	942	402	1050	510
Tram	894	436	1002	544
Total files	**9185**	**4185**	**10265**	**5265**

thesis also reports the results over the evaluation set via the DCASE 2019 competition, even through this evaluation dataset has not been released publicly yet.

Litis-Rouen dataset: This extensive dataset [47], as shown in Table 18, comprises 19 urban scene classes with 3026 segments, divided into 20 training/testing splits. The audio was recorded at a sample rate of 22050 Hz, with each segment duration of 30 seconds. Following the mandated settings, the dataset is separated and organised for 20 times cross validation, reporting the final classification accuracy by averaging over the 20 testing folds.

4.3.2 Setting Hyperparametes and Training Process

The Tensorflow framework is used, and the Kullback-Leibler (KL) divergence loss [116] as in Equation (16) is applied to all of the proposed networks. This is a common loss function for training ASC systems, typically obtaining good performance.

$$LOSS_{KL}(\Theta) = \sum_{c=1}^{C} y_c \log \left\{ \frac{y_c}{\hat{y}_c(\Theta)} \right\} + \frac{\lambda}{2}||\Theta||_2^2, \tag{16}$$

where $LOSS_{KL}(\Theta)$ is the KL loss function, Θ denotes the trainable network parameters and λ denote the ℓ_2-norm regularization coefficient, set to 0.001. C is

Table 18: Litis-Rouen Dataset.

Categories	**Segment No.**
Plane	192
Busy Street	143
Café	120
Car	243
Hall Gate	269
Kid Game	145
Market	276
Metro Pari	139
Metro Rouen	249
Pedestriant Street	122
Plane	23
Pool	155
Quite Street	90
Restaurant	133
Shop	203
Student Hall	88
Train High Speed	147
Train Normal	164
Tube	125
Total files	**3026**

the class number. y_c and $\hat{y}_c$ denote the ground truth and network output at class c, respectively. Experiments use the Adam optimiser [102] to adjust learning rate, with a batch size of 50. Results were obtained after 100 epochs (in practice only a small degree of performance was lost by not continuing beyond this, but it helped significantly to reduce the duration of experiments). Trainable parameters are initialised by a Normal Distribution with mean and variance set to 0 and 0.1, respectively. As aforementioned, mixup data augmentation is applied to enhance the training processes. As regards the training process on the *Encoder*, each of the raw 128×128 dimensional feature was repeated twice by including same-dimension Beta and Uniform Distribution mixup images of the same dimension. It is similar when training the *Decoders* where mixup is applied on the high-level feature vectors prior to the final classifier. In each case, both original and generated mixup data are used in the training processes to improve performance, at the cost of increasing the training time.

4.4 Experimental Results and Comparison

In this section the performance of the *Encoder* network is firstly analysed to specifically understand the contribution made by different spectrogram types, as well as their combinations. The performance of the decoder, thus, is evaluated to assess different back-end classifiers, then the overall performance is compared to a range of state-of-the art methods.

4.4.1 Performance of Each Spectrogram by Class

Firstly, a baseline architecture is proposed and evaluated to determine how different spectrogram types contributed to the performance of different classes. To do this, three C-DNN *Encoder* networks, comprising CNN and DNN-02 blocks, each *Encoder* for an individual spectrogram input, are trained. Meanwhile, another C-DNN encoder network, the entire network as in Figure 14 for spectrogram combination, is also trained. These four trained systems are subsequently used as high-level feature extractors to train the *Decoder* and then to test the overall system. Four different *Encoders*, using the MLP *Decoder* architecture from Section 4.2.2 to assess individual spectrogram performance are used to combine with four *Encoders*. These extensive experiments were conducted using the DCASE 2018 Task 1B Dev. set. To compare performance, class-wise accuracies for the three spectrograms and their combinations are shown in Figure 16, with overall average performance shown at the bottom. Clearly, the combined features performed best overall, with the log-mel and Gamma performing similarly, and both being better than CQT. However, a glance at the per-class accuracy shows some interesting variation. For example, the CQT spectrogram was particularly good at discriminating the *Bus* and *Metro* classes, compared to the other spectrograms. Also, while log-mel and Gamma performances were similar, the former excelled on *Airport* and *Public Square* classes, whereas the latter tended to be slightly

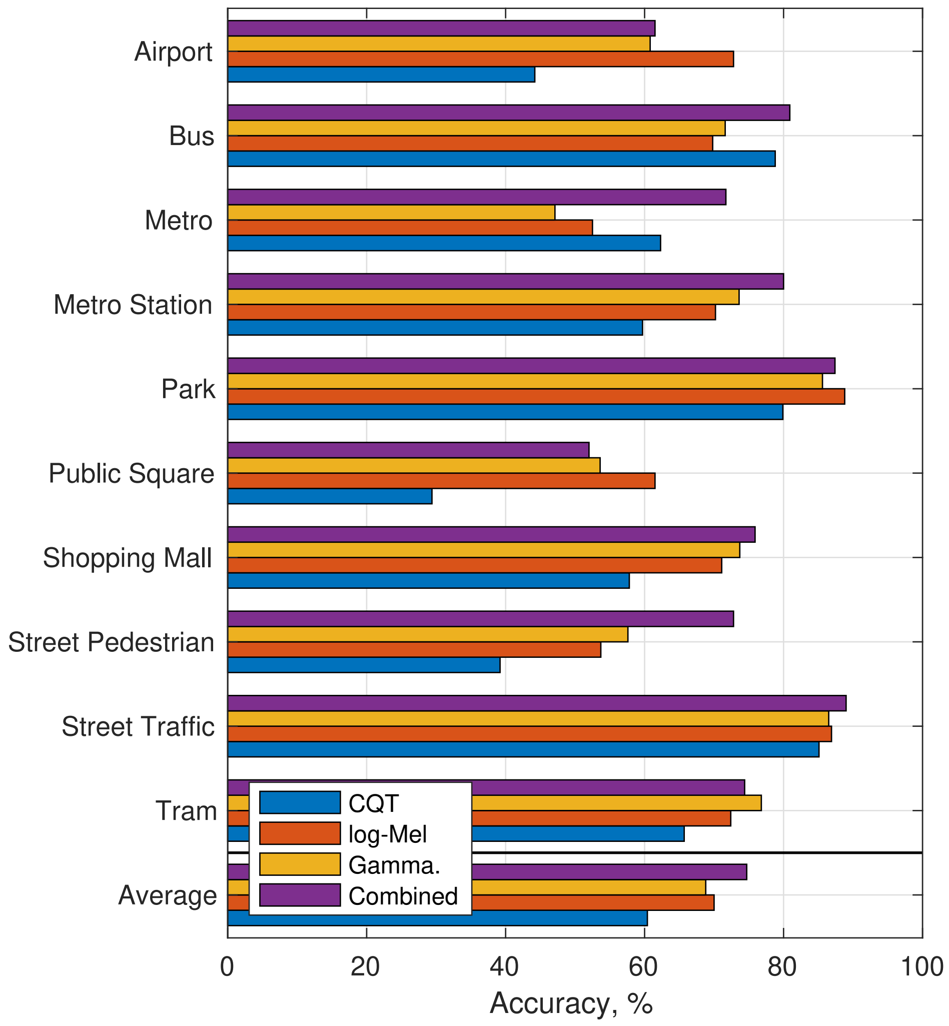

Figure 16: Performance comparison of different spectrograms types, and their combination, for the DCASE 2018 Task 1B Dev. set.

better for classes containing vehicular sounds (with the exception of the *Metro* class). It can be concluded that the three spectrograms represent sounds in ways that have affinity for certain types of sounds (mirroring a conclusion in [119], albeit on very different types of sound data). It is therefore unsurprising that intelligently combining the three spectrograms into a high-level feature vector can achieve significant performance gaining over single spectrograms.

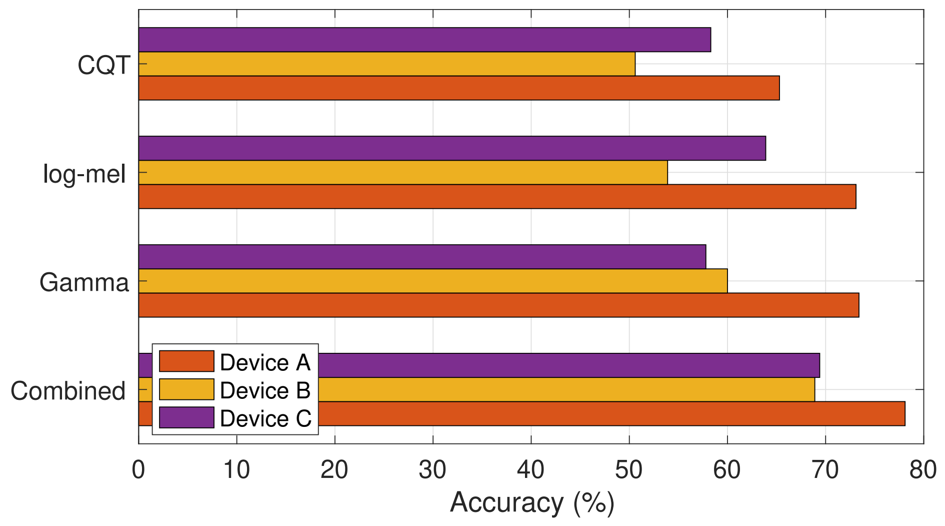

Figure 17: Performance comparison for different recording devices within the DCASE 2018 Task 1B Dev. set.

4.4.2 Spectrogram Performance for Each Device

DCASE 2018 Task 1B includes highly unbalanced data recordings from three different devices as described in Section 3.4.1. The performance of different spectrograms for those three devices is analysed next, results reported as plot in Figure 17. The device with the largest amount of training data (Device A) obviously scored best, achieving the accuracy around 9.0% better than devices B and C. Again, the Gamma and log-mel results were similar, but each 'preferred' a different minority device. Although there were not enough devices included in the test for the evidence to be conclusive, this variability indicates that spectrograms differ in their affinity for different devices (or device locations, or channels). Again, the combined features effectively leveraged the advantages of each spectrogram type.

4.4.3 Spectrogram Performance by Segment Length

Inspired by some recent research considering the ability of systems to recognise a sound class early (or using partial data) [27, 28], this ability for the different spectrogram types is also evaluated. Figures 18 and 19 plot early classification

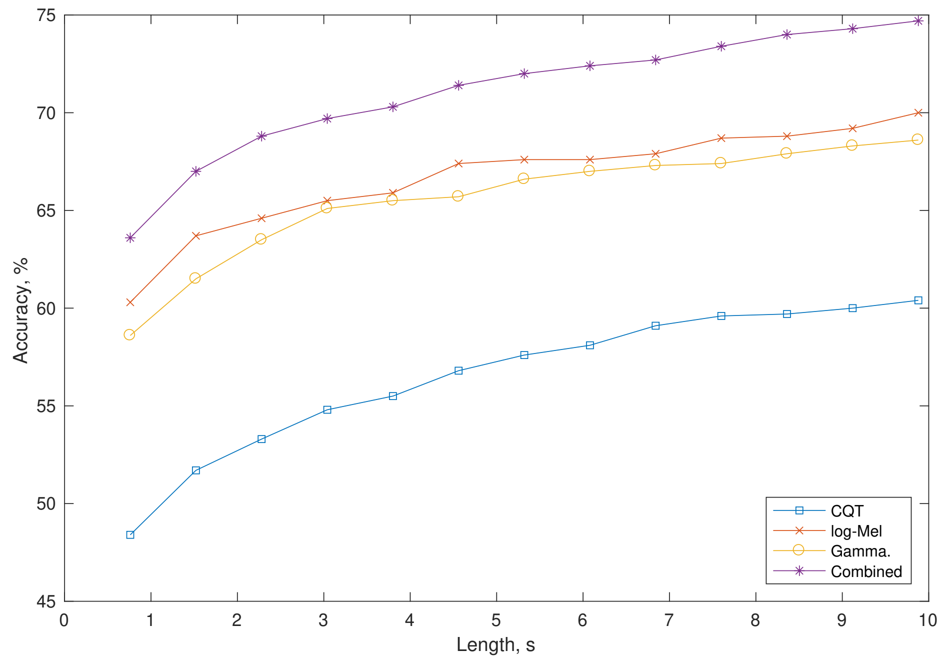

Figure 18: Classification performance as a function of the length of the test signal (second - s) over DCASE 2018 Task 1B Dev. set - all devices.

accuracy for DCASE 2018 Task 1B for all devices and for devices B+C, respectively. Early classification means that class assignment is only performed on the first part of the audio recording, rather than the entire duration (i.e. on cropped audio). Performance is plotted for a number of cropped segment lengths between 1 second and the full 10 seconds. From both plots, immediate observations are that the combined high-level features performed much better than the individual spectrogram types. The CQT performed worst while the other two spectrograms had similar performance (as in the experiments above). Looking closer at Figure 18 (accuracy for all devices), the score for all features continued to climb as duration progressed towards the full 10 seconds. This provides a strong indication that the system was data-constrained and is likely to perform better with longer duration recordings. By contrast, Figure 19 contains indications that the performance of the log-mel and Gamma spectrograms began to plateau as duration exceeded to 5 seconds, indicating that performance might not substantially increase if longer

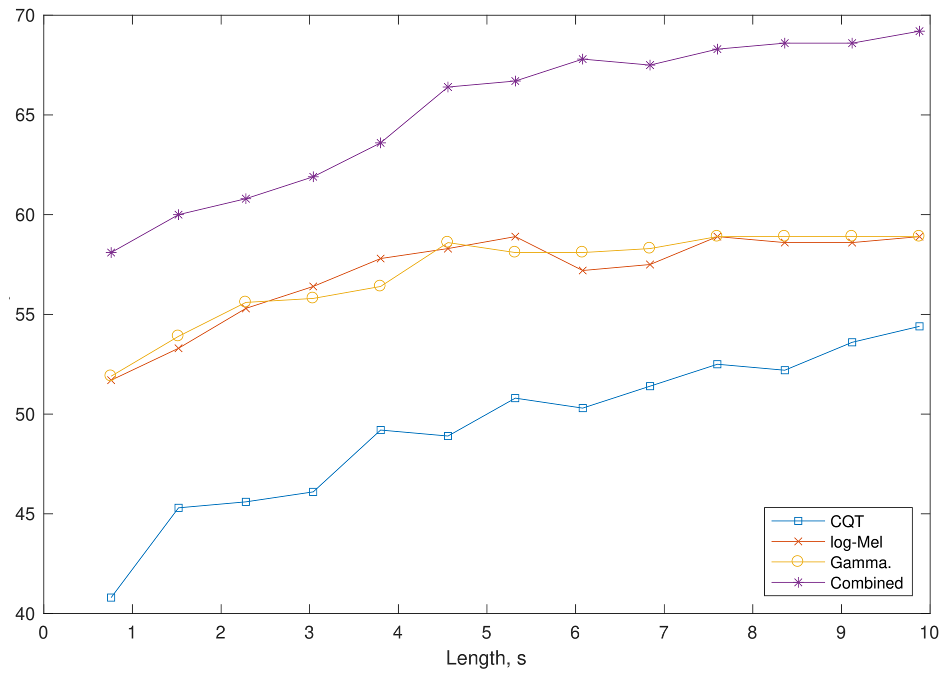

Figure 19: Classification performance as a function of the length of the test signal (second - s) over DCASE 2018 Task 1B Dev. set - devices B&C.

duration recordings were available. However the continued improvement of the CQT representation as length increased gave the combined features an ability to gain higher accuracy from longer recordings: The strength of CQT may lie in the analysis of longer recordings. However, in these experiments, CQT performance lagged the combined features by around 15.0% absolute, with the other spectrograms lagging by only around 5.0% absolute – apart from the area in Figure 19 where they plateaued. Most remarkable is the one with just 2 seconds of input data from a recording, our proposed combined high-level feature was able to match and outperform any of the individual spectrograms operating with the full 10 seconds of input data. This clearly demonstrates a major advantage of the proposed system. It effectively captures the advantages of the individual spectrogram features, which vary in their affinity for different classes and devices, and yields extremely good performance even when a restricted amount of data is available for classification.

4.4.4 Performance of Different Classifiers in The *Decoder*

Three methods were proposed in Section 4.2.2 to incorporate the three high-level spectrogram features into a combined high-level feature in the *Encoders* network. These methods were namely “sum-comb”, “max-comb” and “lin-comb”. To make use of the combined features, three back-end classifier methods for the *Decoders* block are introduced, namely RFC *Decoder*, MLP *Decoder* and MoE *Decoder* in Section 4.2.2. In total, the three classifiers and three combiners yield 9 models to evaluate. In this section, performance among these 9 models are compared, evaluated on the DCASE 2018 Task 1B Dev. dataset. It is noted that the accuracy of the *Encoders* network (i.e. the feature extractor, alone) and the full system accuracy (i.e. incorporating the decoder) are separately reported. Results are presented in Table 19, again split into Device A and Devices B & C performance. Best performance for both device sets, highlighted in bold, was achieved by the MoE *Decoder* classifier with the “lin-comb” combiner. However some interesting trends were evident. Firstly, MLP *Decoder* was only very slightly inferior to MoE *Decoder* for all combiners and device types. Secondly, looking at the *Encoders* network results for the Device A evaluation, the “max-comb” combiner actually outperformed the accuracy of “lin-comb”, although the latter performed best for most of the full systems. This means that the optimal high-level feature combiner for the full system was not the best combiner for loss computation when training the *Encoder* network. However the situation reverses when looking at Devices B & C – an indication that the performance gain of “lin-comb” may have been due to better generalisation.

4.4.5 Per-class Performance of Different *Decoders*

Given that the results presented so far indicate that the **lin-comb** combiner performed best, these high-level features are fed into the three alternative decoders to explore class-by-class performance. Table 21 presents results for DCASE 2018

Table 19: Performance of Encoder/Decoder (%) over DCASE 2018 Task 1B Dev. set.

Device A	RFC *Decoder*	MLP *Decoder*	MoE *Decoder*
sum-comb	71.5/75.6	71.5/72.2	71.5/71.9
max-comb	74.1/75.3	74.1/74.7	74.1/75.5
lin-comb	73.7/75.2	73.7/75.5	73.7/**75.9**
Devices B & C:	RFC *Decoder*	MLP *Decoder*	MoE *Decoder*
sum-comb	63.9/64.4	63.9/65.6	63.9/63.9
max-comb	61.4/65.3	61.4/63.9	61.4/63.9
lin-comb	64.2/68.9	64.2/69.2	64.2/**70.6**

Table 20: Performance comparison (Acc. %) to DCASE 2018 baselines for Task 1B Dev. set on Device A (using "lin-comb" for extracting high-level features in Encoder).

Categories	**D.2018**	**RFC *Decoder***	**MLP *Decoder***	**MoE *Decoder***
Airport	73.4	67.5	60.4	66.8
Bus	56.7	78.5	80.2	80.2
Metro	46.6	67.0	72.8	69.3
Metro station	52.9	84.6	82.6	80.3
Park	80.8	89.7	86.8	88.4
Public square	37.9	47.7	52.8	50.9
Shopping Mall	46.4	74.6	75.3	73.8
Street Pedestrian	55.5	65.6	72.5	71.3
Street Traffic	82.5	91.1	90.7	92.3
Tram	56.5	83.1	79.3	83.1
Average	**58.9**	**75.2**	**75.5**	**75.9**

Table 21: Performance comparison (Acc. %) to DCASE 2018 baselines for Task 1B Dev. set on Devices B+C (using "lin-comb" for extracting high-level features in Encoder).

Categories	**D.2018**	**RFC *Decoder***	**MLP *Decoder***	**MoE *Decoder***
Airport	72.5	55.6	69.4	75.0
Bus	78.3	88.9	86.1	88.9
Metro	20.6	75.0	63.9	66.7
Metro station	32.8	50.0	61.1	50.0
Park	59.2	91.7	91.7	94.4
Public square	24.7	52.8	47.2	47.2
Shopping Mall	61.1	80.6	80.6	80.6
Street Pedestrian	20.8	66.7	75.0	77.8
Street Traffic	66.4	75.0	77.8	77.8
Tram	19.7	52.8	38.9	47.2
Average	**45.6**	**68.9**	**69.2**	**70.6**

Task 1B (Dev. set). Device A and Device B & C results are again shown separately, and the "D.2018" column is the DCASE 2018 baseline. Results show that the three classifiers all outperformed the baseline – with the mixture of experts system improving accuracy by 17.0% and 25.0% absolute, for Device A and

Devices B & C, respectively.

4.4.6 Performance Comparison to State-of-the-art Systems

While performance against the baseline score of DCASE 2018 is good, the same model configuration (i.e. "lin-comb" combiner and MoE *Decoder* back-end classifier) is evaluated on various datasets and competitions, to compare the performance against the state of the art at the time of writing. The results, listed in Table 22, show that the system proposed achieves the highest accuracy for two datasets – achieving 70.6% and 98.9% for DCASE 2018 Task 1B Dev. and LITIS Rouen, respectively. For DCASE 2016, an accuracy of 88.2% was achieved, holding second position on the challenge table, and ranked top-four among state-of-the-art systems. DCASE 2017 performance is a little less competitive at 72.6%. DCASE 2018 Task 1A performance was 77.5%, taking third place on the challenge table. Participating in the recent DCASE 2019 challenge, this system achieved 76.8% and 72.8% for DCASE 2019 Task 1A and 1B, respectively. It should be noted that there is some inconsistency between the accuracies reported in the DCASE 2018 technical reports and those published on the DCASE 2018 challenge website [1]. Therefore, Table 22 carefully reports the accuracies stated in the peer-reviewed published papers and technical reports submitted to the challenge, rather than the figures advertised on the websites, which may differ slightly in some cases.

4.5 Conclusion

This chapter has presented a novel *Encoder-Decoder* deep learning framework applied for ASC. The framework addresses three main factors: low-level feature input, high-level feature extraction, and output classification that affects the final accuracy.

[1]http://dcase.community/challenge2018/

1. Firstly, inspired by the belief that low-level features each contain valuable and complementary information, three different spectrograms (log-mel, Gamma, and CQT) were evaluated, as well as their combination.

2. Hence, the *Encoder* network was proposed to effectively combine three different spectrograms (log-mel, Gamma, and CQT), extracting high-performance high-level feature vectors.

3. Then the *Decoder* was proposed as a final classifier. Three different models were explored and evaluated, the Random Forest Classification (RFC), the MLP-based network, and the Mixture of Expert (MoE).

The final combined system achieved very competitive results on various datasets, and has been evaluated against state-of-the-art systems to prove that the proposed *Encoder-Decoder* framework is both powerful in terms of performance and also general in terms of particular ASC task.

*Table 22: Performance comparison (Acc. - %) of the proposed Encoder-Decoder (E-D) Framework ("lin-comb" + MoE Decoder) to state-of-the-art results, with best performance in **bold** (Upper part: Dataset; Middle part: top-ten DCASE challenges; Lower part: State-of-the-art papers)*

D.2016	**Acc. (%)**	**D.2017**	**Acc. (%)**	**D.2018-1A**	**Acc. (%)**	**D.2018-1B**	**Acc. (%)**	**D.2019-1A**	**Acc. (%)**	**D.2019-1B**	**Acc. (%)**	**LITIS**	**Acc. (%)**
(Eva. set)		**(Eva. set)**		**(Dev. set)**		**(Dev. set)**		**(Eva. set)**		**(Eva. set)**		**(20-fold Ave.)**	
Wei [120]	84.1	Zhao [121]	70.0	Li [103]	72.9	Baseline [104]	45.6	Mingle [122]	79.9	Baseline [43]	61.6	Bisot [123]	93.4
Bae [90]	84.1	Jung [54]	70.6	Jung [66]	73.5	Li [105]	51.7	Wu [124]	80.1	Kong [125]	61.6	Ye [126]	96.0
Kim [127]	85.4	Karol [128]	70.6	Hao [106]	73.6	Tchorz [107]	53.9	Gao [129]	80.5	Waldekar [130]	62.1	Huy [91]	96.4
Takahasi [131]	85.6	Ivan [132]	71.7	Christian [108]	74.7	Kong [109]	57.5	Wang [133]	80.6	Wang [134]	70.3	Yin [135]	96.4
Elizalde [55]	85.9	Park [64]	72.6	Zhang [110]	75.3	Wang [111]	57.5	Jung [136]	81.2	Jiang [137]	70.3	Huy [19]	96.6
Valenti [101]	86.2	Lehner [53]	73.8	Li [112]	76.6	Waldekar [113]	57.8	Huang [138]	81.3	Song [139]	72.2	Ye [140]	97.1
Marchi [11]	86.4	Hyder [141]	74.1	Dang [114]	76.7	Zhao [23]	58.3	Haocong [142]	81.6	Primus [143]	74.2	Huy [20]	97.8
Park [65]	87.2	Zhengh [97]	77.7	Octave [14]	78.4	Truc [16]	63.6	Hyeji [95]	82.5	Hamid [144]	74.5	Zhang [89]	97.9
Bisot [76]	87.7	Han [96]	80.4	Yang [87]	79.8			Koutini [145]	83.8	Gao [146]	74.9	Zhang [81]	98.1
Hamid [52]	89.7	Mun [73]	**83.3**	Golubkov [115]	**80.1**			Chen [74]	**85.2**	Kosmider [147]	**75.3**	Huy [38]	98.7
Mun [94]	86.3	Zhao [59]	64.0	Bai [148]	66.1	Zhao [24]	63.3						
Li [149]	88.1	Yang [60]	69.3	Gao [150]	69.6	Truc [151]	64.7						
Hyder [152]	88.5	Waldekar [153]	69.9	Zhao [24]	72.6	Truc [17]	66.1						
Song [57]	89.5	Wu [63]	75.4	Phaye [25]	74.1	Yang [154]	67.8						
Yin [135]	**91.0**	Chen [92]	77.1	Heo [155]	77.4								
E-D Framework	88.2	*E-D Framework*	72.6	*E-D Framework*	77.5	*E-D Framework*	**70.6**	*E-D Framework*	76.8	*E-D Framework*	72.8	*E-D Framework*	**98.9**

Chapter 5

Two-level Hierarchical Classification

Inspired by the high cross-correlation between sound scenes mentioned in Section 3.5.1, the original “flat”ASC task, i.e. classification of all categories at once, might be better structured into multiple hierarchical sub-tasks operating in a divide-and-conquer manner. This chapter further explores the high cross-correlation between sound scenes, then based on that, it develops a two-level classification scheme for ASC.

In particular, sound scenes, which are expected to be acoustically similar, are firstly grouped into meta categories. The meta-categories constitute the first level of the classification hierarchy. Next, each category within the meta categories is classified by the second level of the two-level classification scheme. The two levels could also be referred to as coarse and fine grained classification. As experiments in this Chapter are conducted on DCASE 2018 Tasks 1A and 1B, the proposed hierarchy scheme is constructed based on them, as shown in Figure 20 (note that meta categories, such as *Indoor, Outdoor* and *Vehicle*, are selected based on analysis of the incorrect classification cases in Section 3.5.1).

The hierarchical classification is performed in a top-down fashion. Firstly,

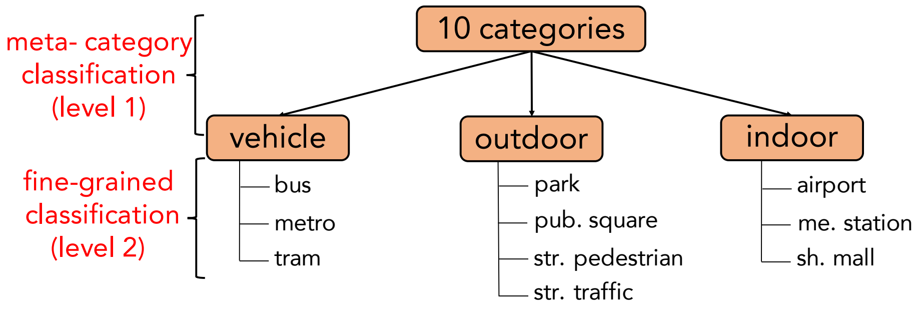

Figure 20: The two-level hierarchy of scene categories constructed by examination of the categories used in the DCASE 2018 dataset.

the meta-categories are classified, followed by the fine-grained classification of the scene categories within each individual meta-category. As a result, four classifiers are trained: one for meta-category classification (referred to as meta-category classifiers), then three are trained for classification of categories within the three meta-categories (namely a “vehicle” classifier, “indoor” classifier, and “outdoor” classifier, respectively). An unseen example will be then be deemed to have classified correctly only if it is correctly classified at both levels of the hierarchy. For example, a “on bus” scene example is correctly classified if it is both correctly classified as “vehicle” by the meta-category classifier and as “bus” by the “vehicle” classifier. Any misclassifcation by one or both of the classifiers will result in the example being wrongly classified overall.

5.1 The Proposed System

5.1.1 High-level Architecture

The high-level architecture of an ASC system applying the hierarchical scheme is described in Figure 21. As regards front-end feature extraction, three types of spectrogram (log-mel, Gamma, and CQT) are used to extract spectrogram information from channel 1. The entire spectrograms are then split into image patches of 128 $\times$ 128 before applying mixup data augmentation.

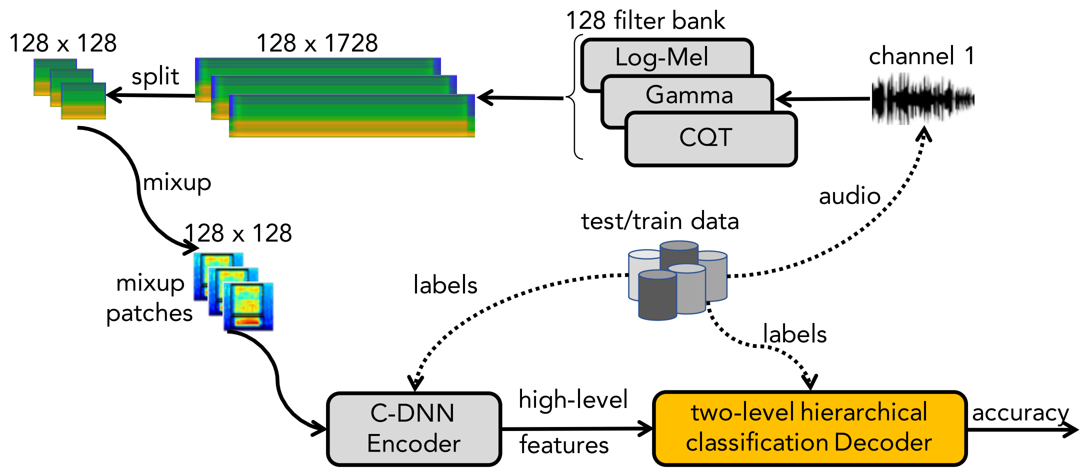

Figure 21: High-level system architecture applying a two-level hierarchical classification scheme.

Other settings for the front-end feature extraction such as filter number, window size, hop size, etc., are re-used from the baseline proposed in Section 3.3. The resulted mixup data is used to train a network for high-level feature extraction, referred as to C-DNN *Encoder*. The high-level features extracted from the C-DNN *Encoder* are fed into the two-level hierarchical classification scheme as described in Figure 20, which in turn then report the final classification accuracy. Compared to the novel *Encoder-Decoder* framework architecture introduced in Chapter 4, the two-level hierarchical scheme proposed in this Chapter takes the role of an MLP *Decoder*, RF *Decoder*, or MoE *Decoder* .

5.1.2 C-DNN *Encoder* Architecture

The high-level feature extractor in this Chapter uses a deep C-DNN as described in Table 23, comprising batch normalization (BN), convolutional (Cv [kernel size] @ kernel number), rectified linear unit (ReLU), average pooling (AP [kernel size]), dropout (Dr) and fully connected (FC) layers.

To clarify it, C-DNN *Encoder* is separated into two parts: the CNN part (the upper of Table 23) for feature learning and the DNN part (the lower of Table 23)

Table 23: The C-DNN architecture used for high-level feature extraction.

Layers	**Output**
Input layer (image patch)	128×128
BN - Cv [3×3] @ 32 - ReLU - BN - AP [2×2] - Dr (10%)	64×64×32
BN - Cv [3×3] @ 64 - ReLU - BN - AP [2×2] - Dr (15%)	32×32×64
BN - Cv [3×3] @ 128 - ReLU - BN - Dr (20%)	32×32×128
BN - Cv [3×3] @ 128 - ReLU - BN - AP [2×2] - Dr (20%)	16×16×128
BN - Cv [3×3] @ 256 - ReLU - BN - Dr (25%)	16×16×256
BN - Cv [3×3] @ 256 - ReLU - BN - AP [2×2] - Dr (25%)	8×8×256
BN - Cv [8×8] @ 256 - ReLU - BN - Dr (30%)	256
Input layer (vector)	256
FC - ReLU - Dr (30%)	512
FC - ReLU - Dr (30%)	1024
FC - Softmax	10

for classification. Both are based on the previous architectures in Chapter 4, but instead of using a Global Average Pooling layer at the output of the CNN as in the *Encoder* of Section 4.2.1, an additional convolutional layer with kernel size of [8×8] is incorporated. This equates to the time-frequency resolution of the output from the previous layer, and is included to capture the interaction across the convolutional channel dimension. In the other words, the final convolutional layer helps to scale temporal and frequency dimensions into one value with trainable parameters learning all pixels of temporal-frequency images.

Once the network has been trained, the feature-learning CNN part of the network is used as a feature extractor and its last convolutional layer is considered to provide high-level features. In this way, when presented with a new input, the high-level feature extractor will process the input starting from the first convolutional layer, through to the final convolutional layer and produce a high-level feature vector of dimension 256.

5.1.3 Two-level Hierarchical Classification as *Decoder*

Most existing works follow a “flat” classification scheme in which all scene categories are classified at once. By contrast, this Section proposes performing the classification hierarchically, as recently introduced in Figure 20. The classifiers

Table 24: MLP-based architecture used in the two-level hierarchical scheme

Layers	**Output Shape**
Input layer	256
FC - ReLU - Dr (30%)	512
FC - ReLU - Dr (30%)	1024
FC - ReLU - Dr (30%)	1024
FC - Softmax	C

involving in the hierarchical classification are realized by Multilayer Perceptron (MLP) based networks. The 256 dimensional high-level features presented in Section 5.1.2 are obtained from the mixup image patches and used to train the MLPs. There are in total four MLPs (one for the meta-categories at the first level and three for fine classification within the meta-category groups of *Indoor, Outdoor* and *Vehicle* at the second level), each comprises four fully connected layers and is parametrized as summarised in Table 24. Note that the MLPs share a common architecture but are trained separately depending on their respective sub-tasks in the hierarchical classification.

The number of categories classified C depends on the specific task in the hierarchical scheme. For example C is 3 for meta-category classification and for the *Vehicle* and *Indoor* group categories. It is 4 for classification of the categories in the *Outdoor* group.

5.2 Experimental Setting

5.2.1 Datasets

To evaluate the two-level hierarchical scheme, DCASE 2018 Task 1A and 1B development datasets [43] are used to conduct experiments. For consistency, the relevant settings used for these datasets are mentioned and reused from Section 3.4.1 and Section 4.3.1.

5.2.2 Setting Hyperparameters and Training Process

Since the labels of the mixup data input are no longer one-hot, the network is trained with Kullback-Leibler (KL) divergence loss [116] rather than the standard cross-entropy loss over mixup training image patches:

$$LOSS_{KL}(\Theta) = \sum_{c=1}^{C} y_c \log\left\{\frac{y_c}{\hat{y}_c(\Theta)}\right\} + \frac{\lambda}{2}||\Theta||_2^2, \tag{17}$$

where Θ denotes the trainable network parameters and λ denotes the ℓ_2-norm regularization coefficient set to 0.001. y_c and $\hat{y}_c$ denote the ground-truth and the network output of class c, respectively.

In addition to the KL-divergence loss, the triplet loss function [156] is additionally employed to train the MLPs in the second-level classifiers to encourage the networks to improve their discrimination power. The motivation is that the triplet loss function has been shown to be efficient in learning a discriminative metric which simultaneously minimises same-category distance while maximising between-category distances. In this way, it enhances Fisher's criterion [156] (i.e. the ratio of the between- class distance to the within-class variance in the feature space).

Suppose that there are two samples from different categories presented to an MLP. The ground-truth label of the first sample is the anchor $\mathbf{a}$, the prediction for the first sample is positive $\mathbf{p}$, and the prediction for a second sample is positive $\mathbf{n}$, then the triplet loss is given as:

$$LOSS_T = \max\{d(\mathbf{a}, \mathbf{p}) - d(\mathbf{a}, \mathbf{n}) + margin, 0\}, \tag{18}$$

where d is the squared Euclidean distance and the $margin$ is set to 0.3.

The final loss function is then a combination of the KL-divergence loss and the triplet loss as follows:

$$LOSSES = \gamma LOSS_{KL} + (1-\gamma) LOSS_{T} \tag{19}$$

The networks are implemented using the Tensorflow framework. The coefficient λ in (17) is set to 0.001, and γ in (33) is experimentally set to 0.2. The network training is accomplished with the Adam optimiser [102] with an initial learning rate of 10^{-4}, a batch size of 100, and a fixed termination after 100 epochs.

5.2.3 Multi-spectrogram Ensemble

As for the comprehensive analysis of low-level features in Chapter 3, using multiple input types, provided a rule of thumb for performance ASC, now all three time-frequency input types are used: log-mel [99], gammatone filter (Gamma) [100], and Constant Q Transform (CQT) [99]. Together these three will enable construction of an ensemble of three systems. The final decision of each classification task (meta-category classification at the first level or the fine-grained second level classification. shown in Figure 20) is obtained by aggregating the individual decisions of the three classifiers (each with one type of spectrogram) in an ensemble. In particular, if $\bar{\mathbf{p}}_{\text{log-mel}}$, $\bar{\mathbf{p}}_{\text{Gamma}}$, $\bar{\mathbf{p}}_{\text{CQT}}$ are probabilities corresponding log-mel, Gamma, and CQT spectrogram input, sum of three probability $\bar{\mathbf{p}}[\bar{p}_1, \bar{p}_2, ..., \bar{p}_C]$ is computed by

$$\bar{\mathbf{p}} = \bar{\mathbf{p}}_{\text{log-mel}} + \bar{\mathbf{p}}_{\text{Gamma}} + \bar{\mathbf{p}}_{\text{CQT}} \tag{20}$$

where $C = 10$ denotes the number of categories classified in DCASE 2018 Task 1A & 1B. Thus, the final classification label is determined as,

$$\hat{y} = \underset{c \in \{1,2,...,C\}}{\operatorname{argmax}} \ \bar{p}_c. \tag{21}$$

where $\hat{y}$ denotes the final label.

5.3 Experimental Results

5.3.1 Performance comparison to DCASE 2018 baseline

Table 25: Performance comparison (in percentage accuracy) between the proposed system (with and without triplet loss), the DCASE 2018 baseline, and the C-DNN Encoder baseline.

Systems Compared	**Task 1A**	**Task 1B**
DCASE 2018 baseline [43]	59.7	45.6
The C-DNN *Encoder*	70.9	61.1
The proposed w/o triplet loss	73.3	**62.2**
The proposed w/ triplet loss	**75.3**	58.9

To evaluate the hierarchical scheme, the C-DNN *Encoder* which uses only the Gamma spectrogram is referred as to the baseline. The performance of the DCASE 2018 baseline, C-DNN *Encoder* baseline, the entire system (applying the two-level hierarchical scheme without ensemble) are compared in Table 25. As can be seen, the proposed system outperforms the DCASE 2018 baseline by a large margin, around 15.6% absolute (with triplet loss) on Task 1A and 16.6% absolute on Task 1B (without triplet loss). Improvements of the individual categories can also be visualised in Figure 22, which compares the proposed system with triplet loss against the DCASE 2018 baseline on Task 1A. It is notable that several categories enjoy a significant gain of more than 20.0%, such as *shopping mall, tram, metro* and *street-pedestrian*. Compared to C-DNN *Encoder*, the proposed system gains an accuracy of 2.4% and 1.1% on Task 1A and Task 1B, respectively, when the triplet loss is not used. When the triplet loss is used, a significant accuracy improvement is seen on Task 1A: 2.4% absolute compared to that without triplet loss and 4.4% compared to the developed baseline thanks to the proposed hierarchical classification scheme. However, using triplet loss seems to be counter-productive on Task 1B as the accuracy is reduced by 3.3% absolute compared to the system without triplet loss. This is presumably due to the device mismatch or the lack of training data on the target devices (device B & C) or both. However, averaging

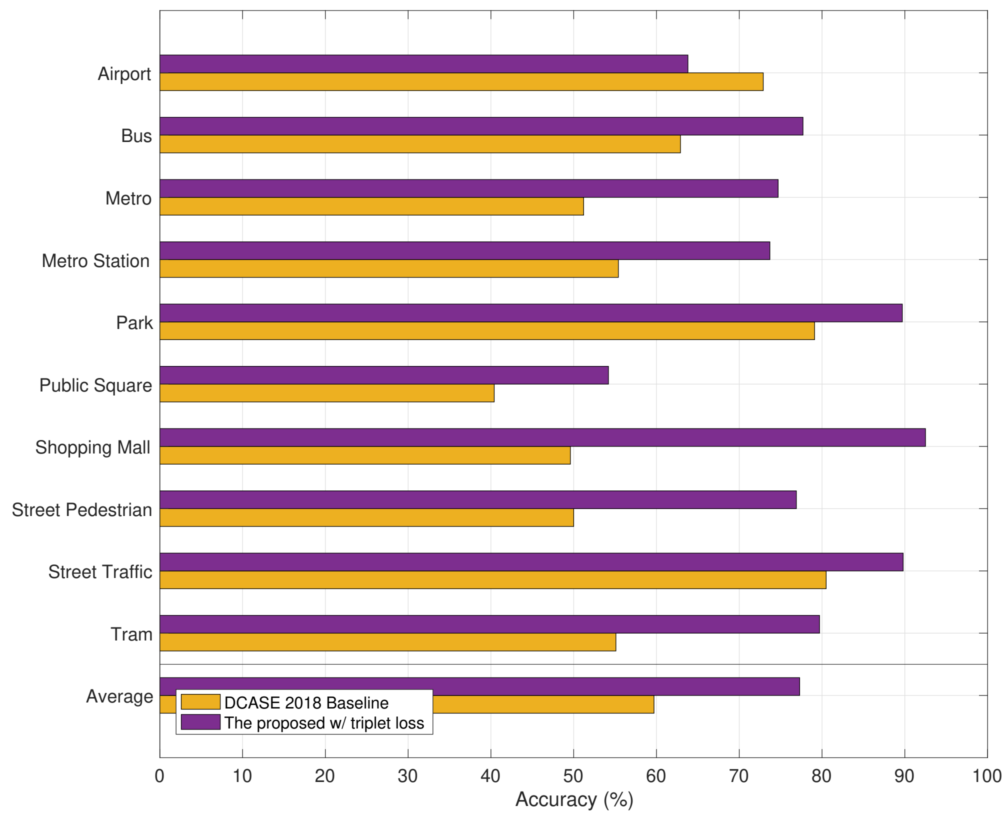

Figure 22: Category-wise performance comparison between the proposed system with triplet loss and the DCASE 2018 baseline on Task 1A.

over all the devices, the proposed system with triplet loss outperforms all other counterparts, as shown in Figure 23. To further shed light on the performance of the classifiers in the proposed hierarchical classification scheme, their confusion matrices are presented in Figure 24. Overall, the meta-categories are discriminated very well by the meta-category classifier, with an average accuracy of 94%. Given the good performance of the meta-category classifier, the test examples are expected to be directed to the correct groups in the lower level. Even though the fine-grained classifiers' performance are not as good as that of the meta-category classifier, this is to be expected since the categories in a group tend to be similar acoustically, however in each group, the fine classification network is able to avoid confusion between its categories and those in other groups.

Further compared to *Encoder-Decored* systems proposed in Chapter 4, while

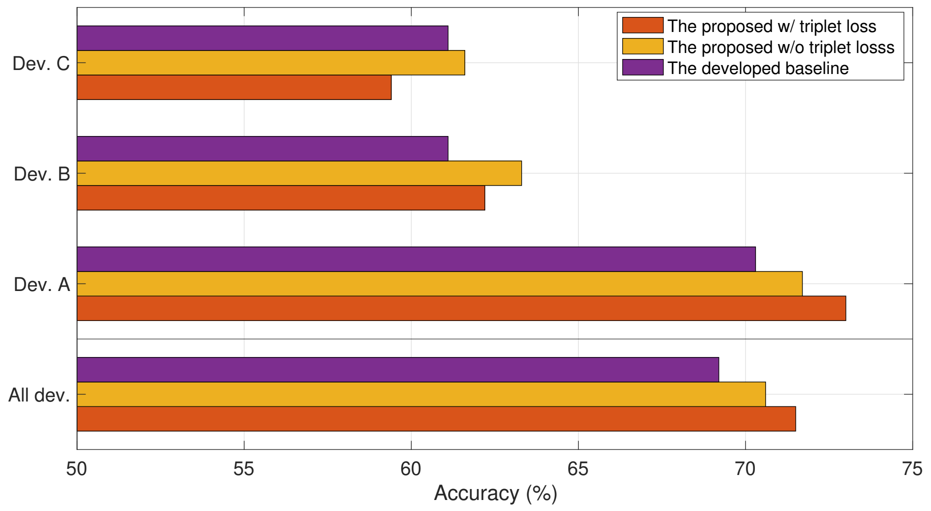

Figure 23: Accuracy obtained by the systems developed in this work on different devices of Task 1B.

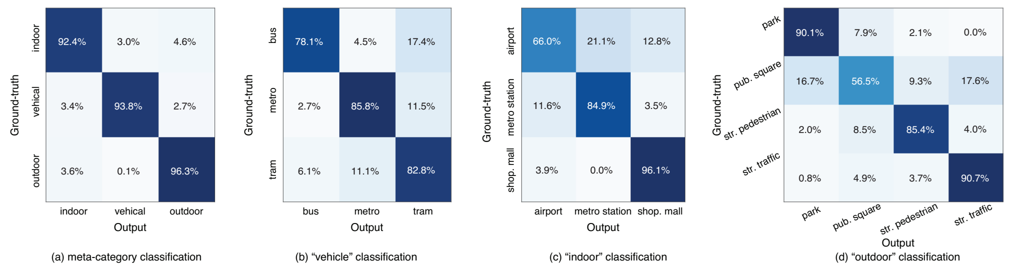

Figure 24: Confusion matrices obtained by different classifiers in the proposed hierarchical classification scheme on Task 1A.

the best *Encoder-Decored* system using lin-comb, MoE Decoder, and three spectrogram inputs achieves the accuracy of 75.9% on DCASE 2018 Taks 1A as shown in Table 19, the two-level hierarchical classification system proposed achieves the competitive result of 75.3% on the same task with only using Gammatone spectrogram input.

5.3.2 Results of Multi-spectrogram Ensemble

Further experiments are conducted over individual time-frequency inputs (i.e. Gamma, log-mel, and CQT spectrograms). The gammatone spectrogram seems to perform best as shown in Figure 25 while the CQT spectrogram performs the

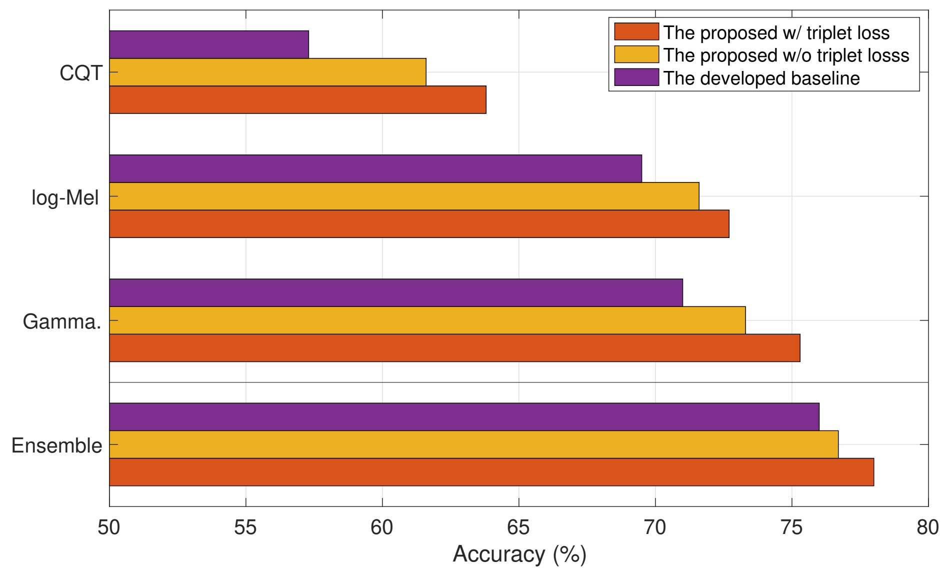

Figure 25: Performance of individual time-frequency representations and their ensemble on Task 1A.

worst. However, aggregation of the classification outputs of all three, results in significant improvements over the individual ones. This is observed over all systems; the proposed system with triplet loss, the proposed system without triplet loss, and the developed baseline. It is expected as different time-frequency representations have been shown to be good for different scene categories, and their individual strength is leveraged in the ensemble to improve the performance gain.

The obtained results are further compared with the previous works (both the DCASE 2018 challenge submission systems and the recent works), providing a comprehensive performance comparison on Task 1A and Task 1B in Table 26. It should be noted that there are inconsistencies between the accuracies reported in the DCASE 2018 technical reports and those published in DCASE 2018 challenge website [1]. The results in Tables 26 are collated from the technical reports which are the original sources of the reported accuracies. For clarity, only top 10 DCASE 2018 challenge submissions are presented in the tables. In the one hand, the proposed system outperforms the recent works (i.e. after the DCASE

[1]http://dcase.community/challenge2018/

Table 26: Comparison between the top-10 DCASE 2018 challenge (top), recent papers (middle), and the proposed system (bottom)

D.2018-1A	**Acc. (%)**	**D.2018-1B**	**Acc. (%)**
Li [103]	72.9	Baseline [104]	45.6
Jung [66]	73.5	Li [105]	51.7
Hao [106]	73.6	Tchorz [107]	53.9
Christian [108]	74.7	Kong [109]	57.5
Zhang [110]	75.3	Wang [111]	57.5
Li [112]	76.6	Waldekar [113]	57.8
Dang [114]	76.7	Zhao [23]	58.3
Octave [14]	78.4	Truc [16]	63.6
Yang [87]	79.8		
Golubkov [115]	**80.1**		
Bai [148]	66.1	Zhao [24]	63.3
Gao [150]	69.6	Truc [151]	64.7
Zhao [24]	72.6	Truc [17]	66.1
Phaye [25]	74.1	Yang [154]	**67.8**
Heo [155]	77.4		
The proposed w/ triplet loss	78.0	*The proposed w/o triplet loss*	66.9

2018 challenge) on Task 1A while retaining as top-4 performer in the context of the DCASE 2018 submission systems. In the other hand, our proposed system achieves very competitive results on Task 1B, achieving an accuracy of 66.9% and outperforming the DCASE 2018 submission systems.

5.4 Conclusion

This chapter has presented an approach that trains deep feature embedding networks to extract high-level features for audio scene signals via a C-DNN based *Encoder* and proposed a novel hierarchical classification scheme to accomplish scene classification. In the classification hierarchy, the similar scene categories are first grouped into meta-categories. Meta-category classification is carried out first, followed by the fine-grained classification within the meta groups. MLPs were trained, with the contribution of triplet loss, to play the role of the classifiers in the classification hierarchy. Experiments on the DCASE 2018 Task 1A and 1B datasets demonstrated that the proposed methods outperform DCASE baseline.

Chapter 6

Respiratory Disease Detection

According to the World Health Organization (WHO) [157], respiratory illness, which comprises lung cancer, tuberculosis, asthma, chronic obstructive pulmonary disease (COPD), and lower respiratory tract infection (LRTI), accounts for a significant percentage of mortality worldwide. Indeed, records indicate that around 10 million people currently have tuberculosis (TB), 65 million have COPD, and 334 million have asthma. Notably, the WHO estimates that about 1.4, 1.6, and 3 million people die from TB, lung cancer or COPD annually, respectively.

To deal with respiratory diseases, early detection is the key factor in enhancing the effectiveness of intervention, including treatment and limiting spread. During a respiratory examination, lung auscultation (listening to the sounds of breathing through a stethoscope) is an important aspect of respiratory disease diagnosis. By listening to respiratory sounds during lung auscultation, experts can recognise adventitious sounds (including *Crackles* and *Wheezes*) during the respiratory cycle. These often occur in those who have pulmonary disorders. If automated methods can be developed to detect such anomalous sounds, it will improve the early detection of respiratory disease and enable screening of a wider population than manual screening.

Inspired by the deep learning techniques that I had developed for effective

Acoustic Scene Classification in Chapters 3, 4 and 5, I decided to apply these advances on a real world problem: respiratory sound analysis. Thus, this chapter introduces a robust deep learning framework aiming to classify anomalies in respiratory cycles for detecting disease from respiratory sound recordings. It evaluates using a standard benchmark, the 2017 International Conference on Biomedical Health Informatics (ICBHI) [1] dataset. The framework proposed for this task is derived from the baseline mentioned in Chapter 3. It begins with front-end feature extraction to transform input sound into a spectrogram representation. Then, a back-end deep learning network is used to classify the spectrogram features into categories of respiratory anomaly cycles or disease classes.

The framework proposed confirms three main contributions towards respiratory-sound analysis. Firstly, it allows an extensive exploration of the effects of spectrogram type, spectral-time resolution, overlapped/non-overlapped windows, and data augmentation, thus indicating which feature has the greatest effect on final prediction accuracy. This leads to a proposal for a novel deep learning system, developing the proposed framework further, which is shown to significantly outperform current state-of-the-art methods. However the complexity of that structure is quite high, and so a Teacher-Student scheme is developed and applied with the aim of achieving a trade-off between model performance and complexity. This additionally helps to increase the potential of the proposed framework for building real-time applications, such as in mobile devices which are constrained in terms of processing power.

Before discussing the framework architecture, state-of-the-art systems used for respiratory sound analysis will be analysed, and thus a benchmark dataset used to conduct experiments as well as specific task definition over this dataset are presented.

6.1 State-of-the-art Respiratory Sound Analysis

6.1.1 Literature Review

Research into the niche domain of automated detection or analysis of respiratory sounds has some precedents [158, 159, 160], but has drawn attention in recent years as robust machine hearing methods have been developed, leveraging on ever more capable deep learning techniques. Like traditional ASC systems, most existing respiratory sound analysis systems tend to rely upon frame-based feature representations such as Mel-Frequency Cepstral Coefficients (MFCC) [161, 162], borrowed from the Automatic Speech Recognition (ASR) and Speaker Recognition (SR) fields. However, Grønnesby *et al.* [163] found that MFCCs did not represent crackles well. They thus replaced them with five-dimensional feature vectors, comprising four time domain features (variance, range, and sum of simple moving average (coarse and fine)), and one frequency domain feature (spectrum mean). Meanwhile, Hanna *et al.* [164] firstly extracted spectral information from bark-bands energy, Mel-bands energy, MFCCs, rhythm features from beat loudness, harmonicity and inharmonicity features, as well as tonal features such as chords strength and tuning frequency. Next, they computed statistical features including standard deviation, variance, minimum, maximum, median, mean, first derivative, second derivative from those features in addition to mean and variance of the raw signal. This extensive list aimed to maximize the chance of achieving a discriminative feature set. To further explore audio features, Mendes *et al.* [165] went further to propose 35 different types of feature, mainly coming from Music Information Retrieval research. Inspired by the finding that only some features contributed to the final result, Datta *et al.* [166] firstly assessed features such as power spectral density (PSD), FFT and Wavelet spectrograms, MFCCs, and Linear Frequency Cepstral Coefficients (LFCCs). Next, they applied a Maximal Information Coefficient (MIC) [167] to score each feature, selecting

Table 27: The state-of-the-art frame-based frameworks

Author	**Front-end Feature Extraction**	**Back-end Classification**
Okubo *et al.* [161]	MFCC	HMM
Kok *et al.* [162]	MFCC, DWT, & Handcrafted features	RUSBoost
Grønnesby *et al.* [163]	MFCC & Handcrafted features	KNN, DT, SVM
Hanna *et al.* [164]	MFCC & handcrafted features	DT
Mendes *et al.* [165]	MFCC & Music based features	LR
Datta *et al.* [166]	PSD, FFT, Wavelet, MFCC, LFCC	SVM
Sengupta *et al.* [168]	MFCC & LBP	KNN, SVM

only the most influencing before feeding into a classifier to improve performance and reduce complexity. Similarly, Kok *et al.* [162] applied the Wilcoxon Sum of Rank test to indicate which features among MFCCs, Discrete Wavelet Transform (DWT) and a set of time domain features (namely power, mean, variance, skewness and kurtosis of audio signal) mainly affected final classification accuracy. Image processing techniques were then employed by Sengupta *et al.* [168], who applied Local Binary Pattern (LBP) analysis on Mel-frequency spectral coefficients (MFSCs) to capture texture information from the MFSC spectrogram, thus obtained an LBP spectrogram. The LBP spectrogram was converted into a histogram presentation before feeding it into a back-end classifier, which was shown to outperform the previous MFCC-based methods. In these systems, the time stream of audio feature vectors is classified by a range of traditional machine learning techniques. These include Logistic Regression (LR) [165], k-Nearest Neighbour (KNN) [163, 168], Hidden Markov Models (HMM) [161, 169, 170], Support Vector Machines (SVM) [163, 166, 168, 171, 172] and Decision Trees (DT) [162, 163, 164, 173].

As we know from earlier chapters, deep learning techniques have achieved strong and robust detection performance for general sound classification [174], [175]. Feature extraction in state-of-the-art deep learning based systems typically involves generating two-dimensional time-frequency spectrograms that are able to capture both fine grained temporal and spectral information as well as present

Table 28: The state-of-the-art spectrogram based frameworks

Author	**Front-end Feature Extraction**	**Back-end Classification**
Perna *et al.* [180]	MFCC	LeNet (CNN)
Aykanat *et al.* [179]	MFCC	LeNet (CNN)
Liu *et al.* [177]	log-Mel	VGG (CNN)
Minami *et al.* [184]	STFT & Wavelet	Parallel VGGs (CNN)
Chen *et al.* [185]	Optimized S-transform	ResNet50 (CNN)
Perna *et al.* [181]	MFCC	LSTM (RNN)
Kochetov *et al.* [183]	MFCC	LSTM & GRU (RNN)
Acharya *et al.* [178]	log-Mel	Hybrid (CNN & RNN)

a much wider time context than single frame analysis. While a variety of spectrogram transformations have been utilised, Mel-based methods such as log-mel spectra [176, 177, 178] and stacked MFCC features [176, 179, 180, 181, 182, 183] are the most popular ones. Some researchers combined different types of spectrogram, e.g. STFT and Wavelet as proposed by Minami *et al.* [184] or optimized S-Transformations in [185]. Although extracting good quality representative spectrograms is very important for a back-end classifier in general, researchers to date have not yet extensively explored the settings used in this step. This applies to both traditional sound classification, as well as to respiratory sound classification.

The most recent deep learning classifiers used with spectrogram input for research into respiratory sound analysis are mainly based on Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), or hybrid architectures. These CNN-based systems span some diverse architectures such as LeNet6 [180, 179], VGG5 [177], two parallel VGG16s [184], and ResNet50 [185]. Inspired by the fact that respiratory indicative sounds such as *Crackle* and *Wheeze* present certain sequential characteristics, RNN-based networks have been developed in order to capture the sequential information. For example, Perna and Tagarelli [181] analysed the use of a Long Short-term Memory (LSTM) network for two tasks of classifying anomalous respiratory sounds and classifying respiratory diseases. By using LSTM and Gated Recurrent Unit (GRU) cells in a RNN-based network, Kochetov *et al.* [183] proposed a novel architecture, namely the Noise Masking

Recurrent Neural Network, which aimed to distinguish both noise and anomalous respiratory sounds. In hybrid architectures proposed in [178], a CNN was firstly used to map a spectrogram input to a temporal sequence. Then, LSTM was used to learn sequence structures before classification takes place via fully connected layers. Compared to traditional machine learning approaches, state-of-the-art respiratory sound detection performance comparisons presented in [181, 184, 185] indicate that deep learning classifiers are robust and effective.

6.1.2 Exiting Issues and Proposed Solution

As the above literature review of respiratory sound analysis systems shows, both frame-based and spectrogram-based systems deployed for respiratory sound classification are very similar to the state-of-the-art ASC systems mentioned in Chapter 2. Basically, the current systems used for respiratory analysis makes use of both machine learning in general, and deep learning techniques in particular, to achieve good results. However, there exist specific issues that differ from the ASC research field. Firstly, the state-of-the-art systems involve ever-increasing model complexity, especially for those employing deep learning models, limiting their potential implementation within mobile or wearable real-time devices. While choosing a platform for ASC applications is flexible and less affected by efficiency in general, the practicalities of implementing respiratory sound analysis on edge devices is an important aspect of the benefit of such systems. For instance, if the function of respiratory sound analysis can be integrated into mobile phones, patients could self-check their situation at home, to track disease or recovery progression, for example. Additionally, analysing respiratory sound on real-time embedded devices helps to reduce the cost of manufacturing devices significantly, thus potentially increasing the ability of observation of lung disease, and the application on a much larger scale. A more serious issue with this research field has been the difficulty of comparing between techniques due to the lack of standardised datasets used by authors for evaluation. Most publications evaluated on proprietary datasets that are unavailable to others [165, 166, 169, 176, 182]. Comparing this to ASC, it is much easier to record an ASC dataset (no ethics approvals required, no access to patients, no infection control issues etc.) and so a number of good ASC dataset have been published for research and academic activities. Having more data, and more diversity of data available, is usually beneficial when building deep learning based systems, but for respiratory sound detection data is lacking.

To tackle these main issues, this chapter proposes a deep learning framework in the following way;

- Firstly to ensure repeatability and ease of comparison, the 2017 International Conference on Biomedical Health Informatics (ICBHI) [1] dataset is used for all experiments. The ICBHI dataset is one of the largest currently available which includes audio recordings. Using this resource, factors can be comprehensively analysed. This includes investigating different types of spectrogram, the use of overlapped or non-overlapped windowing, variable spectrogram patch sizes, the use of data augmentation techniques. In each case, the standard database allows their effects on performance to be precisely pinpointed.

- From this analysis, a deep learning framework is proposed to target two related tasks of anomaly sound classification and respiratory disease detection. Two methods of train/test splitting are used in the literature (namely random 5-fold cross validation and 60/40 splitting as per the ICBHI challenge's recommendation). Both are evaluated here, and compared directly to state-of-the-art systems.

- To aid in the trade-off between performance and complexity, a Student-Teacher scheme is proposed. Specifically, the best deep learning framework, which is used for the task of respiratory disease detection and requires a

Table 29: ICBHI dataset [1]

Database	Testing Set			Traing Set		
	ESSUA	**AUTH**	**All**	**ESSUA**	**AUTH**	**All**
Patients	38	11	49	72	7	79
Recordings	617	64	381	507	32	539
Wheezes	588	61	649	459	42	501
Crackles	273	112	385	1140	111	1215
Crackles & Wheezes	106	37	143	335	28	363
Normal	1216	363	1579	1740	323	2063

large number of trainable parameters, is referred to as the Teacher. Classification information from the Teacher model is extracted and distilled to train another network architecture with fewer trainable parameters, referred to as the Student. Finally, a reduced-size Student network results, and when evaluated, is shown to achieve similar performance to the Teacher, but with significantly lower complexity.

6.2 ICBHI Dataset and Tasks Proposed

6.2.1 ICBHI Dataset

The 2017 ICBHI dataset [1], which was collected from School of Health Sciences, University of Aveiro (ESSUA) an Aristotle University of Thessaloniki (AUTH) as shown in Table 29, provides a large database of labelled respiratory sounds comprising 920 audio recordings with a combined duration of 5.5 hours. The recording lengths are uneven, ranging from from 10 to 90 seconds, and were recorded with a wide range of sampling frequencies from 4000 Hz to 44100 Hz. In total, the dataset contains recordings from 128 patients, who are identified in terms of being healthy or exhibiting one of the following respiratory diseases or conditions: COPD, Bronchiectasis, Asthma, upper and lower respiratory tract infection, Pneumonia, Bronchiolitis. These respiratory condition labels are linked to audio recording files. Within each audio recording, four different types of respiratory cycle are presented – called *Crackle*, *Wheeze*, *Both (Crackle & Wheeze)*, and *Normal*. These cycles, labelled by experts, include identified onset and offset times. The cycles have various recording lengths ranging from 0.2 up to 16.2 seconds, with the number of cycles being unbalanced (i.e. 1864, 886, 506 and 3642 cycles respectively for *Crackle*, *Wheeze*, *Both*, and *Normal*).

6.2.2 Main Tasks Proposed for ICBHI Dataset

Given the ICBHI recordings and metadata, this chapter targets performance over two main tasks.

Task 1, respiratory anomaly classification, is separated into two sub-tasks. The first aims to classify four different cycles (*Crackle*, *Wheeze*, *Both*, and *Normal*). The second is to classify the four types of cycle into two groups of *Normal* and *Anomaly* sounds (the latter group consisting of *Crackle*, *Wheeze*, and *Both*). For convenience, these are named Task 1-1 and Task 1-2, respectively in this thesis.

Task 2, respiratory disease prediction, also comprises two sub-tasks. The first aims to classify audio recordings into three groups of disease conditions: *Healthy*, *Chronic Disease* (i.e. COPD, Bronchiectasis and Asthma) and *Non-Chronic Disease* (i.e. upper and lower respiratory tract infection, Pneumonia, and Bronchiolitis) (Note that individual disease condition is not directly classified as the dataset is unbalanced, especially only 1 patient with Asthma disease and 2 patients with LRTI disease). The second sub-task is for classification into two groups of *Healthy* and *Unhealthy* (comprising the *Chronic* and *Non-Chronic* disease groups combined). These sub-tasks are referred to as Tasks 2-1 and Task 2-2, respectively in this thesis. While Tasks 1-1 and 1-2 are evaluated over individual respiratory cycles, Task 2-1 and 2-2 are evaluated over entire audio recordings.

State-of-the-art published systems that used the ICBHI dataset follow two different approaches to split the database into training and testing portions. The first [170, 172, 173, 184] followed the ICBHI challenge recommendations [1] to divide the dataset into non-overlapping 60% and 40% portions for training and test subsets, respectively. Notably, this avoids a situation in which audio recordings from one subject are found in both of the subsets. Meanwhile, the second [162, 164, 177, 180, 181] randomly separated the entire dataset into training and test subsets, with different ratios.

To evaluate our proposed framework on each task in this chapter, the ICBHI dataset is firstly separated (6898 respiratory cycles for Task 1 and 920 entire recordings for Task 2) into five folds for cross validation. Next, a baseline system is introduced to evaluate the effects of a number of settings and influencing factors as noted above. Due to extensive training times, this initial exploration evaluates on one cross-validation fold. Then, following the initial exploration, two systems are proposed; one for anomaly cycle detection (Tasks 1-1 and 1-2) and the other for respiratory disease detection (Tasks 2-1 and 2-2). Each of those systems is then trained and evaluated with both the full 5-fold cross validation and 60/40 split as the ICBHI challenge's recommendation. Each system is then compared against state-of-the-art methods.

6.2.3 Evaluation Metrics

The baseline and proposed framework variants are assessed using the metrics of *Sensitivity* (Sen.), *Specitivity* (Spec.), and *ICBHI score* [181, 1]. To understand these scores, a confusion matrix for Task 1 as presented in Table 30 is considered. In this case, *C, W, B,* and *N* denote the numbers of cycles of *Crackle*, *Wheeze*, *Both*, and *Normal* respectively, whereas *c, w, b,* and *n* subscripts indicate the classification results. The sums C_t, W_t, B_t and N_t are the total numbers of cycles.

Table 30: Confusion matrix of anomaly cycle classification.

	Crackle	**Wheeze**	**Both**	**Normal**
Crackle	C_c	W_c	B_c	N_c
Wheeze	C_w	W_w	B_w	N_w
Both	C_b	W_b	B_b	N_b
Normal	C_n	W_n	B_n	N_n
Total	C_t	W_t	B_t	N_t

Table 31: Confusion matrix of respiratory disease detection.

	Chronic	**Non-chronic**	**Healthy**
Chronic	C_c	NC_c	H_c
Non-chronic	C_{nc}	NC_{nc}	H_{nc}
Healthy	C_h	NC_h	H_h
Total	C_t	NC_t	H_t

Sensitivity is computed for Task 1-1 (4-class anomaly classification) as:

$$Sensitivity = \frac{C_c + W_w + B_b}{C_t + W_t + B_t} \tag{22}$$

and for Task 1-2 (binary anomaly classification) as:

$$Sensitivity = \frac{C_{c+w+b} + W_{c+w+b} + B_{c+w+b}}{C_t + W_t + B_t} \tag{23}$$

where $C_{c+w+b} = C_c+C_w+C_b$, $W_{c+w+b} = W_c+W_w+W_b$, and $B_{c+w+b} = B_c+B_w+B_b$. Then, *Specificity* can be defined

$$Specificity = \frac{N_n}{N_t} \tag{24}$$

Similarly, Task 2's confusion matrix as shown in Table 31 is considered. In this case, *C, NC* and *H* are the numbers of recordings of the three Task 2 classes. *c, nc* and *h* subscripts indicate the classification results. As before, C_t, NC_t, and H_t are the total numbers of *Chronic*, *Non-chronic*, and *Healthy* recordings, respectively.

For Task 2-1, *Sensitivity* is defined as follows:

$$Sensitivity = \frac{C_c + NC_{nc}}{C_t + NC_t} \tag{25}$$

and for Task 2-2 as:

$$Sensitivity = \frac{(C_c + C_{nc}) + (NC_c + NC_{nc})}{C_t + NC_t} \tag{26}$$

Specificity is then defined as

$$Specificity = \frac{H_h}{H_t} \tag{27}$$

Regarding the composite *ICBHI score*, this represents an equal trade-off between the two metrics and is computed in the same way for each task – namely averaging the *Sensitivity* and the *Specificity* scores.

$$ICBHIscore = \frac{1}{2}(Specificity + Sensitivity) \tag{28}$$

6.3 High-level Framework Architecture

6.3.1 High-level Description

The high-level architecture of the proposed system is described in Figure 26. The architecture is divided into two main parts: front-end feature extraction (the upper part) and back-end deep learning models (the lower part). In general, respiratory cycles in Task 1 or entire audio recording in Tasks 2 are transformed into one or more spectrogram representations. The spectrograms are then split into equal-sized image patches. During training, mixup data augmentation [71, 72] is applied to the patches to generate an expanded set of training data that is fed into a deep learning classifier.

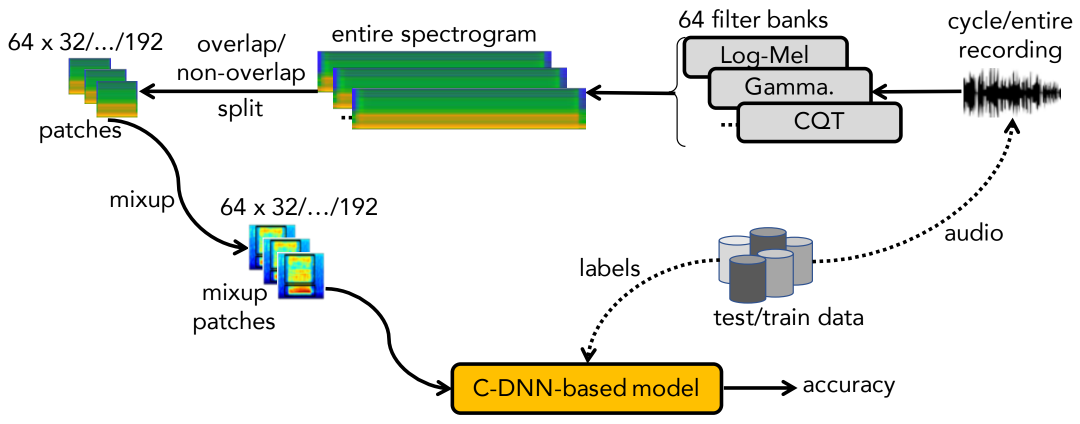

Figure 26: The high-level architecture and processing pipeline of the proposed framework.

6.3.2 Baseline System

From the high-level architecture shown in Figure 26, it can be seen that a variety of factors in front-end feature extraction could affect the performance of the classifier. These include the type of spectrogram used, the size of image patches and their degree of overlap, and the use of data augmentation. These factors are investigated, thus indicate the most influencing factors among those listed above. To limit the investigation scope to manageable proportions, the deep learning architecture assessed is constrained, thus a C-DNN baseline like VGG-7 [84] is proposed, defined below.

The main characteristics and settings of this baseline architecture are listed in Table 32, while the network architecture is presented in Table 33. During processing, all audio recordings are re-sampled (they were, as aforementioned, recorded with various sample rates) to 16000 Hz mono. Since respiratory cycle lengths differ quite widely, short cycles are repeated to ensure that inputs for Task 1 have a minimum length of 5 seconds or longer. This is of course unnecessary for Task 2 which uses entire recordings. Next, each cycle (for Task 1) or recording (for Task 2) is transformed into a spectrogram with 64 features per analysis frame. For example, the log-mel spectrogram is extracted with a window size of 1024

Table 32: Baseline system settings.

Factors	**Setting**
Re-sample	16kHz
Cycle duration (only for Task 1)	5s
Spectrogram	log-mel
Patch splitting	non-overlapped
Patch size	64×64
Data augmentation	None
Deep learning model	C-DNN based architecture

Table 33: Baseline C-DNN network architecture

Architecture	**Layers**	**Output**
	Input layer (image patch)	64×64
Conv. Block 01	BN - Cv [3×3] @ 64 - ReLU - BN - AP [2×2] - Dr (10%)	32×32×64
Conv. Block 02	BN - Cv [3×3] @ 128 - ReLU - BN - AP [2×2] - Dr (15%)	16×16×128
Conv. Block 03	BN - Cv [3×3] @ 256 - ReLU - BN - Dr (20%)	16×16×256
Conv. Block 04	BN - Cv [3×3] @ 256 - ReLU - BN - AP [2×2] - Dr (20%)	8×8×256
Conv. Block 05	BN - Cv [3×3] @ 512 - ReLU - BN - Dr (25%)	8×8×512
Conv. Block 06	BN - Cv [3×3] @ 512 - ReLU - BN - GAP - Dr (25%)	512
Dense Block	FC - Softmax	C

samples, a hop size of 256 samples, and 2048-point FFT, followed by average pooling in the frequency direction to yield a spectrogram with 64 frequency bins. Whichever type of spectrogram is used, the resulting time-frequency output is split into square non-overlapping patches of size 64×64. Since data augmentation is one of factors evaluated, this technique is not applied to the baseline system.

As can be seen from Table 33, the network architecture consists of seven blocks – six are convolutional and one is a dense block. The former blocks comprise batch normalization (BN) layers, convolutional (Cv [kernel size] @ kernel number) layers, rectified linear units (ReLU), average pooling (AP [kernel size]) and global average pooling (GAP) layers, and use dropout (Dr (dropout percentage)). The dense block comprises a fully connected (FC), and a final Softmax layer for classification. C refers to the number of classes, which depend on the specific task being evaluated (i.e. two separate C-DNN models are trained and test with C set to 4 and 3 for Tasks 1 and 2, respectively).

6.3.3 Experimental Settings for The Baseline System

All the systems are implemented using TensorFlow. Network training makes use of the Adam optimiser [102] with 100 training epochs, a mini batch size of 100, and cross entropy loss:

$$LOSS_{EN}(\Theta) = -\sum_{c=1}^{C} y_c \log\{\hat{y}_c(\Theta)\} + \frac{\lambda}{2}||\Theta||_2^2, \tag{29}$$

where Θ are all trainable parameters, C is the number of categories classified, and constant λ is empirically set to 0.001. y_c and $\hat{y}_c$ denote ground truth and predicted results of class c, respectively.

An entire spectrogram or cycle is separated into smaller patches and applied patch-by-patch to the C-DNN model which then returns the probability computed over each patch. The probability of an entire spectrogram is the average of all patches' probabilities. Let us consider $\mathbf{p}^n = (p_1^n, p_2^n, \ldots, p_C^n)$ the probability obtained from the n^{th} out of N patches. Then, the mean probability of a test sound instance is denoted as $\bar{\mathbf{p}} = (\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_C)$ where

$$\bar{p}_c = \frac{1}{N}\sum_{n=1}^{N} p_c^n \tag{30}$$

The predicted label $\hat{y}$ is then determined as

$$\hat{y} = \underset{c\in\{1,2,\ldots,C\}}{\text{argmax}}\ \bar{p}_c. \tag{31}$$

6.4 Analysis of Influencing Factors

By using the baseline system mentioned above, the impact of various factors on performance is investigated below.

6.4.1 Influence of Spectrogram Type

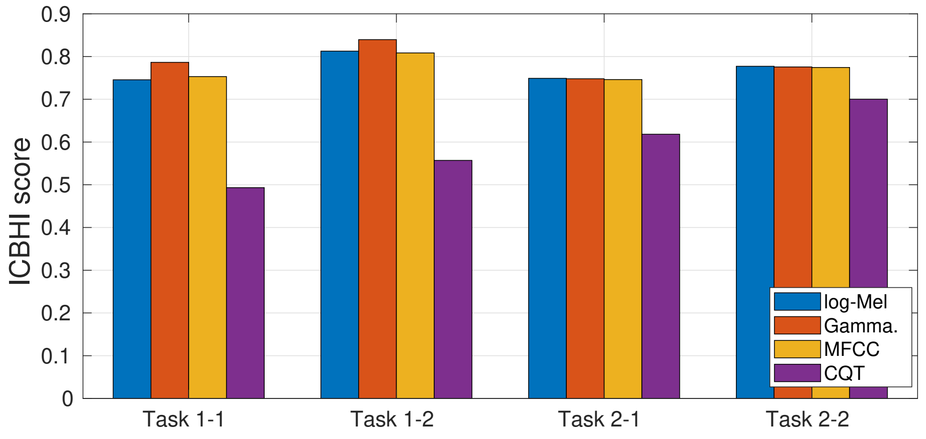

Figure 27: Comparison of baseline performance using different spectrograms.

From previous work on natural sound datasets [29, 21], it is clear that the choice of spectrogram is one of the most important factors that affects final classification accuracy. Therefore, the effect of spectrogram types on ICBHI performance for each task is evaluated. To this end, all settings as described in Table 32 are maintained, but four spectrogram types: log-mel spectrogram, Gamma spectrogram, stacked Mel-Frequency Cepstral Coefficients (MFCC), and rectangular Constant Q Transform (CQT) spectrogram are used. Each of the spectrogram types is evaluated on all four subtasks. Just as in the experiments presented in Chapter 3, while log-mel, MFCC, and CQT spectrograms are generated by using the Librosa toolbox [99], and the Gamma spectrogram by [100] (note that detailed computation of these spectrograms are described in the Appendix).

The obtained results in terms of ICBHI Score are shown in Figure 27, revealing that MFCC, log-mel, and Gamma spectrogram perform competitively, and are much better than CQT for all subtasks. Compared to log-mel, Gamma spectrogram results achieve an improvement of 0.04 for Task 1-1 and 0.03 for Task 1-2. However log-mel slightly outperforms its Gamma counterpart for Task 2. MFCC is, meanwhile, better than log-mel in Task 1-1 (0.01) but the opposite is seen for all other subtasks.

These results suggest that the Gamma spectrogram is optimal for anomaly cycle classification (Task 1) while the log-mel spectrogram works best for detection of respiratory diseases (Task 2). As a result these two spectrograms are adopted in the following experiments for those respective tasks.

6.4.2 Influence of the overlapping degree

Table 34: Baseline performance loss or gain on each subtask when overlapping spectrogram patches are used (ICBHI Score).

Patches	**Task 1-1**	**Task 1-2**	**Task 2-1**	**Task 2-2**
No overlap	**0.79**	**0.84**	0.75	0.77
Overlap	0.78	0.83	**0.77**	**0.79**

As the spectrogram of an entire cycle or audio recoding is large in temporal dimension and is of variable length, they are split into smaller patches of 64×64 for presentation before feeding to the back-end deep learning models. In traditional signal processing systems, overlapping analysis windows are used to prevent occlusion of important features in the original data by edge effects. Therefore, the effect of overlapped or non-overlapped patches on ICBHI performance is examined in this section. Specifically, the baseline with non-overlapping patches (the settings in Table 32) is contrasted to the system with patches overlapped by 50% (note that Gamma and log-mel are applied on Task 1 and Task 2, respectively).

Results shown in Table 34 reveal that Task 1 performs better with non-overlapped patches (subtask scores of 0.79 and 0.84, respectively) while those results for Task 2 performs better with overlapped patches (subtask scores of 0.77 and 0.79, respectively). These results can be explained by two potential factors: firstly different spectrogram types were used in the two tasks, and secondly Task 1 repeats respiratory cycles, whereas Task 2 classifies unrepeated recordings.

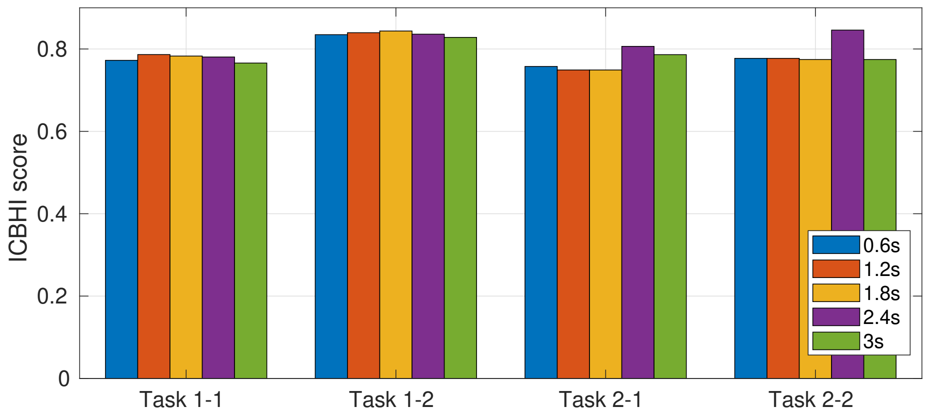

Figure 28: Performance comparison between different time resolutions on each task.

6.4.3 Influence of Time Resolution

The baseline network operates on fixed-size patches where the time span encoded in each patch is defined by its horizontal dimension and sampling rate. Features are presented sequentially, and therefore the time span also defines the temporal resolution of features presented to the classifier. In this section, the effect of different temporal resolution is explored by adjusting patch widths to 0.6 s, 1.2 s, 1.8 s, 2.4 s, and 3.0 s. This is achieved by changing the patch size to be 64×32, 64×64, 64×96, 64×128, and 64×160, respectively, then repeat the experiments for each of them. Note that all settings are reused from Table 32 with exception that Gamma and log-mel spectrograms are used for Task 1 and Task 2, respectively. The frequency resolution (vertical dimension) remains unchanged in each case. The dimension of the network input layer is increased or decreased to accommodate the differing time resolution.

The obtained results are shown in Figure 28 for the four subtasks. As can be seen, patch size of 64×64 (i.e. 1.2 s time resolution) as in the baseline system performs best for Task 1-1 and second best for Task 1-2 (achieving 0.79 and 0.84, respectively). However, a double sized patch, 64×128 (i.e. 2.4 s time resolution) is clearly the best for Tasks 2-1 and 2-2 (achieving 0.81 and 0.85, respectively).

6.4.4 Influence of Data Augmentation

Table 35: Performance (ICBHI Score) with and without mixup data augmentation.

	Task 1-1	**Task 1-2**	**Task 2-1**	**Task 2-2**
Non-mixup	0.79	0.84	0.75	0.77
mixup	**0.80**	**0.85**	**0.84**	**0.85**

Data augmentation (DA) has been shown useful to improve the learning ability of deep learning models in tasks involving natural sound classification [21, 29]. Therefore, DA in the form of mixup [71, 72] is applied and evaluated its effect on respiratory sound classification.

By using two types of Uniform or Beta Distribution to generate mixing coefficient α, this doubles the data size and hence, the training time. Note that in Task 1, the DA mixes the *Normal* class with one of the other classes (since there is already one mixed class in the dataset, i.e. *Crackle & Wheeze*), whereas it randomly mixes samples of all classes for Task 2. After mixup, the generated patches are shuffled and fed into the C-DNN baseline. Since the labels $\mathbf{y}_{mp1}$ and $\mathbf{y}_{mp2}$ of the resulting patches are no longer one-hot encoded, it is, therefore, necessary to replace the cross-entropy loss by the Kullback-Leibler (KL) divergence loss [116]:

$$LOSS_{KL}(\Theta) = \sum_{c=1}^{C} y_c \log \left\{ \frac{y_c}{\hat{y}_c(\Theta)} \right\} + \frac{\lambda}{2} ||\Theta||_2^2. \tag{32}$$

Again, Θ denotes the trainable network parameters and λ denote the ℓ_2-norm regularization coefficient, set to 0.001. C is the number of categories classified, y_c and $\hat{c}_n$ denote the ground-truth and the network output at class c, respectively.

Using the settings in Table 32 with Gamma spectrogram in Task 1 and the log-mel spectrogram in Task 2, the improvement over the baseline ICBHI score for each subtask due to mixup data augmentation can be assessed. Results shown in Table 35 indicate that mixup data augmentation substantially improves the ICBHI score in Task 2 by 0.09 and 0.08 on Tasks 2-1 and 2-2, respectively. However, modest improvements are seen for Task 1.

6.5 Enhanced Deep Learning Framework

Table 36: Deep learning frameworks for Tasks 1 and 2.

Factors	**Anomaly cycle classification**	**Respiratory disease detection**
Resample	16000 Hz	16000 Hz
Cycle duration	5s	N/A
Spectrogram	Gamma	log-mel
Patch splitting	non-overlapped	overlapped
Patch size	64×64	64×128
Data augmentation	Yes	Yes

From the analysis of influencing factors presented above, two systems are proposed. One for Task 1 anomaly cycle classification, and the other for Task 2 respiratory disease detection, both summarised in Table 36. In this section, the performance of the C-DNN architecture is enhanced by incorporating a mixture-of-experts (MoE) technique into the DNN part of the network, leading to a CNN-MoE architecture, similar to that in Chapter 5.

6.5.1 CNN-MoE Network Architecture

According to the C-DNN architecture entailed in Table 33, the first six convolutional blocks are used to map the image patch input to condensed and discriminative embeddings, often referred to as high-level features. The features are then classified by a dense block comprising a fully-connected layer and Softmax. On the basis that the embedding may contain more information than a single fully connected layer can unlock, the dense block in replaced by a mixture-of-experts (MoE) block as shown in Figure 29. The MoE block architecture is reused and mentioned in Section 4.2.2

The proposed systems, as defined in Table 36, are trained with KL-divergence loss [116] (due to the use of mixup data augmentation) and use the same training settings as the previous experiments with the C-DNN baseline.

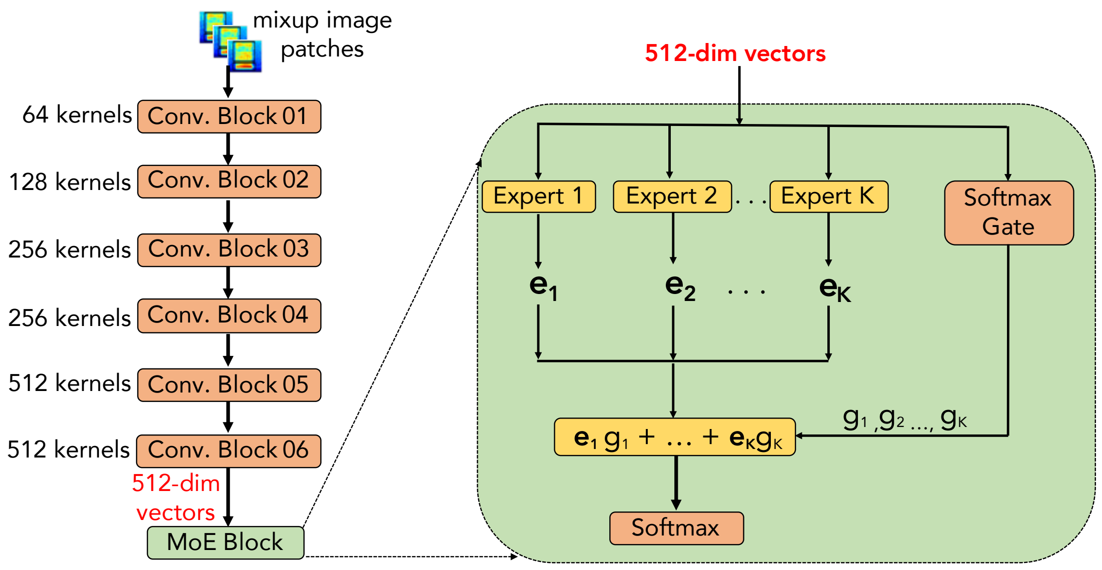

Figure 29: The proposed CNN-MoE architecture.

6.5.2 Performance Comparison

This section firstly compares the performance of using C-DNN and CNN-MoE, analyses if MoE technique is effective to improve the classification accuracy. Next, the best systems proposed are compared to the state of the art.

Comparing C-DNN to CNN-MoE: The efficiency of the MoE technique (experimentally using K=10 experts) is evaluated and compared to the C-DNN system, reporting the performance of both in Table 37 (note that both the systems follows the settings in Table 36, with the back-end classifier being either C-DNN or CNN-MoE – there are thus eight systems in total, two C-DNNs and two CNN-MoEs for each kind of data split). The results in Table 37 clearly indicate that the CNN-MoE systems perform best overall. Although only marginal gains is seen over the C-DNN for Task 1, results in improvement with a margin as large as 0.06 absolute in terms of ICBHI score with both the data splits, 5-fold cross validation and ICBHI challenge's data split, in Task 2.

Comparing to state-of-the-art systems: Next, the proposed framework is contrasted to state-of-the-art systems. For each task, challenge's data split is

Table 37: ICBHI score comparison between the C-DNN and CNN-MoE frameworks over 5-fold cross validation and ICBHI challenge splitting (highest scores in ***bold****).*

Tasks	**C-DNN** (5-fold)	**CNN-MoE** (5-fold)	**C-DNN** (ICBHI)	**CNN-MoE** (ICBHI)
1-1, 4-category	0.77	**0.79**	0.43	**0.47**
1-2, 2-category	0.84	**0.84**	0.53	**0.54**
2-1, 3-category	84.7	**0.91**	0.79	**0.84**
2-2, 2-category	0.86	**0.92**	0.79	**0.84**

Table 38: Comparison against state-of-the-art systems with ICBHI challenge splitting (highest scores in ***bold****).*

Tasks	**Method**	**Spec.**	**Sen.**	**Score**
1-1, 4-category	DT [173]	0.75	0.12	0.43
1-1, 4-category	HMM [170]	0.38	**0.41**	0.39
1-1, 4-category	SVM [172]	0.78	0.20	0.47
1-1, 4-category	CNN-RNN [184]	**0.81**	0.28	**0.54**
1-1, 4-category	**Our system**	0.68	0.26	0.47

evaluated twice – once with the ICBHI challenge train/test split, and once with random splitting (as described in Section 6.2.3). Considering the first splitting method specified in the ICBHI challenge, Table 38 presents scores obtained by the proposed framework and state-of-the-art published systems (where available). It is noted that the proposed framework lies second in terms of Task 1-1 evaluation. Our results for other subtasks were listed in Table 37. Only Task 2-2 is found in the literature (for the ICBHI data split) achieved 0.72 [178], which is surpassed by 0.84 obtained by our system.

Table 39 compares the performance obtained by our system with previously published results that use the random train/test splitting method. For Tasks 1-1 and 1-2, the proposed framework clearly outperforms other systems quite consistently. Meanwhile for Task 2-1 and 2-2 the proposed method also outperforms other systems in terms of overall ICBHI score, but not necessarily simultaneously for both subcomponents of specificity or sensitivity.

*Table 39: Performance comparison between the proposed system and state-of-the-art systems following random splitting (highest scores are highlighted in **bold**).*

Tasks	Methods	train/test	Spec.	Sen.	Score
1-1, 4-category	Boosted DT [164]	60/40	0.78	0.21	0.49
1-1, 4-category	CNN [180]	80/20	0.77	0.45	0.61
1-1, 4-category	CNN-RNN [178]	5 folds	0.84	0.49	0.66
1-1, 4-category	LSTM [181]	80/20	0.85	0.62	0.74
1-1, 4-category	**Our system**	5 folds	**0.90**	**0.68**	**0.79**
1-2, 2-category	Boosted DT [164]	60/40	0.78	0.33	0.56
1-2, 2-category	LSTM [181]	80/20	-	-	0.81
1-2, 2-category	CNN [177]	75/25	-	-	0.82
1-2, 2-category	**Our system**	5 folds	**0.90**	**0.78**	**0.84**
2-1, 3-category	CNN [180]	80/20	0.76	0.89	0.83
2-1, 3-category	LSTM [181]	80/20	0.82	**0.98**	0.90
2-1, 3-category	**Our system**	5 folds	**0.86**	0.95	**0.91**
2-2, 2-category	Boosted DT [164]	60/40	0.85	0.85	0.85
2-2, 2-category	CNN [180]	80/20	0.78	0.97	0.88
2-2, 2-category	RUSBoost DT [162]	50/50	**0.93**	0.86	0.90
2-2, 2-category	LSTM [181]	80/20	0.82	**0.99**	0.91
2-2, 2-category	**Our system**	5 folds	0.86	0.98	**0.92**

6.5.3 Discussion

Comparing Tables 38 and 39, it is notable that those systems following the ICBHI data split (i.e. recordings from the same patient are never found in both train/test subsets) exhibit considerably lower performance over all tasks than those following random splitting. This indicates that the ICBHI dataset presents a very high dependence on patient characteristics, which is likely make respiratory cycle classification challenging in practice.

However, all the results obtained by the proposed framework for Tasks 2-1 and 2-2 (with both splitting methods) exceed 84%. These results for recording-based classification of lung disease – which is highly related to the overall aim of lung disease detection – provide a strong indicator of the robustness of the proposed framework. As does the fact that the same proposed framework is capable of performing well for all subtasks.

6.6 Student-Teacher Scheme for Respiratory Disease Detection

6.6.1 The Proposed Student-Teacher Arrangement

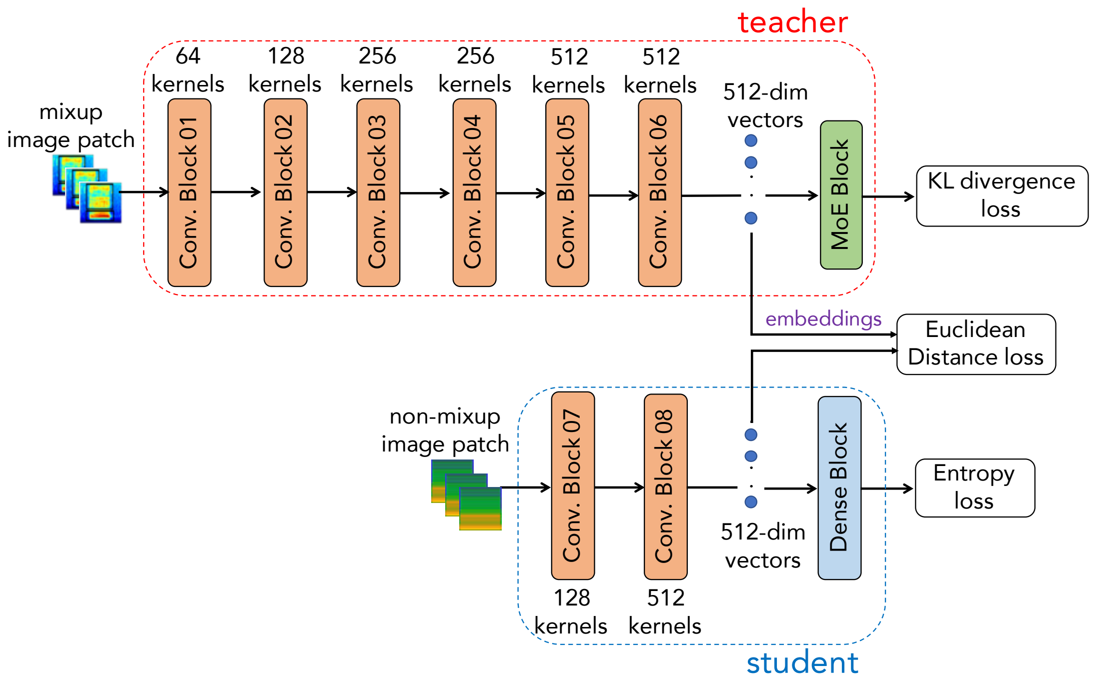

Figure 30: Architecture of the Student-Teacher scheme.

Recent works on sound scene and sound event detection reported the effectiveness of Teacher-Student learning schemes [186, 187]. Among other advantages, these schemes offer a trade-off between model size and performance. Since the complexity of our best model based on the proposed MoE framework may be a barrier to future real-time implementation, it is explored whether a student-teacher scheme can be used to train a network with much lower complexity and perform well on the task of respiratory disease detection (Task 2).

The proposed solution, as shown in Figure 30, comprises two networks, namely the Teacher and the Student. The teacher network re-uses the high-performance CNN-MoE architecture introduced in Section 6.5.1. The student network features

Table 40: The Student network architecture.

Architecture	**Layers**	**Output**
	Input layer (image patch)	64×128
Conv. Block 07	Cv [3×3] @ 128 - ReLU - AP [4×4]	16×32×128
Conv. Block 08	Cv [3×3] @ 512 - ReLU - GAP	512
Dense Block	FC - Softmax	3

a compact architecture, comprising two convolutional blocks (identified *Conv. Block 07* and *Conv. Block 08* in the figure), and a dense block whose configuration is the same as the one in Table 40 (note that the student network does not apply batch normalisation, dropout or mixup data augmentation).

Training the Teacher-Student network is separated into two phases. First, the Teacher is trained as usual. Afterwards, the Teacher's embedding is distilled to the Student's embedding to assist in the Student's learning process. The influence of this knowledge distillation on the student network's performance is empirically investigated. With the presence of this knowledge distillation, training the student network, therefore, aims to minimize two losses: (1) the Euclidean distance $LOSS_{EU}$ between the teacher and student embedding, and (2) the standard cross-entropy loss $LOSS_{EN}$ on the student's classification output. The combined loss function is therefore,

$$LOSS = (1 - \gamma)LOSS_{EN} + \gamma LOSS_{EU} \tag{33}$$

Here, the hyperparameter γ is empirically set to 0.5 to balance the two constituent losses. Other hyper-parameters and settings are inherited from Section 6.5.1.

6.6.2 Results From the Teacher-Student Scheme

The experimental results obtained by the student network in comparison with the teacher network are shown in Table 41. On the one hand, it can be seen that without knowledge distillation from the teacher network, the small-footprint student network obtains a substantially low specificity score, although it maintains

Table 41: Performance comparison between Teacher and Student with and without knowledge distillation.

		Five-fold random split			**ICBHI split**		
Tasks	**Models**	**Spec.**	**Sen.**	**ICBHI Score**	**Spec.**	**Sen.**	**ICBHI Score**
2-1, 3-category	Teacher	0.86	0.95	0.91	0.71	0.98	0.84
	Student w/o knowledge distill	0.43	0.94	0.68	0.41	0.97	0.69
	Student w/ knowledge distill	0.86	0.90	0.88	0.71	0.98	0.84
2-2, 2-category	Teacher	0.86	0.98	0.92	0.71	0.98	0.84
	Student w/o knowledge distill	0.43	0.99	0.71	0.41	0.99	0.70
	Student w/ knowledge distill	0.86	0.96	0.91	0.71	0.98	0.84

Table 42: Model footprint comparison between Teacher and Student

Features	**Teacher**	**Student**
Trainable Convolutional Layers	6	2
Trainable Fully-connected Layers	11	1
Batch normalization	12	0
Number of trainable parameters	4.5×10^6	0.6×10^6
Number of MAC operations	44,886 K	9,513 K

a very good sensitivity. This observation is consistent with the overall ICBHI score and can be explained by the simplicity of the network which results in low learning capacity. On the other hand, distilling knowledge from the teacher significantly boosts the student performance, yielding specificity, sensitivity, and ICBHI scores that are very competitive to those of the teacher network – even though the student network is much smaller and simpler.

Details of the model footprint are shown in Table 42, it can be seen that the Teacher uses six convolutional layers, eleven fully-connected layers and twelve batch normalization layers that together contribute to a large model size with 4.5×106 trainable parameters. Meanwhile, the Student only uses two convolutional layers and one fully-connected layer, requiring only 0.6×106 parameters, approximately one-seventh of the Teacher's. The model footprints also scale in terms of computational cost of multiply-accumulate (MAC) operations during inference. While an inference process on the Teacher costs 44,886 kMAC operations, the Student only costs 9,513 kMAC (the MAC operation computation for a deep learning network is presented in [188]). The inference process for a 20-second long recording in Task 1-1, conducted by a Tesla P100 GPU, takes 0.5 second;

nearly ten times longer than the 0.045 second required for the Student's inference process.

6.7 Conclusion

This chapter has presented a robust deep learning framework for the analysis of respiratory anomalies and detection of lung diseases from lung auscultation recordings. Extensive experiments were conducted with different architectures and system settings using the ICBHI dataset, and two defined tasks related to that. The proposed system is evaluated against existing state-of-the-art methods, outperforming them for most of the challenge tasks. Furthermore, to facilitate implementation in real-time systems, a Teacher-Student learning scheme was employed to significantly reduce model complexity while still achieving very high accuracy. The final experimental results validate the application of deep learning for the timely diagnosis of respiratory diseases, bringing this research area one step closer to clinical applications.

Chapter 7

Conclusion and Future Work

7.1 Summary

Concretely, this thesis has focused on dealing with the task of acoustic scene classification (ASC), and then applied the techniques developed for ASC to a real-life application of detecting respiratory disease.

To deal with ASC challenges, this thesis addresses three main factors that directly affect the performance of an ASC system. Firstly, this thesis explores input features by making use of multiple spectrograms (log-mel, Gamma, and CQT) for low-level feature extraction to tackle the issue of insufficiently discriminative or descriptive input features. Next, a novel *Encoder* network architecture is introduced. The *Encoder* firstly transforms each low-level spectrogram into high-level intermediate features, or embeddings, and thus combines these high-level features to form a very distinct composite feature. The composite or combined feature is then explored in terms of classification performance, with different *Decoders* such as Random Forest (RF), Multilayer Perception (MLP), and Mixture of Experts (MoE). By using this *Encoder-Decoder* framework, it helps to reduce the computation cost of the reference process in ASC systems which make use of multiple spectrogram inputs. Inspired by high-cross correlation among sound categories, and the potentially useful information that this might yield, the architecture is further explored, and a hierarchical classification, referred as to *Decoder*, is proposed to make use of that. The scheme helps to structure the original "flat" ASC task into multiple hierarchical sub-tasks that operates in a divide-and-conquer manner. Since each sub-task is only suitable for some sound categories, a combination of triplet loss and cross entropy loss proves effective to enhance the classification accuracy. To evaluate the *Encoder-Decoder* framework, recently published datasets (Litis Rouen and DCASE 2016 Task 1A, DCASE 2017 Task 1A, DCASE 2018 Task 1A & 1B, DCASE 2019 Task 1A & 1B) are used, and demonstrate very competitive results compared to the state-of-the-art systems. Additionally, the obtained results also indicate that the framework is effective for early detection of sound scenes, which is potentially very useful for real-time applications on edge devices. The results strongly demonstrate that the proposed *Encoder-Decoder* framework is robust for ASC tasks.

Since the proposed techniques applied for general ASC tasks were shown to be highly effective, this inspired an application to a specific real-life application. This was namely the 2017 Internal Conference on Biomedical Health Informatics (ICBHI) respiratory sound dataset. Building upon the proposed ASC framework, the ICBHI tasks were tackled with a deep learning framework, and the resulting system shown to be capable at detecting respiratory anomaly cycles and diseases. The experimental results obtained validated the deep learning techniques used in the general ASC task for the timely diagnosis of respiratory diseases, thus potentially for a wide ranges of clinical applications.

7.2 Future Work

Applying deep learning techniques developed for ASC tasks for specific applications, like the respiratory disease detection task mentioned in Chapter 6, has promise for a number of audio detection problems. However, the challenge to implement complicated deep learning frameworks on edge devices (e.g. mobile and low powered hardware) is the high degree of complexity of the computation required. To deal with such challenges, model compression techniques have drawn increasing attention in recent years. Two main approaches of compression are quantization and pruning. Recently, the Tensorflow framework 2.0 provides a complete guide for both the compression methods mentioned in [189]. The toolbox is very usable and is quick to compress an originally complicated model to a more simple Tensorflow lite model that can be suitable for embedding on a wide range of embedded operating systems. Therefore, one item of future work proposed in this thesis is to implement the deep learning framework applied for respiratory disease detection in Chapter 6 on an embedded or edge hardware platform. Successfully achieving a real-time system for detecting respiratory disease will help patients to self-observe their situation, reduce the cost of fabrication and possibly increase the scale of respiratory disease detection.

As mentioned in Chapter 2, acoustic scene classification (ASC) and acoustic event detection (AED) are two main tasks of the emerging 'machine hearing' research field [3]. Currently, these two tasks are considered to be separate. In particular, while ASC datasets are easily recorded in nature, it is hard to collect AED datasets in real-life environments. Therefore, current AED datasets were synthesised which enable AED and ASC tasks in challenges to be independent yet share the same underlying database. Furthermore, sound events follow certain structures, but this is not always true for sound scenes. Techniques applied to detect sound events and sound scenes are therefore basically different. While CNN-based architectures are explored for sound scenes, analysis of sound events tend to use RNN-based networks which are robust for time sequence information. However, sound context awareness abilities integrated in certain devices in the future should be base on both ASC and AED techniques to achieve high performance. For instance, quite environments such as *in park* or *in home* are very challenging to detect if only based on static sound scene information. However, such quite environments are easier to recognize if specific sound events can be detected, such as *bird song* in a park or sound of *tap water* inside a home. Therefore, if both sound events and scenes can be integrated into sound-based systems, it would be effective at improving sound context awareness. From the analysis above, another idea for future work is exploring the high-cross correlation between ASC and AED to improve the sound context awareness of future sound-based systems.

Appendix A

Mathematical definitions

This appendix provides the detailed derivation of the spectrogram transformations, namely the log-mel, CQT and Gamma used in experiments in Chapter 3, 4, 5, and MFCC mentioned in Chapter 6. Additionally, this appendix details the computation of network layers used in this thesis such as convolutional (Cv), batch normalization (BN), rectify linear unit (ReLU), dropout (Dr), fully connected (FC), and Softmax layers. Note that "$\times$" is used for matrix product.

A.1 Spectrogram Computation

A.1.1 STFT spectrogram

The Short-Time Fourier Transform (SFFT) applies a Fourier Transform to a frame of the time series signal to extract the frequency content of the local section of analysed input. If $\mathbf{s}(n)$ is considered as the digital audio signal, then the STFT spectrogram, denoted by $\mathbf{STFT}[F, T]$, is computed as,

$$\mathbf{STFT} = \sum_{n=1}^{N} \mathbf{s}[n]\mathbf{w}[n]e^{-j2\pi Fn/N} \tag{34}$$

where $\mathbf{w}[n]$ is a window function, typically Hamming. While time resolution (T) of the STFT spectrogram is set by the hop size, the frequency (F) resolution depends on window length and the sample rate of the audio signal.

A.1.2 log-mel spectrogram

To generate a log-mel spectrogram, the section of time series audio being analysed is first transformed into an STFT spectrogram as noted above. Next, a Mel filter bank, which simulates the overall frequency selectivity of the human auditory system is applied. The filter bank uses the frequency warping $F_{mel} = 2595.log_{10}(1+F/700)$ [2] to generate a Mel spectrogram $\mathbf{MEL}[F_{mel}, T]$ (note that frequency resolution F_{mel} depends on the number of Mel filters). Logarithmic scaling is then applied to obtain the log-mel spectrogram.

If $\mathbf{COE_{MEL}}[F_{mel}, F]$ is considered as a matrix storing coefficients of the Mel filters, then log-mel spectrogram $\mathbf{LOG\_MEL}[F_{mel}, T]$ is a matrix computed by a multiplication of the two matrices as follows,

$$\mathbf{LOG\_MEL}[F_{mel}, T] = log_{10}\left(\mathbf{COE_{MEL}}[F_{mel}, F] \times \mathbf{STFT}[F, T]\right) \tag{35}$$

A.1.3 MFCC spectrogram

From log-Mel spectrogram, Discrete Cosine Transform (DCT) is used to extract a sequence of uncorrelated coefficients crossing frequency dimension, reducing log-Mel frequency resolution into smaller space. A pixel value $dct[f_{dct}, t_{dct}]$ of DCT matrix $\mathbf{DCT}[F_{dct}, T_{dct}]$, where F_{dct} and T_{dct} are frequency and time resolutions, is computed by:

$$\begin{aligned} dct[f_{dct}, t_{dct}] = & \left(\frac{2}{F_{mel}}\right)^{\frac{1}{2}} \left(\frac{2}{T}\right)^{\frac{1}{2}} \sum_{f_{mel}=0}^{F_{mel}-1} \sum_{t=0}^{T-1} \\ & \Lambda(f_{mel}) cos\left[\frac{\pi f_{dct}}{F_{mel}}(2f_{mel}+1)\right] \\ & \Lambda(t) cos\left[\frac{\pi t_{dct}}{T}(2t+1)\right] \\ & log{-}mel[f_{mel}, t] \end{aligned} \tag{36}$$

where

$$\Lambda(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases} \tag{37}$$

T, F_{mel}, and $log{-}mel[f_{mel}, t]$ are time resolution, frequency resolution, and magnitude of a pixel of log-mel spectrogram, respectively.

A.1.4 Gamma spectrogram

Gammatone filters are designed to model the frequency-selective cochlea activation response of the human inner ear [190], in which filter output simulates the frequency response of the basilar membrane. The impulse response is given by

$$g[k] = k^{P-1}T^{P-1}e^{-2b\pi kT}cos(2\pi fkT + \theta) \tag{38}$$

where P is the filter order, θ is the phase of the carrier, b is filter bandwidth, and f is central frequency, and T is sampling period. The filter bank is then formulated on the equivalent rectangular bandwidth (ERB) scale [191] as

$$ERB = 24.7(4.37.10^{-3}f + 1) \tag{39}$$

To quickly and conveniently generate the gammatonegram, Ellis *et al.* [100] introduced a toolbox which first transforms the audio signal into STFT spectra as mentioned above. Then, a matrix of gammatone weighting $\mathbf{COE_{GAMMA}}[F_{gamma}, F]$ is applied to the STFT to obtain the Gamma spectrogram.

$$\mathbf{GAMMA}[F_{gamma}, T] = \mathbf{COE_{GAMMA}}[F_{gamma}, F] \times \mathbf{STFT}[F, T] \tag{40}$$

where F_{gamma} is frequency resolution the depends on the number of gammatone filters used.

A.1.5 Constant Q Transform (CQT)

The CQT applies a bank of filters corresponding to tonal spacing, where each filter is equivalent to a subdivision of an octave, with central frequencies given by,

$$F_k = (2^{\frac{1}{b}})^k f_{min} \tag{41}$$

where F_k denotes the frequency of the kth spectral component, f_{min} is the minimum frequency, and b is the number of filters per octave. As the name suggests, the Q value (which is commonly known to be the ratio of central frequency to bandwidth in electrical and control systems), is set to a constant as in,

$$Q = \frac{F_k}{\Delta F_k} = \frac{F_k}{F_{k+1} - F_k} = \left(2^{\frac{1}{b}} - 1\right)^{-1} \tag{42}$$

Like the STFT, the CQT spectrogram $\mathbf{CQT}[F_k, T]$ is extracted using Fourier-based transformation,

$$\mathbf{CQT} = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} \mathbf{s}[n]\mathbf{w}[k, n] e^{-i2\pi \frac{nQ}{N(k)}} \tag{43}$$

where

$$N(k) = Q\frac{fs}{F_k} \tag{44}$$

$$\mathbf{w}[k,n] = \alpha + (1-\alpha)cos\frac{2\pi n}{N(k)-1} \tag{45}$$

and fs is the sample rate of the input signal.

A.2 Computation of Network Layers

if $\mathbf{X}[\mathbf{F},\mathbf{T},\mathbf{C}] \in \mathbb{R}^{\mathbb{F}\times\mathbb{T}\times\mathbb{C}}$ and $\mathbf{x} \in \mathbb{R}^{\mathbb{N}}$ are considered as an input tensor and input vector, where F, T, C are frequency, time, channel dimensions of the input tensor and N is the dimension of input vector, the computation of network layers used in thesis is described in detail as below,

A.2.1 Rectify Linear Unit (ReLU) Layer

Relu function takes each pixel $x_{f,t,c}$ of input tensor $\mathbf{X}$ and thus returns the output $f_{Relu}(x_{f,t,c})$ as,

$$f_{ReLU}(x_{f,t,c}) = \begin{cases} x_{f,t,c}, & \text{if } x_{f,t,c} > 0 \\ 0, & \text{if } x_{f,t,c} <= 0 \end{cases} \tag{46}$$

As regards input vector $\mathbf{x}[x_1,...,x_N]$, this function takes x_n and returns

$$f_{ReLU}(x_n) = \begin{cases} x_n, & \text{if } x_n > 0 \\ 0, & \text{if } x_n <= 0 \end{cases} \tag{47}$$

A.2.2 Dropout Layer

Dropout function f_{Drop} for input tensor $\mathbf{X}$ is computed by,

$$f_{Drop}(\mathbf{X}) = \mathbf{X}\mathbf{D} \tag{48}$$

and for input vector $\mathbf{x}$ as

$$f_{Drop}(\mathbf{x}) = \mathbf{x}\mathbf{d} \tag{49}$$

where $\mathbf{D} \in \mathbb{R}^{\mathbb{F}\times\mathbb{T}\times\mathbb{C}}$ is a matrix with similar dimension of input tensor $\mathbf{X}$ and $\mathbf{d} \in \mathbb{R}^{\mathbb{N}}$ is a vector with similar dimension of input vector $\mathbf{x}$. $\mathbf{D}$ and $\mathbf{d}$ are generated by $\mathbf{D}/\mathbf{d} \sim \text{Bernoulli}(1-p)$ where p is the percentage of input dropped.

A.2.3 Batch Normalization Layer

If a batch of B tensor is described by $\mathbf{X} = \{\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_B}\}$, where $\mathbf{X_b}$ is a tensor input, the batch normalization function takes a tensor $\mathbf{X_b}$, and thus returns an output $f_{Batch}(\mathbf{X_b})$ as

$$f_{Batch}(\mathbf{X_b}) = \lambda \frac{\mathbf{X_b} - \mu}{\sqrt{\sigma + \epsilon}} \tag{50}$$

where μ and σ are defined by

$$\mu = \frac{1}{B}\sum_{b=1}^{B} \mathbf{X_b} \tag{51}$$

$$\sigma = \frac{1}{B}\sum_{b=1}^{B} (\mathbf{X_b} - \mu)^{\mathbf{2}} \tag{52}$$

and λ and ϵ are scale and ship parameters that are learned during training process. In this thesis, the batch normalization function is only applied over input tensors and across the channel dimension.

A.2.4 Fully Connected Layer

In this thesis, the fully-connected layer is applied for input vector. The output vector $\mathbf{y}[y_1, y_2, ..., y_M]$ with output dimension of M is computed by

$$y_m = \sum_{n=0}^{N} w_{m,n} x_n + b_m \tag{53}$$

where pixel $w_{m,n}$ in the coefficient matrix $\mathbf{W}[\mathbf{M}, \mathbf{N}]$ and b_m in bias vector $\mathbf{b}[b_1, ..., b_M]$ are trainable parameters.

A.2.5 Softmax Layer

In this thesis, the Softmax layer is applied on input vectors. The output vector $\mathbf{y} = [y_1, y_2, ..., y_N]$ of this layer is computed by

$$y_n = \frac{\exp(x_n)}{\sum_{n=1}^{N} \exp(x_n)} \tag{54}$$

Note that the output and input vectors of Softmax layer have same dimension of N.

A.2.6 Convolutional Layer

In this thesis, convolutional layer is applied on input tensors. if C' kernels with size of $[K, P]$ are applied on the input tensor $\mathbf{X}[\mathbf{F}, \mathbf{T}, \mathbf{C}]$ in a convolutional layer, the output tensor has size of $\mathbf{Y}[\mathbf{F}, \mathbf{T}, \mathbf{C'}]$ (note that the frequency F and time T dimensions are remained by adding zero padding), a pixel $y_{f,t,c'}$ of output tensor $\mathbf{Y}$ is computed by,

$$y_{f,t,c'} = \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{c=1}^{C} w_{k,p,c,c'} x_{k,p,c} + b_{c'} \tag{55}$$

where pixel $x_{k,p,c}$ is in tensor input $\mathbf{X}[\mathbf{F}, \mathbf{T}, \mathbf{C}]$; pixel $w_{k,p,c,c'}$ in coefficient matrix $\mathbf{W}[\mathbf{K}, \mathbf{P}, \mathbf{C}, \mathbf{C}']$ and $b_{c'}$ in bias vector $\mathbf{b} = [b_1, ..., b_{C'}]$ are trainable parameters.

Bibliography

[1] B. Rocha, D. Filos, L. Mendes, Vogiatzis, *et al.*, "A respiratory sound database for the development of automated classification," in *Precision Medicine Powered by pHealth and Connected Health*, pp. 33–37, 2018.

[2] I. V. McLoughlin, *Speech And Audio Processing: A MATLAB®-based Approach.* Cambridge University Press.

[3] F. Richard, *Human And Machine Hearing.* Cambridge University Press.

[4] S. Ravindran and D. Anderson, "Audio classification and scene recognition and for hearing aids," in *Proc. IEEE International Symposium On Circuits And Systems (ISCAS)*, pp. 860–863, 2005.

[5] J. Xiang, M. F. McKinney, K. Fitz, and T. Zhang, "Evaluation of sound classification algorithms for hearing aid applications," in *Proc. IEEE International Symposium On Circuits And Systems (ISCAS)*, pp. 185–188, 2010.

[6] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *Eurasip Journal On Audio, Speech, And Music Processing*, vol. 2013, no. 1, pp. 1–13, 2013.

[7] B. Clarkson, N. Sawhney, and A. Pentland, "Auditory context awareness via wearable computing," in *Proc. Workshop On Perceptual User Interfaces*, pp. 1–6, 1998.

[8] K. El-Maleh, A. Samouelian, and P. Kabal, "Frame level noise classification in mobile environments," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 237–240, 1999.

[9] A. Jakob, I. M. Stylianos, G. Robert, and L. Hana, “Acoustic scene classification by combining autoencoder-based dimensionality reduction and convolutional neural networks,” in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 7–11, 2017.

[10] E. Marchi, D. Tonelli, X. Xu, F. Ringeval, J. Deng, S. Squartini, and B. Schuller, “Pairwise decomposition with deep neural networks and multiscale kernel subspace learning for acoustic scene classification,” in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 65–69, 2016.

[11] E. Marchi, D. Tonelli, X. Xu, F. Ringeval, J. Deng, S. Squartini, and B. Schuller, “The up system for the 2016 DCASE challenge using deep recurrent neural network and multiscale kernel subspace learning,” tech. rep., DCASE2016 Challenge, September 2016.

[12] A. Vafeiadis, D. Kalatzis, K. Votis, D. Giakoumis, D. Tzovaras, L. Chen, and R. Hamzaoui, “Acoustic scene classification: From a hybrid classifier to deep learning,” tech. rep., DCASE2017 Challenge, September 2017.

[13] N. Moritz, J. Schröder, S. Goetze, J. Anemüller, and B. Kollmeier, “Acoustic scene classification using time-delay neural networks and amplitude modulation filter bank features,” in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 70–74, 2016.

[14] O. Mariotti, M. Cord, and O. Schwander, “Exploring deep vision models for acoustic scene classification,” in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 103–107, 2018.

[15] Y. Sakashita and M. Aono, “Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions,” tech. rep., DCASE2018 Challenge, September 2018.

[16] T. Nguyen and F. Pernkopf, "Acoustic scene classification using a convolutional neural network ensemble and nearest neighbor filters," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 34–38, 2018.

[17] T. Nguyen and F. Pernkop, "Acoustic scene classification with mismatched recording devices using mixture of experts layer," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1666–1671, 2019.

[18] H. Zeinali, L. Burget, and J. Cernocky, "Convolutional neural networks and x-vector embedding for dcase2018 acoustic scene classification challenge," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 202–206, 2018.

[19] H. Phan, L. Hertel, M. Maass, P. Koch, R. Mazur, and A. Mertins, "Improved audio scene classification based on label-tree embeddings and convolutional neural networks," *IEEE Trans. Audio, Speech and Language*, vol. 25, no. 6, pp. 1278–1290, 2017.

[20] H. Phan, O. Y. Chén, P. Koch, L. Pham, I. McLoughlin, A. Mertins, and M. De Vos, "Beyond equal-length snippets: How long is sufficient to recognize an audio scene?," in *Audio Engineering Society Conference: 2019 AES International Conference on Audio Forensic (AES)*, Jun 2019.

[21] L. Pham, I. McLoughlin, H. Phan, R. Palaniappan, and Y. Lang, "Bag-of-features models based on C-DNN network for acoustic scene classification," in *Audio Engineering Society Conference: 2019 AES International Conference on Audio Forensic (AES)*, 2019.

[22] T. Lidy and A. Schindler, "Cqt-based convolutional neural networks for audio scene classification," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2016 (DCASE) Workshop*, pp. 1032–1048, 2016.

[23] R. Zhao, K. Qiuqiang, Q. Kun, D. Mark, and W. Bjorn, "Attention-based convolutional neural networks for acoustic scene classification," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 39–43, 2018.

[24] Z. Ren, Q. Kong, J. Han, M. D. Plumbley, and B. W. Schuller, "Attention-based atrous convolutional neural networks: Visualisation and understanding perspectives of acoustic scenes," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 56–60, 2019.

[25] S. Phaye, E. Benetos, and Y. Wang, "SubSpectralNet using sub-spectrogram based convolutional neural networks for acoustic scene classification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 825–829, 2019.

[26] H. Phan, M. Maass, R. Mazur, and A. Mertins, "Early event detection in audio streams," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2015.

[27] H. Phan, P. Koch, I. McLoughlin, and A. Mertins, "Enabling early audio event detection with neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 141–145, 2018.

[28] I. McLoughlin, Y. Song, L. D. Pham, H. Pham, P. Ramaswamy, and L. Yue, "Early detection of continuous and partial audio events using CNN," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3314–3318, 2018.

[29] L. Pham, I. McLoughlin, H. Phan, and R. Palaniappan, "A robust framework for acoustic scene classification," in *Proc. Annual Conference of*

the International Speech Communication Association (INTERSPEECH), pp. 3634–3638, 2019.

[30] L. Pham, H. Phan, T. Nguyen, R. Palaniappan, A. Mertins, and I. McLoughlin, "Robust acoustic scene classification using a multi-spectrogram encoder-decoder framework," *Digital Signal Processing*, vol. 110, 2021.

[31] L. Pham, I. McLoughlin, H. Phan, R. Palaniappan, and A. Mertins, "Deep feature embedding and hierarchical classification for audio scene classification," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2020.

[32] L. Pham, I. McLoughlin, H. Phan, M. Tran, T. Nguyen, and R. Palaniappan, "Robust deep learning framework for predicting respiratory anomalies and diseases," in *Proc. 42nd Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 164–167, 2020.

[33] L. D. Pham, H. Phan, R. Palaniappan, A. Mertins, and I. McLoughlin, "Cnn-moe based framework for classification of respiratory anomalies and lung disease detection," *IEEE Journal of Biomedical and Health Informatics*, 2021.

[34] L. Pham, H. Phan, R. King, A. Mertins, and I. McLoughlin, "Inception-based network and multi-spectrogram ensemble applied for predicting respiratory anomalies and lung diseases," *arXiv preprint arXiv:2012.13699*, 2020.

[35] L. Pham, A. Schindler, M. Schütz, J. Lampert, S. Schlarb, and R. King, "Deep learning frameworks applied for audio-visual scene classification," *arXiv preprint arXiv:2106.06840*, 2021.

[36] L. Pham, H. Tang, A. Jalali, A. Schindler, and R. King, "A low-compexity deep learning framework for acoustic scene classification," *arXiv preprint arXiv:2106.06838*, 2021.

[37] L. Pham, C. Baume, Q. Kong, T. Hussain, W. Wang, and M. Plumbley, "An audio-based deep learning framework forbbc television programme classification," *arXiv preprint arXiv:2104.01161*, 2021.

[38] H. Phan, O. Chén, L. Pham, P. Koch, M. de Vos, I. Mcloughlin, and A. Mertins, "Spatio-temporal attention pooling for audio scene classification," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3845–3849, 2019.

[39] D. Ngo, H. Hoang, A. Nguyen, T. Ly, and L. Pham, "Sound context classification basing on join learning model and multi-spectrogram features," *arXiv preprint arXiv:2005.12779*, 2020.

[40] H. Phan, L. Pham, P. Koch, N. Q. Duong, I. McLoughlin, and A. Mertins, "On multitask loss function for audio event detection and localization," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 160–164, 2020.

[41] H. Phan, H. Le Nguyen, O. Y. Chén, L. Pham, P. Koch, I. McLoughlin, and A. Mertins, "Multi-view audio and music classification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 611–615, 2021.

[42] A. Mesaros, T. Heittola, and T. Virtamen, "Acoustic scene classification in dcase 2019 challenge: Closed and open set classification and data mismatch setups," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 164–167, 2019.

[43] A. Mesaros, T. Heittola, and T. Virtamen, "A multi-device dataset for urban acoustic scene classification," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 9–13, 2018.

[44] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "Dcase 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 85–92, 2017.

[45] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *Proc. 24th European Signal Processing Conference (EUSIPCO)*, pp. 1128–1132, 2016.

[46] M. Lagrange, G. Lafay, B. Defreville, and J.-J. Aucouturier, "The bag-of-frames approach: a not so sufficient model for urban soundscapes," *The Journal of the Acoustical Society of America*, vol. 138, no. 5, pp. EL487–EL492, 2015.

[47] A. Rakotomamonjy and A. Rakotomamonjy, "Supervised representation learning for audio scene classification," *IEEE/ACM Trans. Audio, Speech and Language*, vol. 25, no. 6, pp. 1253–1265, 2017.

[48] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[49] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in *Proc. 18th European Signal Processing Conference*, pp. 1272–1276, 2010.

[50] D. Smith, L. Ma, and N. Ryan, "Acoustic environment as an indicator of social and physical context," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 241–254, 2006.

[51] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[52] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, “CP-JKU submissions for DCASE-2016: a hybrid approach using binaural i-vectors and deep convolutional neural networks,” tech. rep., DCASE2016 Challenge, September 2016.

[53] B. Lehner, H. Eghbal-Zadeh, M. Dorfer, F. Korzeniowski, K. Koutini, and G. Widmer, “Classifying short acoustic scenes with I-vectors and CNNs: Challenges and optimisations for the 2017 DCASE ASC task,” tech. rep., DCASE2017 Challenge, September 2017.

[54] J. Jee-Weon, H. Hee-Soo, Y. IL-Ho, Y. Sung-Hyun, S. Hye-Jin, and Y. Ha-Jin, “DNN-based audio scene classification for DCASE 2017: Dual input-features, balancing cost, and stochastic data duplication,” in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 59–63, 2017.

[55] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, “Experiments on the DCASE challenge 2016: Acoustic scene classification and sound event detection in real life recording,” tech. rep., DCASE2016 Challenge, September 2016.

[56] S. Abidin, R. Togneri, and F. Sohel, “Spectrotemporal analysis using local binary pattern variants for acoustic scene classification,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2112–2121, 2018.

[57] H. Song, J. Han, and D. Shiwen, “A compact and discriminative feature based on auditory summary statistics for acoustic scene classification,” in

Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 3294–3298, 2018.

[58] H. Phan, P. Koch, F. Katzberg, M. Maass, R. Mazur, and A. Mertins, "Audio scene classification with deep recurrent neural networks," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3845–3849, 2017.

[59] Z. Ren, K. Qian, Y. Wang, Z. Zhang, V. Pandit, A. Baird, and B. Schuller, "Deep scalogram representations for acoustic scene classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 3, pp. 662–669, 2018.

[60] Y. Yang, H. Zhang, W. Tu, H. Ai, L. Cai, R. Hu, and F. Xiang, "Kullback–leibler divergence frequency warping scale for acoustic scene classification using convolutional neural network," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 840–844, 2019.

[61] S. Waldekar and G. Saha, "Wavelet transform based mel-scaled features for acoustic scene classification.," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3323–3327, 2018.

[62] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing," in *A talk at the Stanford Artificial Project*, pp. 271–272, 1968.

[63] Y. Wu and T. Lee, "Enhancing sound texture in cnn-based acoustic scene classification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 815–819, 2019.

[64] S. Park, S. Mun, Y. Lee, and H. Ko, "Acoustic scene classification based on convolutional neural network using double image features," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 98–102, 2017.

[65] S. Park, S. Mun, Y. Lee, and H. Ko, "Score fusion of classification systems for acoustic scene classification," tech. rep., DCASE2016 Challenge, September 2016.

[66] J.-w. Jung, H.-s. Heo, H.-j. Shim, and H.-j. Yu, "DNN based multi-level features ensemble for acoustic scene classification," tech. rep., DCASE2018 Challenge, September 2018.

[67] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2721–2725, 2017.

[68] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[69] T. Zhang, K. Zhang, and J. Wu, "Data independent sequence augmentation method for acoustic scene classification.," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3289–3293, 2018.

[70] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[71] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition," *arXiv preprint arXiv:1711.10282*, 2017.

[72] K. Xu, D. Feng, H. Mi, B. Zhu, D. Wang, L. Zhang, H. Cai, and S. Liu, "Mixup-based acoustic scene classification using multi-channel convolutional neural network," in *Proc. Pacific Rim Conference on Multimedia*, pp. 14–23, 2018.

[73] S. Mun, S. Park, D. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane," tech. rep., DCASE2017 Challenge, September 2017.

[74] H. Chen, Z. Liu, Z. Liu, P. Zhang, and Y. Yan, "Integrating the data augmentation scheme with various classifiers for acoustic scene modeling," tech. rep., DCASE2019 Challenge, June 2019.

[75] G. Mafra, N. Duong, A. Ozerov, and P. Pérez, "Acoustic scene classification: An evaluation of an extremely compact feature representation," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 85–89, 2016.

[76] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Supervised nonnegative matrix factorization for acoustic scene classification," tech. rep., DCASE2016 Challenge, September 2016.

[77] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pp. 41–48, Ieee, 1999.

[78] A. O. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for svm-based speaker recognition," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1471–1474, 2006.

[79] Y. Xu, Q. Huang, W. Wang, and M. D. Plumbley, "Hierarchical learning for dnn-based acoustic scene classification," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 105–109, 2016.

[80] Q. Kong, I. Sobieraj, W. Wang, and M. Plumbley, "Deep neural network baseline for dcase challenge 2016," in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 50–54, 2016.

[81] T. Zhang, K. Zhang, and J. Wu, "Temporal transformer networks for acoustic scene classification.," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1349–1353, 2018.

[82] N. FangLi and D. Shuang, "Acoustic scene classification based on the dataset with deep convolutional generated against network," tech. rep., DCASE2019 Challenge, June 2019.

[83] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceeding of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[84] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[85] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[86] M. K. Patrick, A. F. Adekoya, A. A. Mighty, and B. Y. Edward, "Capsule networks–a survey," *Journal of King Saud University-Computer and Information Sciences*, 2019.

[87] L. Yang, X. Chen, and L. Tao, “Acoustic scene classification using multi-scale features,” in *Proc. Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop*, pp. 29–33, November 2018.

[88] H. Song, J. Han, S. Deng, and Z. Du, “Acoustic scene classification by implicitly identifying distinct sound events,” in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3860–3864, 2019.

[89] T. Zhang, K. Zhang, and J. Wu, “Multi-modal attention mechanisms in lstm and its application to acoustic scene classification.,” in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3328–3332, 2018.

[90] S. H. Bae, I. Choi, and N. S. Kim, “Acoustic scene classification using parallel combination of LSTM and CNN,” tech. rep., DCASE2016 Challenge, September 2016.

[91] H. Phan, L. Hertel, M. Maass, P. Koch, and A. Mertins, “Label tree embeddings for acoustic scene classification,” in *Proc. 24th ACM international conference on Multimedia*, pp. 486–490, 2016.

[92] H. Chen, P. Zhang, and Y. Yan, “An audio scene classification framework with embedded filters and a dct-based temporal module,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 835–839, IEEE, 2019.

[93] A. Singh, A. Thakur, P. Rajan, and A. Bhavsar, “A layer-wise score level ensemble framework for acoustic scene classification,” in *Proc. 26th European Signal Processing Conference (EUSIPCO)*, pp. 837–841, 2018.

[94] S. Mun, S. Shon, W. Kim, D. K. Han, and H. Ko, “Deep neural network based learning and transferring mid-level audio features for acoustic scene

classification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 796–800, 2017.

[95] S. Hyeji and P. Jihwan, "Acoustic scene classification using various pre-processed features and convolutional neural networks," tech. rep., DCASE2019 Challenge, June 2019.

[96] Y. Han and J. Park, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," tech. rep., DCASE2017 Challenge, September 2017.

[97] Z. Weiping, Y. Jiantao, X. Xiaotao, L. Xiangtao, and P. Shaohu, "Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion," tech. rep., DCASE2017 Challenge, September 2017.

[98] 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," tech. rep., DCASE2018 Challenge, September 2018.

[99] McFee, Brian, R. Colin, L. Dawen, D. PW.Ellis, M. Matt, B. Eric, and N. Oriol, "librosa: Audio and music signal analysis in python," in *Proc. of The 14th Python in Science Conference*, pp. 18–25, 2015.

[100] D. P. W. . Ellis, "Gammatone-like spectrogram," http://www.ee.columbia.edu/ dpwe/resources/matlab/gammatonegram.

[101] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," tech. rep., DCASE2016 Challenge, September 2016.

[102] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[103] Y. Li, X. Li, and Y. Zhang, “The SEIE-SCUT systems for challenge on DCASE 2018: Deep learning techniques for audio representation and classification,” tech. rep., DCASE2018 Challenge, September 2018.

[104] T. Heittola, A. Mesaros, and T. Virtanen, “A multi-device dataset for urban acoustic scene classification,” tech. rep., DCASE2018 Challenge, September 2018.

[105] Z. Li, L. Zhang, S. Du, and W. Liu, “Acoustic scene classification based on binaural deep scattering spectra with CNN and LSTM,” tech. rep., DCASE2018 Challenge, September 2018.

[106] W. Hao, L. Zhao, Q. Zhang, H. Zhao, and J. Wang, “DCASE 2018 task 1a: Acoustic scene classification by bi-LSTM-CNN-net multichannel fusion,” tech. rep., DCASE2018 Challenge, September 2018.

[107] J. Tchorz, “Combination of amplitude modulation spectrogram features and MFCCs for acoustic scene classification,” tech. rep., DCASE2018 Challenge, September 2018.

[108] C. Roletscheck and T. Watzka, “Using an evolutionary approach to explore convolutional neural networks for acoustic scene classification,” tech. rep., DCASE2018 Challenge, September 2018.

[109] Q. Kong, I. Turab, X. Yong, W. Wang, and M. D. Plumbley, “DCASE 2018 challenge surrey cross-task convolutional neural network baseline,” tech. rep., DCASE2018 Challenge, September 2018.

[110] L. Zhang and J. Han, “Acoustic scene classification using multi-layered temporal pooling based on deep convolutional neural network,” tech. rep., DCASE2018 Challenge, September 2018.

[111] W. Jun and L. Shengchen, “Self-attention mechanism based system for dcase2018 challenge task1 and task4,” tech. rep., DCASE2018 Challenge, September 2018.

[112] Z. Li, L. Zhang, S. Du, and W. Liu, “Acoustic scene classification based on binaural deep scattering spectra with CNN and LSTM,” tech. rep., DCASE2018 Challenge, September 2018.

[113] S. Waldekar and G. Saha, “Wavelet-based audio features for acoustic scene classification,” tech. rep., DCASE 2018 Challenge, 2018.

[114] A. Dang, T. Vu, and J.-C. Wang, “Acoustic scene classification using ensemble of convnets,” tech. rep., DCASE2018 Challenge, September 2018.

[115] A. Golubkov and A. Lavrentyev, “Acoustic scene classification using convolutional neural networks and different channels representations and its fusion,” tech. rep., DCASE2018 Challenge, September 2018.

[116] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[117] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[118] E. Garmash and C. Monz, “Ensemble learning for multi-source neural machine translation,” in *The 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1409–1418, 2016.

[119] I. McLoughlin, Z. Xie, Y. Song, H. Phan, and R. Palaniappan, “Time–frequency feature fusion for noise robust audio event classification,” *Circuits, Systems, and Signal Processing*, vol. 39, no. 3, pp. 1672–1687, 2020.

[120] W. Dai, J. Li, P. Pham, S. Das, and S. Qu, "Acoustic scene recognition with deep neural networks (DCASE challenge 2016)," tech. rep., DCASE2016 Challenge, September 2016.

[121] S. Zhao, T. N. T. Nguyen, W.-S. Gan, and J. Douglas L., "ADSC submission for DCASE 2017: Acoustic scene classification using deep residual convolutional neural networks," tech. rep., DCASE2017 Challenge, September 2017.

[122] L. Mingle and L. Yanxiong, "The system for acoustic scene classification using resnet," tech. rep., DCASE2019 Challenge, June 2019.

[123] V. Bisot, S. Essid, and G. Richard, "HOG and subband power distribution image features for acoustic scene classification," in *Proc. 23rd European Signal Processing Conference (EUSIPCO)*, pp. 719–723, IEEE, 2015.

[124] Y. Wu and T. Lee, "Stratified time-frequency features for CNN-based acoustic scene classification," tech. rep., DCASE2019 Challenge, June 2019.

[125] Q. Kong, Y. Cao, T. Iqbal, W. Wang, and M. D. Plumbley, "Cross-task learning for audio tagging, sound event detection and spatial localization: DCASE 2019 baseline systems," tech. rep., DCASE2019 Challenge, June 2019.

[126] J. Ye, T. Kobayashi, M. Murakawa, and T. Higuchi, "Acoustic scene classification based on sound textures and events," in *Proc. Of The 23rd ACM International Conference on Multimedia*, pp. 1291–1294, 2015.

[127] J. Kim and K. Lee, "Empirical study on ensemble method of deep neural networks for acoustic scene classification," tech. rep., DCASE2016 Challenge, September 2016.

[128] K. Piczak, “The details that matter: Frequency resolution of spectrograms in acoustic scene classification,” tech. rep., DCASE2017 Challenge, September 2017.

[129] W. Gao and M. McDonnell, “Acoustic scene classification using deep residual networks with late fusion of separated high and low frequency paths,” tech. rep., DCASE2019 Challenge, June 2019.

[130] S. Waldekar and G. Saha, “Wavelet based mel-scaled features for DCASE 2019 task 1a and task 1b,” tech. rep., DCASE2019 Challenge, June 2019.

[131] G. Takahashi, T. Yamada, S. Makino, and N. Ono, “Acoustic scene classification using deep neural network and frame-concatenated acoustic feature,” tech. rep., DCASE2016 Challenge, September 2016.

[132] I. Kukanov, V. Hautamäki, and K. A. Lee, “Recurrent neural network and maximal figure of merit for acoustic event detection,” tech. rep., DCASE2017 Challenge, September 2017.

[133] M. Wang and R. Wang, “Ciaic-ASC system for DCASE 2019 challenge task1,” tech. rep., DCASE2019 Challenge, June 2019.

[134] Z. Wang, J. Ma, and C. Li, “Acoustic scene classification based on CNN system,” tech. rep., DCASE2019 Challenge, June 2019.

[135] Y. Yin, R. R. Shah, and R. Zimmermann, “Learning and fusing multimodal deep features for acoustic scene categorization,” in *Proc. 26th ACM international conference on Multimedia*, pp. 1892–1900, 2018.

[136] J.-w. Jung, H.-S. Heo, H.-j. Shim, and H.-J. Yu, “Knowledge distillation with specialist models in acoustic scene classification,” tech. rep., DCASE2019 Challenge, June 2019.

[137] S. Jiang and C. Shi, "Acoustic scene classification using ensembles of convolutional neural networks and spectrogram decompositions," tech. rep., DCASE2019 Challenge, June 2019.

[138] J. Huang, P. Lopez Meyer, H. Lu, H. Cordourier Maruri, and J. Del Hoyo, "Acoustic scene classification using deep learning-based ensemble averaging," tech. rep., DCASE2019 Challenge, June 2019.

[139] H. Song and H. Yang, "Feature enhancement for robust acoustic scene classification with device mismatch," tech. rep., DCASE2019 Challenge, June 2019.

[140] J. Ye, T. Kobayashi, N. Toyama, H. Tsuda, and M. Murakawa, "Acoustic scene classification using efficient summary statistics and multiple spectro-temporal descriptor fusion," *Applied Sciences*, vol. 8, no. 8, p. 1363, 2018.

[141] R. Hyder, S. Ghaffarzadegan, Z. Feng, and T. Hasan, "BUET bosch consortium (B2C) acoustic scene classification systems for DCASE 2017," tech. rep., DCASE2017 Challenge, September 2017.

[142] Y. Haocong, S. Chuang, and L. Huiyong, "Acoustic scene classification using CNN ensembles and primary ambient extraction," tech. rep., DCASE2019 Challenge, June 2019.

[143] P. Primus and D. Eitelsebner, "Acoustic scene classification with mismatched recording devices," tech. rep., DCASE2019 Challenge, June 2019.

[144] H. Eghbal-zadeh, K. Koutini, and G. Widmer, "Acoustic scene classification and audio tagging with receptive-field-regularized CNNs," tech. rep., DCASE2019 Challenge, June 2019.

[145] K. Koutini, H. Eghbal-zadeh, and G. Widmer, “Acoustic scene classification and audio tagging with receptive-field-regularized CNNs,” tech. rep., DCASE2019 Challenge, June 2019.

[146] W. Gao and M. McDonnell, “Acoustic scene classification using deep residual networks with late fusion of separated high and low frequency paths,” tech. rep., DCASE2019 Challenge, June 2019.

[147] M. Kośmider, “Calibrating neural networks for secondary recording devices,” tech. rep., DCASE2019 Challenge, June 2019.

[148] X. Bai, J. Du, Z.-R. Wang, and C.-H. Lee, “A hybrid approach to acoustic scene classification based on universal acoustic models,” in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3619–3623, 2019.

[149] J. Li, W. Dai, F. Metze, S. Qu, and S. Das, “A comparison of deep learning methods for environmental sound detection,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 126–130, 2017.

[150] L. Gao, H. Mi, B. Zhu, D. Feng, Y. Li, and Y. Peng, “An adversarial feature distillation method for audio classification,” *IEEE Access*, vol. 7, pp. 105319–105330, 2019.

[151] T. Nguyen and F. Pernkopf, “Acoustic scene classification with mismatched devices using cliquenets and mixup data augmentation,” in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2330–2334, 2019.

[152] R. Hyder, S. Ghaffarzadegan, Z. Feng, J. H. Hansen, and T. Hasan, “Acoustic scene classification using a CNN-supervector system trained with auditory and spectrogram image features.,” in *Proc. Annual Conference of*

the International Speech Communication Association (INTERSPEECH), pp. 3073–3077, 2017.

[153] S. Waldekar and G. Saha, "Wavelet transform based mel-scaled features for acoustic scene classification.," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3323–3327, 2018.

[154] L. Yang, L. Tao, X. Chen, and X. Gu, "Multi-scale semantic feature fusion and data augmentation for acoustic scene classification," *Applied Acoustics*, vol. 163, p. 107238, 2020.

[155] H.-S. Heo, J.-w. Jung, H.-j. Shim, and H.-J. Yu, "Acoustic scene classification using teacher-student learning with soft-labels," *arXiv preprint arXiv:1904.10135*, 2019.

[156] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[157] W. H. Organization, "The global impact of respiratory diseases (second edition)," 2017.

[158] H. Polat and İ. Güler, "A simple computer-based measurement and analysis system of pulmonary auscultation sounds," *Journal of medical systems*, vol. 28, no. 6, pp. 665–672, 2004.

[159] R. J. Riella, P. Nohama, R. F. Borges, and A. L. Stelle, "Automatic wheezing recognition in recorded lung sounds," in *Proc. Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC)*, vol. 3, pp. 2535–2538, Sep. 2003.

[160] S. Reichert, R. Gass, C. Brandt, and E. Andrès, "Analysis of respiratory sounds: state of the art," *Clinical medicine. Circulatory, respiratory and pulmonary medicine*, vol. 2, pp. CCRPM–S530, 2008.

[161] T. Okubo, N. Nakamura, M. Yamashita, and S. Matsunaga, "Classification of healthy subjects and patients with pulmonary emphysema using continuous respiratory sounds," in *Proc. Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 70–73, 2014.

[162] X. H. Kok, S. A. Imtiaz, and E. Rodriguez-Villegas, "A novel method for automatic identification of respiratory disease from acoustic recordings," in *Proc. Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2589–2592, 2019.

[163] M. Grønnesby, J. Solis, E. Holsbø, H. Melbye, and L. Bongo, "Feature extraction for machine learning based crackle detection in lung sounds from a health survey," *arXiv preprint arXiv:1706.00005*, 2017.

[164] G. Chambres, P. Hanna, and M. Desainte-Catherine, "Automatic detection of patient with respiratory diseases using lung sound analysis," in *Proc. International Conference on Content-Based Multimedia Indexing (CBMI)*, pp. 1–6, 2018.

[165] L. Mendes, I. M. Vogiatzis, E. Perantoni, E. Kaimakamis, I. Chouvarda, N. Maglaveras, J. Henriques, P. Carvalho, and R. P. Paiva, "Detection of crackle events using a multi-feature approach," in *Proc. Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 3679–3683, 2016.

[166] S. Datta, A. D. Choudhury, P. Deshpande, S. Bhattacharya, and A. Pal, "Automated lung sound analysis for detecting pulmonary abnormalities,"

in *Proc. Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4594–4598, 2017.

[167] D. Reshef, Y. Reshef, H. Finucane, S. Grossman, G. McVean, P. Turnbaugh, E. Lander, M. Mitzenmacher, and P. Sabeti, “Detecting novel associations in large data sets,” *Science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[168] N. Sengupta, M. Sahidullah, and G. Saha, “Lung sound classification using local binary pattern,” *arXiv preprint arXiv:1710.01703*, 2017.

[169] D. Oletic, M. Matijascic, V. Bilas, and M. Magno, “Hidden markov model-based asthmatic wheeze recognition algorithm leveraging the parallel ultra-low-power processor (PULP),” in *Proc. IEEE Sensors Applications Symposium*, pp. 1–6, 2019.

[170] N. Jakovljević and T. Lončar-Turukalo, “Hidden markov model based respiratory sound classification,” in *Precision Medicine Powered by pHealth and Connected Health*, pp. 39–43, Springer, 2018.

[171] G. Serbes, S. Ulukaya, and Y. P. Kahya, “An automated lung sound preprocessing and classification system based onspectral analysis methods,” in *Precision Medicine Powered by pHealth and Connected Health*, pp. 45–49, 2018.

[172] G. Serbes, S. Ulukaya, and Y. P. Kahya, “An automated lung sound preprocessing and classification system based on spectral analysis methods,” in *Precision Medicine Powered by pHealth and Connected Health*, pp. 45–49, Springer, 2018.

[173] B. M. Rocha, D. Filos, L. Mendes, G. Serbes, S. Ulukaya, Y. P. Kahya, N. Jakovljevic, T. L. Turukalo, I. M. Vogiatzis, E. Perantoni, *et al.*, “An open access database for the evaluation of respiratory sound classification algorithms,” *Physiological measurement*, vol. 40, no. 3, p. 035001, 2019.

[174] H. Zhang, I. McLoughlin, and Y. Song, “Robust sound event recognition using convolutional neural networks,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, no. 2635, pp. 559–563, Apr. 2015.

[175] I. McLoughlin, H.-M. Zhang, Z.-P. Xie, Y. Song, and W. Xiao, “Robust sound event classification using deep neural networks,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 23, pp. 540–552, Mar. 2015.

[176] L. Shi, K. Du, C. Zhang, H. Ma, and W. Yan, “Lung sound recognition algorithm based on vggish-bigru,” *IEEE Access*, vol. 7, pp. 139438–139449, 2019.

[177] R. Liu, S. Cai, K. Zhang, and N. Hu, “Detection of adventitious respiratory sounds based on convolutional neural network,” in *Proc. International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, pp. 298–303, 2019.

[178] J. Acharya and A. Basu, “Deep neural network for respiratory sound classification in wearable devices enabled by patient specific model tuning.,” *IEEE transactions on biomedical circuits and systems*, vol. 14, pp. 535–544, 2020.

[179] M. Aykanat, Ö. Kılıç, B. Kurt, and S. Saryal, “Classification of lung sounds using convolutional neural networks,” *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 65, 2017.

[180] D. Perna, “Convolutional neural networks learning from respiratory data,” in *Proc. IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 2109–2113, 2018.

[181] D. Perna and A. Tagarelli, “Deep auscultation: Predicting respiratory anomalies and diseases via recurrent neural networks,” in *Proc. 32nd IEEE*

International Symposium on Computer-Based Medical Systems (CBMS), pp. 50–55, 2019.

[182] E. Messner, M. Fediuk, P. Swatek, S. Scheidl, F. Smolle-Juttner, H. Olschewski, and F. Pernkopf, "Crackle and breathing phase detection in lung sounds with deep bidirectional gated recurrent neural networks," in *Proc. Annual International Conferences of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 356–359, 2018.

[183] K. Kochetov, E. Putin, M. Balashov, A. Filchenkov, and A. Shalyto, "Noise masking recurrent neural network for respiratory sound classification," in *Proc. International Conference on Artificial Neural Networks*, pp. 208–217, 2018.

[184] K. Minami, H. Lu, H. Kim, S. Mabu, Y. Hirano, and S. Kido, "Automatic classification of large-scale respiratory sound dataset based on convolutional neural network," in *Proc. 19th International Conference on Control, Automation and Systems (ICCAS)*, pp. 804–807, 2019.

[185] H. Chen, X. Yuan, Z. Pei, M. Li, and J. Li, "Triple-classification of respiratory sounds using optimized s-transform and deep residual networks," *IEEE Access*, vol. 7, pp. 32845–32852, 2019.

[186] H.-S. Heo, J.-w. Jung, H.-j. Shim, and H.-J. Yu, "Acoustic scene classification using teacher-student learning with soft-labels," *arXiv preprint arXiv:1904.10135*, 2019.

[187] L. Gao, H. Mi, B. Zhu, D. Feng, Y. Li, and Y. Peng, "An adversarial feature distillation method for audio classification," *IEEE Access*, vol. 7, pp. 105319–105330, 2019.

[188] M. Taghavi and M. Shoaran, "Hardware complexity analysis of deep neural networks and decision tree ensembles for real-time neural data classification," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pp. 407–410, 2019.

[189] Google, *TensorFlow Model Optimization Toolkit*, 2020. Available at `https://www.tensorflow.org/model_optimization`.

[190] R. D. Patterson, "Auditory filters and excitation patterns as representations of frequency resolution," *Frequency selectivity in hearing*, 1986.

[191] B. R. Glasberg and B. C. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hearing research*, vol. 47, no. 1-2, pp. 103–138, 1990.