2019-1

Investigation into the Perceptually Informed Data for Environmental Sound Recognition

Chenglin Kang

Technological University Dublin

Recommended Citation

Kang, C. (2019) Investigation into the Perceptually Informed Data for Environmental Sound Recognition , *Dissertation M.Sc. in Computing (Data Analytics), TU Dublin, 2019.*

Investigation into the Perceptually Informed Data for Environmental Sound Recognition

Chenglin Kang

A dissertation submitted in partial fulfilment of the requirements of Dublin Institute of Technology for the degree of

M.Sc. in Computing (Advanced Software Development)

**

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Knowledge Management), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: **_____Chenglin Kang______________**

Date: **22 01 2019**

ABSTRACT

Environmental sound is rich source of information that can be used to infer contexts. With the rise in ubiquitous computing, the desire of environmental sound recognition is rapidly growing. Primarily, the research aims to recognize the environmental sound using the perceptually informed data. The initial study is concentrated on understanding the current state-of-the-art techniques in environmental sound recognition. Then those researches are evaluated by a critical review of the literature.

This study extracts three sets of features: Mel Frequency Cepstral Coefficients, Mel-spectrogram and sound texture statistics. Two kinds machine learning algorithms are cooperated with appropriate sound features. The models are compared with a low-level baseline model. It also presents a performance comparison between each model with the high-level human listeners.

The study results in sound texture statistics model performing the best classification by achieving 45.1% of accuracy based on support vector machine with radial basis function kernel. Another Mel-spectrogram model based on Convolutional Neural Network also provided satisfactory results and have received predictive results greater than the benchmark test.

Key words: *Environmental sound recognition, Sound Texture Statistics, Mel-spectrogram, Supervised Machine Learning, SVM, CNN*

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere thanks to my supervisor Sean O'Leary for his professionalism of audio processing and precise guidance throughout this dissertation.

I would like to thank the lecturers and students at Advanced Softeware Development stream for their support over the courses.

I would also like to thank my colleagues at Waratek company, especially my manager Turlough Whelan who tolerated my study leaves without complaint.

I am grateful to my friends, especially Yiqian Ch'ng, Liping Zhang, Victor Santiago, all of whom rendered their help during the period of my dissertation work.

Finally, I would like to acknowledge with gratitude, the support and love of my family – my parents, Tongping Hu and Yuanfa Kang; my sister, Yingrong Lv.

TABLE OF CONTENTS

TABLE OF CONTENTS**

TABLE OF CONTENTS**

List of figures**

LIST OF FIGURES

LIst of Tables**

LIST OF TABLES

List of Acronyms**

LIST OF ACRONYMS

ANN	Artificial Neural Network
AUC	Area Under the Curve
ASC	Audio Signal Classification
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
ERB	Equivalent Rectangular Bandwidths
ESC	Environmental Sound Classification
ESR	Environmental Sound Recognition
FFT	Fast Fourier Transform
mAP	mean Average Precision
HMM	Hidden Markov Model
MFCC	Mel Frequency Cepstral Coefficients
RBF	Radial Basis Function
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
LTS	Long-term Statistics

1 INTRODUCTION

1.1 Background

Audio Signal Classification (ASC) is the task of extracting relevant features from the input sound and identifying into which of a set of classes the sound is most likely to fit at the output (Gerhard, 2003). The existing ASC systems are mainly used for characterising three types of audio signal: speech, music, environmental sounds. Speech and music signals are two categories that have been traditionally focused on and extensively studied (Chachada & Kuo, 2013). A considerable amount of research has been made towards Environmental Sounds Recognition (ESR) over the past decade, also various independent areas of sonic studies have integrated to deal with aspects of ESR such as: acoustics, psychoacoustics, electroacoustics, taxonomy, statistics and machine learning. Nevertheless, the activity is relatively low compared to speech or music (Chu, Narayanan, & Kuo, 2009).

The demand of ESR is rapidly growing as it plays a critical role in perfecting IoT systems. According to a report by the IoT Analytics Agent (Lueth, 2018), the total number of IoT devices reached 7 billion in the second Quarter of 2018. A simple vision-based device would lose their utility when the visual information is insufficient or absent. To meet the system requirement of robustness, ESR is indispensable part for robots enhancing their context awareness and mitigating the dependency on vision. Furthermore，video as a multimodal medium which contains audio signal become an indivisible part of today’s big data. The 2015–2020 Cisco Visual Networking Index report estimates that, by 2020, compressed video bitstreams will occupy more than 82% of all IP traffic, with one million minutes of video crossing the network every second (Cisco, 2015). The sustained increasement is a booming demand for ESR techniques to exploit abundant multimodal clues and automate the classification processes.

The typical workflow of an ESR task deals with feature extraction. It can be divided into two categories: stationary (frequency-based) feature extraction and non-stationary (time-frequency based) feature extraction (Cowling & Sitte, 2003). In its infancy, ESR adopted stationary feature extraction methods from speech or music recognition to produces an overall result detailing the frequencies contained in the entire signal (Cowling & Sitte, 2002). However, most of the environmental sounds, such as sea waves, do not have meaningful stationary features such as phonemes, melody and rhythm. Also, environment sounds are more complex than music due to noises. In contrast, non-stationary feature extraction identifies frequency as occurring in discrete time units. Recent researches in ESR focused on capturing non-stationary features over a long period, which aids understanding of the signal.

1.2 Research Problem

Most of the environmental sounds like dog barks, drillings and sea waves can be recognised by temporal homogeneity through human cochlea, because they are produced by a concurrence of many similar acoustic events that overlap in time. Those sounds are defined as “sound textures”, corresponding to the visual textures that have been studied for decades (Heeger & Bergen, 1995; PortillaEero & Simoncelli, 2000). The constituent sound features, and their relationships can be captured by the marginal statistics of individual frequency sub-bands. However, hearing science has neglected them for very long time. There are only a few studies imply the potential of statistical model in the computational audio community (Arnaud & Popat, 1998; Dubnov, Bar-Joseph, El-Yaniv, Lischinski, & Werman, 2002; Athineos & Ellis, 2003)

McDermott et al. (2009) suggested using time-averaged statistics to capture the constituent sound features. By imposing the statistics of a Gaussian noise sound, they successfully synthesized 168 enviromental sounds， proving enviromental sounds contain sufficient statistical structures. Moreover, Ellis, Zeng, and McDermott (2011) investigated the automatic classification ability of sound texture statistics with a Support Vector Machine (SVM). They found the performance was as well as the conventional statistics based on Mel Frequency Cepstral Coefficients (MFCC) covariance. Nonetheless, they did acknowledge the investigation was not ideal, since the dateset that they used was not crisply distinguished. For instance, a class like "indoor-noisy" may consist of restaurant babble or machine noise without distinguiding between them. Futher work is required to assess statistics features on a more precise categorized dataset which contains over a wider range of sounds.

The SVM is a frequenctly used supervised learning model in ESR research. It benefits classifying the sound features with vectors such as MFCC. Like most of the sound features, convolutional neural network (CNN) has been frequently applied in speech recognition since 2009. CNN paradigm has proved highly successful in a number of classification tasks, but it has slowly begun in the ESR area since the last three years (Piczak, 2015). Both machine leaning techniqes yielded very good results in various research and showed the most potential for developing high performance ESR models.

The primary research question that is planned to be addressed in the current study can be consisely stated as follows –

> *"To what extent can a perceptually informed model significantly enhance the classification accuracy when compared to a Mel Frequency Cepstral Coefficients model based on Support Vector Machine?"*

The null hypothesis (H_0) may be expressed as:

> *"A perceptually informed model does not significantly enhance the classification accuracy when compared to a Mel Frequency Cepstral Coefficients model based on Support Vector Machine."*

Conversely, the alternative hypothesis (H_A) is stated as:

> *"A perceptually informed model significantly enhance the classification accuracy when compared to a Mel Frequency Cepstral Coefficients model based on Support Vector Machine."*

1.3 Research Objectives

The aims and objectives of the research are:

1. Critically review the literature regarding environmental sound taxonomy, sound features, sound texture statistics and classification models.

2. Carry out experiments to analyse the sound texture statistics and Mel-spectrogram for ESR.

3. Develop a classification model using MFCC with SVM as a baseline system.

4. Evaluate the results by comparing the statistical results with the baseline system hereby testing hypothesis H_0.

5. Identify the limitations of this research study and suggest areas of future research to build on this study.

1.4 Research Methodologies

The research methodology used in this study is quantitative research. Secondary data from a well-labelled environmental sound dataset is used for sound feature extractions. that experimentally develops multiple classification models, and quantitatively assesses their performance against a set of test data. The quantitative results are tested for significance, and the outcome is used to confirm or reject the research hypothesis.

1.5 Scope and Limitations

Auditory scene is a high-level environmental sound and could be the single signal mixed by a entire group of sounds that a listener hears in everyday situation at any one moment. It closely connects with graphical contexts (beach, park, road, etc…), social situations in indorr or outdoor lications (restaurant, office, home, market…) or transprtation groud (car, bus, tramway…) (Rakotomamonjy, 2017). In terms of scope from data perspectives, this study just focused on unsophsticated environmetal sounds without dependent on the contexts. Due to the time and computing constraints of the experiment, the study had to limit the number of environmental sound types to 50.

From the sound feature perspectives, there are plenty of sounds featues on various domains in ESR field. Multiple sound feature extraction methodologies and plenty of machine learning models were discovered from the literature in order to gain better insights from the data. The scope of this study was restricted to develop two classification models , using two of the popular techniques - MFCCs and sound texture statistics. The classification models were not optimised individually, because the main goal of the research is to compare their classification capabilities. Therfore, identifying the most capable enviromental sound feature is out of the scope.

1.6 Dissertation Outline

The rest of the dissertation is structured as follows: Chapter 2 provides a critical overview of the literature and provides necessary background information on environmental sound taxonomy and datasets. It also assesses current research on data understanding, sound features, classifiers and evaluation methods. Chapter 3 discusses the methodological approach, with reference to techniques from the literature. Chapter 4 includes the implementation and results. Chapter 5 discusses and critically assesses the findings. Chapter 6 concludes the paper by summarising the main points of the study. It gives some thoughts on future research directions. The full set of results are contained in Fig 4.8. The python scripts for experiment implementation are provided in Appendix C.

2 LITERATURE REVIEW

The following literature review is organised into two main parts – "Environmental Sound Feature Extraction" and "Environmental Sound Feature Analysis and Classification". This chapter starts by introducing the taxonomy for environmental sound research. It covers a guide though some well-known datasets. This section after that introduce the classical environmental sound features extracted in different domains (*i.e.*, temporal, frequency, cepstral).

As the project is deeply rooted in machine recognition, the chapter presents an up-to-date state-of-the-art review of the ESC model's performances, main audio feature extraction techniques, and machine learning algorithoms. In particular, the MFCC will be introduced as a traditional baseline system; the sound texture statistics will be evaluated as the currently leading methodology. This literature review assumes the reader has a certain scale of knowledge in the machine learning field. Hence it would not present the additional explanation of the algorithmic design of machine learning. Meanwhile, the history and some of the current challenges are highlighted.

2.1 Taxonomy for Environmental Sounds

Environmental sound comprises all types of sound in general. To date, environmental sounds do not have a will-defined structure or definition, because the relationship is not exclusive between itself and music/speech. For example, the street music could be considered as a kind of environmental sound. Because of the pervasiveness, taxonomical categorisation would be the typical pre-processing of ESR. The taxonomies of environmental sound are usually formed into an abstraction hierarchy with sound descriptors. A standardized taxonomy could address the difficulty of comparing the ESR results when the semantic groups may vary from study to study. Schubert (Schubert, 1975) and Bregman (Bregman, 1994) claims " identification of sound sources and the behaviour of those sources is the primary task of the auditory system". Environmental sound categorisation has garnered increased research attention within the ecological approach to auditory perception and in the field of soundscape research (Neuhoff, 2004).

Schafer (1993) formed the basis by dividing enviromental sounds into six categories: “natural”, “human”, “society”, “mechanical”,“silence”, and “indicators”. In 1997, many researchers (David, 1997; Dubois, 2000, Guastavino, & Raimbault, 2006; Gastellego, & Fabre, 1997) tend to have one primary element with spontaneous descriptors. However, the auditory signal classes often range broadly with non-exclusive relationships. The oversimplified terms could mislead it to the issues of overlap, for instance, it is not valid when a system separates “cat sounds” from “purr”. In order to aid the accuracy of recognition, multiple organisational principles have been proposed to classify environmental sounds. The hierarchy structural sort the environmental sounds into a superordinate level(e.g. Sounds of things), basic level (e.g. Vehicle), and subordinate level (e.g. Motor vehicle), corresponding to Rosch’s prototype theory of natual categories (Trudeau & Guastavino, 2018). With the rapid growth of ecological psychology in urban soundscapes, positive judgments were used to investigate everyday listening by Guastavino (Guastavino. 2006). It built complex phrases which integrating notions of time, location and activities such as “riding motorcycles at Bastille on Saturday night” (Guastavino, 2007). The perceptual study on how people perceive environmental sounds helps the taxonomy in evolving.

2.2 ESR Datasets

There are only a few publicly available datasets with highly scientific taxonomies in this field of research. The high cost of manual classification and annotation limits the dataset developments in both number and size. This section gives a brief overview of several frequently used datasets.

FreeSound

FreeSound project was started in 2005 by the Music Technology Group of Pompeu Fabra University. With the Creative Commons licenses, it allows users to upload, download, and even rate sounds. It also provides a API which researchers can retrieve similar sounds and retrieve automatically extracted features from audio files, . Thus, it became the biggest collaborative database of audio snippets. Many famous environmental sound databases were the subset of FreeSound or inspired by it, such as UrbanSound8k, ESC-50.

UrbanSound8K

UrbanSound8K is a fundamental dataset with real field-recordings of the urban environments selected from FreeSound project. Salamon et al. manually checked over 60 hours of audio by listening and inspecting the user-provided metadata then resulting 1302 variable length recordings with timestamps for sound events and salience annotations. After that, recordings were separated into 8,000 labelled slices. UrbanSound8K also contains a taxonomy with 4 top-level groups: human, nature, mechanical and music, which are common to most previously proposed taxonomies. Fig. 2.1 represents the principles and the construction of the 101 classes.

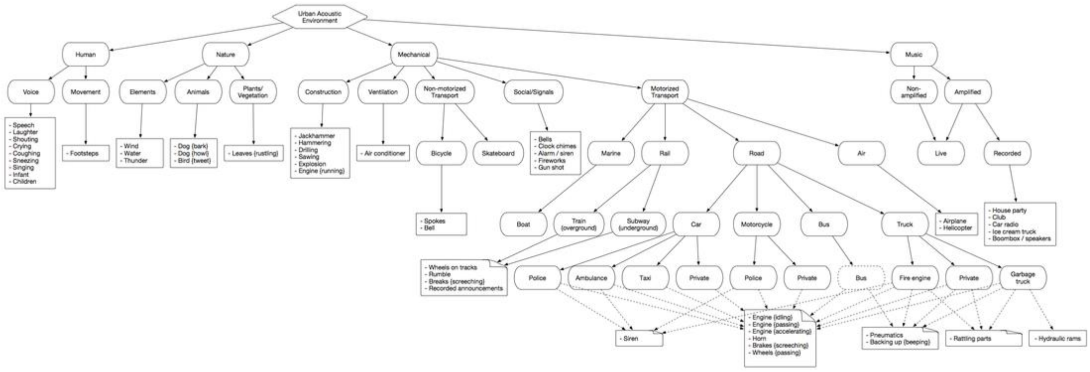


Fig. 2.1 Urban Sound Taxonomy

AudioSet

Since its inception in 2017, the AudioSet database has been the largest audio dataset to date. It includes 1,789,621 audio segments in 10-seconds long of YouTube videos and a taxonomy with 632 audio classes guided by the literature and manual curation. The taxonomy is called the Audio Set Ontology which uses spontaneous descriptors with a maximum hierarchical depth of 6 levels. Comparing to UbanSound8k with meticulous lexica such “Walking on leaves”, AudioSet ontology simplifies it as “Walk, footsteps”. Fig. 2.2 shows the 50 first- and second-level classes in the ontology.

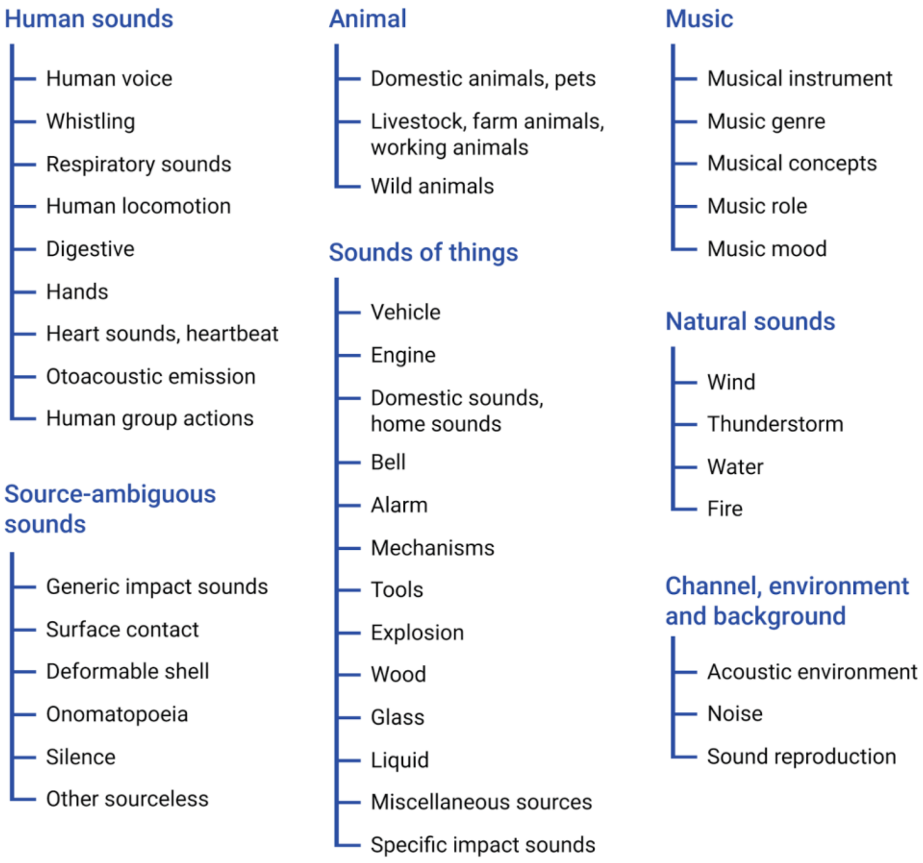


Fig. 2.2: AudioSet ontology

ESC

The ESC dataset is a freely available project made by Karol J. Piczak to facilitate open research initiatives. Over 250,000 environmental recordings are collected through the FreeSound project and unified into 5 seconds long, 44.1 kHz sample rate. It composed of two subsets. ESC-50 contains 2000 manually annotated clips, while ESC-US is a compilation of 250,000 clips with metadata (tags/sound descriptions) which are not verified individually by the dataset author (ESC: Dataset for Environmental Sound Classification). It also provides an estimation of human-level performance as a baseline approaches against machine classification. This study uses the ESC-50 database for the model training and testing. More details about ESC-50 will be provided in the Section 3.2.

2.3 Data Understanding

In constast to the time-varying aspects of most environmental sounds, non-staionary feature extration is considered as more appropriate in classifying enviromental sounds (Bountourakis, Vrysis, & Papanikolaou, 2015). Due to the nature of enviromental sounds, a audio signal could be a set of infinite sinusoidal curves which computer can hardly computed. The process of spliting the signal into discrete time frames is the prerequisite for non-stationary feature extraction, because it allows frequencies to be identified as occurring in a particular area of the signal. The duration of a frame is often in the range of 10-30 ms. In order to analyse the spectrum, a window function (i.e. Fast Fourier Transform) is often applied to reduce the ripples of the sine waves on either side and smooth the signal for further feature extractions. Framing-based processing often implies a Hanning or a Hamming window to get a pulse like Fig. 2.3 below.

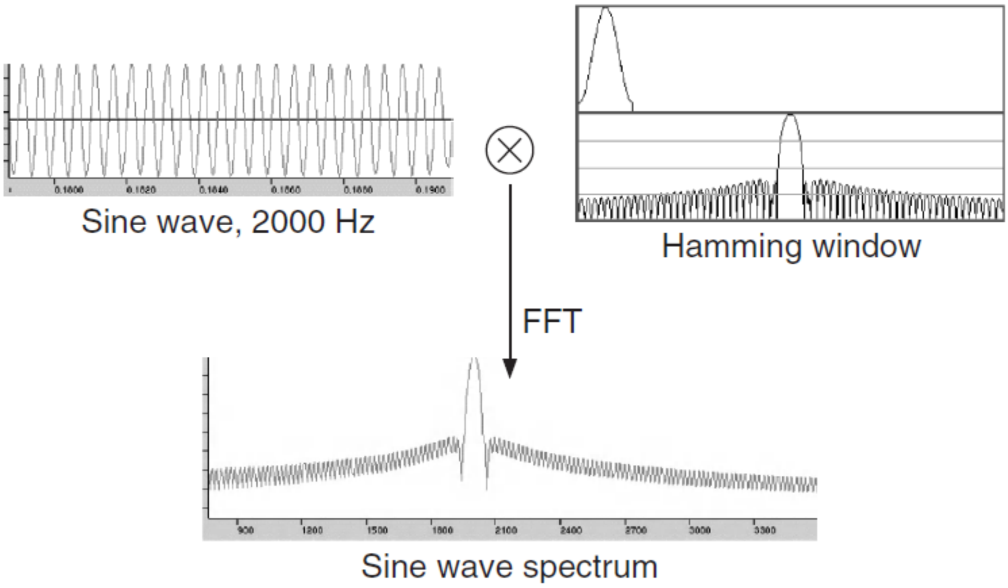


Fig. 2.3 Effect of applying a window in the time domain

The preferred choice of sample rate is 44,100 Hz which identical to an audio CD quality in most of the environmental sound datasets. Regarding the sample rate of the signal, a frame size of 256, 512, or 1024 samples with some degree of overlapping between adjacent frames, such as 25% or 50%, to prevent loss of information around the edges of the window (Sharan & Moir, 2016). There are three commonly used time-segment processing schemes (Chachada & Kuo, 2013): framing-based processing, sub-framing-based processing, and sequential processing. A typical sequential process which can be seen from Fig. 5 segments a signal into 20-30 ms long with 50% overlap. Therefore, the sequential signal model like the Hidden Markov Models [1] (HMM) could capture the inter-segment correlation and the long-term variations of the sound.

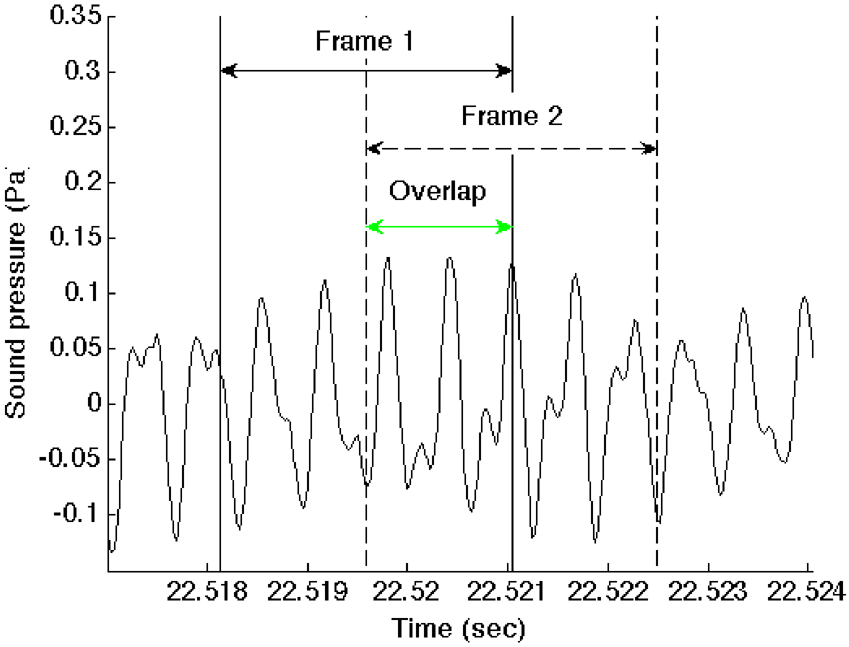


Fig. 2.4 Two analysis frames and the overlap

2.4 Environmental Sound Feature Extraction

In the respect of most ESR systems, feature extraction and sequential learning methods are the keys to maximise the performance and stability. This section covers commonly used techniques for ESR processing. In the view of fact that the audio signal carries overly redundant and irrelevant information, the goal of feature extraction has

[1] HMM is a statistical model which can make predictions for the future of the process based solely on its unobserved (i.e. hidden) states.

generally been to filter out the excess information and obtain compact feature vectors of the salient characteristics of the environmental sound (Alías, Socoró, & Sevillano, 2016). Owing to feature vectors have high dimensionality issues called "curse of dimensionality" by Bellman (2010), data dimentionality reduction usually would be the following process of extraction. Over the past few decades, many variants of Fourier analysis, filter banks and cepstral vectors have been used for environmental sound feature extraction.

2.4.1 Types of Sound Feature

Feature extraction approaches differ on the domain of operation, ranging from the classic frequency and cepstral domains to the derivation of features based on the recent sound representations (Alías, Socoró, & Sevillano, 2016). Time domain, frequency domain, and cepstral domain are the primarily applied in ESC systems. Fig. 6 below is a taxonomy illustrating the relationship between the prevalent sound features and the corresponding domains. A detailed taxonomy of features was given in Appendix A.

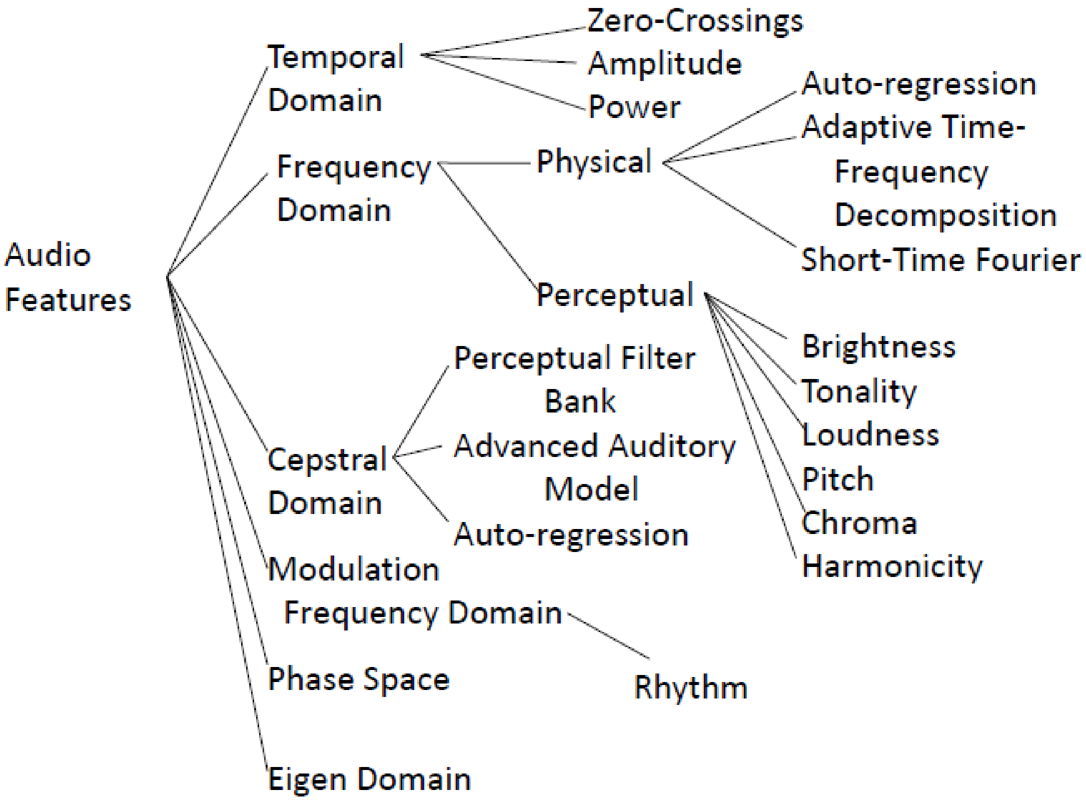


Fig. 2.5 Taxonomy of audio features

- Temporal domain – represents the relatively straightforward features such as amplitude, power and zero-crossing rate[2]. Simplex time-based features are often not capable to drive a classifier (Gerhard, 2003).

- Frequency domain - is broadly categorised as perceptual and physical (Sharan & Moir, 2016). Perceptual features rely on the ways used by human to classify sounds such as pitch, loudness, and timbre. Comparing to the perceptual features, physical features are relatively easier to extract and recognized by a machine, because they are usually obtained from the Shor-Time Fourier Transform (STFT) and can be directly measured without human biases. Thus, they contribute the largest set of audio features reported in the literature (Mitrović, Zeppelzauer, & Breiteneder, 2010). Also, the statistical restults of individual frequency channels are captured at this domain.

- Cepstral domain – is compact representations of the spectrum and provide a smooth approximation based on the logarithmic magnitude (Alías, Socoró, & Sevillano, 2016). Perceptual filter banks-based cepstral features often simulate and synthesize the frequency selectivity of the cochlea. It comprises the famous Mel Frequency Cepstral Coefficients and their variants such as Equivalent Rectangular Bandwidths (ERB) (Moore, Peters, & Glasberg, 1990), Bark (Zwicker, 1961), critical bands (Greenwood, 1961) and octave-scale (Maddage, Xu, Kankanhalli, & Shao, 2004).

2.4.2 MFCC Features

MFCCs have consistently shown a good performance in sound classification. In the early 2000s, the European Telecommunications Standards Institute standardised an MFCC algorithm as the principal data reduction tool to be used in mobile networks

[2] Zero-crossing rate is extracted from time domain but captures the frequency information of the signal.

(Pearce, 2003). Due to the lack of a standard database, many researchers chose MFCCs to benchmark the performance of new classification approaches. Hence, MFCCs has been widespread in every aspect of environmental sound.

At the initial stage, researchers were focusing on using MFCC to recognise specific animal species such as Canada goose (C. Kwan, 2006), frog (Huang et al., 2009). Cai et al. (2007) developed a real-time model for bird species classification. A multilayer perceptron neural network was used to learn the pattern of MFCCs vectors. The study presents that the number of hidden units in a neural network plays an essential role in the performance. An optimal recognition rate of 86.3% was achieved when the number of hidden units around 80. However, the rate almost remained unchanged when the number of hidden units was increasing to 160.

Temko and Nadeu (2006; 2009) conducted a sequence of experiments focusing on the indoor-sounds. They built two MFCC-based classifiers: SVM with decision surfaces; Gaussian mixture model[3] (GMM) with probability distributions and compared the classification capability by the confusion matrix. In those tests, the SVM model had the best results with 88.29% classification rate. For the audio scene recognition, Eronen et al. (2006) investigated 24 classes of ambient sounds such as restaurant, office and train. Through training a five-component GMM based on the MFCCs for each class, they obtained the GMM model recognition rate of 63% which was superior than 61% using the 1-NN classifier. Afterwards, Chu et al. (2009) proposed the matching pursuit (MP) algorithm to extract multiple time-domain features, then learn the pattern combined with MFCCs. The algorithm yielded outstanding results – averaged accuracy rate of 83.9% in fourteen classes. The classification rates of 7 classes are more than 90%.

[3] GMM is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

Subsequently, MFCCs have expanded to soundtrack classification. In 2010, Lee and Ellis adopted Eronen et al.'s (2006) model as a baseline comparison system. They introduced a novel technique - probabilistic latent semantic analysis (pLSA) for classifying consumer video clips based on their soundtracks. They also compared MFCC frame reduction performance of three different techniques: Single Gaussian modelling (1G), Gaussian mixture modelling, and pLSA of a Gaussian component histogram. After comparing the average precision and accuracy rate, they concluded the pLSA model gave the best results consistently, nonetheless the margin of improvement was too small to carry conviction.

2.4.3 Sound Texture Statistical Features

Sound texture originates from sound synthesis. A storm sound could be regarded as the hybrid of rain falling and wind blowing. The rain falling sound can be further broken down into myriad water drop sounds. Base on the decomposability, Saint-Arnaud & Popat (1995) define sound textures in two levels: the low-level sound atoms (features), and the high-level periodic and stochastic distributions of sound features. The sound texture statistics model the distributions.

In the early stage, Markov chain[4] debuted as the prime statistical estimate in music and speech resynthesize. Voss and Clarke (1975) investigated the long-time power-spectrum of environmental sounds by Markov process, then found that energy falls off with increased frequency according to a 1/f law. However, the important limitation is the second-order statistic can only obtain a inadequate marginal distribution when the sound amasses on low-energy bands. Furthermore, inspired by image texture analysis, EI-Yaniv and Dubnov (1999) applied a Markovian unsupervised clustering algorithm to sound textures, achieving a discrete statistical model of a sequence of paths through

[4] Markov chain shares the same principle with HMM model. The only difference is the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters, while in the HMM, the state is not directly visible.

a wavelet tree[5] representation. Even though their results demonstrated a high-quality resynthesized jazz ensemble, it was the recombination of different segments of the musical instruments instead of working from the low-level sound textures.

To cover the weakness of the second-order statistics and extract the highly kurtotic of energy in sub-bands, McDermott et al. (2009) applied the neurophysically motivated statistics to noise filtering synthesis. They segmented the signal into frames by sequential processing with 50% overlap rate. Then a cascade of two kinds of filter banks narrowed down the signal to mimic the psychoacoustical cochlear crital bands, which conformed to the signal process from the cochlea through the thalamus. The set of marginal moments (mean, variance, skew, and kurtosis, and correlations) were used to calculate the envelopes of the histogram. Finally, by modifying a white noise signal according to the desired statistic moments as the descriptor of the energy distribution. The synthesize model produced very compelling results and revealed the underlying invariances of sound texture which can be obtained by the right statistics.

2.5 Model Performance and Issues

After the features extracted from the labelled training samples, the essential task of sound classification is to learn consistent sound feature representations by a well-formulated mathematical framework. Most of the formal training algorithm are model-based, such as SVM, ANN, HMM, GMM.

In order to compare the performance of commonly employed models for ESR, Cowling and Sitte (2003) presented a comprehensive comparative study of both stationary and non-stationary features combined with 10 models. Table 2.1 below shows a part of the performance related to MFCC and Long-term Statistics (LTS) based on the spectrogram. The study gave a general performance outline of each combination. From the point of view of MFCC, the GMM model performs better than

[5] The wavelet tree is a succinct structure for multi-scale decomposition of the signal and can be viewed as a complete tree.

the ANN model. Overall, the MFCC based models outperform the statistics based model like HMM and LTS. Due to it's a self-recorded database with insufficient environmental sound, the author noted that it is too small to make a meaningful comparison, and statistical techniques need to be revisited in the future.

The most relevant work in regard to the objectives of the thesis is the research done by Ellis et al. (2011). They examined the sound texture statistical techniques with 6630 soundtracks for the TRECVID 2010 Multimedia Event Detection task. They developed three SVM classifiers based on three feature sets: second-order statistics of MFCC features; statistical moments proposed by McDermott et al. (2009); the combination of the first two feature sets. The combination system outperformed in every system with averaged accuracy of 75.5%. The study also provided the performances of each subset of the texture feature blocks, which demonstrated the higher order moments are better than the mean subband energies. In conclusion, all the reviews showed that any techniques alone cannot achieve successful recognition rates. Most of the state-of-the-art ESR models tend to use greedy schema to integrate abundant sound features. See Table 2.1 for a summary of the average accuracy of each model referenced by this chapter.

Study	Year	Dataset(s)	Feature	Classifier	Classification Accuracy
Cowling & Sitte	2003	Self-recorded database consists of 8 classes like *Footsteps on leaves*, *Footsteps on glass*.	MFCC	ANN	37.5%
			MFCC	GMM	46%
			FT	LTS	29%
			Power FT	LTS	29%
Chu, Narayanan, & Kuo	2009	BBC SoundEffects, FreeSound	MFCC +MP	GMM	83.9%
Karbasi,	2011	BBC SoundEffects,	MFCC	GMM	62.69%

Ahadi, & Bahmanian		FreeSound		SVM	75.49%
			ΔMFCC	GMM	41.65%
				SVM	70.10
Cai, Ee, Pham, Ro, & Zhang	2007	Self-recorded dataset consists of 14 bird species	MFCC	HMM + ANN	86.8%
Ellis, Zeng, & McDermott	2011	TRECVID 2010	Statistical moments	SVM	72.5%
			MFCC	SVM	73.8%
			Statistical moments + MFCC	SVM	75.5%
Lee & Ellis	2010	1,873 sound clips extracted from 4,539YouTube videos	MFCC	GMM	87.3%
				1G	85.2%
				pLSA	88.9%

Table 2.1: Literature Review of studies

2.6 Evaluation and Results

In terms of statistical measures, many researchers chose to use measures such as precision and recall, which are two widely used statistical criteria. Precision can be seen as a measure of exactness or fidelity, whereas recall is a measure of completeness. Researchers use varying evaluation techniques for their models. However, the standard statistical methods are used. The most common evaluation methods used in sound tagging area are F-score measure and Receiver operating characteristic (ROC) curves.

F-measure is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score. The F-score can be interpreted as a weighted average of the precision and recall, where an F score reaches its best value at 1 and worst score at 0 (Yong & Ying, 2010). From the year 2006, Temko and Nadeu (2006; 2009) chose F-measure to compare their discriminative capability in the application. In 2010, Cheng et al. stated that the results of MFCCs with GMM are promising by F-measure. For wood detection, Yella et al. present an F-score comparison of several pattern recognition techniques combined with various stationary feature extraction techniques for classification of impact acoustic emissions (Yella, Gupta, & Dougherty, 2007). Measurements showed that any technique alone cannot achieve successful recognition rates.

ROC curve is a graphical plot of the sensitivity, or true positive rate vs. false positive rate. The ROC can also be represented equivalently by plotting the fraction of true positives out of the positives vs. the fraction of false positives out of the negatives. The ROC is also known as a Relative Operating Characteristic curve, because it is a comparison between two operating characteristics (True Positive Rate & False Positive Rate) as the criterion changes. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. Hershey et al. calculated the balanced average across all classes of Area Under the Curve (AUC), which is the area under the Receiver Operating Characteristic (ROC) curve, and mean Average Precision (mAP) (Hershey, et al., 2016). The evaluation results calculated over the 100K balanced videos. It shows that all CNN models beat the baseline model.

2.7 Conclusion

This chapter has critically examined the many sound features currently affecting ESC researches. It clearly exhibits there are various methodologies were taken to solve the seemingly intractable sound classification problem. Comparative studies reduce uncertainty and aid focusing the research efforts on the algorithms, features and methodological approaches that will offer the best opportunity for ESC.

3 DESIGN AND METHODOLOGY

This chapter presents the plan and the design methodology for the current study. Several generally accepted data mining methodologies were used to construct a robust data mining workflow. The key stages are Data Understanding; Data Preparation, Feature Extraction, Feature Reduction, Data Partitioning, Modelling and Evaluation. The brief methodology is provided in the next Section.

3.1 Overview of Methodology

The three key steps for an environmental sound classification (ESC) system are signal pre-processing, feature extraction, and classification. Fig. 3.1 describes a model of a statistical pattern recognition employed in the most ESC applications. Firstly, the time-series audio signals in the trainning set are segmented into smaller frames, often into the duration of 10-30 ms. Features are extracted form each frame for analysis. A algorithm based classifier learn to match the feature patterns with correspodding sound descriptors. After training, the classifier was given task to make decision using the statistics absorbed from the test dataset.

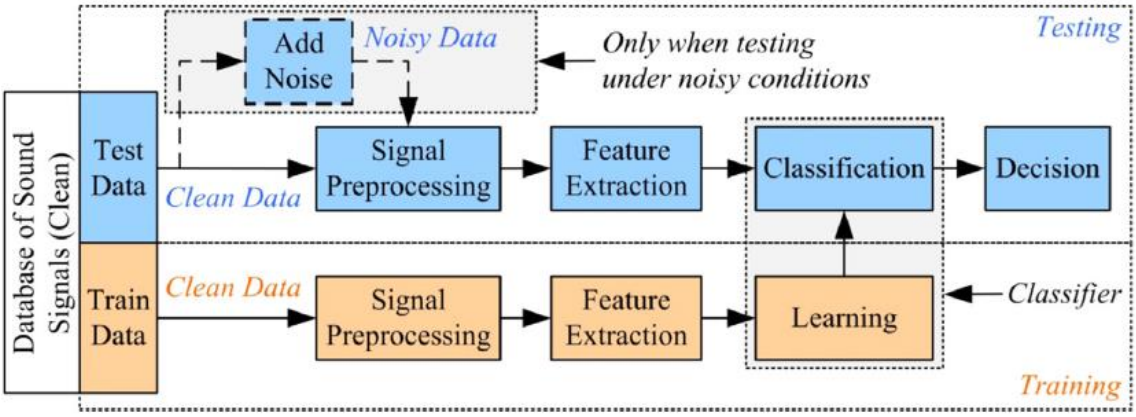


Fig. 3.1 Model of a statistical pattern classifier

The main phases of the methodology are briefly:

1. Data understanding – A well-labelled environmental sound database is required for the classifier training. This phase introduces the ESC-50 datasets as the meta data for the project, as well as the details of data categories, data file format, sample rate and the sound duration etc.

2. Data transformation – In order to extract the sound features, each sinusoidal signal was decomposed into a sequence of consecutive windows. Then a STFT transform translates each window from time domain to frequency domain, resulting a two-dimensional array which represents the power spectrum of the sound clip.

3. Environmental sound feature extraction – Three sets of features were extracted: MFCCs and their derivatives (ΔMFCC), Mel-spectrogram and sound texture statistics. The phase explains the theories behind each feature and explicates the equations which are used to compute the values.

4. Data modelling and classification – Each set of sound features mentioned above was modelled by an appropriate machine learning algorithm. Three combinations are listed in the following table 3.1

Sound Features	**Machine Learning Algorithms**
MFCCs and their derivatives (ΔMFCC)	SVM with linear kernel
Sound Texture Statistics	SVM with radial basis function kernel
Mel-spectrogram	CNN

Table 3.1 Models

5. Performance evaluation - The 5-fold cross-validation separates database into tanning set and testing set. The experiment results were evaluated by the results of human listeners. The hypotheses were tested by the performance differences of the models with the MFCC baseline model.

3.2 Data Understanding

3.2.1 ESC-50 Dataset

This study uses the manually labelled ESC-50 database provided by Karol J. Piczak, which was introduced in Section 2.2. The database is an open-source project hosed by GitHub for download and maintenance. It consists 2000 recordings that organized into 50 semantical classes (with 40 examples per class) and loosely arranged into 5 major categories: animals; natural soundscapes & water sounds; human non-speech sounds; interior/domestic sounds; exterior/urban noises. Partial ESC-50 category with 15 classes is displayed by Table 3.1. The detailed table of categories is given in the Appendix B Table B.1.

Animals	**Natural soundscapes & water sounds**	**Human, non-speech sounds**	**Interior/domestic sounds**	**Exterior/urban noises**
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren

Table 3.2 Partial ESC-50 categories

3.2.2 Data Transformation

As discussed in Section 2.4, environmental sound frequencies are measured by applying the Fourier Transform. In this research, the STFT transform was used to convert the audio to the frequency domain and result in a complex-valued function of frequency. The real part of the results stands for the magnitude of the signal frequencies. The imaginary part represents the phrase offsets of the set of sinusoidal signals. Thus, the frequency domain allows the research to visualise the sounds across multiple dimensions and preform operations on it. To compute the three-dimensional array STFT {*x*(*t*)} (τ, ω) of the signal *x*(*t*), the usual mathematical equation is shown in Equation 3.1.

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t}\,dt$$

Equation 3.1 STFT

Where the *w(t)* is the window function with length M, usually a Hamming window or Hann window cantered around zero. R is the hop size between successive FFT frames. The FFT function *X* (τ, ω) takes the time axis τ and the frequency axis ω as parameters. Fig. 3.1 illustrates a normative STFT process which is a series of Fast Fourier Transforms (FFT) spaced evenly in time.

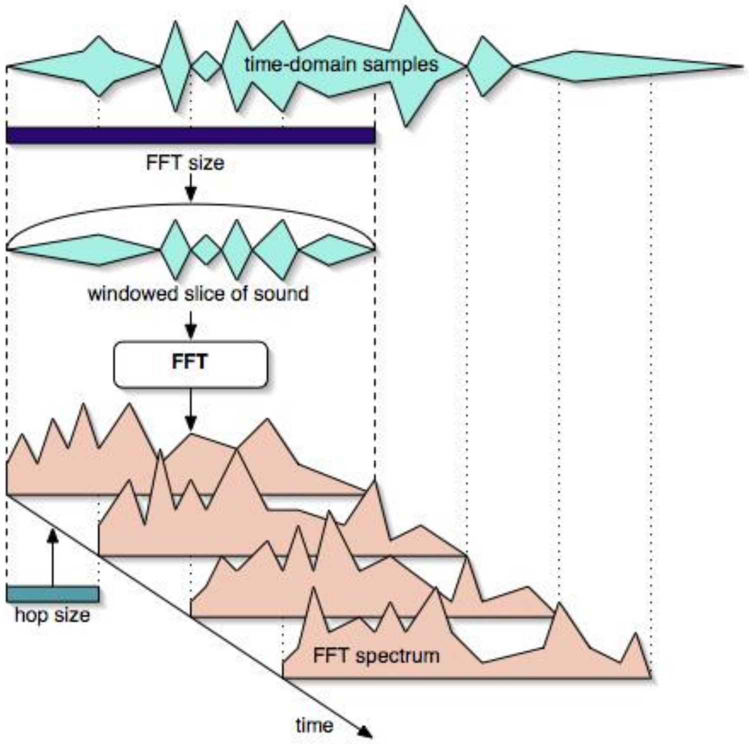


Fig. 3.2 A STFT Process

3.3 Environmental Sound Feature Extraction

3.3.1 MFCC Features

MFCCs and its derivatives (ΔMFCC, ΔΔMFCC) are often regarded as data dimensionality reductions based on Mel-Filterbanks. Because human ears are sharper at listening to sounds in lower frequencies than high frequencies, Mel-frequency scale crudely approximate the perceived frequency in the inner hair cells in the cochlea to the organ of Corti. From the mathematics perspective, Mel-frequency scale basically is a logarithmic spiral. The formula for converting from frequency to Mel-Frequency scale is shown in the Equation 3.2:

$$M\,(f) = 1125\,\ln(1 + f/700\,)$$

Equation 3.2

The equation is plotted in Fig 3.2

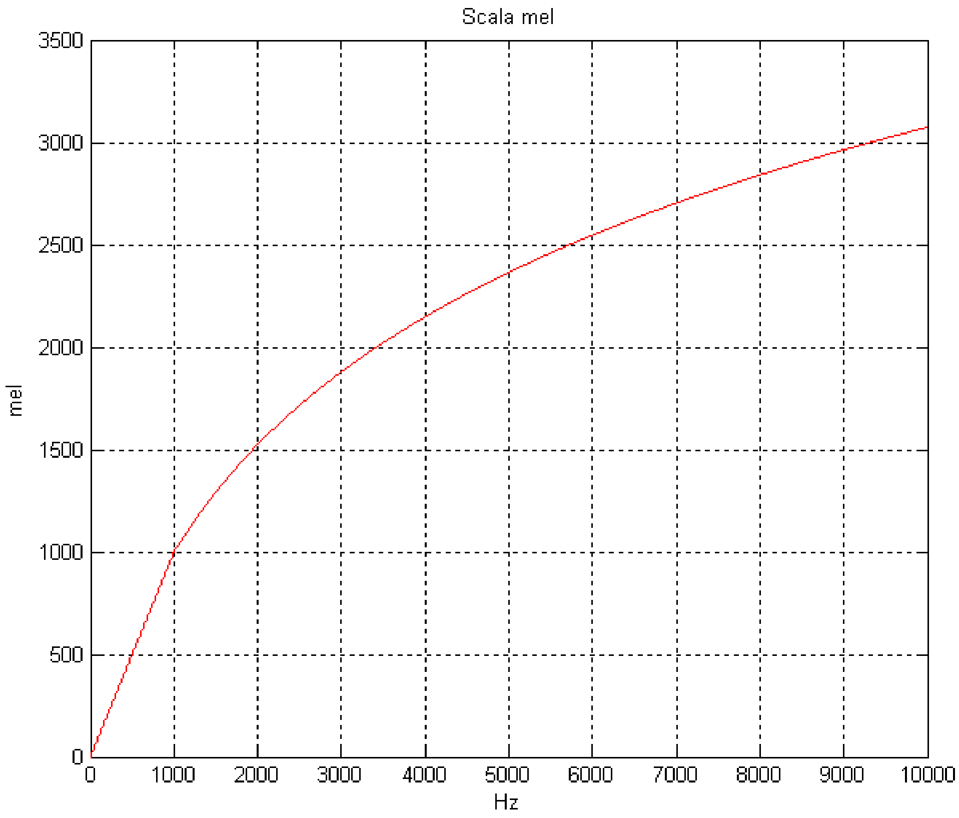


Fig 3.3 Mel Scale

MFCC is usually derived using Mel-filterbanks, which is a set of 20 - 40 overlapped triangular filters are illustrated in Fig. 3.2. To remove the extra energies, Mel-filterbanks function as bandpass filter by multiplying each filterbanks H_i () with power spectrum S(*n*). A logarithm would be used to filter the loudness that human hearing cannot perceive.

$$Y(i) = \sum_{k=0}^{N/2} \log|s(n)| \cdot H_i\left(k \cdot \frac{2\pi}{N'}\right)$$

$$\widetilde{Y}(k) = \begin{cases} Y(i) & ,k = k_i \\ 0 & ,otherk \in [0, N'-1] \end{cases}$$

$$c_s(n) = \frac{2}{N'} \cdot \sum_{i=1,2,\ldots,N_{cb}} \widetilde{Y}(k_i) \cdot \cos\left(k_i \cdot \frac{2\pi}{N'} n\right)$$

Equation 3.3

Where *Y* (*i*) is the filtered energies, N_{cb} is the number of Mel-filterbanks. So, the MFCCs can be calculated by the Equation 3.3 above. The Discrete Cosine Transform (DCT) transforms the complex number results to real numbers.

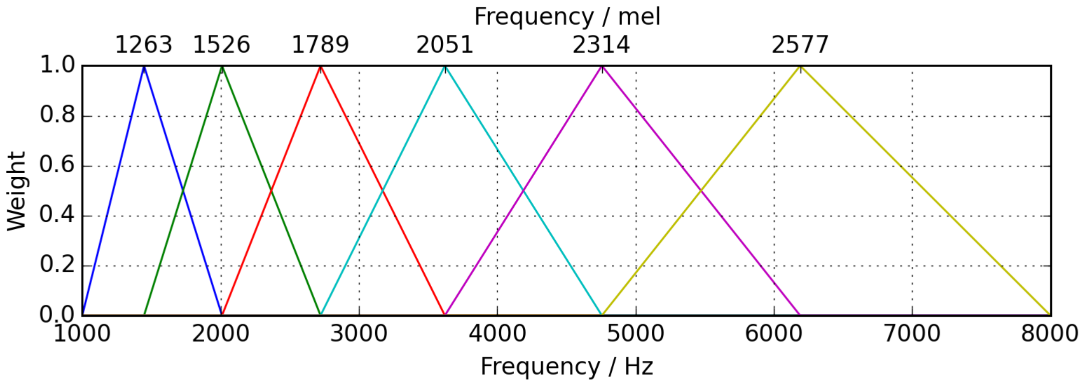


Fig. 3.4 Mel-Filterbanks

3.3.2 Mel-spectrogram Features

After computing a STFT transform, the squared magnitude of the audio signal was obtained. The results can be used to plot a three-dimensional spectrogram with the time axis τ and the frequency axis ω, which represents the spectrum of frequencies as they vary with time. For the convenience of display, the common spectrum was compressed into two Dimensions, which represent the squared magnitude by the intensity or the gradation of colour. For instance, the yellow lines in Fig 3.4 indicate the power peaks of a helicopter sound clip. They also mean several sound textures playing at the same periods.

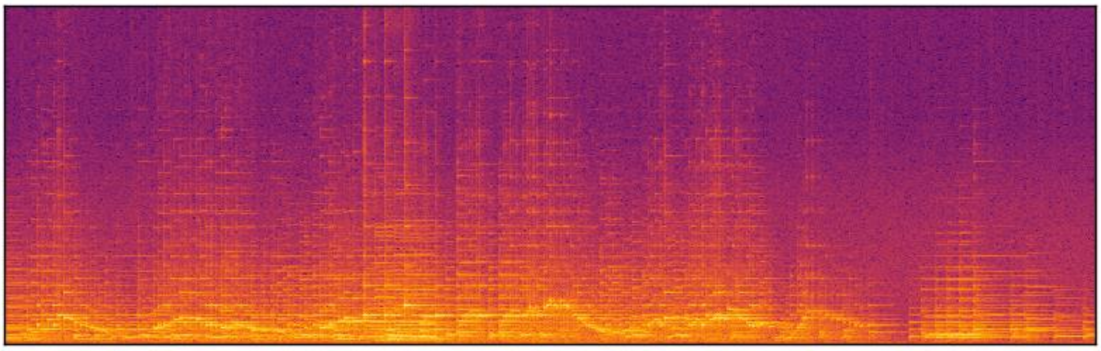


Fig. 3.5 Spectrogram of Helicopter Sound

The CNN classifier requires the conspicuous spectrogram structures to achieve better results. Therefore, the study transformed the raw spectrograms into Mel-spectrogram by applying Mel-filterbanks. The Mel-spectrogram of the helicopter sound is more recognizable than the spectrogram for identification. See Fig 3.4.

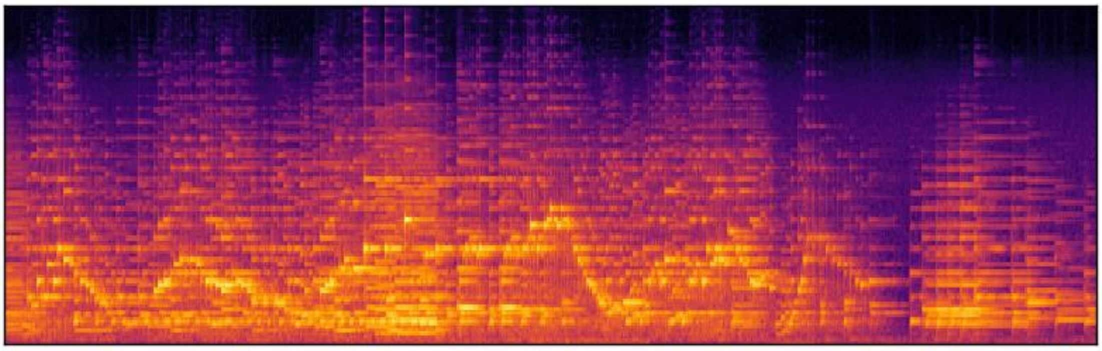


Fig 3.6 Mel-spectrogram of helicopter sound

3.3.3 Sound Texture Statistical Features

Following on from the discussion in Section 2.4.4, the three-dimensional Mel-spectrogram can be broken into several sub-bands along the frequency axis ω, which resulted the histograms of magnitude. The envelope of the histogram and the correlation between sub-band envelopes were testified to be ponderable by McDermott and Simoncelli (2011) The envelopes were analysed as the texture representation by the four marginal moments (mean, variance, skew and kurtosis). The *k* is an ordinal number corresponding to the kth sub-band envelopes in the is represented by s_k *(t)*. The *w(t)* denotes windowing function. The equations are listed below:

$$M1_k = \mu_k = \sum_t w(t) s_k(t),$$

Equation 3.4 Mean

$$M2_k = \frac{\sigma_k^2}{\mu_k^2} = \frac{\sum_t w(t)(s_k(t) - \mu_k)^2}{\mu_k^2},$$

Equation 3.5 Variance

$$M3_k = \frac{\sum_t w(t)(s_k(t) - \mu_k)^3}{\sigma_k^3},$$

Equation 3.6 Skew

$$M4_k = \frac{\sum_t w(t)(s_k(t) - \mu_k)^4}{\sigma_k^4} \quad k \in [1 \ldots 32]$$

Equation 3.7 Kurtosis

In 1999, Nelken at al. (1999) found the cross-band correlations between the envelopes, or “co-modulations”, were universal in the natural sounds. Then McDermott and

Simoncelli (2011) agreed with that and proved the co-modulations are the major source of variation among sound textures. To provide a qualitative from of correlation matrix, this research calculated the co-modulations of each envelope with a subset of eight of its neighbours. See Equation 3.8

$$C_{jk} = \sum_{t} \frac{w(t)\left(s_j(t) - \mu_j\right)\left(s_k(t) - \mu_k\right)}{\sigma_j \sigma_k},\ j,\ k \in [1 \ldots 32]$$

Equation 3.8 Co-modulation

The modulation power is the last statistical parameter to capture. First, a FFT was used to transform the magnitudes into a modulation spectrum. The magnitudes were splinted into 6 sub-bands. Each band is octave-wide spanning 0.5-1 Hz, 1-2 Hz, 2-4 Hz, 4-8 Hz, 8-16 Hz, and 16 Hz to the Nyquist rate of 32 Hz. Finally, the proportions of total power are calculated by each band as shown in Equation 3.9.

$$M_{k,n} = \frac{\sum_t w(t) b_{k,n}(t)^2}{\sigma_k^2},\ k \in [1 \ldots 32],\ n \in [1 \ldots 20].$$

Equation 3.9 Modulation power

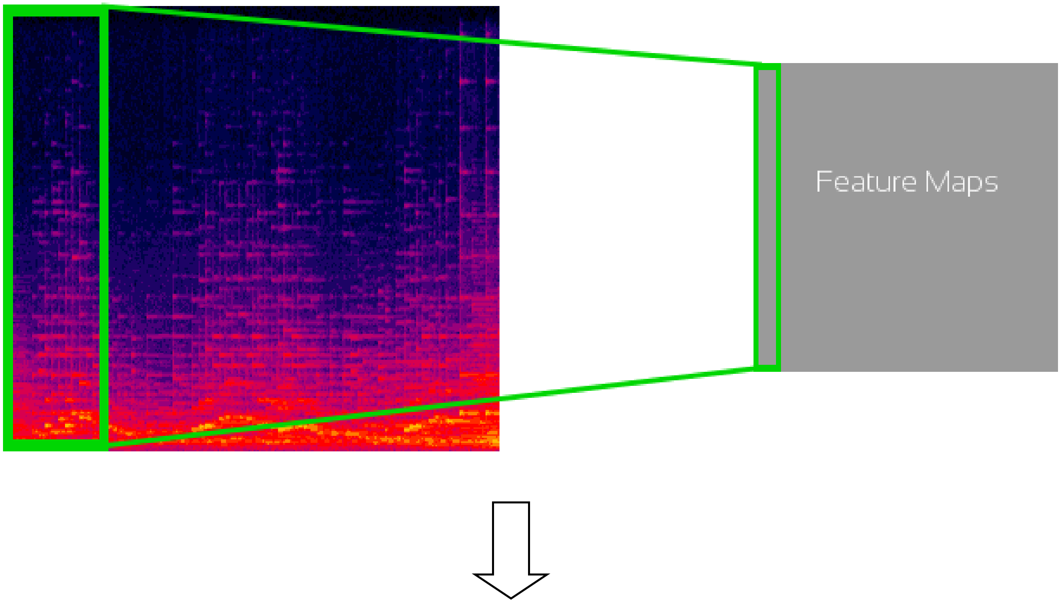
Finally, the statistical relationships between all the sub-band envelopes were analysed.

3.4 Data Modelling and Classification

The objective of the research is to carry out an evaluation of machine learning techniques to investigate the classification capability of different environmental sound features. In this stage, two kinds of machine learning methodology were utilized to train the classification models.

The first technique to be deployed is SVM. A SVM with linear kernel was used to train the baseline model with MFCC features. The goal for the baseline model is to get a general benchmark of the dataset, without optimizing for the maximum classification accuracy. Another SVM with radial basis function (RBF) kernel was used to work with the sound texture statistical features. The RBF kernel, also called Gaussian kernel, supports full covariance matrices. Therefore, this model is capable to calculate the Euclidean distance between the statistical feature X and Y, for each pair of rows x (*i.e. marginal moments, envelope correlations*) in X and y in Y.

The third model is based on a typical CNN for the Mel-spectrogram image classification. The problem for this research is the dataset is fairly small for a proper CNN training. To address the problem, the layers with the basic functions like edge detection and shape detection were transformed from a pre-trained model called Inception[6], which has been trained in a large image dataset called ImageNet[7], to this CNN model. The CNN architecture consists of number of layers: input layer, pooling layers, hidden layers and output layer. The Mel-spectrogram and their deltas as a 2-channel input to the CNN. See Fig 3.5



[6] Inception is an experimental Google product: https://github.com/google/inception

[7] ImageNet is available with the following link http://www.image-net.org/

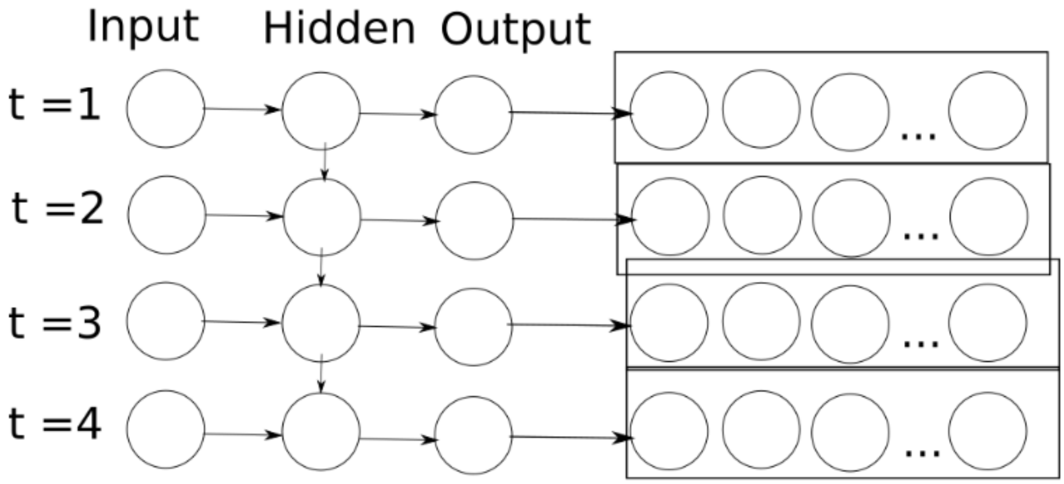


Fig. 3.7 CNN architecture

3.5 Performance Evaluation

Supervised Machine Learning methodology was required to split the dataset into a training set and test set. It could prevent the test leaking into the training set and resulting the false alarm with a surprisingly high accuracy. Due to the usability, k-fold cross validation is commonly used methodology to compare models for a given classification problem. As mentioned in Section 3.2.2, the ESC-50 database initially split data into 5 unique groups. Thus, this research took advantage of that and uses 5-fold cross validation. The cross-validation process was repeated 5 times. At each time, these 4 group were modelled as training data by the above discussed machine learning models, while the left group was retained as the validation data for testing. Every group is used for validation exactly once. The overall performance is the mean value of the 5 results. It measures the fitness of a classification model. The positive or negative results of classification tabulated and displayed as the confusion matrix.

Furthermore, a human classification model was used as a high-level reference object to compare with the other three models which based on the perceptually informed data. The data were collected form Karol J. Piczak’s experiment, which tested the sound classify abilities of several participants by the sounds in ESC-50 database, then received around 4000 judgments which is also tabulated as the confusion matrix. It provides a rough estimate of human capabilities in recognizing environmental sounds.

Accepting or rejecting the null hypothesis will be based on the evaluation measure calculated in the next chapter.

4 IMPLEMENTATION AND RESULTS

This chapter outlines how the experiments were carried out, based on the research methodologies discussed in the previous chapter. The first three sections describe the practical steps taken to complete the data understanding and the sound feature extractions. The last section shows partial results with a limited discussion as guidance. The full set of results are provided in the Appendix B. The python scripts for experiment implementation are listed in Appendix C.

4.1 Data Understanding

The recordings are unified into 5 seconds long, 44,100 Hz sampling rate, single-channel (mono) clips. The clips use the Waveform Audio File Format, commonly known as the filename extension "wav". They were lossy compressed at 192 Kbit/s by Ogg Vorbis[8]. The total sized of the database is roughly 843 MB.

The database provided a XML file which describes: file ID; category name; category ID; original source ID from the FreeSound project and the file sequence letter indicating the file's position in the original sources. Table 3.2 shows tree samples of the XML file. The filename follows the naming convention below:

{Folder ID} - {Source ID} – {Sequence Letter} – {Category ID}.wav

The last two samples come from the same "clapping" recording, thus they share the same source file ID.

Filename	Folder ID	Category ID	Category	Source file ID	File Sequence
1-100038-A-14.wav	1	14	chirping_birds	100038	A
1-104089-A-22.wav	1	22	Clapping	104089	A

[8] Ogg Vorbis is an open-source software that produce smaller files at higher quality while comparing to Windows Media Audio.

1-104089-B-22.wav	1	22	Clapping	104089	B

Table 4.1: The XML file samples

The clips were divided into 5 uniformly sized folders for comparable cross-validation, making sure that the clips from the same original source file are contained in a single folder. As mentioned in Section 3.2.1, ESC-50 consists 2000 clips organized into 50 semantical classes. In other words, each folder has 8 clips per class and 400 clips in total. Accordingly, the training set has 32 clips per class and 1600 clips in total which have a duration of 8000 seconds. The summary of environmental sound raw data for each cross-validation is shown in Table 4.2.

	Clips per class	**Clip duration per class (s)**	**Samples per class**	**Total Clips**	**Total duration (s)**	**Total samples**
Training	32	160	7,056,000	1,600	8,000	352,800,000
Testing	8	40	1,764,000	400	2,000	8,820,000
Total	40	200	8,820,000	2,000	10,000	441,000,000

Table 4.2 Summary of ESC-50 data

4.2 Data Preparation

Mel-spectrogram

To prepare the data for the experiment, several data preparation processes were carried out. The first step was to transform the data from time domain to frequency domain. The research experimented with the sequential processing for data segmentation. Hence, the selected hop size is 512 samples equated to a quarter of the FFT window size, which determines the 75% overlap. The FFT window size is 2048 frequency bins from 0 Hz to the sampling frequency. The STFT transform has been performed by a python library called Librosa[9] which is a frequently-used tool in audio processing. The function *librosa.feature.melspectrogram* firstly computed the magnitude spectrogram *S* by FFT, then mapped the *S* on to the Mel-scale by *mel_f.dot(S^2)*, finally called the function librosa.filters.mel creating 128 Filterbanks to combine FFT bins into Mel-frequency bins. The python script is illustrated below:

```
self.melspectrogram = librosa.feature.melspectrogram(audio.raw,
                                                     sample_rate = 44100,
                                                     fft_window_size = 2048,
                                                     hop_kength = 512,
                                                     power = 2)
```

The thumbnails of Mel-spectrogram and sinusoid waves plotted in figures below, which covers the 5 main categories.

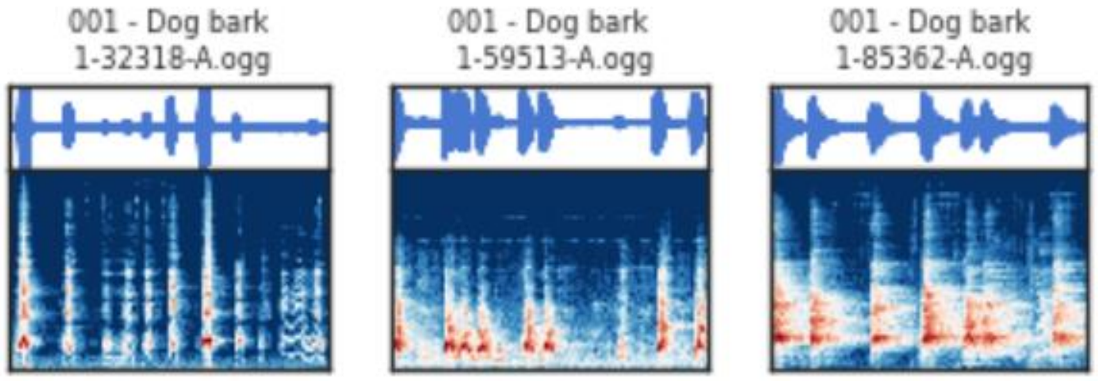


Fig 4.1 Dog

[9] Librosa is available by the following link: https://librosa.github.io/

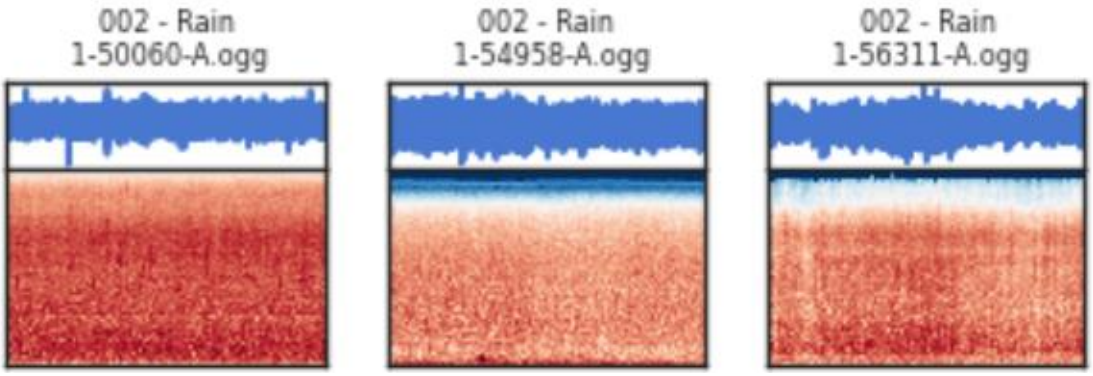


Fig 4.2 Rain

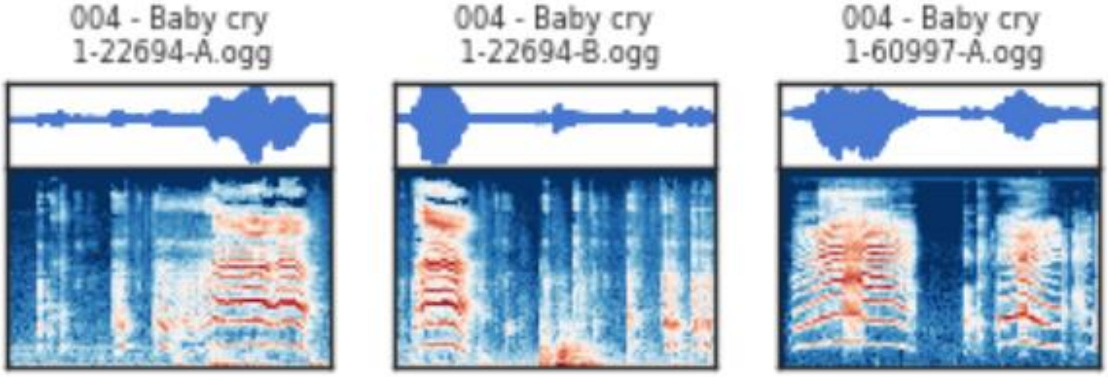


Fig 4.3 Baby cry

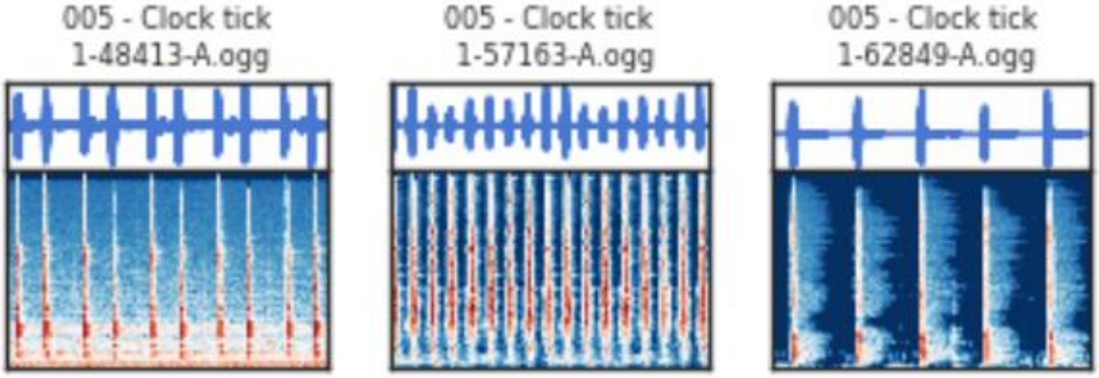


Fig 4.4 Clock

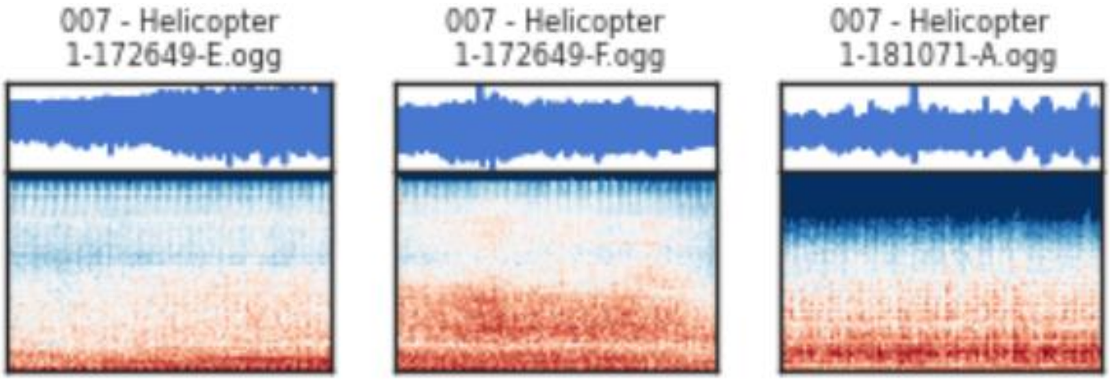


Fig 4.5 Helicopter

MFCC

Similarly, this research utilized librosa package to calculate MFCCs. At the outset, the function librosa.amplidude_to_db convert the Mel-spectrograms to decibel units. Then 13 numbers of MFCCs and ΔMFCC were obtained by the function librasa.feature.mfcc and librosa.feature.delta. The mean values of MFCC were used to train the baseline system. The MFCC distributions of a "Crying baby" clip is shown in the Fig 4.6.

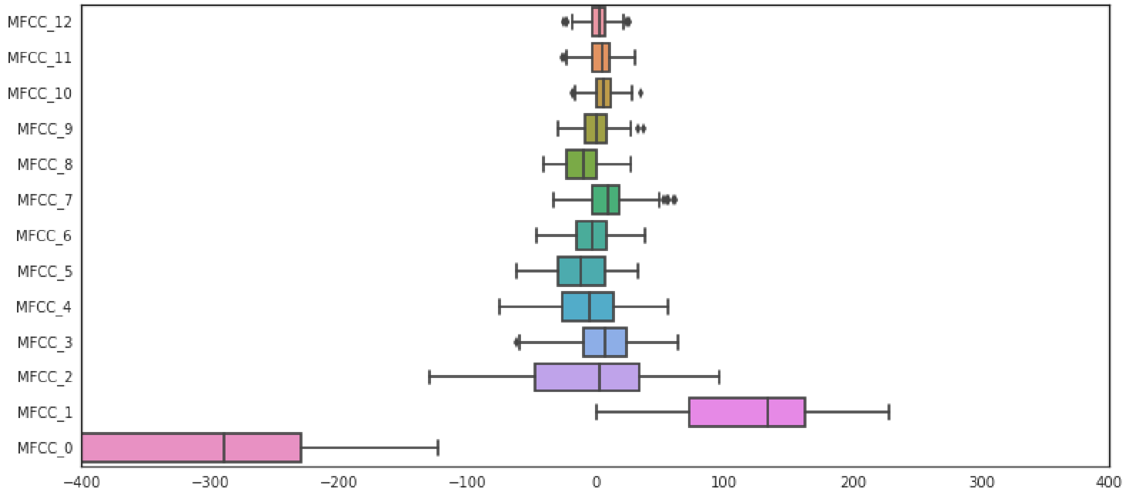


Fig 4.6 Example of MFCC distributions

Sound Textual Statistics

Implementation and results**

When a magnitude spectrogram *S* was mapped on to the Mel-scale, it has been broken into 18 sub-bands along the frequency axis ω. The 4 marginal moments of each sub-bands results a 18*4 feature block. Then a 18*6 modulation power block were extracted by FFT. Finally, the normalized co-modulations of each envelope gave 138 dimensions. Consequently, every clip has been transformed into 18*4+18*6+138 = 318 dimensions. The example results are shown in the Fig 4.7.

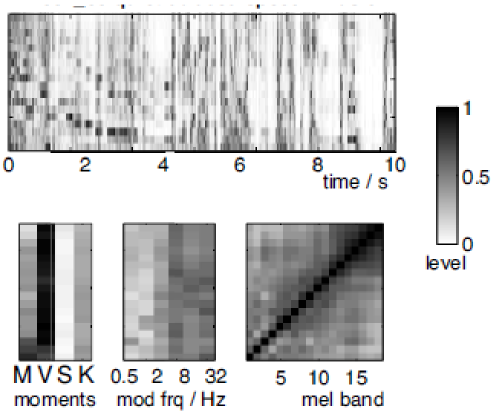


Fig 4.7 Example of sound texture statistics

4.3 Results

This section discusses the key results from the experiments. The positive or negative results of classification tabulated and displayed as the recall for each classifier. The results of a human classification model are also provided. The 5-cross validation results are listed in Table 4.3.

	SVM + MFCC(baseline)	**SVM + Statistical features**	**CNN + Mel-spectrogram**
Fold 1	30.0%	45.1%	38.5%
Fold 2	32.5%	49.5%	39.7%
Fold 3	34.0%	43.7%	39.2%

Fold 4	34.7%	46.0%	40.5%
Fold 5	30.0%	45.2%	39.7%
Average	32.2%	45.1%	39.5%

Table 4.3 Results of 5-cross validation results

The full confusion matrix is too huge to display in this chapter. So, the recall of ten classes are presented for human listener.

Human Listener

Baby cry	Chainsaw	Clock tick	Dog bark	Fire crackling	Helicopter	Person sneeze	Rain	Rooster	Sea waves
Baby cry	**100.0%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Chainsaw	0.0%	**98.3%**	0.0%	0.0%	0.0%	1.5%	0.0%	0.0%	0.2%
Clock tick	0.0%	0.0%	**99.7%**	0.0%	0.0%	0.0%	0.0%	0.3%	0.0%
Dog bark	0.0%	0.0%	0.0%	**99.8%**	0.0%	0.0%	0.2%	0.0%	0.0%
Fire crackling	0.0%	0.2%	0.7%	0.2%	**87.4%**	0.2%	0.0%	11.1%	0.0%
Helicopter	0.0%	4.8%	0.0%	0.2%	0.4%	**91.9%**	0.0%	0.8%	0.0%
Person sneeze	0.4%	0.0%	0.0%	0.0%	0.0%	0.0%	**99.6%**	0.0%	0.0%
Rain	0.0%	0.6%	0.0%	0.0%	6.7%	0.6%	0.0%	**89.7%**	0.0%
Rooster	0.0%	0.0%	0.0%	0.2%	0.0%	0.0%	0.0%	0.0%	**99.8%**
Sea waves	0.0%	1.8%	0.0%	0.4%	0.0%	0.4%	0.0%	6.2%	0.0%

Table 4.4 Recall

The performance of each model in 50 classes are plotted in the Fig 4.8. The blue triangle denotes human performance. The green square denotes the baseline MFCC + SVM classifier. The yellow hexagon denotes the Mel-spectrogram classifier. Finally, the red pentagon denotes the sound texture statistics classifier.

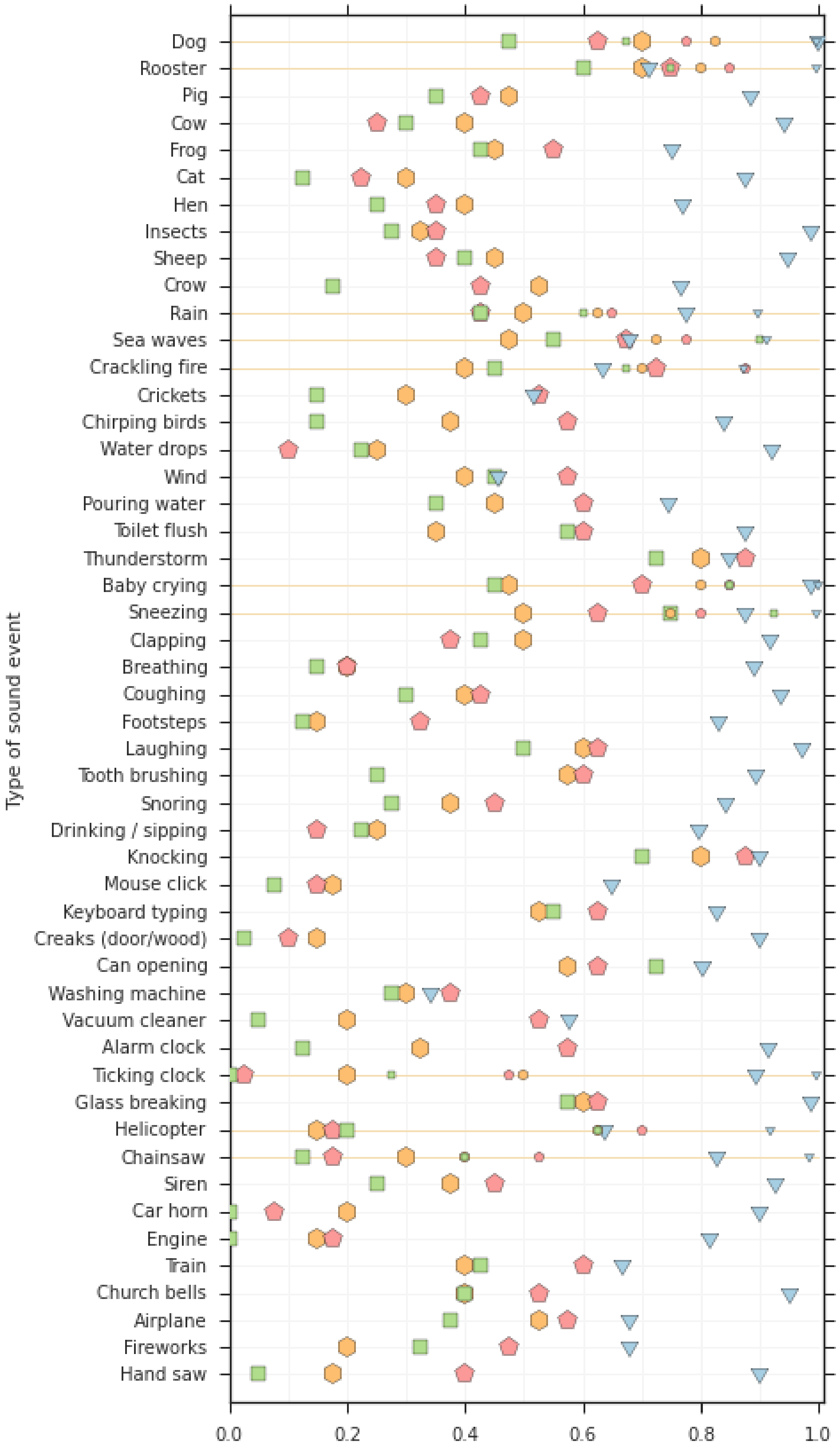


Fig 4.8 Performance

5 ANALYSIS, EVALUATION AND DISCUSSION

This chapter performs an in-depth analysis of the experiment and the results obtained from the design implementation as stated in the previous chapter. The key findings are summarised. The performance of sound texture statistics and Mel-spectrogram will be compared to evaluate the hypothesis. Several categories will be discussed individually. The chapter concludes by stating the strengths and limitations of the experiment.

5.1 Summary of Key Findings

The results prove that the SVM classifier has superior classification performance than the CNN model based on Mel-spectrogram, when used to classify environmental sound using sound texture statistical features.

5.2 Analysis

The research analyses the high-level performance of human listeners as benchmark reference at first. The average accuracy across all categories is 81.3%. The recall for each class varies between 34.1% and 100%. Based on the recall rates, the 50 categories are split into three difficulty levels:

	Recall	**Categories**
Easy level	90% < Recall <= 100%	Church bell; Clapping; Clock alarm; Coughing; Cow; Crying baby; Dog; Glass breaking; Insects flying; laughing; Sheep; Siren; Water drops
Average level	70% < Recall <=90%	Breathing; Brushing teeth; Can opening; Car horn; Cat; Chainsaw; chirping birds; Clock tick; Crow; Door - wood creaks; Door knock; Drinking – sipping; Engine; Footsteps; Frog;

		Hand saw; Hen; Keyboard typing; Pig; Pouring water; Rain; Rooster; Sneezing; Soring; Thunderstorm; Toilet flush
Difficult level	Recall < =70%	Airplane; Crackling fire; Crickets; Fireworks; Helicopter; Mouse click; Sea waves; Train; Vacuum cleaner; Washing machine; wind

Table 5.1 Difficulty levels

The unusual performance for the baseline classifier occurred at “Helicopter” and “Fire cracking”. Those two classes are ranked as difficult by the human listeners. However, there are not much distinction between the accuracies of two models. The question can be addressed through the Fig 5.1. It illustrated the relations between the mean values of $MFCC_1$ and $MFCC_2$. The purple circles represent the fire cracking sounds. The green stars denote the helicopter sounds. Most of those are spread on the fringe of the clusters. It would be one of the potential reasons that make the feature more recognizable.

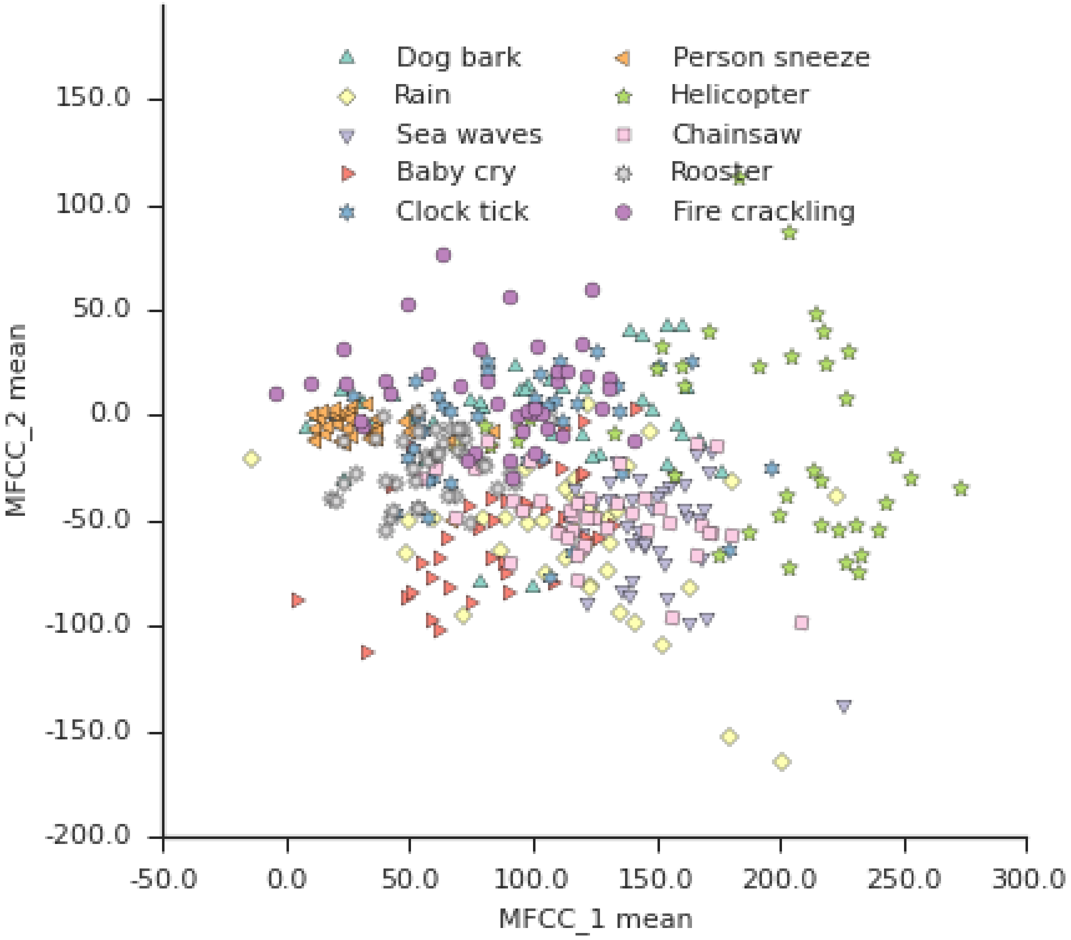
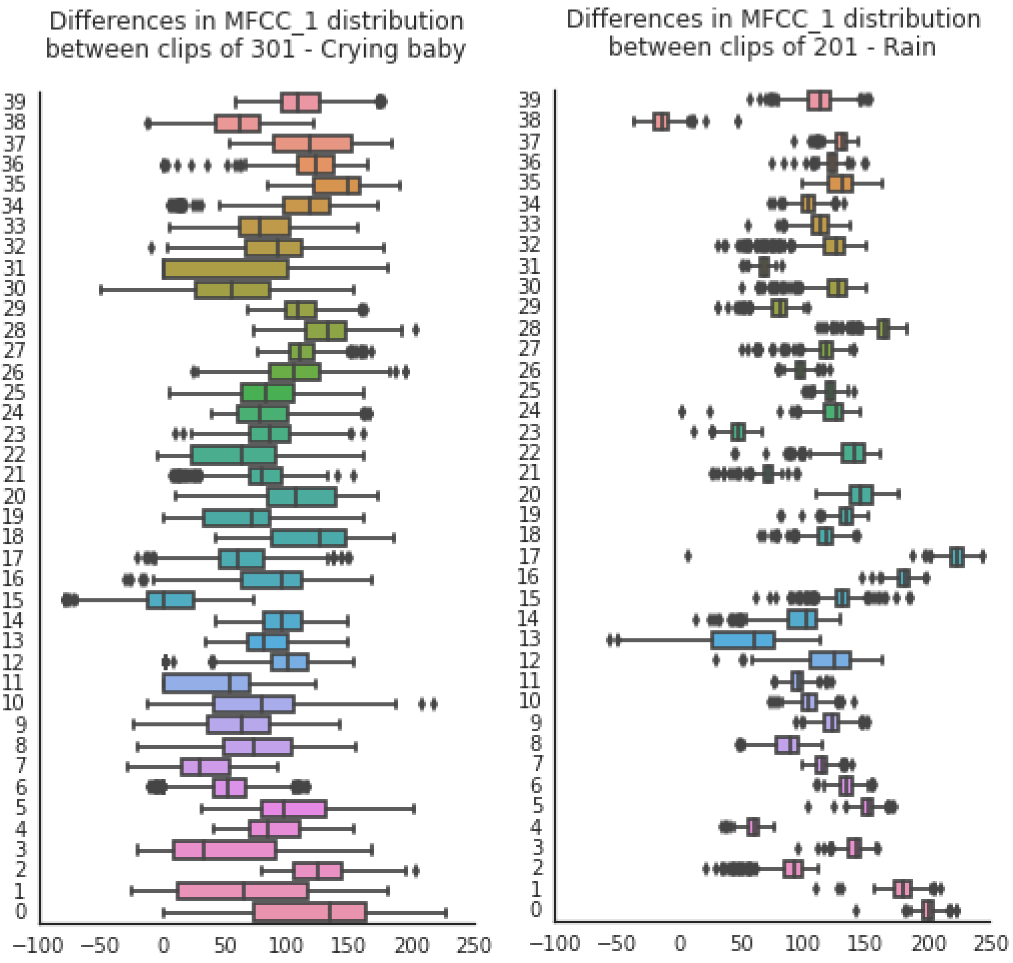


Fig 5.1 MFCC1 / MFCC2

Likewise, the statistical classifier also outperformed at main categories of “Natural soundscapes” and “Urban noise”. Most of the difficult level sub-classes reside in these two main categories. In order to find the reason behind the outstanding performance of statistics features in sound textures, it is requisite to explore the underlayer structures of environmental sounds. In particular to that, the analysis of MFCCs would be helpful to understand the characteristic of sounds. Through the $MFCC_1$ distribution figures of two classes, the repetitive sound textures of rain are concentrated around the mean value, while the baby crying sounds with more variable sound texture are dispersion around the mean value. This fact may indicate that highly homogeneous sound texture is a sensible feature for statistics.


Differences in MFCC_1 distribution between clips of 301 - Crying baby

Differences in MFCC_1 distribution between clips of 201 - Rain

Fig 5.2 MFCC1 distributions

As opposed to the previous two classifiers, the Mel-spectrogram classifier performed poorly on the difficult level classes. However, it outplayed at “Animal” and “Human non-speech sound” easy level categories. By observing the Mel-spectrogram listed in the Section 4.2, there are three Mel-spectrograms per class. The relatively difficult sounds such as rain and helicopter represent no clear boundary between colours and the power peaks are in pairs of spots, due to lack of harmonic. The colour edge patterns are distinctive shown in the easy level classes. All three thumbnails show that the shape of the power peak is presented as triangles for “dog bark” class. Similarly, the power peaks of “baby crying” are formed in several asymmetry lines.

5.3 Hypothesis Evaluation

The null hypothesis (H_0) of the current experiment is restated below:

> *A perceptually informed model on the ESC-50 dataset does not yield a different classification accuracy that is significantly greater than the SVM + MFCC baseline model, with a p value < 0.05.*

The alternative hypothesis (H_A) is restated below:

> *A perceptually informed model on the ESC-50 dataset yields a different classification accuracy that is significantly greater than the SVM + MFCC baseline model, with a p value < 0.05.*

In the Section 4.4, the results of each classifier created were listed. The results show that the statistical SVM classifier has superior performance compared with the baseline MFCC + SVM classifier whether for the overall results or the results of a specific class. Moreover, the differences in the performance are statistically significant with the p value of 0.005834, which is quite less than 0.05. In consequence, the alternative hypothesis H_A is accepted, while the null hypothesis H_0 can be rejected.

5.4 Strengths and Limitations

This research contributes to the limited literature on the ESC field. It is the only research to compare the sound texture statistical features with the Mel-spectrogram. The results revealed the strengths and drawbacks of each technique. The unique results were discussed individually. It provides fresh evidence for the potential of the perceptually informed data and biomimicry technology. Finally, it is one of the few papers that transform the sound recognition problem to image recognition with CNN architectural.

This study used ESC-50 database which has 2000 clips. One of the possible deficiencies of this dataset is the limited number of clips available per class. This is related to the high cost of manual annotation and extraction, and the decision to maintain strict balance between classes despite limited availability of recordings for more exotic types of sound events., the transfer learning was deployed to help the CNN model detect colours. It could produce a slight bias. A larger dataset might further improve the results of CNN model by expand the training set.

The size and shape of the analysis FFT window can be varied. A smaller (shorter) window will produce more accurate results in timing, at the expense of precision of frequency representation. A larger (longer) window will provide a more precise frequency representation, at the expense of precision in timing representation. The size of the FFT window is 2048 samples. It is the trade-off between precision and accurate.

6 CONCLUSION

This chapter performs a review of the current study. It reiterates the research question and all the different stages involved in answering it. The objectives of the research and all the important phases are quickly walked through. Additionally, the contributions of the research are also stated. The chapter concludes by highlighting the areas of further research.

6.1 Research Overview

Primarily, the research aimed to recognize the environmental sound using the perceptually informed data. The initial study was concentrated on understanding the current state of the art techniques in environmental sound recognition. Then those current research on ESR were evaluated by a critical review of the literature.

After chose the suitable database for the research, the next main area of focus in the research was to design the structure of experiments. Many decisions have been made during that phrase, such as the sound features for the baseline system. Three kinds of sounds features were extracted based on the perceptually informed data. Two kinds of machine learning algorithms cooperated with appropriate sound features. Finally, both these sound features can be proved effective for the experiment. The following depicts the stages followed as an aim to answer the research question:

	Stage	**Notes**
1	Performed extensive study on the existing literature of ESR.	Gaps have been identified in the research domain
2	A solution was designed to address the gaps in the ESR research.	The primary motive of the design was to investigate the perceptually informed

Conclusion**

		data.
3	The solution was implemented primarily based on the design and methodologies.	
4	Evaluate the results by comparing with multiple baseline systems	
	Future areas of research are identified to extend the field of study. Multiple recommendations on the study have also been made	

Table 6.1 Stages

6.2 Problem Definition

Based on the literature review, a gap in the current body of knowledge was exposed. The research work sought to empirically determine the strengths and limitations of perceptually informed data in the ESR area. The research question investigated in the study stated below:

> *"To what extent can a perceptually informed model significantly enhance the classification accuracy when compared to a Mel Frequency Cepstral Coefficients model based on Support Vector Machine?"*

Conclusion**

6.3 Future Work and Recommendations

Extending the investigation to a larger environmental sound database, such as Urbansound8k, Audioset. The commercial environmental dataset could also worth to explore. If the database contains recordings over 1000 per class, it will offer opportunities for the CNN Mel-spectrogram recognition and eliminate the bias of transfer learning.

Due to the goal of this study is to investigate perceptually informed data, the SVM and CNN models are respectively adopted from Sklearn and Tensorflow. There are rooms to improve the classification accuracy for each model, by tuning the arguments and optimising the structures.

Future efforts should also consider the impact of FFT window size. There are many studies proved that the correlation between sample rate and the window size has a remarkable impact on the sound recognition performance. How perceptually informed data would respond to various combination between window size and sample rate is worth to investigate.

BIBLIOGRAPHY

Alías, F., Socoró, J. C., & Sevillano, X. (2016). *A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds.* Barcelona: Ramon Llull University.

Arnaud, N. S., & Popat, K. (1998). *Analysis and synthesis of sound textures.* Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Athineos, M., & Ellis, D. P. (2003). Sound texture modelling with linear prediction in both time and frequency domains. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (pp. 648-516). New Paltz: IEEE. doi:10.1109/ASPAA.2003.1285816

Bellman, R. (2010). *Dynamic Programming - Princeton Landmarks in Mathematics.* Princeton: Princeton University Press.

Bountourakis, V., Vrysis, L., & Papanikolaou, G. (2015). Machine Learning Algorithms for Environmental Sound Recognition: Towards Soundscape Semantics. *AM '15 Proceedings of the Audio Mostly 2015 on Interaction With Sound* (pp. 54-61). New York: ACM.

Bregman, A. S. (1994). *Auditory Scene Analysis: The Perceptual Organizations of Sound.* Montreal: A Bradford Book.

C. Kwan, K. C. (2006). An Automated Acoustic System to Monitor and Classify Birds. *EURASIP Journal on Advances in Signal Processing, 352*(4), 1569-1665.

Cai, J., Ee, D., Pham, B., Ro, P., & Zhang, J. (2007). Sensor Network for the Monitoring of Ecosystem: Bird Species Recognition. *International Conference on Intelligent Sensors, Sensor Networks and Information* (pp. 293-298). Melbourne: IEEE. doi:10.1109/ISSNIP.2007.4496859

Bibliography**

Chachada, S., & Kuo, C. C. (2013). Environmental sound recognition: A survey. *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* (pp. 1-9). Kaohsiung: IEEE. doi:10.1109/APSIPA.2013.6694338

Chu, S., Narayanan, S., & Kuo, C. J. (2009). Environmental Sound Recognition With Time–Frequency Audio Features. *IEEE Transactions on Audio, Speech, and Language Processing, 17*(6), 1142-1158. doi:10.1109/TASL.2009.2017438

Cisco. (2015, 11 30). *Cisco Visual Networking Index: Forecast and Trends, 2015-2020.* Retrieved from Cisco: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf

Cowling, M., & Sitte, R. (2002). *Recognition of Environmental Sounds Using Speech Recognition Techniques* (Vol. 703). Boston: Springer.

Cowling, M., & Sitte, R. (2003). Comparison of Techniques for Environmental Sound Recognition. *Pattern Recognition Letters, 24*(15), 2895-2907.

Dubnov, S., Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., & Werman, M. (2002). “Synthesizing sound textures through wavelet. *IEEE Computer Graphics and Applications, 22*(4), 38-48. doi:10.1109/MCG.2002.101669

EI-Yaniv, R., & Dubnov, S. (1999). Granular Synthesis of Sound Textures using Statistical Learning. *Proceedings of the International Computer Music Conference* (pp. 86-90). Jerusalem: ICMC.

Ellis, D. P., Zeng, X., & McDermott, J. H. (2011). Classifying soundtracks with audio texture features. *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 5880-5883). Pragee: IEEE. doi:10.1109/ICASSP.2011.5947699

Eronen, A., Peltonen, V., Tuomi, J., Klapuri, A., Fagerlund, S., Sorsa, T., . . . Huopaniemi, J. (2006). Audio-based context recognition. *IEEE Transactions on Audio, Speech, and Language Processing, 14*(1), 321-329. doi:10.1109/TSA.2005.854103

Bibliography**

Gerhard, D. (2003). *Audio Signal Classification: History and Current Techniques.* Regina: Department of Computer Science, University of Regina.

Gerhard, D. (2006). Audio Signal Classification: History and Current Techniques. *Pattern Recognition,* 682-694.

Greenwood, D. (1961). Critical Bandwidth and the Frequency Coordinates of the Basilar Membrane. *The Journal of the Acoustical Society of America, 33*, 1344-1356.

Guastavino, C. (2006). The Ideal Urban Soundscape: Investigating the Sound Quality of French Cities. *Acta Acustica united with Acustica, 92*(6), 945-951.

Guastavino, C. (2007). Categorization of environmental sounds. *Canadian Journal of Experimental Psychology, 61*(1), 54-63.

Guastavino, C. (2007). Categorization of environmental sounds. *Canadian Journal of Experimental Psychology, 61*(1), 54-63.

Heeger, D. J., & Bergen, J. R. (1995). Pyramid-based texture analysis/synthesis. *SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (pp. 229-238). New York: ACM. doi:10.1145/218380.218446

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Aren Jansen, Moore, R. C., . . . Wilson, K. (2016). *CNN Architectures for Large-Scale Audio Classification.* New York: arXiv.

Huang, C., Yang, Y., Yang, D., & Chen, Y. (2009). Frog classification using machine learning techniques. *Expert Systems with Applications: An International Journal, 36*(2), 3737-3743. doi:10.1016/j.eswa.2008.02.059

Karbasi, M., Ahadi, S. M., & Bahmanian, M. (2011). Environmental sound classification using spectral dynamic features. (pp. 1-5). Singapore: IEEE. doi:10.1109/ICICS.2011.6173513

Lee, K., & Ellis, D. P. (2010). Audio-Based Semantic Concept Classification for Consumer Video. *IEEE Transactions on Audio, Speech, and Language Processing, 18*(6), 1406-1416. doi:10.1109/TASL.2009.2034776

Maddage, N., Xu, C., Kankanhalli, M., & Shao, X. (2004). Content-based music structure analysis with applications to music semantics understanding. *In Proceedings of the 12th ACM International Conference on Multimedia, October 10 - 16*, 112-119.

McDermott, J. H., & Simoncelli, P. E. (2011). Sound Texture Perceprion via Statistics of the Auditory Periphery: Evidence from Sound Synthesis. *Neuron, 71*(5), 927-940. doi:10.1016/j.neuron.2011.06.032

McDermott, J. H., Oxenham, A. J., & Simoncelli, E. P. (2009). Sound Texture Synthesis via Filter Statistics. *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (pp. 297-300). New Paltz: IEEE.

Mitrović, D., Zeppelzauer, M., & Breiteneder, C. (2010). Features for Content-Based Audio Retrieval. *Advances in Computers*, 71-150.

Moore, B., Peters, R., & Glasberg, B. (1990). Auditory filter shapes at low center frequencies. *The Journal of the Acoustical Society of America, 88*, 132-140.

Nelken, I., Rotman, Y., & Yosef, O. B. (1999). Responses of auditory-cortex neurons to structural features of natural sounds. *Nature*, 154-157. Retrieved from https://doi.org/10.1038/16456

Neuhoff, J. G. (2004). *Ecological Psychoacoustics.* Amsterdam: Brill.

Pearce, D. (2003, 5 23). *Details of 'RES/STQ-00044a' Work Item*. Retrieved 11 17, 2008, from European Telecommunications Standards Institute: https://portal.etsi.org/webapp/workprogram/Report_WorkItem.asp?wki_id=18820

Bibliography**

Piczak, K. J. (2015). Environmental Sound Classification with Convolutional Neural Networks. *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING* (pp. 17-20). Boston: IEEE.

PortillaEero, J., & Simoncelli, P. (2000, 10). A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. *International Journal of Computer Vision, 40*(1), 49-70.

Rakotomamonjy, A. (2017). Supervised Representation Learning for Audio Scene Classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25*(6), 1253-1265. doi:10.1109/TASLP.2017.2690561

Saint-Arnaud, N., & Popat, K. (1995). Analysis and Synthesis of Sound Textures. In D. Wang, & G. J. Brown, *Computational auditory scene analysis: Principles, Algorithms, and Applications* (pp. 293-308). Hillsdale: Wiley-IEEE Press.

Schafer, R. M. (1993). *The Soundscape: Our Sonic Environment and the Tuning of the World.* Merrimac: Destiny Books.

Schubert, E. D. (1975). The role of auditory perception in language processing. *Reading, perception and language: Papers from the World Congress on Dyslexia*, 62-75.

Sharan, R. V., & Moir, T. J. (2016). An overview of applications and advancements in automatic sound. *Neurocomputing, 200*, 22-34.

Temko, A., & Nadeu, C. (2006). Classification of acoustic events using SVM-based clustering schemes. *Pattern Recognition, 39*(4), 682-694.

Temko, A., & Nadeu, C. (2009). Acoustic Event Detection in Meeting-Room Environments. *Pattern Recognition Letters, 30*(14), 1281-1288.

Trudeau, C., & Guastavino, C. (2018). Classifying Soundscapes Using a Multifaceted Taxonomy. *EuroNoise 2018 Crete* (pp. 2487-2492). Hersonissos: European Acoustic Association.

Voss, R. F., & Clarke, J. (1975). `1/fnoise' in music and speech. *Nature, 258*(5533), 317-318. doi:10.1038/258317a0

Yella, S., Gupta, N. K., & Dougherty, M. S. (2007). Comparison of pattern recognition techniques for the classification of impact acoustic emissions. *Transportation Research Part C: Emerging Technologies, 15*(6), 345-360.

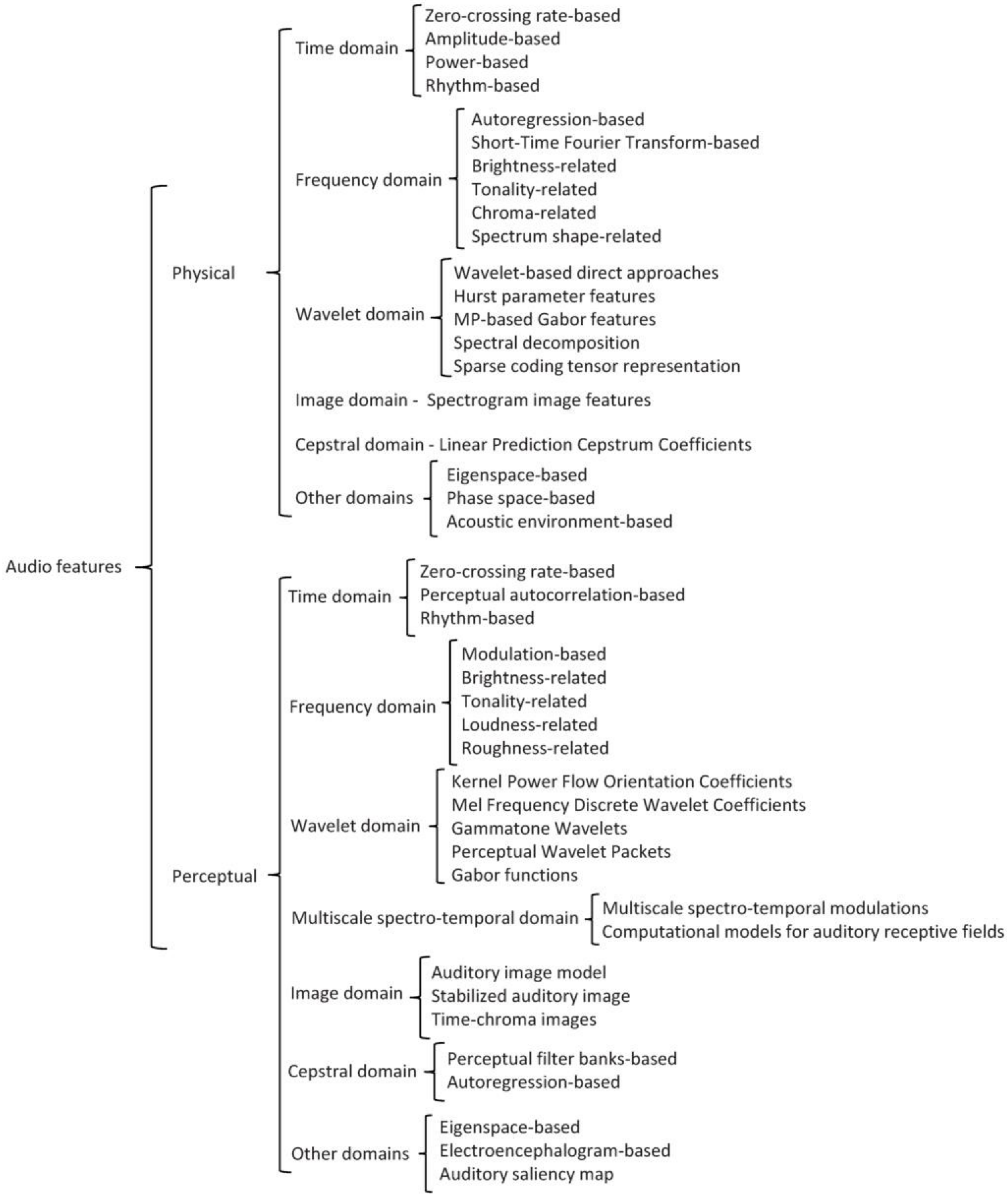
Yu, G., & Slotine, J.-J. (2009). Audio Classification from Time-frequency Texture. *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 1677-1680). Taipei: IEEE. doi:10.1109/ICASSP.2009.4959924

Zwicker, E. (1961). Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen). *The Journal of the Acoustical Society of America, 33*, 248.

Appendix A**

APPENDIX A

Fig A.1 Taxonomy of sound features



APPENDIX B

Table B.1: Categories of ESC-50

Animals	**Natural soundscapes & water sounds**	**Human, non-speech sounds**	**Interior/domestic sounds**	**Exterior/urban noises**
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

APPENDIX C

Experiment Implementation by Python

```
import numpy as np
import pydub
import librosa
import os
import IPython
import pandas as pd
import matplotlib as plt


class Clip:
    """A single 5-sec long recording."""

    RATE = 44100  # All recordings in ESC are 44.1 kHz
    FRAME = 512  # Frame size in samples

    class Audio:
        """The actual audio data of the clip.

            Uses a context manager to load/unload the raw audio
data. This way clips
            can be processed sequentially with reasonable memory
usage.
        """

        def __init__(self, path):
            self.path = path

        def __enter__(self):
            #For fixing the runtime warning: Couldn't find ffmpeg
or avconv
            pydub.AudioSegment.converter = "C:\\Program Files
(x86)\\ffmpeg\\bin\\ffmpeg.exe"
            # Actual recordings are sometimes not frame accurate,
so we trim/overlay to exactly 5 seconds
            self.data = pydub.AudioSegment.silent(duration=5000)
            self.data =
self.data.overlay(pydub.AudioSegment.from_file(self.path)[0:5000])
            self.raw = (np.fromstring(self.data._data,
dtype="int16") + 0.5) / (0x7FFF + 0.5)  # convert to float
            return (self)

        def __exit__(self, exception_type, exception_value,
traceback):
            if exception_type is not None:
                print exception_type, exception_value, traceback
            del self.data
            del self.raw

    def __init__(self, filename, category):
        self.filename = os.path.basename(filename)
        self.path = os.path.abspath(filename)
        self.directory = os.path.dirname(self.path)
        self.category = category

        # print ("Clip name is " + self.filename + "\n" +
        #        "Clip path is " + self.path + "\n" +
        #        "Clip directory is " + self.directory + "\n" +
        #        "Clip category is " + self.category) + "\n"

        self.audio = Clip.Audio(self.path)

        with self.audio as audio:
            self._compute_mfcc(audio)

    def _compute_mfcc(self, audio):
        # MFCC computation with default settings (2048 FFT window
length, 512 hop length, 128 bands)
        self.melspectrogram =
librosa.feature.melspectrogram(audio.raw, sr=Clip.RATE,
hop_length=Clip.FRAME)
        self.logamplitude =
librosa.amplitude_to_db(self.melspectrogram)
        self.mfcc = librosa.feature.mfcc(S=self.logamplitude,
n_mfcc=13).transpose()
        self.mfcc_delta = librosa.feature.delta(self.mfcc)

    @classmethod
    def _get_frame(cls, audio, index):
        if index < 0:
            return None
        return audio.raw[(index * Clip.FRAME):(index + 1) *
Clip.FRAME]

    def __repr__(self):
        return '<{0}\\{1}>'.format(self.category, self.filename)


def load_dataset(name):
    """Load all dataset recordings into a list from a csv file"""

    clips = []

    df = pd.read_csv('meta\\esc50.csv', skipinitialspace=True,
usecols=['filename', 'category'])
    # subclasses = df['category'].drop_duplicates().tolist()

    for clip in df.values:
        # print("Loading " + clip[0] + " in \"" + clip[1] + "\"
category \n")
        clips.append(Clip(name + '\\' + clip[0], clip[1]))

    IPython.display.clear_output(clips)
    print('\n All {0} recordings loaded. \n'.format(name))

    return clips
```

```
def create_set(clips):
    cases = pd.DataFrame()

    for i in range(0, len(clips)):
            case = pd.DataFrame([clips[i].filename],
columns=['filename'])
            case['category_name'] = clips[i].category

            mfcc_mean = pd.DataFrame(np.mean(clips[i].mfcc[:, :],
axis=0)[1:]).T
            mfcc_mean.columns = list('MFCC_{} mean'.format(i) for
i in range(np.shape(clips[i].mfcc)[1]))[1:]
            mfcc_std = pd.DataFrame(np.std(clips[i].mfcc[:, :],
axis=0)[1:]).T
            mfcc_std.columns = list('MFCC_{} std dev'.format(i)
for i in range(np.shape(clips[i].mfcc)[1]))[1:]
            case = case.join(mfcc_mean)
            case = case.join(mfcc_std)
            cases = cases.append(case)
    print cases
    return cases


def plot_single_clip(clip):
    col_names = list('MFCC_{}'.format(i) for i in
range(np.shape(clip.mfcc)[1]))
    MFCC = pd.DataFrame(clip.mfcc[:, :], columns=col_names)

    f = plt.figure(figsize=(10, 6))
    ax = f.add_axes([0.0, 0.0, 1.0, 1.0])
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    ax.set_frame_on(False)

    ax_mfcc = add_subplot_axes(ax, [0.0, 0.0, 1.0, 0.75])
    ax_mfcc.set_xlim(-400, 400)

    plt.title('Feature distribution across frames of a single clip
({0} : {1})'.format(clip.category, clip.filename),
              y=1.5)
    sb.boxplot(MFCC, vert=False,
order=list(reversed(MFCC.columns)), ax=ax_mfcc)



import numpy as np
from numpy import transpose as tp
import scipy.signal as sig
import scipy.stats as scistat
import filterbanks as fb


class SoundTexture(object):
    """
```

Appendix C**

```
    Based on Josh McDermott's Matlab toolbox:

http://mcdermottlab.mit.edu/Sound_Texture_Synthesis_Toolbox_v1.7.z
ip

    y = audio file
    fs = sample rate
    """
    def __init__(self, y, fs):
        self.y = y
        self.fs = fs
        # default settings:
        self.desired_rms = .01
        self.audio_sr = 20000
        self.n_audio_channels = 30
        self.low_audio_f = 20
        self.hi_audio_f = 10000
        self.use_more_audio_filters = 0
        self.lin_or_log_filters = 1
        self.env_sr = 400
        self.n_mod_channels = 20
        self.low_mod_f = 0.5
        self.hi_mod_f = 200
        self.use_more_mod_filters = 0
        self.mod_filt_Q_value = 2
        self.use_zp = 0
        self.low_mod_f_c12 = 1
        self.compression_option = 1
        self.comp_exponent = .3
        self.log_constant = 10 ** -12
        self.match_env_hist = 0
        self.match_sub_hist = 0
        self.n_hist_bins = 128
        self.manual_mean_var_adjustment = 0
        self.max_orig_dur_s = 7
        self.desired_synth_dur_s = 5
        self.measurement_windowing = 2
        self.imposition_windowing = 1
        self.win_steepness = .5
        self.imposition_method = 1
        self.sub_imposition_order = 1
        self.env_ac_intervals_smp = np.array([1, 2, 3, 4, 5, 6, 7,
9, 11, 14, 18, 22, 28, 36, 45, 57, 73, 92, 116, 148, 187, 237,
301])  # in samples
        self.sub_ac_undo_win = 1
        self.sub_ac_win_choice = 2
        self.num_sub_ac_period = 5
        # allocate memory:
        self.mod_c2 = []
        self.mod_c1 = []
        self.env_c = []
        self.subband_ac = []
        self.mod_power_center_freqs = []
        self.mod_c2_center_freqs = []
        self.mod_c1_center_freqs = []
        self.audio_cutoffs_hz = []
        self.subband_mean = np.zeros(self.n_audio_channels + 2)
        self.subband_var = np.zeros(self.n_audio_channels + 2)
        self.subband_skew = np.zeros(self.n_audio_channels + 2)
        self.subband_kurt = np.zeros(self.n_audio_channels + 2)
        self.env_mean = np.zeros(self.n_audio_channels + 2)
        self.env_var = np.zeros(self.n_audio_channels + 2)
        self.env_skew = np.zeros(self.n_audio_channels + 2)
        self.env_kurt = np.zeros(self.n_audio_channels + 2)
        self.subband_hist = np.zeros([self.n_audio_channels + 2 +
1, self.n_hist_bins])
        self.subband_bins = np.zeros([self.n_audio_channels + 2 +
1, self.n_hist_bins])
        self.env_hist = np.zeros([self.n_audio_channels + 2,
self.n_hist_bins])
        self.env_bins = np.zeros([self.n_audio_channels + 2,
self.n_hist_bins])
        self.env_ac = np.zeros([self.n_audio_channels + 2,
self.env_ac_intervals_smp.shape[0]])
        self.mod_power = np.zeros([self.n_audio_channels + 2,
self.n_mod_channels])
        self.subband_ac_power = np.zeros(self.n_audio_channels + 2)
        # calculate stats:
        self.orig_sound, self.ds_factor = self.format_orig_sound()
        self.measurement_win =
self.set_measurement_window(self.orig_sound.shape[0],
self.measurement_windowing)
        self.measure_texture_stats(self.orig_sound,
self.measurement_win)

    def format_orig_sound(self):
        orig_sound = self.y
        if orig_sound.ndim == 2:
            orig_sound = (orig_sound[:, 0] + orig_sound[:, 1]) / 2
# if stereo convert to mono
        if self.fs != self.audio_sr:
            orig_sound = sig.resample(orig_sound,
int(orig_sound.shape[0] * self.audio_sr / self.fs))
        if np.remainder(orig_sound.shape[0], 2) == 1:
            orig_sound = np.concatenate([orig_sound,
np.array([0])])
        ds_factor = self.audio_sr / self.env_sr
        new_l = int(np.floor((orig_sound.shape[0] / ds_factor / 2)
* ds_factor * 2))
        orig_sound = orig_sound[:new_l]
        orig_sound = orig_sound /
np.sqrt(np.mean(np.square(orig_sound))) * self.desired_rms
        return orig_sound, ds_factor

    def set_measurement_window(self, sound_length,
windowing_option):
        if windowing_option == 1:
            measurement_win = np.ones([int(sound_length /
self.ds_factor), 1])
        elif windowing_option == 2:
            temp =
self.make_windows_rcos_flat_no_ends(int(sound_length /
self.ds_factor), int(np.round(sound_length / self.audio_sr)),
self.win_steepness)
            measurement_win = np.sum(temp, 1)
        else:
            raise Exception('measurement_win must be 1 or 2')
        return measurement_win

    @staticmethod
    def make_windows_rcos_flat_no_ends(signal_length_smp, num_secs,
ramp_prop):
        num_secs = num_secs + 2
        if ramp_prop == 0.5:
            ramp_length_smp = int(np.floor(signal_length_smp /
(num_secs - 1)))
            flat_length_smp = 0
        elif ramp_prop < 0.5:
            flat_length = signal_length_smp / (num_secs * (1 -
ramp_prop) / (1 - 2 * ramp_prop) - ramp_prop / (1 - 2 * ramp_prop))
            ramp_length_smp = int(np.floor(flat_length * ramp_prop
/ (1 - 2 * ramp_prop)))
            flat_length_smp = int(np.floor(flat_length))
        else:
            raise Exception('ramp_prop must be less than .5')
        windows = np.zeros([signal_length_smp, num_secs])
        windows[:flat_length_smp, 0] = 2
        windows[flat_length_smp: flat_length_smp + ramp_length_smp,
0] = np.cos(np.linspace(1, ramp_length_smp, num=ramp_length_smp) /
ramp_length_smp * np.pi) + 1
        start_pt = flat_length_smp
        for n in range(0, num_secs - 2):
            windows[start_pt:start_pt+ramp_length_smp, n+1] =
np.cos(np.linspace(-ramp_length_smp+1, 0, num=ramp_length_smp) /
ramp_length_smp * np.pi) + 1

windows[start_pt+ramp_length_smp:start_pt+ramp_length_smp+flat_len
gth_smp, n+1] = 2

windows[start_pt+ramp_length_smp+flat_length_smp:start_pt+2*ramp_l
ength_smp+flat_length_smp, n+1] = np.cos(np.linspace(1,
ramp_length_smp, num=ramp_length_smp) / ramp_length_smp * np.pi) +
1
            start_pt = start_pt + flat_length_smp +
ramp_length_smp
        windows[start_pt:start_pt+ramp_length_smp, num_secs-1] =
np.cos(np.linspace(-ramp_length_smp + 1, 0, num=ramp_length_smp) /
ramp_length_smp * np.pi) + 1
        windows[start_pt + ramp_length_smp:signal_length_smp,
num_secs-1] = 2
        windows = windows[:, 1:-1]
        windows = windows / 2
        return windows

    @staticmethod
    def stat_central_moment_win(x, n, win, x_mean=-99):
        win = win / np.sum(win)
        if x_mean == -99:
            x_mean = np.sum(win * x)
        if n == 1:
            m = x_mean
        elif n == 2:
            m = np.sum(win * ((x - x_mean) ** 2))
            m = np.sqrt(m) / x_mean
        elif n == 3:
            m2 = np.sum(win * ((x - x_mean) ** 2))
            m = np.sum(win * ((x - x_mean) ** 3)) / (m2 ** (3.0 /
2.0))
        elif n == 4:
            m2 = np.sum(win * ((x - x_mean) ** 2))
            m = np.sum(win * ((x - x_mean) ** 4)) / (m2 ** 2)
        else:
            raise Exception('input value of n not recognised')
        return m

    @staticmethod
    def shift_s(s, num_samples):
        if num_samples == 0:
            new_s = s
        elif num_samples < 0:
            new_s = np.concatenate([s[-num_samples:], np.zeros(-
num_samples)])
        else:
            new_s = np.concatenate([np.zeros(num_samples), s[:-
num_samples]])
        return new_s

    def stat_env_ac_scaled_win(self, f_env, sample_spacing, use_zp,
win):
        if use_zp != 0:
            raise Exception('zero padding not implemented')
        win = win / np.sum(win)
        ac_values = np.zeros(sample_spacing.shape[0])
        for p in range(0, sample_spacing.shape[0]):
            num_samp = sample_spacing[p]
            meanf_env = np.mean(f_env[:, p])
            mf_env = f_env[:, p] - meanf_env
            env_var = np.mean(mf_env ** 2)
            ac_values[p] = np.sum(win * (self.shift_s(mf_env, -
num_samp) * self.shift_s(mf_env, num_samp))) / env_var
        return ac_values

    @staticmethod
    def stat_var_win(s, win):
        win = win / np.sum(win)
        w_var = np.sum(win * (s - np.sum(win * s)) ** 2)
        return w_var

    def stat_mod_power_win(self, s, mod_subbands, use_zp, win):
        if use_zp != 0:
            raise Exception('zero padding not implemented')
        win = win / np.sum(win)
        s_var = self.stat_var_win(s, win)
        mp = np.sum(np.dot(win[:, None], np.ones([1,
mod_subbands.shape[1]])) * (mod_subbands ** 2), 0) / s_var
        return mp

    @staticmethod
    def stat_mod_c2_win(subbands, use_zp, win):
        if use_zp != 0:
            raise Exception('zero padding not implemented')
        win = win / np.sum(win)
        analytic_subbands =
np.transpose(sig.hilbert(np.transpose(subbands)))
        n = analytic_subbands.shape[1]
        c2 = np.zeros([n-1, 2])
        for k in range(0, n-1):
            c = (analytic_subbands[:, k] ** 2) /
np.abs(analytic_subbands[:, k])
            sig_cw = np.sqrt(np.sum(win * (np.real(c) ** 2)))
            sig_fw = np.sqrt(np.sum(win *
(np.real(analytic_subbands[:, k+1]) ** 2)))
            c2[k, 0] = np.sum(win * np.real(c) *
np.real(analytic_subbands[:, k+1])) / (sig_cw * sig_fw)
            c2[k, 1] = np.sum(win * np.real(c) *
np.imag(analytic_subbands[:, k + 1])) / (sig_cw * sig_fw)
        return c2

    @staticmethod
    def stat_corr_filt_win_full(f_envs, use_zp, win):
        if use_zp != 0:
            raise Exception('zero padding not implemented')
        win = win / np.sum(win)
        cbc_value = np.zeros([f_envs.shape[1], f_envs.shape[1]])
        meanf_envs = np.mean(f_envs, 0)[None, :]
        mf_envs = f_envs - np.dot(np.ones([f_envs.shape[0], 1]),
meanf_envs)
        env_stds = np.sqrt(np.mean(mf_envs ** 2, 0))[None, :]
        cbc_value[:, :] = np.dot(np.transpose((np.dot(win[:, None],
np.ones([1, f_envs.shape[1]]))) * mf_envs), mf_envs) /
np.dot(np.transpose(env_stds), env_stds)
        return cbc_value

    @staticmethod
    def autocorr_mult(x):
        xf = np.transpose(np.fft.fft(np.transpose(x)))
        xf2 = np.abs(xf) ** 2
        cx2  = np.transpose(np.real(np.fft.ifft(np.transpose(xf2))))
        cx = np.zeros_like(cx2)
        for j in range(0, cx2.shape[1]):
            cx[:, j] = np.fft.fftshift(cx2[:, j])
        return cx

    def autocorr_mult_zp(self, s, win_choice, undo_win):
        n = s.shape[1] - 2
        s_l = s.shape[0]
        wt = np.linspace(1, s_l, num=s_l) / s_l
        if win_choice == 1:  # hanning
            w = 0.5 - 0.5 * np.cos(2 * np.pi * wt)
        elif win_choice == 2:  # rect
```

```
            w = np.ones_like(wt)
        elif win_choice == 3:  # hamming
            w = 0.54 - 0.46 * np.cos(2 * np.pi * wt)
        elif win_choice == 4:  # hamming
            w = 0.6 - 0.4 * np.cos(2 * np.pi * wt)
        elif win_choice == 5:  # welch
            w = np.sin(np.pi * wt)
        else:
            raise Exception('window type not recognised')
        s_w = s * np.dot(np.transpose(w[None, :]), np.ones([1,
n+2]))
        s_wp = np.vstack([np.zeros([int(s_l / 2), int(n + 2)]),
s_w, np.zeros([int(s_l / 2), int(n + 2)])])
        w_p = np.vstack([np.zeros([int(w.shape[0] / 2), 1]), w[:,
None], np.zeros([int(w.shape[0] / 2), 1])])
        ac = self.autocorr_mult(s_wp)
        if undo_win:
            w_ac = self.autocorr_mult(w_p)
            ac = ac / np.dot(w_ac, np.ones([1, int(n + 2)]))
        ac = ac[int(s_l / 2):int(3 * s_l / 2), :]
        return ac

    def measure_texture_stats(self, sample_sound, measurement_win):
        # Construct the filter banks
        if self.use_more_audio_filters == 0:
            if self.lin_or_log_filters == 1 or
self.lin_or_log_filters == 2:
                filt_bank =
fb.EqualRectangularBandwidth(self.orig_sound.shape[0],
self.audio_sr, self.n_audio_channels, self.low_audio_f,
self.hi_audio_f)
            elif self.lin_or_log_filters == 3 or
self.lin_or_log_filters == 4:
                filt_bank = fb.Linear(self.orig_sound.shape[0],
self.audio_sr, self.n_audio_channels, self.low_audio_f,
self.hi_audio_f)
            else:
                raise Exception('filter type not recognised')
        else:
            raise Exception('double and quadruple audio filters
not implemented')
        self.audio_cutoffs_hz = filt_bank.cutoffs
        filt_bank.generate_subbands(sample_sound)
        subbands = filt_bank.subbands  # [:, 1:-1]
        subband_envs = tp(np.absolute(sig.hilbert(tp(subbands))))
        if self.compression_option == 1:
            subband_envs = subband_envs ** self.comp_exponent
        elif self.compression_option == 2:
            subband_envs = np.log10(subband_envs +
self.log_constant)
        subband_envs = sig.resample(subband_envs,
int(subband_envs.shape[0] / self.ds_factor))
        subband_envs[subband_envs < 0] = 0
        if self.use_zp == 1:
            mod_filt_length = subband_envs.shape[0] * 2
        elif self.use_zp == 0:
            mod_filt_length = subband_envs.shape[0]
        else:
            raise Exception('use_zp input not recognised')
        if self.lin_or_log_filters == 1 or self.lin_or_log_filters
== 3:
            const_q_bank = fb.ConstQCos(mod_filt_length,
self.env_sr, self.n_mod_channels, self.low_mod_f, self.hi_mod_f,
self.mod_filt_Q_value)
        elif self.lin_or_log_filters == 2 or
self.lin_or_log_filters == 4:
            const_q_bank = fb.LinConstQCos(mod_filt_length,
self.env_sr, self.n_mod_channels, self.low_mod_f, self.hi_mod_f,
self.mod_filt_Q_value)
        else:
            raise Exception('lin_or_log_filters input not
recognised')
        env_ac_bank = fb.EnvAutocorrelation(mod_filt_length,
self.env_sr, self.n_mod_channels, self.low_mod_f, self.hi_mod_f,
self.mod_filt_Q_value, self.env_ac_intervals_smp)
        octave_bank = fb.OctaveCos(mod_filt_length, self.env_sr,
self.n_mod_channels, self.low_mod_f_c12, self.hi_mod_f)
        if self.lin_or_log_filters == 1 or self.lin_or_log_filters
== 3:
            mod_c1_bank = octave_bank
            c1_ind = 1
        elif self.lin_or_log_filters == 2 or
self.lin_or_log_filters == 4:
            mod_c1_bank = fb.LinearOctaveCos(mod_filt_length,
self.env_sr, self.n_mod_channels, self.low_mod_f_c12,
self.hi_mod_f)
            c1_ind = 0
        else:
            raise Exception('filter type not recognised')
        # Now calculate the stats
        self.subband_mean = np.mean(subbands, 0)
        self.subband_var = np.var(subbands, 0)
        self.mod_c2 = np.zeros([self.n_audio_channels + 2,
octave_bank.N - 1, 2])
        self.mod_c1 = np.zeros([subband_envs.shape[1],
subband_envs.shape[1], mod_c1_bank.N - c1_ind])
        for j in range(0, self.n_audio_channels + 2):
            self.subband_skew[j] = scistat.skew(subbands[:, j])
            self.subband_kurt[j] = scistat.kurtosis(subbands[:, j],
fisher=False)
            self.env_mean[j] =
self.stat_central_moment_win(subband_envs[:, j], 1,
measurement_win)
            self.env_var[j] =
self.stat_central_moment_win(subband_envs[:, j], 2,
measurement_win, self.env_mean[j])
            self.env_skew[j] =
self.stat_central_moment_win(subband_envs[:, j], 3,
measurement_win, self.env_mean[j])
            self.env_kurt[j] =
self.stat_central_moment_win(subband_envs[:, j], 4,
measurement_win, self.env_mean[j])
```

```
            temp, bins = np.histogram(subbands[:, j],
self.n_hist_bins)
            temp = temp.astype(float, copy=False)
            bins = bins.astype(float, copy=False)
            bins = (bins[:-1] + bins[1:]) / 2  # get bin centres
            self.subband_hist[j, :self.n_hist_bins] = temp /
np.sum(temp)
            self.subband_bins[j, :self.n_hist_bins] = bins
            temp, bins = np.histogram(subband_envs[:, j],
self.n_hist_bins)
            temp = temp.astype(float, copy=False)
            bins = bins.astype(float, copy=False)
            bins = (bins[:-1] + bins[1:]) / 2  # get bin centres
            self.env_hist[j, :self.n_hist_bins] = temp /
np.sum(temp)
            self.env_bins[j, :self.n_hist_bins] = bins
            env_ac_bank.generate_subbands(subband_envs[:, j])
            f_env = env_ac_bank.subbands
            self.env_ac[j, :] = self.stat_env_ac_scaled_win(f_env,
self.env_ac_intervals_smp, self.use_zp, measurement_win)
            const_q_bank.generate_subbands(subband_envs[:, j])
            mod_subbands = const_q_bank.subbands
            self.mod_power[j, :] =
self.stat_mod_power_win(subband_envs[:, j], mod_subbands,
self.use_zp, measurement_win)
            self.mod_power_center_freqs =
const_q_bank.center_freqs
            octave_bank.generate_subbands(subband_envs[:, j])
            mod_c2_subbands = octave_bank.subbands
            self.mod_c2[j, :, :] =
self.stat_mod_c2_win(mod_c2_subbands, self.use_zp, measurement_win)
            self.mod_c2_center_freqs = octave_bank.center_freqs[:-
1]
        # compute subband envelope, modulation band correlations
        self.env_c = self.stat_corr_filt_win_full(subband_envs,
self.use_zp, measurement_win)
        f_envs = np.zeros_like(subband_envs)
        for k in range(0, mod_c1_bank.N - c1_ind):
            for i in range(0, subband_envs.shape[1]):
                mod_c1_bank.generate_subbands(subband_envs[:, i])
                f_envs[:, i] = mod_c1_bank.subbands[:, k + c1_ind]
# exclude first
            self.mod_c1[:, :, k] =
self.stat_corr_filt_win_full(f_envs, self.use_zp, measurement_win)
        self.mod_c1_center_freqs = mod_c1_bank.center_freqs
        # subband autocorrelation
        sub_ac_n_smp = np.round(self.num_sub_ac_period /
self.audio_cutoffs_hz * self.audio_sr)
        sub_ac_n_smp[sub_ac_n_smp > self.num_sub_ac_period / 20.0
* self.audio_sr] = self.num_sub_ac_period / 20.0 * self.audio_sr
        temp = self.autocorr_mult_zp(subbands,
self.sub_ac_win_choice, self.sub_ac_undo_win)
        l2 = subbands.shape[0]
        c2 = l2 / 2
        for k in range(0, self.n_audio_channels + 2):
            self.subband_ac.append(temp[int(c2 -
sub_ac_n_smp[k]):int(c2 + sub_ac_n_smp[k] + 1), k])
            self.subband_ac_power[k] = np.sum(self.subband_ac[k]
** 2)  # used in SNR calculation
        amp_hist, amp_bins = np.histogram(sample_sound,
self.n_hist_bins)
        amp_bins = (amp_bins[:-1] + amp_bins[1:]) / 2  # get bin
centres
        self.subband_hist[self.n_audio_channels +
2, :self.n_hist_bins] = amp_hist
        self.subband_bins[self.n_audio_channels +
2, :self.n_hist_bins] = amp_bins
```