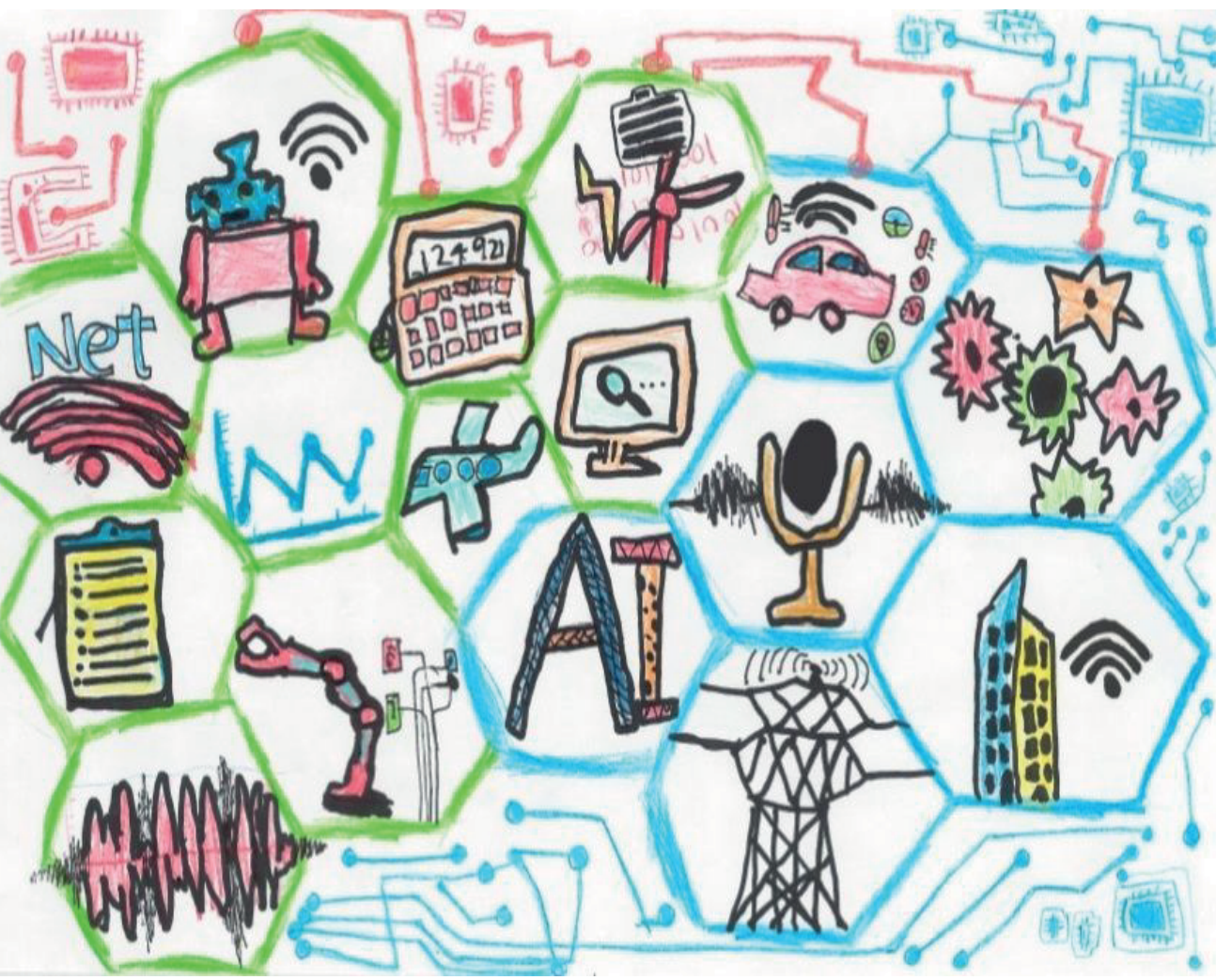
SEEING WITH THE SOUND

Sound-based human-context recognition using machine learning

Wei Wang
PhD Thesis

October 2021



UNIVERSITY OF TWENTE.

SEEING WITH THE SOUND

SOUND-BASED HUMAN-CONTEXT RECOGNITION USING MACHINE LEARNING

Wei Wang

SEEING WITH THE SOUND

SOUND-BASED HUMAN-CONTEXT RECOGNITION USING MACHINE LEARNING

DISSERTATION

to obtain
the degree of doctor at the Universiteit Twente,
on the authority of the rector magnificus,
prof. dr. ir. A. Veldkamp,
on account of the decision of the Doctorate Board
to be publicly defended
on Wednesday 13 October 2021 at 12.45 hours

by

Wei Wang

born on the 14th of September, 1985
in Hubei, China

This dissertation has been approved by:

Supervisor
prof.dr.ir. P.J.M. Havinga

Graduation Committee:

Chair / secretary:	prof.dr. J.N. Kok
Supervisor:	prof.dr.ir. P.J.M. Havinga
Committee Members:	prof.dr. D.K.J. Heylen dr.ir. L.J. Spreeuwers prof.dr. Y. Bapat prof.dr. D. Das

ISBN 978-90-365-5293-6
DOI: 10.3990/1.9789036552936
https://doi.org/10.3990/1.9789036552936

"You're only as good as your weakest link in the ecosystem of sound, of audio."

Jimmy Iovine

UNIVERSITY OF TWENTE

Abstract

PS Group
EWI

Doctor of Philosophy

by Wei Wang

This thesis presents a research on how to use sound to detect the contexts of human, i.e. what are people doing or how many people are there. Many modern applications such as smart buildings, healthcare or retails heavily depend on these human-context information to make our lives better. Take smart buildings as an example, energy can be saved by turning lights off when no one is around, comfort can be improved when room temperature is automatically adjusted according to the activities of people. Numerous sensors are substantially used in this domain, such as simple PIR sensors that detect human presence, camera systems that monitor and recognize very detailed human actions, wearable devices that track user movement and identity, etc. None of these sensors and technologies is perfect, and each has pros and cons in different aspects. For instance, PIR sensors are cheap and non-intrusive but only give binary presence information. Cameras can identify human activities info at a fine-grained level, but are more expensive, privacy-risky and require line-of-sight. Wearable devices are diverse, but all of which need careful maintenance and often let users feel intrusive and troublesome.

Our research addresses the above-mentioned challenges by using sound sensors to detect human-context information indoor. Sound is everywhere and has many advantages such as rich in information, no line-of-sight problem, etc. Audio sensors or microphones are also very suitable for indoor applications as they are cheap, small and easy to install. On the other hand, sound also has obvious challenges such as noise interference and the overlap of multiple sounds. In addition, sound-based applications in buildings may need some more considerations, such as privacy concerns and resource constraints of devices. To study and address the impact of noises and overlapping sounds, our research is conducted on different scenarios and datasets, i.e. from clean sounds in quiet environments to overlapping sounds in crowded environments. In order to tackle the challenges in different scenarios, several methodologies are carefully designed and compared. Together with the performance or accuracy, we also compare the memory and time cost to show how they fit resource-constraint devices. In our research, an innovative lightweight CNN-based model is also proposed to balance performance and complexity. In some experiments, we strip the voice bands from audio inputs to explore the possibilities of using low privacy-risk data while still maintaining high accuracy.

In this thesis, we use sound to detect human activities and the number of people, which are the two very basic elements of human-context. Our research questions are as follows:

1. How can sound be used to recognize indoor human activities with resource-constraint devices while preserving privacy?
2. How can sound be used to count the number of people in indoor environments?

To address the above problems, we start with classifying single-source sounds that are related to human activities. Next we propose a multi-microphone model for noisy environments. This model first decomposes overlapping sounds and then recognize each of the decomposed signals. Thirdly, we provide a crowd-activity based solution for crowded environments. Instead of recognizing individual sound events, we think it better to estimate the ratio of different sounds in order to sense the human-context in the crowd. Finally, we proposed a voice-based people counting model. We count people from both estimating speakers in multi-speaker sounds and clustering single-speaker sounds. In this task, the key voice-feature is not from engineered features but is retrieved from transfer-learning.

In a brief conclusion, our study has shown that sound is suitable for both human activities recognition and people counting applications in the context of quiet environments. When environments become nosier, sound based models become less reliable and also require much more resources.

Acknowledgements

This research work is part of the project COPAS (Cooperating Objects for Privacy Aware Smart public buildings). My sincere thanks goes to NWO (Dutch Research Council) and Meiti (the Indian Funding Agency) for funding my research. Also thanks everyone that has helped me survive my PhD life. Throughout the writing of this dissertation I have received many support and assistance.

In particular, I would first thank my supervisor, Professor Paul Havinga, whose expertise and insight was invaluable in formulating the research questions and objectives. The discussion with you always helps me to keep my research on the right track when I feel confused about the bigger plan. I would also thank my daily supervisor, Professor Meratnia Nirvana, for the daily supervision and feedback on my daily work.

Besides the advisors, I would particularly thank my colleagues and friends Fatjon.Seraj and Kyle.Zhang for the technical and mental support. My thanks also goes to other colleagues: Viet Duc, Jacob, Etto, Okan, Eyuel, Rob, Yanqiu for the insightful and joyful daily discussions, and our secretary Nicole for all the valuable help.

At last, I would like to thank my family members, my parents - Chuanhua Wang, Xiping Song, my wife - Qing Chen and my lovely kids - Damon and Stefan. Thank you all for the practical and spiritual support every day.

Contents

List of Figures

List of Tables

Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
GMM	**G**aussian **M**ixture **M**odel
HDF5	**H**ierarchical **D**ata **F**ormat **V**ersion 5
HMM	**H**idden **M**arkov **M**odel
LPC	**L**inear **P**redictive **C**oding
LPCC	**L**inear **P**redictive **C**epstral **C**oefficients
LSF	**L**ine **S**pectrum **F**requencies
MAE	**M**ean **A**verage **E**rror
MFCC	**M**el-**F**requency **C**epstral **C**oefficients
NMF	**N**on-negative **M**atrix **F**actorization
PCM	**P**ulse-**C**ode **M**odulation
RBF	**R**adial **B**asis **F**unction
ReLU	**R**ectified **L**inear **U**nit
RNN	**R**ecurrent **N**eural **N**etwork
STE	**S**hort **T**ime **E**nergy
STFT	**S**hort-**T**ime **F**ourier **T**ransform
SVM	**S**upport **V**ector **M**achine
TE	**T**emporal **E**ntropy
WSN	**W**ireless **S**ensor **N**etwork
ZCR	**Z**ero **C**rossing **R**ate

Chapter 1

Introduction

In this chapter, we will first introduce the high-level background and goals of our research, which mainly aims for the smart buildings. This is because our research work is carried out under the COPAS project, although sound-based activity recognition can be applied in many different domains. Subsequently, we put forward the two research questions and the corresponding hypotheses. In the end, there is an outline about all the rest chapters.

1.1 Indoor human activity recognition for energy efficiency

1.1.1 Energy efficiency in smart buildings

Recent studies have shown that people spend 80%∼90% of their time indoor, therefore the indoor environmental quality has a huge impact on the comfort, health and productivity of the occupants [1]. This indoor environmental equality includes air quality, light, acoustic and thermal comfort. In order to improve the indoor environment, systems such as lights and HVAC (Heat, Ventilation, Air-conditioning and Cooling) are used everywhere in daily life. The downside of using these systems is that they normally rely on quite heavy energy. In Europe, around 40% of the entire energy consumption is represented in commercial and residential buildings at the year 2009 [2]. A report from 2010 says that US had 114 million households and more than 4.7 million commercial buildings, which consumed more energy than the transportation or industry sectors, accounting for nearly 40% of the total national energy use [3]. In China, buildings accounted for more than 20% of the total national energy consumption by the year 2009, and the proportion is estimated to gradually increase to 35% at 2020 [4]. Among all these consumptions, HVAC is one of the largest energy consumption factors, accounting for 60% of total energy consumption and half of the overall utility cost in a building.

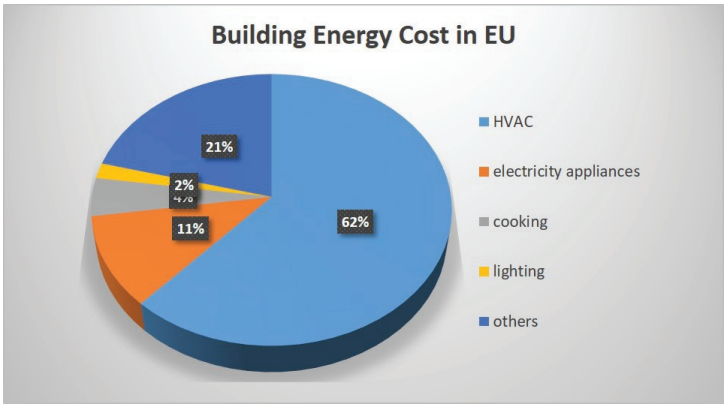


FIGURE 1.1: Energy consumption for different functions in buildings

In hotter areas such as the Middle East, this proportion is as high as 70% [5]. Figure 1.1 shows the proportion of energy consumption in EU buildings, where HVAC is the largest energy consumption factors [6]. In addition to obvious financial concerns, the use of energy also affects everyone's life, as many problems would arise: greenhouse gas emissions, acid rain, climate change, and the unsustainable energy sources. Energy efficiency impacts positively the portfolio of both buildings owners and utility companies, with benefits ranging from reduction in utility bills to grid stability and sustainability [7].

One can improve the building energy efficiency from various aspects, such as increasing the wall insulation or using better fuels and more efficient heat engines, etc. Of all these measures, the concept of "smart buildings" has emerged recently, referring to building automation systems (BAS) with automatic and intelligent control functions. Smart buildings aim to efficiently reduce energy consumption and operating costs without compromising the comfort of occupants, which mainly rely on computer science and information technologies [7].

In order to achieve the *"smartness"* and save energy, there are four basic components a smart building should consist of:

1. *Sensors.* Firstly, the building needs sensors to collect extensive and relevant ambient information. Modern HVAC control systems rely on a lot of environmental parameters such as temperature, humidity and CO_2 density for the actuators to react properly. Smart lights can detect the presence of humans with PIR sensors. Smart windows can use sensors to detect light intensity in order to adjust the sunshades accordingly.

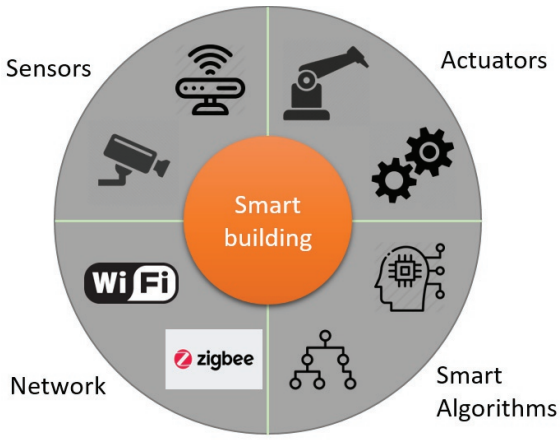


FIGURE 1.2: The basic needs of smart building systems

2. *Connectivity.* A smart building should be a connected building, both internally and externally. Through ubiquitous network connections, devices can communicate with each other inside the building, or communicate externally with other facilities through the Internet. Wireless technologies such as WiFi, BLE or Zigbee which allow easy installations and few costs are particularly helpful and popular nowadays.

3. *Smart algorithms.* After the sensor data have been collected and gathered, an intelligent algorithm should learn from them and issue commands to actuators to automatically react to changes of environment and occupants. Depending on the complexity of data and problem, these algorithms can range from rigid if-then rules to complex machine learning models.

4. *Actuators.* At last, an actuator receives the commands from the control system and is responsible for the regulation of the system, such as opening a valve or turning on an engine. It is the mechanical part of a system through which a control system act upon.

Figure 1.2 shows the four basic needs of the smart buildings systems.

It can be seen that data or information technology is the core of the entire smart building concept. Without incoming data, a system can only work according to the preset schedules, thus no intelligence is needed at all. The sensed data can be of various categories, such as environmental parameters, presence of users, or even the status of the sensors themselves. In practice, this 'smartness' from the sensed data can save energy from the following aspects:

1. *Environmental sensing:* A smart building should use sensors to continuously monitor the environment and adjust its behavior accordingly. Smart lights can be turned off when it detects no people are staying around. Smart windows are able to lighten or darken depending on sunlight intensity to save the illumination cost. As a heavy energy consumer, modern HVAC control systems can save energy from sensing a lot of environmental parameters such as temperature, humidity and CO^2 density, etc. All of these factors need be considered in order to calculate the required heat load precisely.

2. *Fault detection:* Malfunctioning of building energy utilities can waste more than 20% of the entire energy cost [8]. An investigation report concluded that there exist 13 types of faults in all kinds of building equipment, including material leakage, control system error, wrongly configuration, misplaced sensors or wires, etc [9]. Huge energy would be wasted if these failures are not handled properly efficiently. With historical and ongoing sensed data, data-driven models can efficiently diagnose the system and even detect problems proactively. Moreover, the proactive fault detection can also help reduce downtime and improve the quality of service.

3. *Global optimization:* Through ubiquitous network connections, devices can communicate with each other inside the building and with facilities far away. Instead of working separately, edge units of HVAC can sync the data and let the central system sets the actions of everyone, in order to achieve the global optimum on overall cost. By communicating with city power grid, a smart building can also save the cost by limiting its energy consumption during the peak hours of electricity supply.

4. *Demand-driven services:* Many of the devices in buildings are human-centric and only need to provide services while there are occupants. Lights should be turned off when no one is nearby in order to increase lifespan and cost efficiency. HVAC control systems are much more complicated, which also need to consider the warmup latency and the accompanying energy cost. Therefore, it does not only passively adjust the settings to current status and also need to predict the future occupancy pattern according to historical data.

1.1.2 Energy efficiency using human activity information

Among all the sensed data introduced in the previous section, the data about people is the focus of our research. Essentially, human comfort is the reason these systems exist in the first place. Knowing the human activity, location and mobility patterns inside the building can substantially increase the effectiveness of such solutions.

Again take HVAC as an example, it is a trend nowadays to adopt more and more human sensing technologies in its control systems. Smart HVAC systems in residential buildings may identify the users, and regulate according to their preferred temperature automatically [10, 11]. Some smart HVAC systems can also differentiate different human activities such as sleeping, cooking or reading. Each of these activities has its impact on human thermal comfort and needs different thermal and ventilation requirements [12, 13]. Moreover, an energy efficient HVAC system should be demand-driven, where occupancy detection and indoor occupant positioning are the key technologies of it [13]. In addition to the mere yes/no occupancy, the estimation of inhabitants-number also has an impact on the heat load and ventilation rate, thus is important to smart HVAC automation control[14]. In brief, various kinds of information about human activities ranging from location, behavior to the number of people can be utilized for demand-driven services.

1.1.3 Techniques for indoor human activity recognition

Numerous technologies can be used to detect human presences and activities automatically. Such technologies include camera systems that recognize human presence and actions from images, wearable devices that track users movement, PIR sensors that give binary information about people presence [15][16][17]. These sensors observe very different physical quantities and detect human activities from various aspects. Every sensor has its pros and cons in different aspects, and not every sensor is suitable for smart building applications. In general, passive and non-intrusive sensors are preferred than the wearable devices and intrusive sensors. This is especially the case in many public buildings where users of the buildings are not the owners. A customer has little concern on the utility cost of the grocery store, therefore would hardly bother participating in any troublesome activities.

To simply detect human presence, a PIR sensor is a quite efficient and common solution. Using PIR has many benefits such as being cheap, easy to deploy and privacy friendly [18]. A typical digital PIR sensor measures the infrared radiation and fires a signal when it senses a noticeable discrepancy, which indicates human presence since the discrepancy is most likely caused by humans [19]. However, a simple digital PIR can not detect occupancy accurately when people do not move, e.g. sleeping or sitting still. Another good option for presence detection is the light barrier, especially in security related applications. A light barrier is normally a pair of transmitter and receiver, where the transmitter projects an array of light beams to the receiver. It would fire a signal to the control system when an object breaks one or more of the beams [20]. Other similar

sensors that use transmitter and receiver pairs include laser scanners and ultrasonic sensors. It is worth mentioning that the light barriers can only detect the very small area covered by the beams. The aforementioned sensors can only detect whether there are people, but in many applications the number of people is also an important input. Wireless sensors such as WiFi or BLE receivers can estimate the number of people around. Thanks to the popularity of smartphones, nowadays nearly everyone has a smartphone which constantly scan WiFi access points. As the SSID of each smartphone is unique, a WiFi server can sniff the broadcast beacons in order to count people. However, this method falls short if other non-smartphone wireless devices are around, or not everyone carries a smartphone. A more challenging problem is that the distance between a transmitter the receiver is difficult to estimate, so that whether the sniffed device is inside or outside the building is hard to know [21]. A camera system is a more handy option in counting people, since seeing is believing. Thanks to the booming machine learning technologies, it is possible to detect and track people automatically in images accurately. To detect persons in an image, one first needs to train an object classification model (normally based-on deep learning) from extensive sample images in different categories [22]. Given an image that mainly containing one big object, this trained model is able to tell whether it is a cat, a car or a person. Secondly, this object classification model is to be combined with some searching framework to find possible locations of objects in an image (for example Faster-RCNN [23]). These object detection models nowadays can easily find people and count them in an image, be them walking, sitting or even partly blocked. Apart from normal cameras, thermal cameras which form thermal images just like common camera forms visual light images, are also widely used in human detection and tracking [24, 25]. The aforementioned computer-vision techniques can easily be adapted and applied to thermal images, as the thermal images and optical images resemble in the outline sketch.

Cameras can also be used for human activity recognition, which is a more complex problem. The goal of human activity recognition is to examine activities from video sequences or still images, and to correctly classify them into predefined activity categories [26]. The definition and categorization of human activities itself is a challenging job. Some human activities such as 'reading' can be identified instantly, some consist of a sequence of different actions such as cooking. Some activities are the sub-categories of others, such as 'kicking' to 'playing football'. In an overview paper [27], six levels of human activities are defined depending on the complexity of activities. These activities include gestures, atomic actions and group actions, etc, which are shown in Figure 1.3. The methods and features used to reason human activities also vary a lot. Some methods use the spatiotemporal features, some use Hidden Markov models, and some use high-level reasoning methods by modeling the motion of human body parts, etc [27].

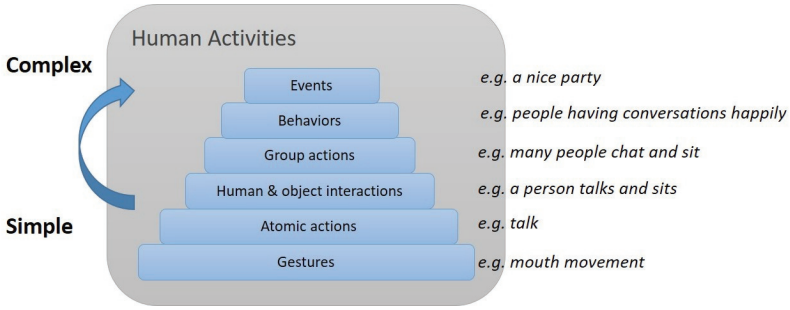


FIGURE 1.3: A hierarchical categorization of different human activities

In this review paper [27], Gestures are defined as primitive movements of the body parts of a person that may correspond to a particular action of this person. Atomic actions are movements of a person describing a certain motion that may be part of more complex activities. Human-to-object or human-to-human interactions are human activities that involve two or more persons or objects. Group actions are activities performed by a group or persons. Human behaviors refer to physical actions that are associated with the emotions, personality, and psychological state of the individual. Finally, events are high-level activities that describe social actions between individuals and indicate the intention or the social role of a person.

Nevertheless, the common components of most camera-based methods are: 1) background subtraction, 2) human tracking and 3) human action and object detection.

In this thesis, we are going to use sound, or acoustic sensors to detect human activities. Compared to the other sensors, sound offers many advantages such as pervasive, rich in information, no line-of-sight problem, etc. A sound sensor or microphone can be both cheap and small enough to be barely noticed. Despite so many advantages, there are also obvious challenges in using sound for smart building applications. Sound sensors may induce critical privacy issues, since people could feel stalked when microphones are around them. The processing of sound signals is also very complicated, especially when affected by noises and reflections. Although many research initiatives have focused on environmental sound recognition for general purposes, very few are designed for human activity detection.

TABLE 1.1: Sensors for collecting ambient information

	info level	cost	identification	counting	localization	activity	unobtrusive
PIR	L	L	-	-	-	-	+
light barrier	L	L	-	-	-	-	+
RFID	L	L	+	+	+	-	-
Wifi, BLE	M	L	+	+	+	-	+
ultrasonic	M	M	-	+	+	-	+
motion sensor	M	L	-	-	-	+	-
microphone	H	M	+	+	+	+	+
Thermal camera	H	H	-	+	+	+	-
video camera	H	H	+	+	+	+	-
L=low, M=Medium, H=high.							

1.2 Human activity recognition using sound

1.2.1 The scope, challenges and intuition

Sound is complex and ubiquitous, and there are many different ways of adopting sound in smart building applications. It can be as simple as a lamp embedded with a sound-sensor switch which merely uses the volume, or as complex as a voice assistant in smartphone which can 'understand' human languages and execute commands accordingly. Some applications also combine sound with other data to predict human behaviors, for example a HVAC system can combine microphones with humidity, CO_2 and other sensors to get a full image of the human behaviors [28]. In the sensor-fusion works, sound data is only one of the multiple sources so that the fusion model rather than the audio-processing model is the focus of research.

To provide a clear scope, our research focuses on the data technologies and algorithms, which extract and reveal the info related to human activities, based on time and frequency audio features. Each main chapter works specifically on one type of human recognition problem, or one type of information. Undoubtedly, these extracted information can of course be fused to serve a larger goal. However, these fusion processes may vary depending on the specific applications or scenarios, so are they outside the scope of this research.

Our research is aimed at smart building scenarios and IoT devices, hence many restrictions and preferences exist. Firstly, our solutions should only require regular and cheap microphones for sound recording. This means the microphone arrays with complex hardware are not considered, as they are normally too expensive and intrusive for IoT application scenario. For the same reasons, we do not require high-quality microphones either, which means the output sound may not have a high SNR (Signal Noise Ratio). Secondly, our algorithms or software should be lightweight in memory and computationally efficient, as IoT devices are normally resource-constrained. In practice, we have used raspberry pi 3B and zero for testing and deployment, both of which are capable of real time processing. In order to evaluate the results with the same experimental setup, all the experiments carried out in this thesis use raspberry pi 3B platform and cheap USB microphones. Any hardware equivalent or more powerful than this setting should work well with our algorithms. For algorithms or models with the similar performances, we prefer the ones with significant less memory and calculation time. Thirdly, privacy is also a big issue in indoor environment, no one wants their conversations to be tapped. Although concrete measures, such as data encryption or network security are not in the scope of this thesis, we still need to consider and minimize the risks which might be

incurred from the system. This is particularly important in a quiet environment, where people's conversations can be easily recognized.

As for all the researches that extract info out of sound, they mainly divide sound into three categories: speech, environmental sound and music sound. Of the three sound types, we mainly use the environmental sound and context-independent speech sound. Environmental sound refers to all the non-speech and music sound made by objects or humans surrounded us in the environment. Obviously, environmental sound can have an unlimited size in categories so that the researches are highly domain oriented. It is therefore hard to create a general model which understands every single sound event. Of all the environmental sound events, we only choose the most common ones that represent indoor human activities, such as walking, door slamming or hands clapping. Beside environmental sound, we also use speech sound in some of our works, as human voice is efficient for distinguishing and counting people. The researches on speech sound has gained a lot of attractions and has achieved great progress within the last decade, benefiting from the advances in deep learning and big data. Speech recognition and speaker recognition are the two most popular research fields. Speech recognition is an interdisciplinary field of computational linguistics and sound processing which develops technologies that enables the recognition and translation of spoken language into text. Speaker identification, on the other hand, refers to identifying the speaker, rather than what they are saying. Our research on speech sound is close to speaker identification, which needs to differentiate people according to their voice.

1.2.2 Research questions and hypothesis

The semantics of 'human activity' can relate to the researches of the presence, physical actions, mobility or identity of people. In this thesis, we mainly focus on activity recognition and people counting, i.e.: what are people doing and how many people are there? Identifying an individual person is not included in our research because of the potential privacy risks. We also believe other research objectives such as the presence or the mobility pattern can be achieved through continuously counting of people in every place, which of course need multiple devices and additional post-processing.

In this thesis, we would answer the following two main research questions, each of which consists of several sub-questions:

Q 1: How can environmental sound be used to recognize human activities automatically and efficiently?

Q 1.1: How can sound activity recognition methods be privacy-preserving and efficient to fit smart building applications?

Q 1.2: What is the impact of noises and overlapping sounds to sound-based activity recognition?

Q 1.3: How can sound be used to reveal human activities in a crowded environment?

Q 2: How can speech sound be used to count people automatically?

Q 2.1: What audio features are highly related to human voice and can be used to differentiate different people?

Q 2.2: What is the impact of the overlapping talking sound to speech-based people counting? Can the people-counting model work well in a crowded environment?

To solve the above research questions, several hypotheses are made:

H 1: Through automatically classifying sounds related to human activities into predefined classes, it is able to recognize human activities by sound in both quiet and noisy environments.

H 1.1: Using machine learning models and carefully extracted audio features, sound events can be automatically classified by a very lightweight model. Meanwhile, privacy risks can be largely reduced by using edge computing and voice bands stripping, without losing much accuracy.

H 1.2: Through decomposing the overlapping sound signals, multiple sound events can be simultaneously classified in a noisy environment.

H 1.3: When the sound is too noisy to be decomposed into its sub-components, contextual info can still be obtained from the sound in through the statistical features.

H 2: Through extracting and clustering the voice features, the speech sound can be used to count people in the scenario where everyone speaks.

H 2.1: A sophisticate speaker-identification model trained from substantial human voices can extract very general voice features. When projecting human voices onto this feature set, sounds from the same person should get similar values, while sounds from different people should get a larger difference in value.

H 2.2: A model that estimates how many people are talking simultaneously can increase the people counting accuracy. It can also filter out the 'multi-talkers' from 'single-talker' sound to reduce the errors of the model in ***H 1.1***.

1.3 Thesis Contributions and Organization

The remainder of the thesis is organized as follows:

Chapter 2 presents an overview of related works that focus on applying sound in smart building applications. These works vary a lot from targeted scenarios, applications and also methodologies. We would also compare these works and discuss the pros and cons of each type of methods.

Chapter 3 presents all the audio dataset used in our research, which consists of both environmental sound and speech sound. Most of these data are collected from open and free public datasets so that no privacy concern is raised. Aside with the raw audio data, this chapter also includes the common preprocessing methods shared by all the models throughout our research.

Chapter 4 and Chapter 5 present our research on sound-based activity recognition in quiet and crowded environments, which together answer the research question **Q 1**. In these two chapters, we mainly use machine learning techniques to identify environmental sound events that can reveal different human activities. Sounds in **Q 1.1** contain clean single event and is researched in Chapter 4, where the focus is not only on performance but also on privacy and the method complexity. Chapter 5 answers question **Q 1.2** and **Q 1.3** where the environmental sound is noisy and with overlapping sound events. This problem is much harder and no perfect solutions exist when sounds become very noisy. Two different solutions are proposed by us from different aspects. Section 5.1 aims to classify every sound source in an overlapping sound event, by decomposing the overlapping signals first. Section 5.2 aims to estimate the proportion of each type of events in a crowded environment, which reveal the information of crowd behaviors.

Chapter 6 addresses the people counting problem of **Q 2** with a two-step methodology. The model in Section 6.3.2 aims to count people by distinguishing their voices, using the speech clips that contain a single speaker. This voice-feature set used in this method is acquired from transfer-learning, which reuses the knowledge from a well-trained speaker-identification model. In Section 6.3.3 we propose a model to directly estimates how many people speak simultaneously from noisy speech clips. This model is less accurate in people counting, but is a good complement to the voice-distinguishing model of Section 6.3.2. It can helps to filter out 'multi-talker' speech clips, which can cause errors to the voice-distinguishing model.

Chapter 7 gives the overall conclusion of the works on each research problem, and the future work as well.

Chapter 2

Related works using sound in smart buildings

This chapter presents a study of the related works that apply sound in human activity recognition and people counting in separate sub-chapters. These works vary a lot from targeted scenarios, applications and methodologies. Therefore, it is hard to fully cover them from every aspect. In this chapter, the emphasis is on the methodologies and algorithms, as which are the most relevant to our research. We will first present the classic researches of the problem, and then move forward to list the categories of the cutting-edge methods.

2.1 Sound-based activity recognition

Classification of human activities based on sound falls under category of *'Environmental Sound Recognition'* (ESR). ESR aims to automatically detect and identify audio events from captured audio signal, where the events could be anything from the environment. By using ESR on audio events related to human activities, IoT devices can 'hear' what people are doing and save energy for buildings. Compared to other areas in audio presence such as speech recognition, ESR researches has received less attention in general. As environmental sound has an arbitrary size of categories, very few of the ESR researches are specifically focused on human activity related events. In order to provide a broad overview, we will list and compare the most promising ESR works in recent years. These works are described in the following two sections, from the models that work in quiet environments to the ones targeting crowded environments. These works are separated according to the scenarios mainly because the challenges and solutions, which are the essence to investigate, differ a lot in the two scenarios. In a quiet environment where

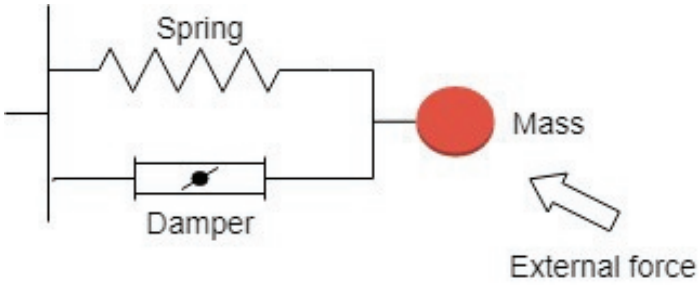


FIGURE 2.1: A theoretical resonance model to show how objects make sound

every event is clearly heard, a good classification model is the major key. On the other hand, a crowded environment needs to handle overlapping sound events, where signal decomposition is often needed.

Martin et al. [29] has given a comprehensive and in-depth analysis of the sound source recognition problem. This work was strong in the analysis of mathematics and signal processing. Figure 2.1 shows a simplified and general physical model of how objects vibrate to make sound. According to the geometry and material properties, a sound source's vibrating body can be viewed as a resonator coupled with the external force. Without any external force, the mass will vibrate with a fixed frequency, which is the natural frequency of the system. With external forces, the response would be a range of frequency bands that centered at the natural frequency. From this analysis, the pitch frequency and its brightness were suggested to be the key for humans to recognize a sound source.

2.1.1 In quiet environment

Researchers have been interested in automatic audio recognition for a long time. However, it seems not to be a trivial task for computers although human hearings can easily identifying what's happening.

Classic models based methods

In the early days, sound source recognition is mainly based on all kinds of signal processing and the characters analysis of the specific sounds.

The Sound Understanding Testbed (SUT) [30] was one of the first to recognizes environmental sounds. They employed several levels of feature abstraction, from power intensity envelope to "peaks" representing narrow-band portions of the spectrum and "contours" made up of groups of peaks with similar frequencies. With a Gaussian Mixture Model

(GMM), they could get around 60% accuracy on 40 household events recorded in their laboratory.

Along this path, a lot of works have evolved with better feature combinations and more efficient machine learning classification models. With the means of classifying short sound events, these works target either specific applications or general research purpose. Cowling et al. [31] compared several up-to-date feature extraction and classification techniques typically used in speech recognition for general environmental sound recognition. In his research, tempo-frequency features such as short time Fourier transformation (STFT) were found to perform better than stationary features such as Mel-frequency cepstral coefficients (MFCC). As for the classification models, they also used several techniques, and they found Dynamic time warping (DTW) working better than GMM in the task. Lin et al. [32] combined several features from both wavelets and Fourier transformation to build a sound source recognition system. They also used Support Vector Machines (SVM) instead of GMM as the classification model and showed good results. Until then, SVM was only designed for binary-class classification and they proposed a tree structure of multiple binary-SVMs for the multi-class classification. Yaniv et al. [33] used a combination of sound and floor vibrations to automatically detect people fall events to monitor elderly people living alone. Sound event length, energy, and MFCC features were extracted and used for event classification. In fall detection, vibration sensor data is the key source and sound was used as an effective aid to correct false-positive predictions. Tran et al. [34] also proposed SVM for online sound event classification, but with their new probabilistic distance kernel. This method has comparably good performance with very short latency and low computation cost compared to most other SVM based approaches. In order to increase the efficiency for realtime applications, they only use the simple STE (Short Time Energy) feature as the input. However, sound power is highly related to the distance of the sound source, using this feature alone could be highly biased in sound recognition task. As a broad range of feature sets have been used in these works, Dalibor et al. [35] compiled a survey on the audio feature extraction techniques for audio retrieval systems. They proposed a taxonomy of audio features together with the relevant application domain. Temporal domain, frequency domain, Mel-frequency and harmonic features were the most popular features in environmental sound recognition. This work has provided an extensive collection of audio features for environmental events recognition, which can benefit researchers in this domain.

As most works classify each sound event independently, some suggest combing the context information would increase the performance. These works view the context as the prior distribution of sound events. Intuitively, clapping sound happens more often in a classroom than on the streets. Heittola et al. [36] developed a context-dependent sound

event detection system. With a hidden Markov model (HMM) model, it first gives the context a label which provides prior probabilities to improve event detection accuracy. The system also adopted an iterative algorithm based on the original HMM to model the overlapping sound events. Cai et al. [37] proposed a key audio effect detection framework for context inference, where movies are used as the dataset. The advantage of their model over others is to first infer the context by carefully picked and distinguishable key sounds events (like explosion and gun shot means an action movie). However these 'key events' need manually and carefully chosen and is hardly scaleable. Chu et al. [38] proposed an environmental sound recognition model to classify events within 14 scenarios. Their innovation was a matching pursuit algorithm to extract features from time-frequency domain to complement MFCC. Compared to previous works, they employed a very rich feature-set to help improve the classification accuracy. For these context-based researches, HMM is the most commonly used model, where the prediction of one event depends on both itself and its previous events. HMM models can be called stateful or sequential models while SVM and decision-tree are stateless models. From applications perspective, context-based methods are not quite useful for real time applications. Moreover, for most IoT applications, the context is fixed and doesn't require to be detected.

In brief, the performance of the classic models based methods highly rely on the engineered features, or signal-processing. This is because classic models can easily overfit to high dimensional features, where audio data is too large and requires deep compression. Therefore, it is difficult for classic models to strike a balance between learning complex representations and being stable. In this sense, a good feature is more important than a good model. Using decision tree as an example, it is simple to understand and to interpret, which also requires little data preparation, however is hard to learn concepts such as XOR or multiplexer problems.

Deep learning based models

With the rapid development of neural network technology in the field of image processing and speech recognition, many works have also applied different types of neural network models to ESR problems. Being different from classic models based methods, deep neural network models could easily deal with very high-dimensional features or even raw data. Deep learning models are also very flexible and scaleable. User can choose different frameworks such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) layers to learn image and linguistic knowledge respectively.

Khunarsal et al. [39] used multiple full-connected deep neural network layers to classify short audio events, with the STFT spectrogram as inputs. Their method achieved an overall accuracy of 85% on a set of 20 different audio events. Multiple full-connected

layers can represent very complex non-linear information, as long as there are enough hidden layers. The shortcoming of full-connected layers based models are also obvious: they are too computationally expensive, and they could easily learn the noise rather than the real pattern.

Parascandolo et al. [40] used a bi-directional RNN to classify real life sound events. This method used a frame-in-frame-out framework just as the full-connected neural network in [39]. In RNN networks, a state information exists which remembers past or foresees future inputs. The output of each frame depends on both the current frame and this inner-state information to make the prediction more comprehensive and accurate. RNN is especially popular in linguistic information learning since the sentiment of a sentence develops through the connection of words. However,RNN as a sequential model may be over complicated for sound events, which can be easily identified without a context.

Piczak et al. [41] applied CNN, a method usually applied to image processing, to classify environmental sound and achieved results comparable with traditional methods (64.5% accuracy for 10 classes). CNN is best known for learning shift-invariant features from the input images, while the spectrogram of an audio is a perfect image-like feature. Compared the real life images, the spectrogram 'images' do not have much details to be identified. The envelope or the general shape of the spectrogram is enough to classify different sound events. As a result, they only use two stacked CNN layers to extract the shape information, which is much less than the layers in typical image processing frameworks. Many other works also proposed CNN for sound events classification, mainly with different settings including hidden-layers, filter-size or activation functions [42] [43]. While the spectrogram resembles a grey image which has no depth, [44] increased the model robustness against noises via stacked spectrograms which are just like RGB images.

The difference of event-level and frame-level recognition is shown in 2.2, where CNN is typically the former type and RNN belongs to the later.

A summary of the mentioned works are listed in Table 2.1, which contains the targeted scenarios, the used feature and model of each work.

2.1.2 In crowded environment

Thanks for the many contributions of researchers, sound event classification in a quiet environment is no more a theoretical hard challenge. However, in real life the environment is often very noisy, many sound events would happen simultaneously. In this case, a robust system must be able to classify the overlapping events, just like human

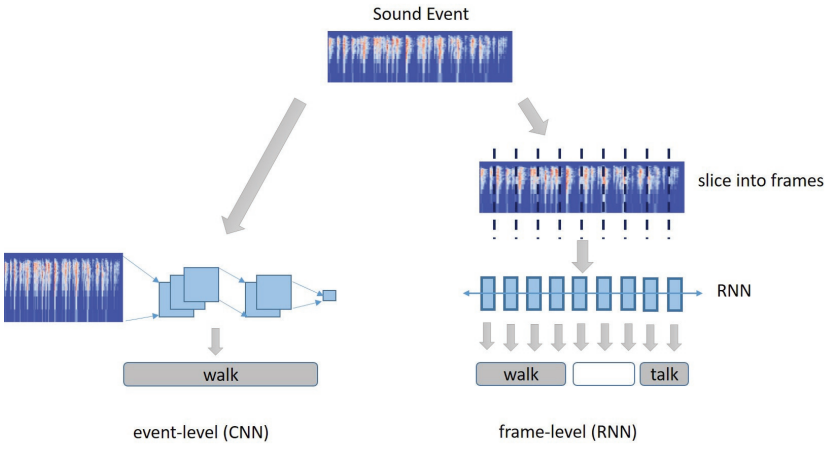


FIGURE 2.2: An example of event-based recognition and frame-based recognition

TABLE 2.1: A summary of the works used for sound event classification in quiet environments

References	Scenario /Data Source	Context/Non-context	Online /oFfline	Features	Models
[31]	General	N	F	FFT subband,STFT subband,MFCC	DTW, GMM
[45]	Conference Room	N	O	Volume	SMM
[33]	People Falling	N	O	Length,STE,MFCC	HMM
[46]	General	N	O	STE	SVM
[41]	General Urban	N	F	Mel bands	CNN
[39]	BBC recordings	N	F	STFT	DNN
[42]	General	N	F	STFT	CNN
[43]	General Urban	N	F	Mel-bands	CNN
[37]	Movie	C	F	Spectral Flux,Harmonicity,MFCC	SVM, BN
[38]	General	C	F	MFCC,temporal signatures,time-frequency	SVM, BN
[36]	Transport,Shop,Open Space	C	F	MFCC	GMM,HMM

C = Context, E = Event, BN = Bayes Network, SMM = Semi-Markov Model, GMM = Gausian Mixture Model, DTW = Dynamic time warping

hearings. Single sound classification, from the methodology perspective, is a very typical machine learning problem where most recent works share a similar pipeline. However Overlapping events classification, on the other hand, is much more complicated and the solutions may vary greatly.

Multi-label learning

Multi-labelling means a classifier can output multiple positive classes. Basically, multi-labelling based methods just extend the single sound classification models for overlapping sounds.

It is very easy to design a multi-labelling model with neural network, the only difference from a single-labelling model is the output layer. Cakir et al. [47] used the output layer activation function with a sigmoid function which bounds every output-node value to (0,1). With a post-processing step that binarizes the outputs by thresholding, this model can identify multiple types of sound events happening at each time frame. Their

neural network model consists of multiple full-connected layers and use MFCC features as the inputs. Their results showed that the accuracy of classifying two overlapping events was quite high. Adavanne et al. [48] used multi-channel microphones and RNN (Recurrent Neural Networks) for sound event detection, the output layer also used a sigmoid activation function. The RNN layer is used to create a frame-in-frame-out model, where the prediction is performed on each short frame, not on the entire event. In this case, the prediction of one frame definite gets more accurate with the information from previous frames. The advantage of a frame-in-frame-out model is that the input length of an event can be variable.

For classic models, there is no common and easy solution for designing a multi-labelling system. Temko [49] combined decision tree and SVM into a confusion model to detect and classify overlapping sounds. In order to classify the detected overlaps, a confusion-based clustering scheme is used. A big decision tree is trained with one SVM model at each node, where each SVM finds the best way of splitting the classes at a given node into two clusters with the least mutual confusion. In that way, each leaf node is a possible single or overlapping class label. Tran et al. [50] invented improved this method by transforming the frequency subbands to a new domain, making the features easier to be classified. However, the scalability of these methods is quite limited, as the model complexity rapidly increases with the class numbers. Moreover, these models only support two events overlapping in practice.

In conclusion, it is much easier and more scaleable to design a multi-labelling system with deep learning than with classic models. Multi-labelling models address the overlapping problem merely through mass training, which also has an obvious flaw: the training data needs to cover all the possible mixtures in order to be representative. Theoretically, a multi-labelling system learns an overlapping event as if it's a new event. More specifically, a multi-labelling model can not recognize an overlapping event $A + B$, if it has only seen individual A and B before. To avoid bias, the training data needs to be incredibly large when more than two events overlap.

NMF based Signal Decomposition

The second method to address overlaps is using signal decomposition, where non-negative matrix factorization (NMF) is one of the most popular algorithms. NMF is a group of algorithms in multivariate analysis and linear algebra where a big matrix X is factorized into two smaller matrices W and H, with the property that all three matrices have no negative elements. This non-negativity makes the resulting matrices much smaller in dimension so that is easier to inspect. In the processing of audio spectrograms, non-negativity is inherent to the data being considered. Figure 2.3 shows the basic principle of NMF. In audio signal decomposition, matrix V is the spectrogram of m time frames

and n frequency bands, matrix W represents the r extracted sub-components, while matrix H indicates the amplitude of the sub-components at each time frame. One sound source may consist of one of or multiple components, i.e. multiple columns in W.

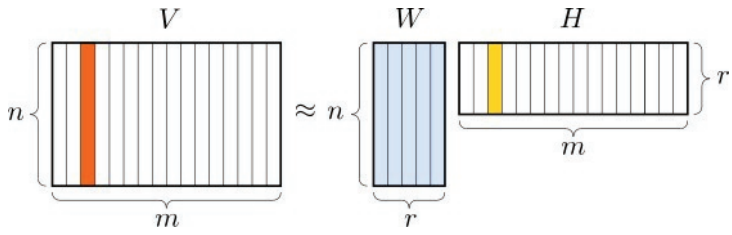


FIGURE 2.3: The basic principle of non-negative matrix decomposition

Heittola [51] used NMF to transform the input audio into four components, where different sound events are separated into different components for recognition. Applying additional constraints, such as sparsity on the NMF can improve the decomposition [52] and by treating the components as a coupled matrix factorization problem [53], the problem can be transformed into supervised learning that automatically maps the audio class label to each component. NMF based methods have their own drawbacks for instance the feature-length (i.e. the complexity) is hard to control when the number of samples becomes large and the method predicts the class label for short frames thus, losing the global characters of the event. Compared to environmental sound classification, NMF is more popular for recognizing musical instruments, where each instrument makes stable and very distinctive frequencies.

Multi-channel inputs

Instead of using single-channel sound, some works use multiple microphones or microphone-arrays to decompose overlapping sounds. A microphone array is multiple closely distributed microphones operating in tandem. Given a fixed physical relationship in space between individual microphones, well designed algorithms can locate sound sources and filter noises. Microphones-arrays can achieve a high localization accuracy even for overlapping sound and are commonly seen in robotic applications [54–56]. Although the purpose of these works is to perform sound localization rather than classification, one can easily use the decomposed sound signals for classification.

Most of these localization algorithms are based on the direction-of-sound (DoA) principle. As the location of every microphone in a microphone-array is known precisely, the slight difference of received signals among each microphone can be leveraged. On top of DoA, recent researches also use a DNN classification model to detect the source presence likelihood at a fixed set of angles according to the classification results [57–59].

In one of these works, Takeda [57] use a supervised learning DNN network to decompose the eigenvalue of the spatial correlation matrix. The goal of this work is to separate speech sound and locate the speaker from a noisy environment filled with other sound sources. With enough training, the DNN model can estimate the posterior probability of sound positions at every possible location. On the input side, this DNN model uses several concise features that are extracted from the STFT spectrogram. In practice, this method requires a long period of time as input for pooling and may not fit for real time applications. He et al. [58] later designed a DNN model with a likelihood-based encoding of the network output, which naturally allows the detection of an arbitrary number of sources. This DNN used the cross-correlation of sub-bands information as features and achieved better localization in sound mixtures than pure spectrograms. In [60], CNN and RNN are combined with multi-channel inputs, with CNN being the first layer to learn intra-channel features and the following RNN layers to learn inter-channel features.

Being obtrusive and expensive, microphone-array methods are ill-suited for IoT applications. Instead, acoustic sensor networks are used to discreetly track human activities by leveraging the time-of-arrival (ToA) of the sound from the source. The implementation of these networks varies from simple localization using energy-based threshold [61] to estimate the ToA and applying an audio event classification algorithm [62] for sound classification. Sometimes the quality of the network does not provide a reliable ToA estimation [63] thus, new algorithms are required to compensate for these uncertainties. However, it is rare to see acoustic sensors be used to locate and separate overlapping sound sources in noisy environments.

As a summary of the aforementioned works, the difficulty of sound events recognition has been greatly eased by the rapid development of deep learning techniques. Many models such as CNN and RNN that aim for speech recognition have achieved good results in classifying environmental sounds. However, in order to apply these models to real applications, the complexity and privacy issues of the models still need to be considered. In real applications, the greatest challenge is probably the overlap of sounds. It is even difficult for human hearings difficult to distinguish many overlapping sounds. Depending on different scenarios and degrees of sound overlap, different kinds of solutions are applicable. In less crowded places where only a few sounds occasionally overlap, a multi-labelling model could be a good choice because of its simplicity. This method falls short when the environment becomes more crowded, since its complexity grows exponentially when the number of events increases. In more crowded environments, it is better to decompose the overlapping sounds into their sub-components before classifying each event. One can choose NMF-based models for mono-channel signals and DoA-based methods for multi-channel inputs. In general, the signal decomposition accuracy highly

depends on the complexity of the signals. It is fairly easy to decompose sounds from music instruments since each produces a clear and single frequency sound. In contrast, the sound events in real life are much harder to decompose since they normally consist of multiple frequency bands and last much longer. As a matter of fact, there is no perfect solution yet to decompose and identify each event from overlapping sounds, especially in a crowded scenario such as a party or a restaurant.

2.2 Sound-based people counting

In this chapter, we will present the sound based people counting works. Counting people using sound is much harder than using images or even WiFi signals. A snapshot of a room tells exactly how many people and who are there in it. Also with today's computer vision technology, state-of-the-art object detection models can quickly detect humans with close to 100% accuracy. However, there are yet no perfect solutions of counting people with sound to be both accurate and fast. Depending on whether individuals are 'identified' , there are two different types of sound-based people-counting techniques.

Counting without identification

In this section, we will introduce the methods that can count how many people are speaking simultaneously, caring not the individuals' identity.

Human hearings can easily identify if there are multiple people speaking simultaneously. Whatsmore, we can even focus on and pick out the content of interest. However, this ability is a difficult problem for computers and has been an important and long-term goal in speech recognition research. Bronkhorst has [64] provided a comprehensive survey on how to select multi-talker speeches from the audio stream. This research covered a broad range of areas such as psychoacoustics, semantics and the human auditory system.

One type of people counting works is by decomposing the overlapping and classify sound sources of environmental sound, where each sound source corresponds to a person (they assume all sounds are made by people). These methods are quite similar to the ones used in overlapping sound classification, which has been stated in the previous chapter. The exception is that these works only need to count, but not to separate or reconstruct the signals of each sub-component. For example, Mirzaei et al. [65] counted people by evaluating the differences of phase and amplitude between received spectrums by microphones. Walter et al. [66] counted multiple sound sources with a Bayesian infinite Gaussian mixture model. Signal decomposition requires specific signal processing methods specific to an individual task, making these models efficient only in a quiet environment with few people. However, counting people through counting sound sources

is error-prone and hardly scalable. This only works with very few people where there are not too many events to detect. Not to mention that multiple persons can make the same sound or vice versa.

The other type of works uses speech sound instead of environmental sound to count people, i.e. to count speakers. Since everyone's voice has its own characteristics, many propose to count simultaneous speakers from an overlapping sound clip. Arai et el. [67] proposed single-channel sound to count people leveraging speeches energy pattern. This early research was mainly based on signal processing. Their method seems to work well in an experiment with a relatively small amount of data and limited speech text. It is hard to estimate the results if the model is used on text-independent speech sound. Andrei et al. [68] proposed an algorithm that counts concurrent speakers through correlating single audio frames of multi-speaker mixtures with a set of single-speaker utterances. This work used STFT spectrogram as the features and dynamic time warping model. Their experiments granted averagely 70%+ accuracy for counting up to ten simultaneous speakers. Stoter et al. [69] later proposed a much more complicated RNN(recurrent neural network) model on the same problem and achieved better performance based on the huge training dataset.

Despite the good performance of counting concurrent speakers in testbeds, this however has a big flaw in real applications. The counting without identification could not well proximate the real number of people since not everybody speaks simultaneously. Moreover, the number of speakers would become impossible to estimate when it reaches a certain level. It is just impossible to tell if a mixed sound contains 8 or 10 speakers, even for people blessed with outstanding hearings.

Counting by identification

While an instant counting through overlapping sounds only gets vagarious results, some proposed to identify each individual in a continuous process. More specifically, these methods normally use speech sound or voice to identify people, as everyone's voice has some unique characters. Instead of estimating how many are talking in a very noisy environment, we can count by making notes everyone is talking in turn. By contrast, when it comes to differentiate people, environmental sounds are not as handy as for activity recognition.

The key core of distinguishing people's voices is the extraction of highly voice-correlated features. This task is well-discussed in the research domain of speaker diarization. Speech diarization is the technique without knowing speakers' voice beforehand, hence belongs to unsupervised learning. It is commonly used as part of the job in speech recognition where the speech of multiple speakers in a conference or broadcast news

need to be separated [70]. Undoubtedly, audio features that are popularly used in speech recognition such as MFCC and LPCC are also widely used in speaker diarization. But in this specific domain, a more popular engineered feature that correlates with speaker identity is I-vector, applied in smart voice applications to verify the host's voice from a random speech pool [71]. The only disadvantage of the I-vector feature is that it needs the prior knowledge of speakers and is mainly used to verify a certain speaker's voice which is not suitable in our case.

Here are several examples of using speaker diarization in counting people. Friedland [72] proposed a multi-modal approach to detect and count speakers in meeting recordings. Several acoustic features and compressed video features are combined to double ensure accuracy. Their work also presented a systematic investigation of the speaker discriminability of 70 long-term audio features, experimental results show that despite the dominance of short-term cepstral features in speaker recognition, many long-term features can provide significant information for speaker discrimination. With several hours of real meeting dataset, they grant an overall error of 21%+ in speaker counting. Xu et al. [73] proposed a smartphone based speaker counting system. This system mainly extracts MFCC and other engineered short-term audio features of a short sound clip as the voice features. These voice features are clustered in real time for estimating the number of speakers, where each cluster represents a person. A customized on-the-fly clustering algorithm was used in the research since this work was targeting a real-time application. In several experiments with up to 10 persons intermittently talking, this smartphone-based people counting system granted an average error of 1.5 (person).

Both of these models depend on the assumption that everyone speaks and no one rushes to talk. However, there is always overlapping talking sound in real applications. These overlapping speech sounds would be falsely clustered as new persons since the extracted features appear new to the clustering algorithm. Another weakness of the above researches is the audio features they used. Engineered features used in [72][73] are not specifically designed for speaker identification, and one can imagine the MFCC features of the same speaker might alter dramatically along with different contents or contexts. A more popular engineered feature that correlates with speaker identity is I-vector, applied in smart voice applications to verify the host's voice from a random speech pool [71]. The I-vector feature however needs the prior knowledge of speakers and is mainly used to verify a certain speaker's voice which is not suitable in our case. There has been tremendous progress in applying neural network technologies in natural language processing. It is therefore the trend to use deep belief networks and convolutional neural network (CNN) to extract voice features, which may be capable of differentiating between a large amount of speakers [74] [75] [76].

To conclude, in order to develop a speech-based people counting system, there is still much to be improved. Firstly, a preprocessing step should be employed to filter the environmental sound and the overlapping speech sound. This would allow a clean data input to the voice-clustering process. Secondly, there is no perfect feature-set in this task yet. A more dedicated, specifically designed feature-set from deep transfer learning might trump the traditional engineered feature-set.

Chapter 3

The sound dataset used in this research and the preprocessing

This chapter will briefly introduce the dataset used in our research, which is the common foundation of all the experiments and evaluations. Our dataset consists of both real-life recordings and simulation data. The simulations are mainly used in crowded environment analysis where data and ground truth are hard to obtain. In addition to the raw audio data, this chapter also includes the common preprocessing steps used throughout the thesis, while more details will be covered in the later chapters.

3.1 Audio dataset collection

3.1.1 The public and website datasets

For a long time, researchers have shown great interest in audio processing, especially speech recognition. One of the contributions as well as the foundations of these researches is the extensive and labelled datasets used for experimental evaluation. Many of these datasets are packaged and published for research purposes, which can be downloaded for free. To save time and make our results comparable with other works, we will first select and utilize some well-documented and highly reputable datasets in our research. There are two types of sound used in our research: the human speech sound and the environmental sound. From the rich online datasets of human speech sound, we have chosen the following three:

1. The *TIMIT* [77] corpus of reading speech is an early recorded (in the year 1993) database which has been widely used in many different works. It contains broadband recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences.

2. The *Librispeech* [78] is a corpus derived from reading English audiobooks, which was published in 2015 by Vassil Panayotov and Daniel Povey. The data contains approximately 1000 hours of data, which is more favourable than TIMIT, especially for deep learning models.

3. The *Libricount* [79] is a dataset created by mixing the data of Librispeech into overlapping speech sounds. This dataset is to simulate the scenario of a cocktail party, which is mainly used for people counting rather than content recognition. Being

All of the listed datasets are carefully segmented and aligned, meaning that the start and end of each word's syllable are marked, normally at milliseconds precision. Regarding privacy issues, our research has virtually very little worries because we only analyze the voice but not the speech content, i.e. our research is text-independent. Although any language can fit our research, we only choose English speaking datasets for simplicity.

As for environmental sound datasets, fewer public datasets are available since this research field is not as popular as speech recognition. Moreover, the environmental sound is quite broad in categories, making it hard to find datasets specially designed for human activity detection purposes. Therefore, we do not only focus on the packaged research datasets, but on the available online resources that contain the data we needed. There are many websites that collect extensive audio recordings under broad categories, which are mainly uploaded by volunteers. From these websites, people can find and download the wanted data through the hashtags on the audio clips.

We mainly use the following online audio data for the activity recognition application:

1. The *TUT16* [80] is a *published dataset* for environmental sound research, consisting of binaural recordings from 15 different acoustic environments. A subset of this database contains annotations for individual sound events, specifically created for sound event detection of residential areas and home environments.

2. *freesound.org* [81] is a *website* with a huge collaborative database of real-life audio recordings. This database is under Creative Commons license and allows the audios to be used for research purposes.

3. *freesfx.co.uk* [82] is a *website* with a database for all kinds of sound effects, including sounds of everyday items like doorbells, car horns, or telephone rings, etc. This database can be used for both research and commercial purpose.

There are both pros and cons of using the website-databases. On the positive side, the data include many different categories of sound related to human activities. Whatsmore, the audio clips are recorded by different volunteers from different environments so that they are sufficiently diverse and can make the trained model more general. On the negative side, the format and length of these audio clips are not uniform so that we need to preprocess them carefully. Another big issue is that there is no easy way to download these data, since we have no direct access through their database.

The performance of machine learning models, especially deep learning models, largely relies on extensive and high-quality data samples. In total, we need to download thousands of samples for the training and test of machine learning models. In order to efficiently download a huge amount of data, we used a web-crawling technique to save the repetitive work of clicking download buttons manually.

Web crawling techniques (also known as spidering) use programs or automated scripts to browse the World Wide Web in a methodical, automated manner. It is most widely used by search engines, which regularly and densely browse a large amount of websites on the internet to provide up-to-date data. After browsing a website, crawlers normally gather specific types of information from Web pages, such as harvesting e-mail addresses or telephone numbers (sometimes for spam purposes though). Some web crawlers such as search engines even cache a whole copy of the visited pages to provide fast searches through indexing the downloaded pages directly. On a smaller scale and a slower pace, crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code.

Figure 3.1 demonstrates the basic idea of how we use web-crawling. Most modern websites such as 'www.freesound.org' support quite a flexible way of indexing their data. The first indexing method is through the search box, which allows one to search for the audios with a title that contains the typed text. The second indexing method is through the pre-labelled tags on the audios, where one audio can have multiple tags. In either way, the filtered results will be listed on multiple pages, where each page shows the detailed description and the download links of 10∼20 audios. By clicking the download link of an audio sample, one can then download it into his local disk. Since the listed data is strictly structured, a web-crawler can easily traverse the searched results with simple logic. In practice, we use 'pyscraper', a python-based framework to implement the web-crawler. Using a framework has many advantages over directly programming

with URL libraries, e.g. it schedules all the requests and responses asynchronously and automatically bypass the duplicated download links. This framework is built based on pure URL requests and HTML parsing, and no browser-GUI or rendering is needed. These characters make it fast in execution time, and the downside is that it is hard to simulate button clicks or handle dynamic HTML pages.

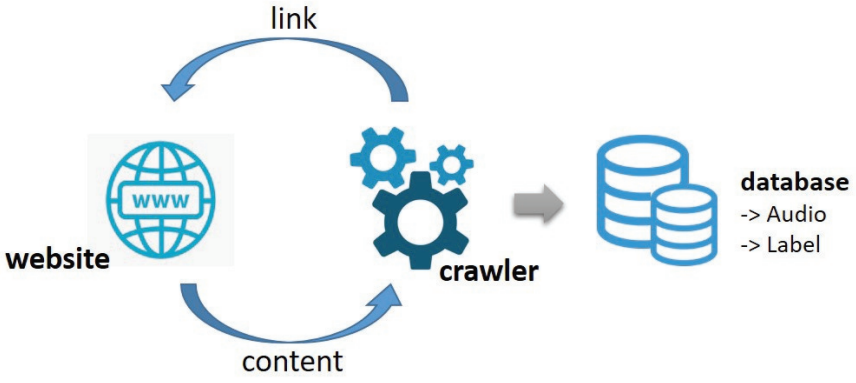


FIGURE 3.1: Automation-scripts can greatly help with data collection

Using 'freesound.org' as an example, our web-crawler script consists of the following steps:

1. The script should first login the website, as this website only grants the download permission for logged in users. With the cookies taking care of by the framework, one can easily implement the login function with the framework's API.

2. Find and jump to the corresponding URL address of 'browsing' the audios with the desired tags. For example, to browse audios with the 'walk' tag, one can jump to the URL address: 'https://freesound.org/search/?g=1&q=&f=tag:%22walk%22'.

3. With the returned results shown in multiple pages, we next figure out the URL address to each page. For example, to browse the results in page 3, we can simply browse: 'https://freesound.org/search/?q=&g=1&f=tag%3A%22walk%22&page=3'.

4. The script then traverses through all the pages and download all the audios within the download links. All the downloaded audios are saved according to the tags.

5. To get more audio samples, we can repeat steps 2-4 with different tags that correspond to the same sound event. For example, 'walk' and 'footsteps' both correspond to the walking sound. The audios with both tags would not be downloaded repeatedly as the framework can remember and automatically skip the duplicated addresses.

3.1.2 The simulation data

We mainly use simulation for creating complex and overlapping sound events in crowded environments, which are blended from the single-event recordings. Compared to single events, it is much more difficult to collect and label sound events in a crowded environment. Firstly, to label the ground truth we need to use surveillance cameras recording everything, which has privacy issues in a public building. Secondly, even with the video recordings, it is cumbersome and complicated to extract the exact duration of each sound event precisely, especially in a room with many people.

Therefore, it is more feasible and accurate to use simulation data for sound in crowded environments. In order to blend two sound events into an overlapping sound event, one may think that simply adding the two signals temporally is feasible. However, sound events created like this does not perfectly resemble the sound from a real indoor environment. Many factors need to be considered for the indoor sound simulation, such as sound attenuation, white noise, reflection and multi-path, etc. In our research, we use the simulation tool *pyroomacoustics* to simulate the indoor sound. Pyroomacoustics is a simulation tool for the indoor acoustic environments, which can simulate the sounds people hear in complex environments. Users can customize the room size, place sound events in the room, and then obtain the mixed sound stream heard at any location. Both 2D and 3D rooms are supported in this package, where we only simulate the 2D space for simplicity. An example of our data simulation is shown in Figure 3.2 where multiple walking, talking and bell sounds are heard by a microphone in the room centre.

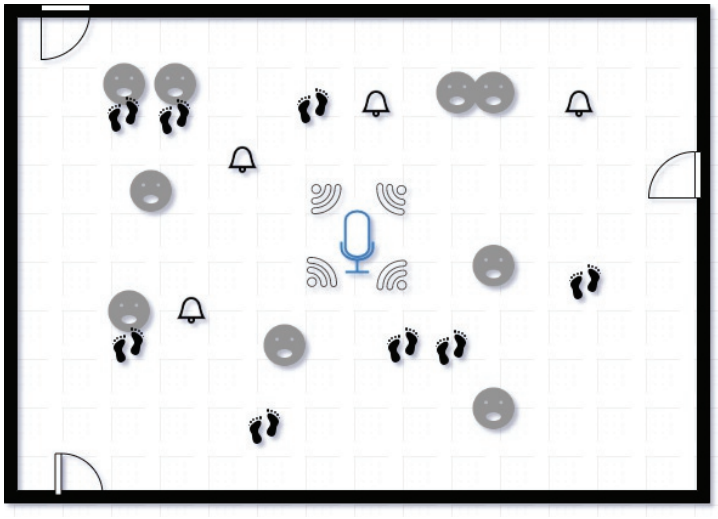


FIGURE 3.2: An example of the data simulation by tool 'pyroomacoustics'

Table 3.1: The overview of our audio dataset

	speech			environmental			
name	TIMIT	Librispeech	Libricount	TUT 16	freesound.org	freesfx.co.uk	pyroomacoustics
size	∼250 min	∼1000 min	∼200 min	∼50 min	∼350 min	∼100 min	∼800 min
type	single	single	overlapping	single	single	single	overlapping
source	public	public	public	public	website	website	simulation

An overview of our datasets with the basic information is shown in Table 3.1.

3.2 The common pre-processing steps

Data preprocessing in general is a technique that involves transforming raw data into structured format and filtering out the errors. It serves to prepare a correct, unified and easy to use dataset for further analysis. Audio recordings in daily life can contain a lot of noise, and have different recording formats, which can create troubles to data analysis models. In this section, we will mainly introduce the common preprocessing steps for all the datasets, which include format unification, noise filtering and HDF5 format conversion. Other uncommon preprocessing steps such as voice stripping and segmentation will be described later in each chapter.

3.2.1 Format Unification

The data format of the audio from different datasets can vary a lot and it first needs to be unified for later processing. Data format unification is a basic need for machine learning models and many other data analytic methods. Without data unification, two similar data samples processed by the same pipeline could get very different results.

The audio format here mainly includes the following parts:

1. File format: We convert all the audio files of our datasets to *.wav* (Waveform Audio File) format. *.wav* typically uses linear pulse-code modulation (LPCM) bitstream encoding and is the raw and uncompressed audio format. Although being large, a *.wav* file retains all the samples of an audio signal, which makes the data processing models more accurate and also easier to interpret. We also set the LPCM encoding format to 16 bits per sample, which means each data sample is represented by two bytes.

2. Audio Channel: All the audios in the public datasets we use are mono-channel (single channel) sound. This is reasonable since the sound from a single channel is enough to tell what is going on. However, some of the website audios are recorded with stereo-channels (two channels), which makes the data non-unified. In the environmental sound, the difference between the two channels is very small, therefore we always discard the right channel and keep only the left channel sound.

3. Sampling Rate: Sampling rate is the number of samples of audio carried per second, measured in Hz or kHz (one kHz being 1024 Hz). A larger sampling rate means

more details in an audio clip of the same length. Sampling rate also determines the maximum audio frequency. Theoretically, the maximum frequency that can be reproduced by the sampled data is half the sampling rate. We use two different sampling rates in the datasets according to the sound type: the environmental sounds are sampled at 44KHZ, while the speech sounds are sampled at 20KHZ. This is because speech sound is usually less than 8KHZ, while environmental sound (such as door slam or ring bell) can reach very high-frequency bands. It is therefore not necessary and a waste of resources to use the same sampling rate for both data.

3.2.2 Convert Raw audio to HDF5

In this section, we will read all the raw audio files and convert them into several large HDF5 (Hierarchical Data Format 5) files. This process is mainly used to speed up audio data reading during model training, and is not needed in test and deployment phases. In our research, we will use many machine learning models (mostly deep learning) to process the data, and this method saves a large amount of training time.

Machine learning models, especially deep learning models, can take a very long time to train until they get good results. During the model training process, all data samples are repeatedly fed into the model as inputs, where each repetition is normally called an episode. This means a model which trains 1000 episodes would also read the entire dataset 1000 times. With a relatively small amount of data, data access speed is not an issue since one can loads all the data into memory before training starts. However, audio files are very large and cannot be fully loaded into memory, so we need a more efficient method instead of reading raw audio data each time.

We apply the same method on the entire dataset: firstly read all the audio files into arrays, and then store all the indexed arrays in one single HDF5 file. Among all the available technologies including caching memory pool, or other databases, we choose this method since it is both fast and flexible which perfectly meets our needs. The HDF5 is an open source file format that supports large, complex, heterogeneous data storage and file-system alike data index. The HDF5 format can be thought of as a file system contained and described within one single file, where one can access different data in it just liking accessing different files on a computer. For example, we first read the wave file storing at "*pathA/01.wav*" into an array Arr_01, and then store this array into the "*pathB/all_audios.hdf5*", with the data index of the same name: "*pathA/01.wav*". With proper wrappings, one can read the Arr_01 in HDF5 with the original file name: "*pathA/01.wav*", just like reading the original audio file. This file-system alike index

undoubtedly makes programming much more flexible and easier than using SQL-based languages.

In practice, we use python for programming and the h5py package as the HDF5 library. The access speed is faster than raw audio reading for two reasons: Firstly, the audio files are already read into python arrays so that it saves the .wav format parsing time. Secondly, HDF5 itself is very fast in indexing and reading large arrays. HDF5 uses a "chunked" on-disk data format, which makes reading an arbitrary slice cost very similar time for the best and worst case, while which takes much longer in the worst case when reading from the on-disk data. An example is shown in Figure 3.3 to better understand the process.

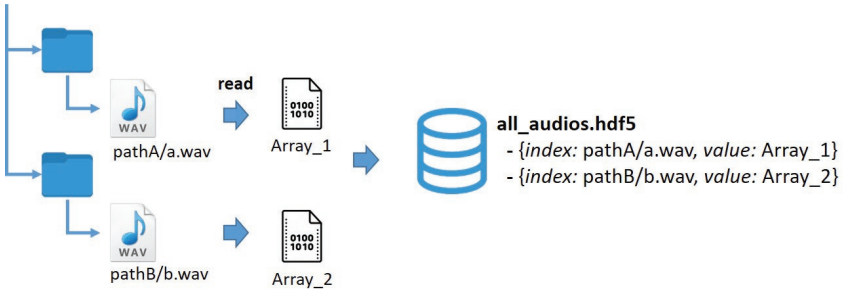


FIGURE 3.3: Raw audios are converted to a single HDF5 file

3.2.3 Noise filtering

In everyday life, one of the biggest challenges of sound-based applications is noise. Compared to the two-dimensional images we see, noise is more pervasive in the one-dimensional sound we hear and is more easily to bring down our perceptual ability. Filtering the noise, therefore, seems to be an important step to improve the audio data quality. However, sound noise has many types and should be treated accordingly. Some related even suggest not to filter noise in sounds, since models trained from noisy data are also less susceptible to the noises in real-life environments.

People from different domains have different definitions for sound noise. From human hearings perspective, noise is the unwanted sound judged to be unpleasant, loud or disruptive to hearing. In audio engineering, noise can refer to the unwanted residual electronic signal that gives rise to acoustic noise, heard as a hiss or a buzz. While in our research, noise refers to any sound apart from the wanted information (i.e. the events), be it the white noise or the buzz from the recording device.

We categorize the noise in our data into two types:

1. *Background Noise* is the sum of the sounds from the environment which is different from the events we monitor. In indoor environments, it can come from air conditioners, window vibration, water waves or the traffic noise from nearby streets. Background noise in everyday life is unavoidable and it normally occurs continuously in a similar pattern. Techniques such as *noise gate* are able to find this pattern and remove them from the signal.

   When the environment becomes more crowded, it becomes very similar to Gaussian noise, which is the combination of many different small noises and none is dominant. One can basically find no pattern in the Gaussian noise so that it is also very hard to remove them. On the other hand, in many of our experiments, Gaussian noise basically makes very little difference in the results.

2. *Recording Device Noise* is the noise generated from the recording devices, which is more especially obvious from low quality microphones. It can come from interference from the inside circuits or wireless signal around the microphone. This noise exists continuously and mainly concentrates in the low-frequency bands, which is normally called buzz or hum.

   In the public datasets such as TUT and TIMIT, the recording devices are normally very good and there is very little buzz sound. In the online dataset, the quality of audio data is quite different, some of which may have very strong buzz noise.

To get a better quality, we use a noise-profile based technique to cancel both types of noises in the audios. To use this technique, one needs to first select a short and silent moment from an audio stream to build the profile of the noise. Subsequently, this profile is applied to the entire stream to reduce non-signal noise sound. This two-step process of using the noise-profile algorithm is shown in Figure 3.4.

This technique is also called spectral noise gating, as it 'opens' at the signal frequency bands and 'closes' at the noise frequency bands. Spectral noise gating first calculates the frequency spectrum of the background noise in the selected snippet. That forms a fingerprint of the static background noise in the sound file. The algorithm compares the noise spectrogram to those of each short segment of the sound file. Any frequency bands that aren't sufficiently louder than their average levels in the fingerprint are reduced in volume. In this way, the major frequency bands of a door slam sound or human voices are preserved, but hiss, hum and other steady noises can be minimized. This technique works best when the background noise tends to be static and the selected noise profile can largely represent the noise pattern. Undoubtedly, noise can only be reduced but not completely cancelled, and a heavy degree of noise reduction also brings in signal distortion. Nevertheless, noise reduction is not the focus of our research either. The

goal of noise filtering is to control the noise at a moderate and equal level for all data samples, so that the processing models would not be less affected. As noise filtering is not the focus of this research, we just need a moderate and practical solution and do not further investigate the theory behind.

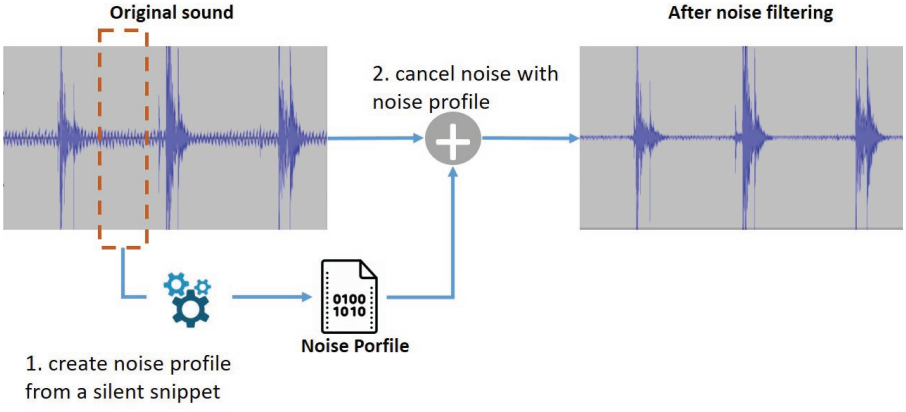


FIGURE 3.4: A demonstration of using noise-profile to cancel noise

Chapter 4

Sound-based activity recognition in non-noisy environments

In this chapter, we will present the principles and solutions for detecting and recognizing human activities in a quiet environment. Here quietness means there are not many human activities so that most sound events can be clearly identified and there is very little overlapping in sounds, such as in many small offices or residential houses. To tackle the problem, two lightweight methods are introduced separately, each based on a classic and neural network machine learning model. The main content of this chapter is published in PETRA 2019 [83].

4.1 The fundamentals of sound-based activity recognition

Human activities are always accompanied by some kind of sound such as walking, talking, door slamming, etc. It is therefore easy for humans to be aware of the things going on around just through listening to the interesting sound events. Although automatically identifying and distinguishing sound events is a trivial task for humans, computer systems have not been able to achieve comparable accuracy until recently.

Classification of human activities based on sound falls under the category of *'Environmental Sound Recognition'* (ESR). ESR aims to automatically detect and identify audio events from captured audio signals [84]. In recent years, most ESR models use machine learning models such as SVM, HMM or Deep learning for the sound classification problem as they are significantly better than methods that merely use signal processing. As aforementioned in Chapter 2, ESR models were mostly spawned by offline applications such as movie-type classification [37] or video-context classification [85]. These

proposed audio-processing methods would focus primarily on precision and very little on the memory or computing cost. As a result, these models are not quite suitable for resource-constraint IoT devices in smart building applications. Apart from the low-cost requirement, another issue with the sound-based systems in smart buildings is privacy, as no one wants to be constantly tapped. In this chapter, our work will fill in the gaps between the current models and the needed solutions for smart building applications.

A general real-time system of sound event classification is shown in Figure 4.1. As is shown, we first preprocess the data which would segment the continuous audio stream into discrete segments or events. Audio features are then extracted from each event to classify the event into predefined classes. Regarding the models, we mainly investigate the low-cost models for meeting the needs of smart buildings. We will introduce the classic and deep learning based methods separately later in Chapters 4.2 and 4.3. This separation is due to the significant differences in the design and principles of the two methods.

Although the specific features and models differ a lot for different methods, our preprocessing steps in the pipeline are similar. In this work, preprocessing mainly consists of two steps: voice stripping and audio segmentation, which will be discussed in the next two subsections.

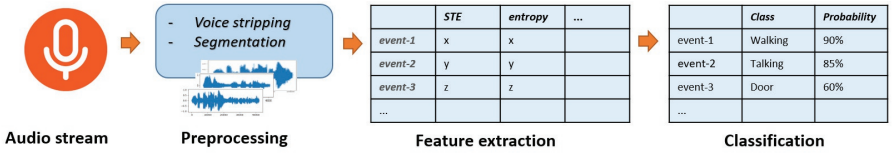


FIGURE 4.1: The real-time audio events classification system

4.1.1 Preprocessing: voice stripping

Audio data is of high privacy concern so that needs to be handled carefully. It is also one of the significant advantages of edge computing, i.e. the entire data processing pipeline is placed in the edge device. A weaker protection against potential malicious network attacks is the use of data compression, which means that the compressed data instead of the raw data is being transmitted to the server for processing [86]. Whatsmore, these compressed data can be intentionally distorted in order to further increase the security[87].

However, these protections all aim at the network attacks and are not capable of protecting the edge devices themselves. What if the edge devices are hacked and modified to upload the raw audio data somewhere? Even if the hackers are not considered, would

employees worry about these microphones being used as surveillance tools by the administrator, who deliberately leak the data to the big boss?

In this sense, a even better approach is to protect privacy from the very beginning, i.e. not to record people's conversations. As human conversations are one of the most critical privacy concerns in indoor environments, we propose to strip them from the beginning. In our research, the human voice stripping is implemented by a software band-stop filter while in a real world implementation this can be implemented by acoustic sensor physically, so that the privacy is protected at the device layer.

A typical band-stop filter can achieve this function, which let pass only the bands from zero up to its lower cut-off frequency F_{low} and the bands above its upper cut-off frequency F_{high}. This results in a rejection of frequencies in the band F_{low}:F_{high}. In our case F_{low} = 300Hz and F_{high} = 3KHZ, this range is often referred to as the voice bands [88], where most people's voice are concentrated in. In a practical experiment, we selected several audio samples in the dataset to see how the speech sounds after removing the speech band. By filtering out this frequency band, even if there are residual bands in the signal, the speech content becomes highly blurred and unrecognizable.

Figure 4.2 and 4.3 shows one speech sound and one door sound before and after voice-bands truncation in time and frequency domain. Obviously stripping the voice bands could deprive a lot of information of both sound events, resulting in a more challenging classification problem. On the other hand, although a speech sound can lose a lot of information from voice stripping, there is still energy in other frequency bands, making the classification feasible.

4.1.2 Preprocessing: segmentation

Audio stream is large and continuous, which is very different from the discrete data generated by common IoT sensors such as PIR, motion sensors or vibration sensors. For example, the data with 44kHZ sampling rate and 16-bit LPCM would generate 88k bytes each second. An early step of sound event recognition, therefore, is to segment the continuous audio stream into small discrete segments and analyze them sequentially.

There are several benefits of applying the commonly used segmentation technique. Firstly, a quiet environment most of the time has only background sound which is unnecessary to be processed. The segments that are detected as background noise can be directly discarded at an early stage to save calculating resources. Secondly, apart from the 'detection' purpose, a large amount of continuous data is also hard for data processing models to handle. A typical machine model takes a small and equal size of data

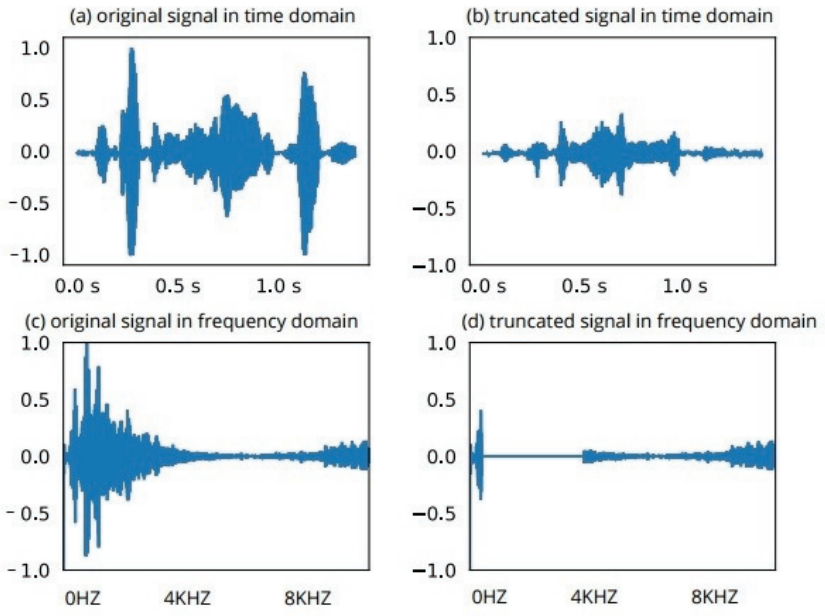


FIGURE 4.2: Voice bands truncation of a speech sound

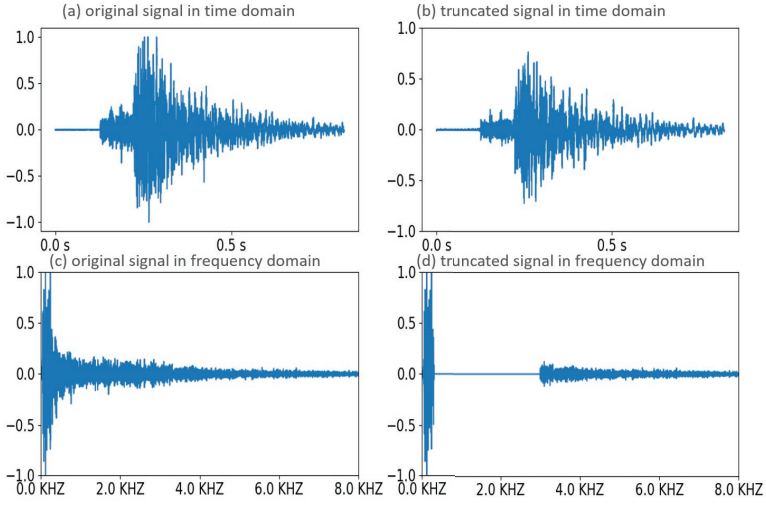


FIGURE 4.3: Voice bands truncation of a door-slam sound

points as the input and predict the results according to the pattern inside. Too much input data once a time can make the model hard to generalize and over complicated.

There are normally two types of segmentation algorithms based on the segment length, i.e. the fixed-length segmentation and flexible length segmentation. The fixed-length segmentation technique simply cuts the audio stream into end to end connected segments of the same length. The segments are then determined to be either 'active' events or 'silence' with power thresholding or other metrics. This segment length is an empirical value that normally ranges from $2 \sim 5$ seconds depending on the specific sound events in different researches.

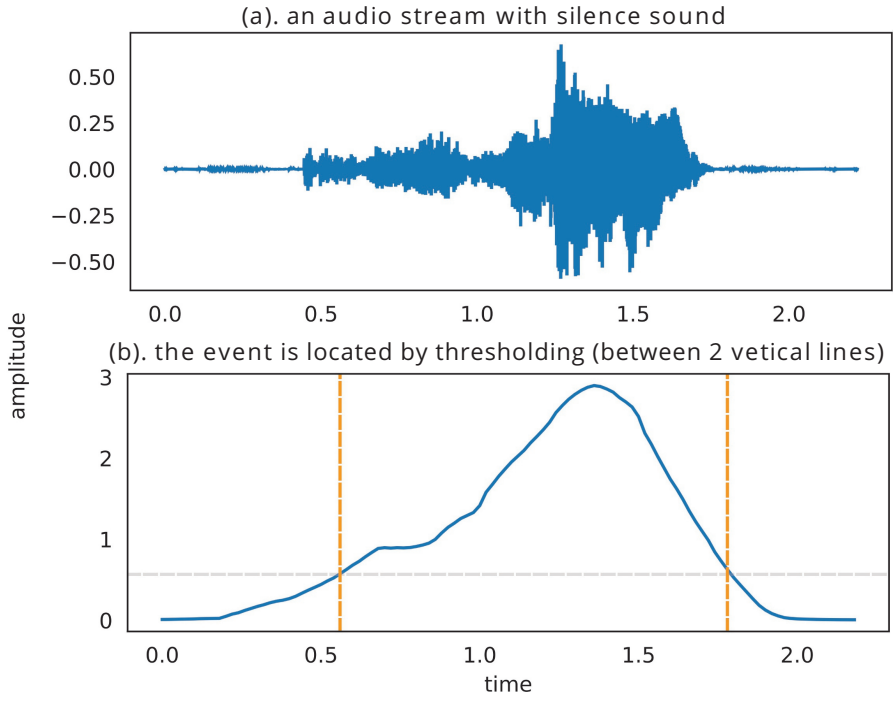


FIGURE 4.4: An example of flexible-length audio segmentation

The flexible-length segmentation technique is more complicated which needs to find the start and end of a sound event precisely. Our algorithm of the flexible segmentation works as follows:

1. The audio stream is smoothed by 'moving average' algorithm in time domain [89]. The smoothed stream is then cut into fixed short frames (20ms) while the power is calculated for each frame.

2. The frames with power higher than a threshold are labelled as 'active'. The threshold can be a preset static value or dynamic adjusting.

3. Adjacent 'active' frames are combined to form an event.

4. Events shorter than a given duration are dropped, while the long frames are truncated such that the events are between 1 to 3s. The reason to choose this duration range is from practical experience, as humans can identify sound segments with such length quite well.

Figure 4.4 shows an example of the flexible-length segmentation results.

Both fixed-length and flexible segmentation techniques have their pros and cons and should be selected based on the scenarios and data-processing models. The flexible-length segmentation detects and extracts the precise duration of an audio event. This makes it theoretically better for the activity recognition task than the fixed-length method, which often split a single audio event into halves or falsely merge two different events. However, there are also problems that make the flexible-length segments less favourable than the simple fixed-length strategy. One of the challenges is that the boundary of an event is hard to locate when the environment becomes noisy especially when multiple events happen simultaneously. Another problem of flexible-length segmentation lies in the subsequent data processing models. Most commonly used machine learning models such as SVM, Decision Tree or Deep Neural Network only accept fixed-length inputs. To adapt to these models, we either need to pad or truncate the segments to a unified length or extract some statistic features which then become irrelevant to the original data length. Both of these two segmentation strategies are used in our following chapters for different reasons and will be discussed accordingly. A comparison of flexible and fixed-length segmentation algorithms on the same signal is shown in Figure 4.5, which shows that the flexible-length algorithm can locate events more accurately.

In brief, we will only use flexible-length segmentation in Section 4.2 and will use fixed-length segmentation for all other models. This is mainly because the method in Section 4.2 uses the statistic features whose length is independent of the length of audio data. In this case, it is more important to accurately extract the event duration than to uniform the data length. By contrast, for deep learning models that are able to handle high-dimensional inputs, it is better to input frame-based features so that the data length will be consistent.

4.2 With Classic models

In this section, we will present the audio events classification solutions based on the classic models, i.e. the non-deep-learning model. Although consensus has been reached that deep learning models can generally achieve better classification accuracy, classic models are still widely used because they need much less memory and calculation power.

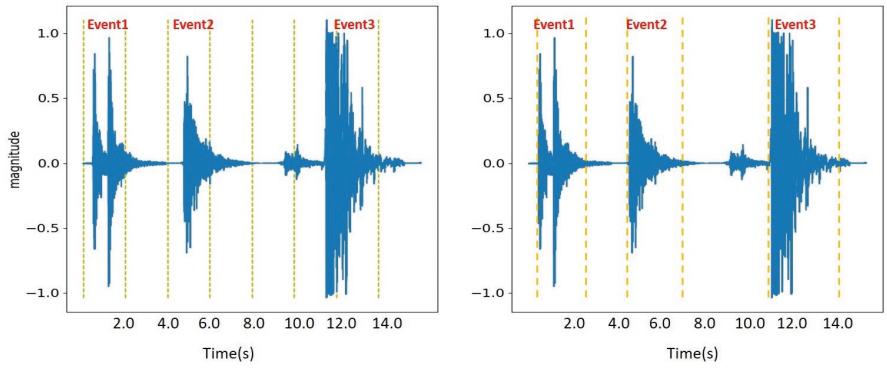


FIGURE 4.5: A comparison of the two segmentation algorithms (left:fixed-length, right:flexible-length)

This lightweight function is especially suitable for IoT devices because it means cheaper prices and longer battery life.

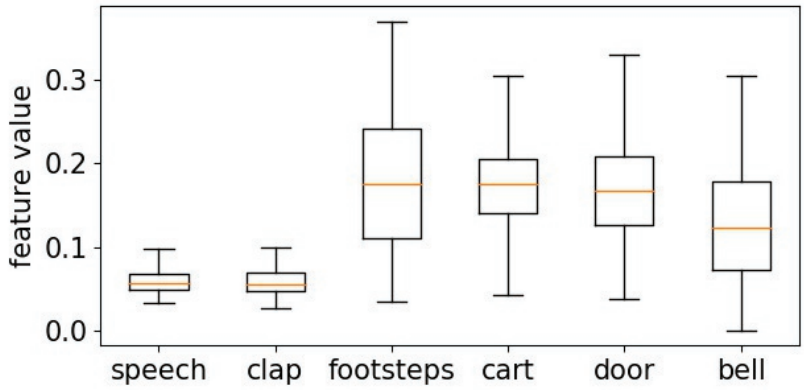


FIGURE 4.6: Feature 'frequency-spread' on frame basis

4.2.1 The Features

Audio data is a continuous stream of high sampling rate information. This stream of continuous data can be transformed into a reduced set of features, which contain the most important and the most relevant information for the classification task.

A single feature from a single domain only represents limited information, thus it will be hard to classify. However, by combining multiple features, the class characterization becomes conspicuous[90]. In this work, we first cut the audio stream into smaller frames of fixed length and partly overlapped (e.g. *20ms* frame length with *10ms* overlap). We

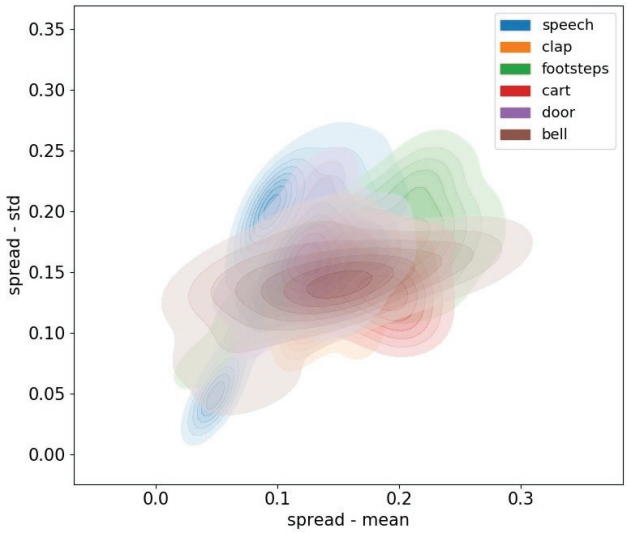


FIGURE 4.7: Feature 'frequency-spread' statistics on event basis

then calculate the statistics (mean and variance) of all frame features as the representation of the whole event. There are several benefits for the aggregation of short frames: firstly the features extracted from any event have the same length, i.e. independent of the event duration, secondly the statistics stand for global representation which treats the event as a whole. Figure 4.8 shows the feature extraction flow.

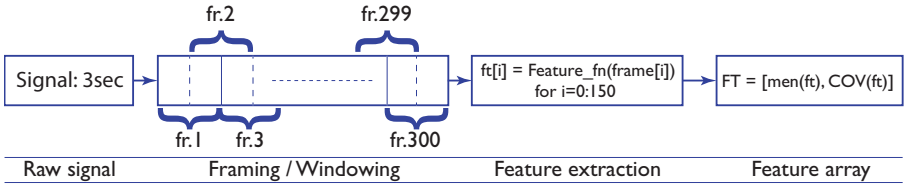


FIGURE 4.8: The statistic audio-features extraction

Figure 4.6 and 4.7 show the box-plot of per-frame value and the scatter-plot of statistic values of features: 'spectral-spread'. At the first glance, basically, no conclusion can be drawn from a single frame feature value, but the statistic feature of an event shows a much clearer pattern and would be easier to classify.

While there are numerous audio features, we mainly select features that prove to be highly relevant in audio recognition tasks. We also choose features from different domains and with different characters since machine learning works better by combining features with low correlation. The features used in our paper are listed in Table 4.1, including temporal-domain, frequency-domain and spectrogram-based features [35].

A spectrogram is the spectrum of frequencies of a signal as it varies with time. The visualization of the spectrogram is a graph with two geometric dimensions: x-axis represents time, y-axis represents frequency and the color intensity implies the amplitude at that specific coordinate of time and frequency. It is commonly derived from the STFT transformation, which cuts an audio stream into trunks of equal length and then takes the Fourier transformation over each short trunk. *Mel spectrogram* is another commonly used audio spectrogram, which is the product of the Mel-filterbanks and the STFT spectrogram [35]. Although the STFT do not compress the data, it provides a good base to many good audio features such as MFCC and LPCC. To provide an intuitive impression, we choose a representative sample from each class and plot the signal and some spectrogram features, as shown in Figure 4.9.

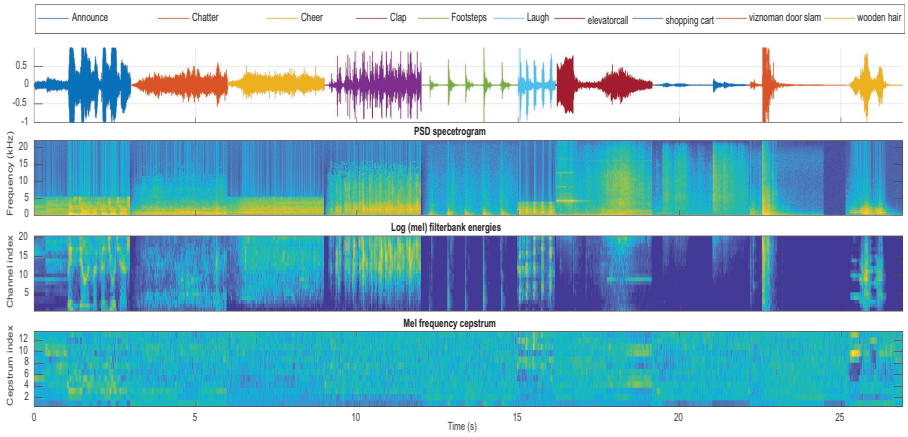


FIGURE 4.9: An example of spectrogram features from each sound event class

A brief description of our selected features is as follows:

Time Domain Features

(i) **Short-time energy** (STE) is a widely used feature in audio analysis which describes the energy of signal, calculated by mean-square of signal per frame [35].

(ii) **Zero crossing rate** (ZCR) is the rate at which the signal changes from positive to negative or vice versa. This feature is one of the simplest and widely used in speech recognition, which characterizes the dominant frequency of signal[91].

(iii) **Temporal entropy** (TE) is the entropy of temporal domain per frame, which characterizes the dispersal of acoustic energy [92].

Frequency Domain Features

(iv) **Spectral centroid** is calculated as the weighted mean of the frequencies while magnitudes are the weights. It indicates where the "center of mass" of the spectrum is located. Sometimes the median is used for "center" rather than mean [35].

(v) **Spectral spread** is the magnitude-weighted average of the differences between the spectral components and the spectral centroid, together they describe how disperse and wide the frequency bands are [35].

(vi) **Spectral entropy** is calculated as the entropy of spectrum it reflects the flatness but spectrum [92].

(vii) **Spectral flux** also describes the flatness in spectral domain, but across frames. It calculates as the 2-norm Euclidean distance between the power spectrum of adjacent frames [92].

(viii) **Spectral rolloff** represents the point where N% power is concentrated below that frequency. Spectral rolloff is extensively used in music information retrieval [35].

Spectrogram Features

(ix) **Mel-frequency cepstral coefficients (MFCC)** is the cepstral representation of Mel-Frequency. Compared to original linear frequency bands, Mel-Frequency is equally spaced on the Mel scale, which approximates the human auditory system's response. MFCC describes the spectral envelope and is commonly used in speech recognition [38]. MFCC is commonly derived with the following steps [93]:

- Take the Fourier transform of (a windowed excerpt of) a signal.
- Map the powers of the spectrum obtained above onto the Mel scale, using triangular overlapping windows.
- Take the logs of the powers at each of the Mel frequencies.
- Take the discrete cosine transform of the list of Mel log powers as if it were a signal.
- The value of MFCC is the amplitude of the resulting spectrum.

(x) **Linear Predictive Coding (LPC)** is an auto-regression model in which a predictor estimates a sample by linear combination of previous sample values (4.1), s is the signal sequence and a_1 to a_p are the coefficients. LPC can be used in audio compression and speech recognition to represent the spectral envelope. However, LPC is prone to a disturbance where small changes in LPC value would cause large deviation in spectrum [35].

$$s[n] \approx a_1 * s[n-1] + a_2 * s[n-2] + ... + a_p * s[n-p] \tag{4.1}$$

TABLE 4.1: The feature list used by our classic model

Domain transformation			
Nr.	Temporal	Frequency	Spectrogram
1	ZCE	centroid	MFCC
2	STE	medium	LPCC
3	TE	spread	LSF
4		entropy	Chromogram
5		flux	
6		rollof	

(xi) **Linear predictive cepstral coefficients (LPCC)** is the cepstral representation of LPC [35].

(xii) **Line Spectrum Frequencies (LSF)** is the roots of 2 polynomials decomposed from the LPC polynomial [35]. Both LPCC and LSF are derived from LPC, they are alternative representation of LPC, thus containing the same amount of information. However for LPC, a small disturbance from input may incur a big difference to output, while LPCC and LSF are more robust in this aspect, which make them better for the classification task.

(xiii) **Chromagram** is a well-established tool for processing and analyzing music data that capture harmonic and pitch characters from the sound. Apart from music applications, Chroma features are also powerful mid-level feature representations in content-based audio retrieval or audio matching [35].

4.2.2 The classification models

After features have been extracted, a classifier should be learned so that new coming data can be classified. This classification function which is also called representation, has different forms for each model, e.g. a search tree in decision tree or a hyperplane in SVM. Learning is therefore a process performed by searching through the representation space to find the best hypothesis that better fits the solution.

Classification algorithms can be categorized into two types [94]:

I. *stateless algorithms*, in which events to be classified are treated as unrelated to each other. Most machine learning models such as GMM, SVM, Decision Tree and the common neural network models are all stateless models.

II. *stateful algorithms*, in which the algorithms treat the events as related to each other and put them into a context while updating the memory. A stateful model works best in

the scenario where the output is not only decided by current input, but also by previous input in the timeline (state). Stateful models such as HMM and RNN are most famous for language modelling.

In environmental sound processing, stateless models are widely used in sound events classification, while stateful models are preferred by context classification tasks. In our task, we think there is no need to remember the past information, since it is enough for human beings to tell what is happening when a sound event is heard, even without much context.

While there might be a plethora of machine learning algorithm variations with different types of representation space, the effectiveness for different scenarios can only empirically be confirmed. Hence, we select several commonly used algorithms to conduct an empirical comparison and find the best candidate for the problem. The chosen stateful models in this chapter are listed below:

1. ***Decision tree*** is a non-parametric supervised learning method used for both classification and regression [90]. This model predicts the value of a target variable by learning simple if-then rules (i.e. decisions) inferred from the data features.

   Decision tree is simple to understand and interpret since the trees can be visualized. A decision tree classification model also supports multi-classes outputs. On the other hand, decision tree can be unstable that small variations in the data might generate a completely different tree. This is partly because decision tree is normally trained through greedy search, through which only local optimum is found. As is built by merely if-then rules, decision tree can also be hard to learn and express many complex concepts such as XOR, parity or multiplexer problems. Figure 4.10 shows a simple example of the decision tree used to decide whether to play outside, given all the inputs.

2. ***Random Forest*** is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [90]. The model built from several estimators independently and then to average the predictions is called ensemble model. On average, the combined estimator is usually better than any of the single base estimators because its variance is reduced. A decision tree is normally created through greedy search and typically exhibit high variance and tend to overfit. By taking an average of those predictions, random forests achieve a reduced variance by combining diverse trees. In practice, the variance reduction is often significant hence yielding an overall better model.

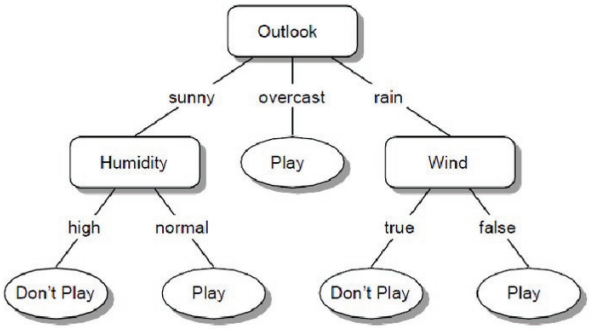


FIGURE 4.10: An example of decision tree

3. ***Mixed Gaussian*** model assumes that all the data samples are generated from a mixture of several different Gaussian distributions with unknown parameters (i.e. the mean and variance) [90]. 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. A Gaussian mixture model normally uses the expectation-maximization (EM) algorithm for training which maximizes the overall probability of all samples and all classes. As this algorithm maximizes only the likelihood, it will not bias the means towards zero, or bias the cluster sizes to have specific structures that might or might not apply. One major weak point of the Gaussian model is the setting of components numbers, which must be decided manually and through external cues.

4. ***Naive Bayes*** is a supervised learning algorithm based on applying Bayes distribution with the "naive" assumption of conditional independence between all the features given the class label [90]. Mathematically, if we have the data sample with the feature set: $\{x_1, x_2, ..., x_n\}$ and want to predict the class label: y, the problem equals to predicting the probability of:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

If we assume all the features x_i are independent, we will have:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Finally, the prediction of the class $\hat{y}$ is calculated by:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

In practise, we normally assume the distributions are all Gaussian, where:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ and μ can be estimated using maximum likelihood through training. Naive Bayes classifier is both fast and simple compared to many other sophisticated methods, which is because it decouples the joint distribution to many one dimensional distributions. Naive Bayes is a famous algorithm for document classification and spam filtering, where the assumption of features-independence applies.

5. ***SVM*** is an powerful classification model which is especially good for high dimensional features input. It may still work in cases where the feature dimension is greater than the number of samples. A support vector machine model builds one or multiple hyperplanes in a high dimensional space to separate the samples into classes [95]. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class, since in general the larger the margin the lower the generalization error of the classifier. These hyperplanes are actually generated by nearest training data points, which are also called the support vectors. A demonstration of the SVM margin and hyperplane is shown in Figure 4.11.

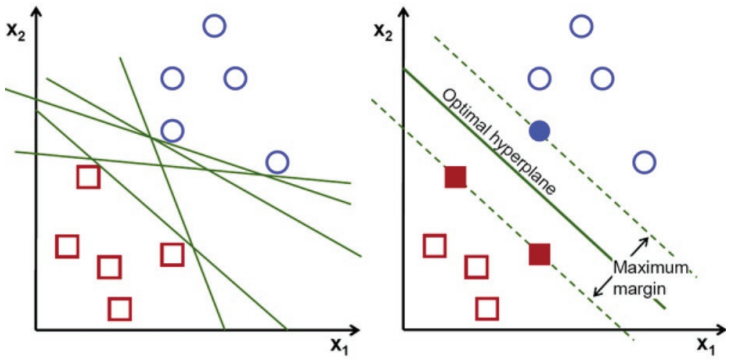


FIGURE 4.11: The SVM classifier explained (left: many possible hyperplanes. right: the hyperplane that maximize the margin)

Because the hyperplane only separates a space into two halves, a basic SVM algorithm can only output two classes. To support multiple classes outputs, we can use the SVM-SVC algorithm, which uses multiple "one-against-one" approaches for multi-class classification. For the SVM-SVC algorithm, if n is the number of

classes, then $n \times (n-1)/2$ binary classifiers are constructed and the final decision is the class that wins the majority of votes. Another important component of the SVM algorithm is its kernel function. A linear kernel function transforms the data into a different hyperspace where the samples can be better separated than the original space. The typical kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid. The most used type of kernel function is RBF, which return the inner product between two points, resembling a notion of similarity. This RBF kernel is widely favored since it can project data into very high-dimensional spaces with little computational costs. A RBF kernel function is defined as:

$$RBF(x, x') = e^{(-\gamma \|x - x'\|^2)}$$

, where x' is the support vector and γ is a coefficient. Intuitively, γ indicates how far the influence of a single training example reaches. If γ is very large, the influence will be very little for samples other than the support vector itself, so that the results can be easily overfitting.

4.3 With deep learning

With the rapid development of neural network technology in the field of image processing and speech recognition, many works have also applied different types of neural network models to this task. As mentioned in Chapter 2, these models can be based on full-connected, CNN or RNN frameworks. Generally, neural network models could outperform the classic models as long as there are enough training data. One reason is that classic models can easily overfit complex audio data so that the input features are normally highly compressed, resulting in a severe loss of information already.

As introduced in Section 2.1.1, many deep learning based models have been proposed for sound events recognition, most of them are overly complex for devices with limited resources. Many works simply borrow the frameworks from computer vision or speech recognition, and often require millions of weights. The high complexity not only limits the application scenarios, but also makes a model difficult to interpret or debug. To fill this gap, we propose a carefully designed CNN-based model that contains only two main hidden layers and thousands of weights. In addition to a concise framework, a shared weights strategy is also used in the output layer to further reduce weights and prevent overfitting.

The specific flow chart of our sound event classification model is demonstrated in Figure 4.12. The data flow is shown on the left and the processing functions are shown on the

right. Here we use the fixed-length segmentation algorithm, so the length of each input sound clip is identical, e.g. 2 seconds. In our recordings, the sound events longer than 2 seconds are truncated and the ones shorter than 2 seconds are padded with zeros. Even though the sound events are not perfectly extracted, they can still be well classified with the power of deep learning models.

Out of each input sample, we first extract the Mel-bands feature, which is a two-dimensional image-like feature. With this feature as input, the deep learning model then gives the probability of each event class in the output layer. Then, the sound event falls into the class with the highest probability. We choose this feature and model combination mainly because it is highly efficient in terms of model size and computational density. More details will be discussed in the following subsections.

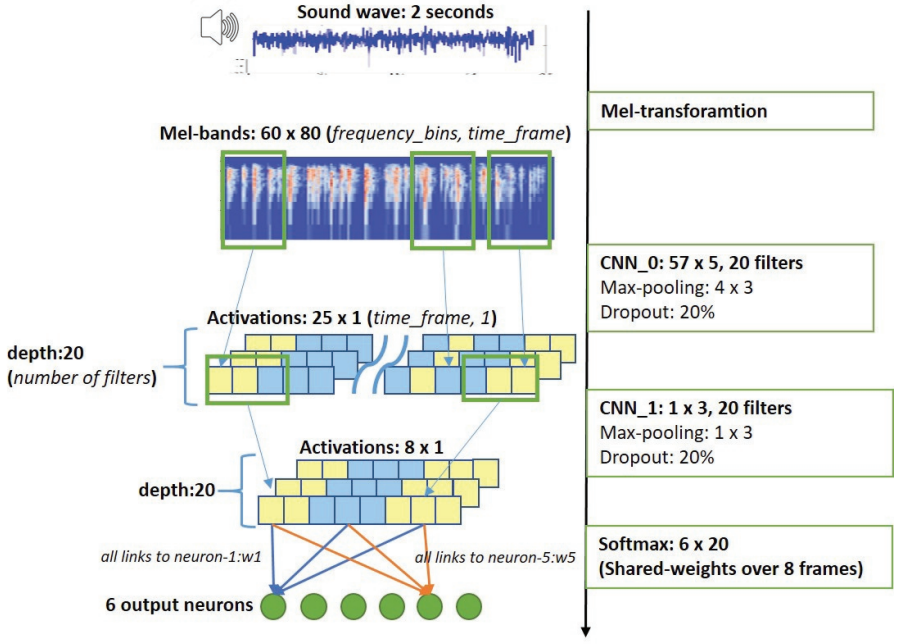


FIGURE 4.12: The CNN-based sound event classification model: CNN-L2
(left) - The neurons of each layer
(right) - The functions of each layer

4.3.1 The features

As is shown in Figure 4.12, we mainly extract the image-like Mel bands features from audio clips, which is a typical input of the CNN-based model. The *Mel bands* feature is the product of the Mel-filterbanks and the Short-time Fourier transform (STFT)

spectrogram. A spectrogram can be easily visualized as an image to understand how the frequencies of a signal varies with time. The x-axis and y-axis refer to the time-frame and frequency, the color is used to indicate the amplitude of the signal. In this figure, blue color refers to low amplitude while red color refers to high amplitude.

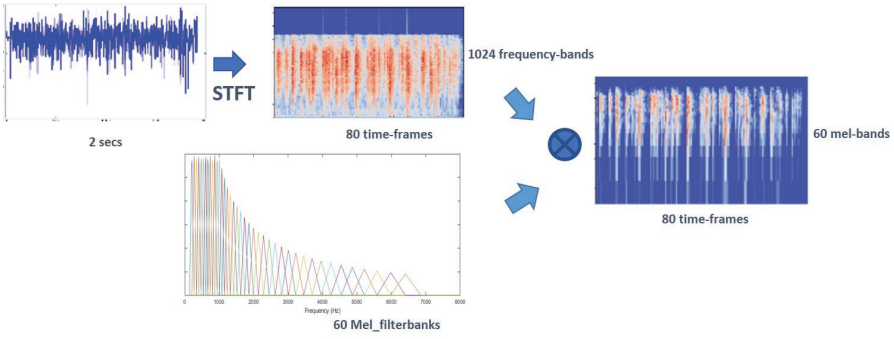


FIGURE 4.13: The Mel-bands feature extraction flow. The redder the color, the greater the amplitude.

Mathematically, the STFT algorithm first divides an audio stream into trunks of equal length, which normally overlap, and then computes the Fourier transform separately on each short trunk. Often, a window function ω (e.g. Hann window or Gaussian window) is applied on each trunk to smooth the signal. Let $x(n)$ be the data signal and let m be the trunk index. STFT of can be expressed as:

$$STFT\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n}$$

Where X is the discrete FFT function. The spectrogram is then the amplitude of the STFT result.

$$\text{spectrogram}(m,\omega) = |\text{STFT}(m,\omega)|^2$$

With the STFT bands, the Mel bands are just a simple transformation of it, which aims to make each band sounds equal in distance to human listeners. Firstly, the mel-scale frequency M is converted from the original frequency f by:

$$M = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

The final step to computing filter banks is applying triangular filters (e.g. 60 filters), on the Mel-scale frequency to the extract frequency bands.

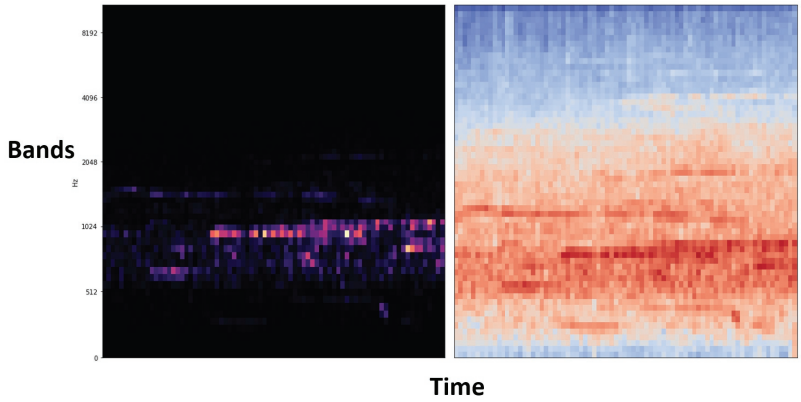


FIGURE 4.14: Mel(left) and Log-Mel(right) bands feature of the same sound event

In this work, we set the frame-length to $50ms$ and half overlapped and the Mel-bands size to 60, the output feature of a 2 seconds audio stream is then of size $60(bands) \times 80(frames)$. The method to extract Mel-bands features is shown in Figure 4.13. STFT bands and Mel bands are both commonly used features in audio processing. Out of the two features, we consider Mel bands as the better one for sound classification for two reasons: First, Mel bands is more condensed thus would help solve the overfitting problem. Second, the information of the sound events is more evenly distributed in each band, which is better for the classification model.

Apart from the original Mel bands, many works also suggest that the log-form Mel bands features performs generally better in experiments [41, 47]. An example of the original and the log-form Mel bands features of the same audio sample is shown in Figure 4.14. One can see that Log-Mel bands is obviously more colourful and contains more details than Mel-bands feature. This is because the logarithmic function can smooth the signal, thereby compressing the range of values, resulting in an effect similar to data normalization. In Section 4.4.3, we will further compare and analyze the difference between Log-Mel and Mel bands features through the experimental results.

In brief, deep learning models do not need highly compressed engineered features or structured data. It is the model which needs to be sophisticatedly designed and let it find the statistic pattern in the data.

4.3.2 The classification model: CNN-L2

A typical neural network model consists of one input and one output layer and multiple hidden layers. We name our model *CNN-L2* since it has two hidden layers, both of which

are convolutional neural network layers. Three of the most commonly used layers up to date probably are: fully connected layer, convolutional layer and recurrent layer. Each layer has its own suitable use cases. A full-connected layer is most commonly used for data classification, which basically connects every neuron in this layer to every neuron in the previous layer. A convolutional layer, in another hand, is widely used in image processing, while a recurrent layer is most popular for linguistic information modelling. Of the three layers, our model will mainly use convolutional layers and full-connected layers. We do not consider using the recurrent layer because it is mainly used in stateful and sequential data and is also very computationally costly. Generally speaking, with a similar model size, the recurrent layer has the highest computational cost while the convolutional layer has the lowest [96].

4.3.2.1 Convolutional neural network layer

A CNN-layer consists of multiple 2-dimensional filters of the same size, which is also called the *filter-size* of a layer. During the forward pass, each filter is convolved across the width and height of the input volume to produce a 2-dimensional output of that filter. After training, the network would learn filters that 'activate' when they detect some specific shapes at any position in the input. Since the filter size is irrelevant to the size of the input size, it greatly reduces the weights and calculation density compared to the full-connected layer. In image-processing problems, CNN layers are well known to be able to detect the shift-invariant features from any part of a picture. With the stacking of multiple CNN layers, the output image of one CNN layer is also the input image of the following CNN layer. These images would gradually become smaller in the forward direction which represents more condensed and higher-level features, as demonstrated in Figure 4.15.

In neural network, a CNN-layer is also commonly followed by a maxpooling-layer. A maxpooling layer applies a max filter to (usually) non-overlapping subareas of the input neurons. It can downscale the output-size from the CNN-layer and helps avoid overfitting by suppressing the influence of the smaller outputs.

In addition to the max-pooling layer, we also attach a dropout layer to each CNN layer, which is a simple yet efficient technique for reducing overfitting in neural networks. In the training process, neurons of the dropout layer would be randomly set to zero at a given ratio (i.e. 20%) so that it looks like some neurons are dropped, which makes the training process noisier so that the learned neuron values are more sparse.

Both the maxpooling and dropout layers contain very little to no weights, so that for counting weights we only need to consider the CNN-layer. A CNN-layer in total has the

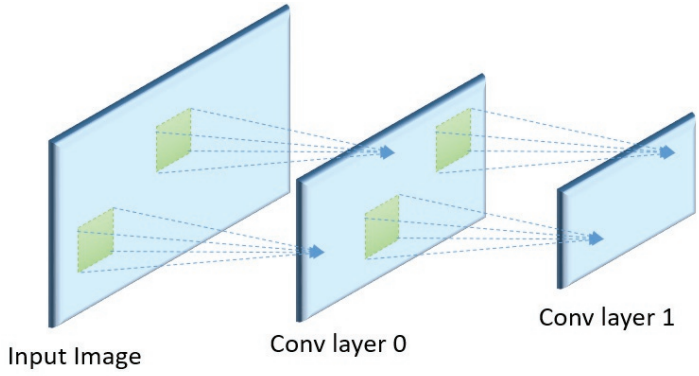


FIGURE 4.15: The Convolutional Neural Network layers explained

weights of: $height \times width \times filters \times depth$, where the 'depth' refers to the input image number of the previous layer, e.g. for a RGB picture it is 3 and for a preceding CNN layer it is its filter-number.

4.3.2.2 The Activation Functions

Both of the two CNN layers in our model use the popular $ReLU$ activation function. ReLU is a non-linear function which is widely used as the activation function in neural network [97]. The basic ReLU function has the form of:

$$ReLU(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

Compared to the traditional sigmoid activation function, ReLU has several advantages such as faster computation, more efficient gradient propagation and avoid the gradient vanishing problem.

We use the softmax activation function in the output layer, which gives the prediction of the probabilities of each event class. The softmax function is specifically designed for probability predictions which makes the outputs positive and sum to one:

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{4.3}$$

, where x is the linear activation before softmax. For a classification problem, there are equal number of neurons and the classes in the output layer where each neuron outputs

the probability of that class. In the test session, the neuron with the biggest output value would be the predicted event class.

4.3.2.3 The model design insights

As shown in Figure 4.12, our *CNN-L2* model mainly contains two CNN layers and one full-connected output layer. This model framework is more concise than the ones proposed from other papers such as [41][47]. The intuition behind each layer is explained below with both words and figures:

Firstly, our model starts with two CNN layers to extract useful features from the image-like spectrograms. Since the CNN filters convolve through the whole image and only accept a small area once a time, the filter size is irrelevant to the input size. Secondly, we do not add additional full-connected hidden layers after the CNN-layers. Usually, people use multiple full-connected hidden layers to connect the last CNN-layer and the output layer [41]. These full-connected layers together can build complex non-linear functions so as to map the CNN output-shapes to the final classes. However, in our model, the last CNN layer only outputs a small vector which is generally sequence-invariant to the sound classes prediction. Therefore, we believe a linear function can easily solve this problem so that no more hidden layers are needed. More details of each layer are explained below:

1. ***CNN_0*** is the first CNN-layer that extracts the basic shapes from the input Mel-bands spectrogram. Unlike an image-processing problem which normally uses small a filter-size (i.e. 5×5), we use a very large filter-size in the first layer (i.e. 57×5), which covers most of the frequency bands. In image-processing problems, CNN layers are well known to be able to detect the shift-invariant features from any part of a picture. This tall-shaped filter makes the detected features only shift-invariant in time-domain, since the same shape in two different frequency bands actually sounds very different. As is shown in Figure 4.16, a tall filter that covers the entire frequency bands would solve this problem.

   Following this CNN layer is a maxpooling and a dropout layer which are very basic layers to reduce noise and overfitting. The maxpooling layer has the size of $4x3$ so that after pooling the output is of size $\{1x27x20\}$, corresponding to $\{frequency, frame, filters\}$ respectively. In the output neurons, the frequency dimension has vanished (reduced to 1) and the original time-frames have shrunk from 80 to 27.

   To get a more intuitional view, Figure 4.17 shows the plots of some learned filters. In this figure, the weights are all between -0.3 to 0.3, where the red spots are to

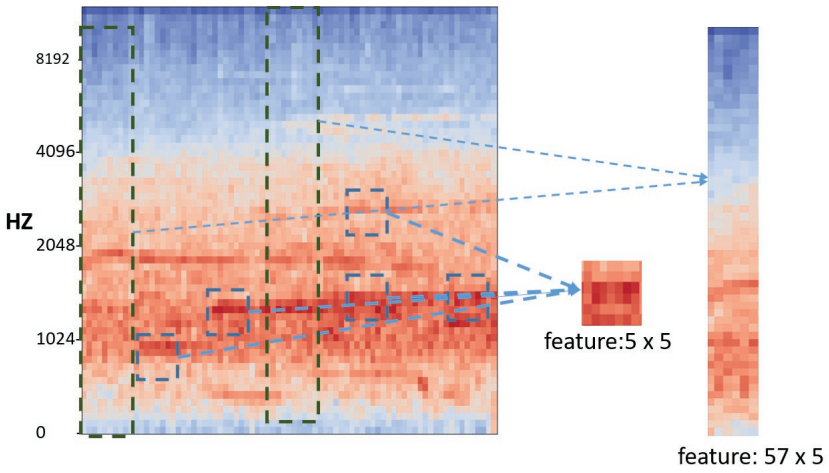


FIGURE 4.16: 'tall' filters are preferred in the first CNN layer: small filters would find features at different frequency bands which sound very different though

be amplified and the blue spots are to be suppressed. Take the bottom-right filter as an example, it is clearly filled with blue stripes in the lower frequency bands which means it only activates at high-frequency inputs.

2. *CNN_1* is the second layer that detects the higher level features by combining the smaller features detected from CNN_0. We used 20 filters in this layer each with the feature-size 1×3, which means the features detected in CNN_1 is three times longer than CNN_0 in the time domain. Similar to CNN_0, we also used ReLU activation function and attached a maxpooling layer and a dropout layer subsequently.

   Plotting the filter weights in CNN_1 however, does not provide a straightforward sense about the learned features since these filters do not directly apply to the inputs but on the outputs of previous layers. Mathematically, the relationship between the output of CNN_1 and the input is:

$$O_{cnn1} = F_{cnn1}(F_{dropout}((F_{pooling}(F_{cnn0}(I)))))$$

   , where I is the initial input spectrogram and O is the output and all F are the functions of neurons. As we can see, merely plotting F_{cnn1} does not give a clear idea of what this layer really expects from the input.

   To visualize and interpret the functions of higher layers such as CNN_1, a special visualization technique called activation maximization can be used [98]. The idea behind activation maximization is simple in hindsight : 'generate' an input I^* that maximizes the filter outputs, i.e. to compute:

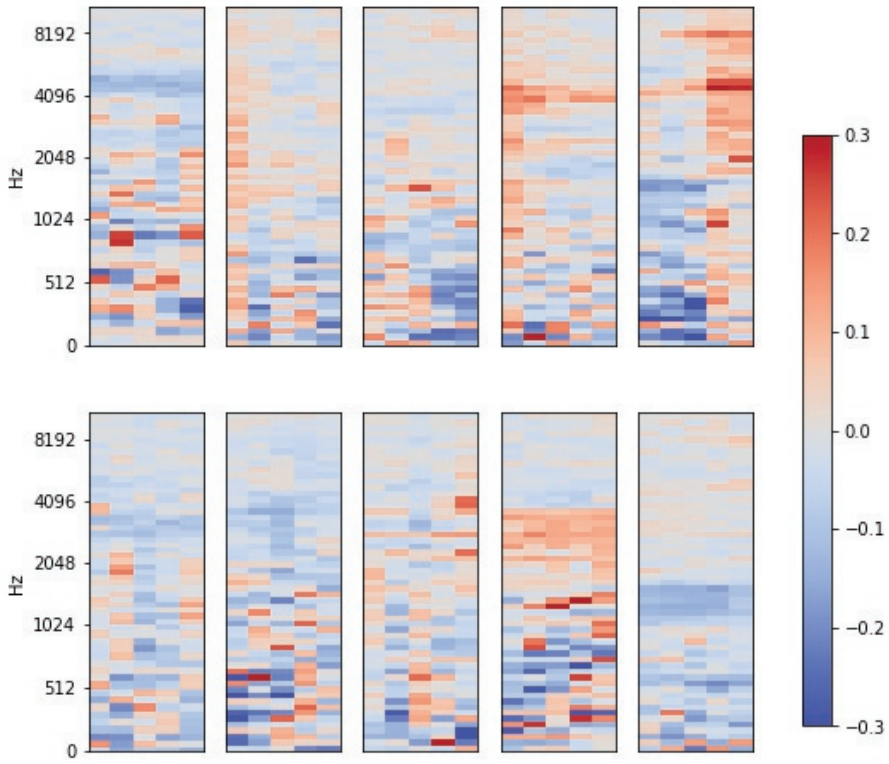


FIGURE 4.17: Some samples of learned filters from CNN_0 layer

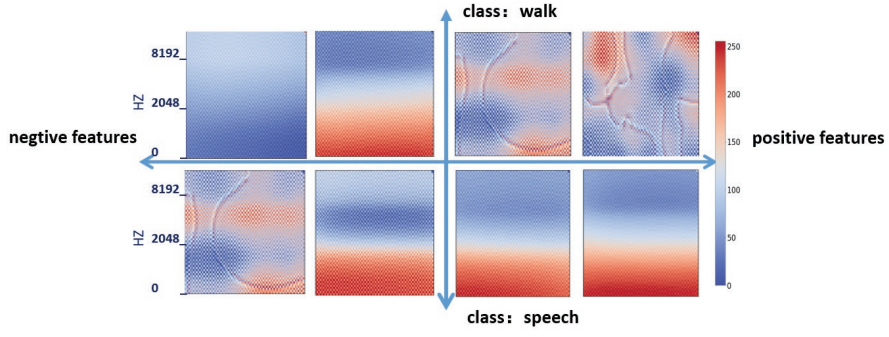


FIGURE 4.18: 'Activation maximization' of the learned filters from CNN_1 layer: what are the most and least wanted inputs of a sound event

$$I^* = \arg\max_I O_j(I) \tag{4.4}$$

, where O_j is the output of the jth filter from CNN_1 layer with respect to input I. This is a non-convex optimization problem but the local minimum can be found

through gradient descent method, where the $\partial(O_j)/\partial(I)$ is used to update the input I iteratively.

With this activation maximization technique, we are able to visualize what each learned filter would extract from the inputs.

To get a clear sense, several most representative filters out of the 20 filters are plotted according to their contribution to two event classes, i.e. each filter is either 'most' or 'least' wanted by the event class 'walking' and 'talking'. As is shown in Figure 4.18, the right quadrant means the most wanted filters while the left quadrant means the least wanted ones, the upper quadrant means the filters are for class 'walking' while the lower quadrant ones are for class 'walking'. (We will explain in the next paragraph how to calculate which filter is wanted by which sound event.) From this figure, one can clearly see that the walking-class prefers the tall cylinder-like inputs and the talking-class prefers the low-frequency bands and flat-shaped inputs.

3. The output layer is a full-connected layer with *Softmax* activation function, which gives the prediction of the probabilities of each event class.

   The previous CNN_1 layer has the output of size 8×20, where 8 can be interpreted as the activations in 8 sequential time segments, each is $500ms$ long ($500ms \times 8 = 4s$ plus spectrogram-frames are half-overlapping). For a typical full-connected layer, there is a weight linking every neuron and the previous layer output, i.e. we would have $6(class) \times 8(time) \times 20(filters)$ weights. To further reduce the weights and prevent overfitting as while, we adopted a shared-weights strategy on this layer. The idea is that for sound event classification, the CNN_1 layer outputs are basically time-invariant for event prediction. In another word, even if we shuffle the $500ms$ sound frames of a sound event randomly, we can still identify if it is a speech sound or a bell ringing. According to this, we force the weights to be shared over the time-axis so as to reduce the weight-size to: 6×20. This shared-weights mechanism does not only reduce the model size, but also increases the robustness since we have removed the redundant flexibility on the weights.

   As is shown in Figure 4.19, in a weight $W_k^{i,j}$, $\{i, j, k\}$ means $\{time, filters, class\}$ respectively. The shared-weights trick has eliminated time i, so that there is only one weight W_k^j for each CNN_1 filter j and output-class k. This trick actually makes CNN_1 layer easier to debug and interpret: the bigger the W_k^j, the more positive contribution the filter i makes to class j and vice-versa. Therefore, combining with the previous activation maximization and W_k^j we can easily interpret how the model makes the prediction in the output layer, as is shown in Figure 4.18.

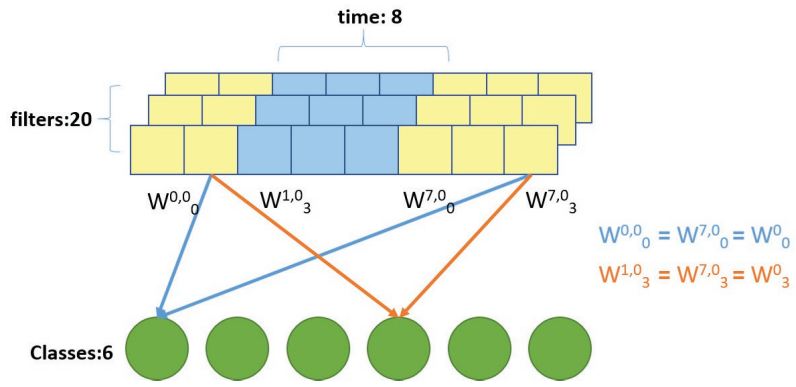


FIGURE 4.19: Shared-weights in time-axis to reduce weights and mitigate overfitting

Using this framework and the given hyperparameters above, the weight-size of each layer is:

1. CNN_0 : $57 \times 5 \times 20 = 5700$
2. CNN_1 : $3 \times 1 \times 20 \times 20 = 1200$
3. Softmax : $20 \times 6 = 120$

To summarize, via a much smaller framework and shared-weights trick, our model has only approximately 7K weights, which is much smaller than the million-weights sized model proposed by [41][99].

4.4 Experimental Evaluation

In this section, we will present the experimental results over our labelled audio dataset. In order to have extensive and representative audio data samples, we will use data from both public datasets and websites, as stated in Chapter 3. The entire dataset is split into 2 parts: training and test set. For the training sets, 5 folds cross-validation is used to build the best fitting model, which is subsequently applied to the test set for validation.

We will first present the results of the methods based on classic models and the deep learning model in Section 4.4.2 and Section 4.4.3, respectively. For each method, we have conducted comparative experiments to find the best setting for each component. After the best possible settings of each method have been found, we will also compare the two methods in Section 4.4.4, using the same metrics.

The evaluation metrics include both performance and efficiency since both aspects are important to our research. The performance is represented by the overall classification *accuracy* and *F1-score*. For a multi-class classification problem, *accuracy* is the simply proportion of correctly classified samples to all samples. F1-score is also known as balanced F-score or F-measure, it is the harmonic mean of precision and recall, to punish the extreme errors:

$$F1\_score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F1-score is initially defined for binary classifiers, i.e. whether the output is true or false. In our multi-class classification problem, the overall F1-score is the mean of F1 on each output class. Accuracy is preferred when each class is equally important while F1 fits better for imbalanced datasets. As our dataset is by large balanced since every sound event class is equally important, it is safe to say that accuracy is a better index than F1-score.

The complexity of an algorithm normally includes its memory cost and running time. For the memory cost, we mainly evaluate neural network models by giving their weights-number as it is the dominant index of memory cost. The memory of classic models however is harder to compare theoretically since different models have different structures. In our practical programs, all the classic machine learning models appear to be very cost-effective. All of these trained models take less than 10 kilo-bytes when being saved on disk as loadable files, however, the additional costs in the library are hard to estimate. The complexity of running time is defined as the reasoning time of a unit length audio sample, i.e.:

$$C_t = \frac{T_{reasoning}}{T_{audio}}$$

, where $T_{reasoning}$ is the time to predict an audio sample while T_{audio} is the length of the audio. C_t only includes the test time since it affects the time delay and is the constraint in real-time IoT applications.

After the statistical evaluation, some discussions are given to gain instructive insights about how to build real-time indoor environmental sound recognition applications.

4.4.1 Dataset

As is stated in Chapter 3, our datasets for experiments come from both public datasets (*TUT, NUY*) and websites (*www.freesound.org, www.freesfx.co.uk*). The audio data events need to be common indoor sound and should be highly relevant to human activities information. According to this criteria, we have selected 6 types of sound events:

speech, door, footsteps, clapping, cart-wheels, bell ringing. Initially, we had chosen more events, such as chair moving and people cheering. However, after looking into the dataset, we merged and truncated some classes since some sounds always occur simultaneously. For example, the cheering sound is always accompanied by the applause sound. Since these sounds are hard to separate, we merge them together into one class.

The original datasets are all audio streams, where the duration of each event is precisely labelled in milliseconds. A continuous audio stream is however not easy to evaluate or process. Using the methods stated in Section 4.1.2, the very first step we take is to segment the streams into discrete sound clips, where each clip contains exactly one event.

For the classic models, each audio clip is extracted precisely based on the labels, since we use the flexible-length segmentation method. To filter the relatively bad data points, events with less than 1-second length are dropped. In contrast, deep learning models require consistent input data length. In this case, we slice the audio stream into 2-second long audio clips where each clip contains one single event. The original events longer than 2 seconds from the stream are truncated, while the shorter ones are padded with silence. After segmentation, we further choose 1000 good sound events for each event class to make an even distribution.

4.4.2 Evaluation of the classic models

In this section, we will use experimental results to evaluate the results based on classic models. During the experiments, different settings such as features combinations, models and their augments are tested, so as to find the best one for the application.

Among all these components, we will put more effort on the features rather than the classification models. Compared to the classic machine learning models, classic models are not complex and sophisticated in general. The knowledge or weights trained from most classic models can be stored with very little memory, which definitely can not store very extensive information. Take decision tree as an example, the information or parameters that needed be stored is in the tree nodes, where a common 5-level tree has $2^5 = 32$ nodes in maximum. From the data-representation perspective, this small amount of information can be very hard to represent complex and extensive data patterns. In this sense, we think it is more important to find good features that highly correlates with the output than to find a model that perfectly fits the data.

4.4.2.1 Feature Comparison

In this subsection, we first compare different features from performance and complexity perspectives.

The entire reasoning time of a sample mainly consists of the $T_{feature\_extraction}$ and $T_{model\_prediction}$. In our experiments, we find that $T_{model\_prediction}$ for the classic models are much smaller than $T_{feature\_extraction}$, thus can be ignored. The complexity C_t is thus simplified as:

$$C_t = \frac{T_{feature\_extraction}}{T_{audio}} \tag{4.5}$$

To clarify, the values of both $T_{feature\_extraction}$ and T_{audio} are from experimental results, through a single thread program running on Raspberry-Pi 3B platform of 700 MHz CPU. A real-time application should definitely control this complexity lower than 1.0, i.e. make feature extraction time smaller than audio length. Through all the comparative experiments, we choose SVM (with RBF kernel) model together with feature MFCC as the baseline. This baseline function has been used in both environmental sound classification and sound context classification and has achieved good results [35][34].

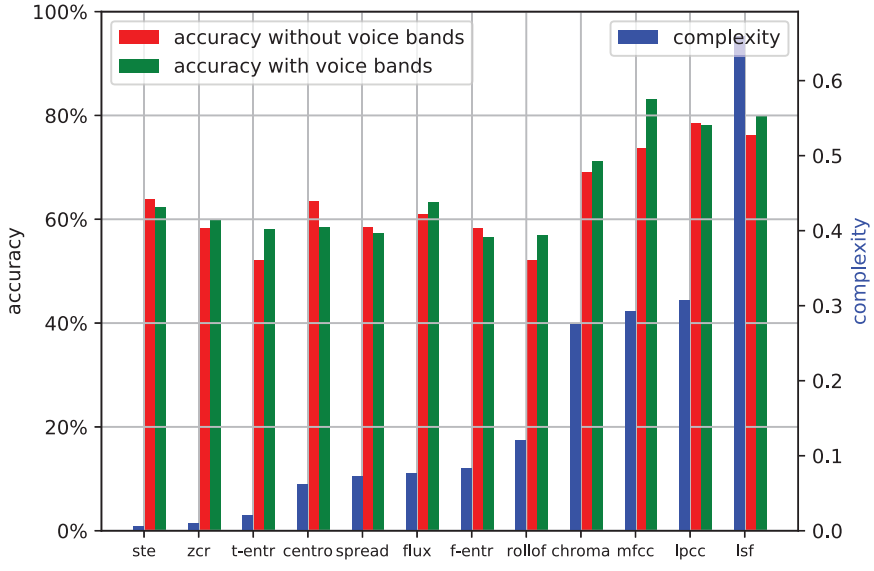


FIGURE 4.20: Performance and complexity of using each audio feature

Figure 4.20 shows the performance of each single feature. To get an intuition of how voice bands affect the results, we tested the data before and after voice stripping. Results show that the MFCC of the baseline is the best before voice bands truncation, while LPCC is the best after truncation. These results show a significant performance drop of MFCC after voice bands truncation (from 82% to 77%), which is reasonable because MFCC is designed for human speech recognition as such it provides better resolution in lower frequency bands. While LPCC portraits the smoothed spectral envelope for entire frequency bands without discrimination, it works equally good with or without voice bands.

After the evaluation of single features, Figure 4.21 shows the results of multi-features (or feature combination). In this figure, each broken line shows the greedy search results for finding the 'best feature combination', where each dot represents one iteration. The two separate dots that are not on the broken lines show the all-features-combination result for comparison. In machine learning, multiple features could be combined together to reach better performances, as more features provide more information. However, this does not mean the more features the better, with a fixed number of training samples. The predictive power would normally first increase as feature numbers go up then decreases [100], mainly because duplicated and irrelevant features only introduce noise to the model. The problem incurred by high dimensional features is also called 'curse of dimension' [100]. In offline audio classification, this problem is normally addressed by dimensionality reduction techniques such as Principle Component Analysis(PCA) [101]. PCA is the process that computes the principal components and using them to perform a change of basis on the data, normally using only the first few principal components and ignoring the rest. By projecting data points onto only very few principal components, the data dimension is reduced while preserving as much variation as possible. The first principal component can equivalently be defined as a direction that maximizes the variance of the projected data.

This kind of pipeline of "getting all features" and "pruning redundant information" however would cost a lot of time and is not fitting for real-time applications. In order to save time and computational cost, it is better to proactively select the good features. While brute force all the combinations is too time costing, we use heuristic greedy search to find the local optimum instead. It starts with selecting the best single feature, then at each iteration one more feature is added depending on the classification accuracy. In both cases, t combination of features can improve the performance significantly. Looking into the voice-bands truncated case, the best combination contains 4 features: LPCC, spectral-flux, STE and time-entropy. This result matches our expectations, as these features portrait different perspectives of the audio signals. The classification accuracy would increase to more than 90% if voice bands are reserved, which is 4.7% higher than

using voice-stripped signals. In both cases, the highest performance can be obtained by combining 4 different features, each starts with a cepstral feature-vector. Not surprisingly, using all-features for model input is not preferred, as high dimension data will cause overfitting problems and is much less efficient. The calculation complexity of features combination is smaller than the simple accumulation of each since they share some steps such as FFT for all frequency-domain features.

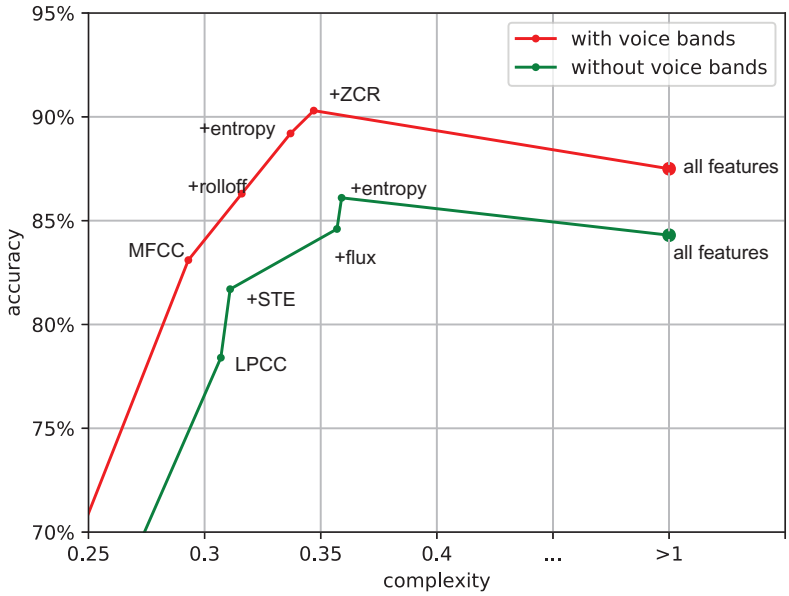


FIGURE 4.21: The greedy search results for the best feature combination

In speech recognition, common techniques such as STFT are used to extract spectrum from overlapping frames, normally with half or 3/4 frame length overlapped. This is because audio signals especially speech are highly time varying, and need more sophisticated analysis to reflect the details. However, using overlapping frames also duplicate or quadruple the complexity. We compared the results from different overlapping frame lengths, with the best features combination found above. Figure 4.22 shows the results with different overlapping frame lengths. Results show that it is best to use half-frame length overlap, which leads to roughly the same accuracy as 3/4 overlapping, but is much more efficient.

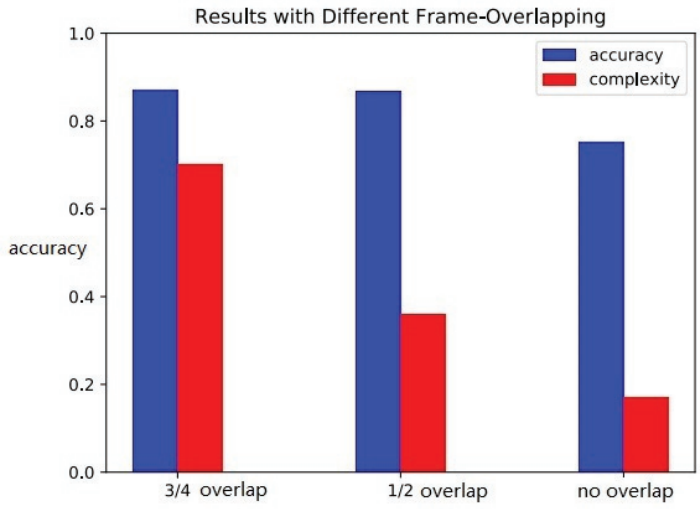


FIGURE 4.22: Frame overlapping test with SVM and LPCC feature

4.4.2.2 Models comparison

In this section, we compare the performance between 6 stateless classification models with the same input features: LPCC. The performance of different classifiers is shown in Figure 4.23. Results show that SVM(RBF kernel) is the best, and the second best is random forest. GMM performs the worst among all in this experiment, which basically suggests that it is hard to find one or multiple perfect samples that best represent a sound event.

4.4.2.3 More detailed results

In our application, knowing the classification results is not enough, we also need to score how reliable the results are. The high score classification results are kept, while the low score ones will be discarded. This is because there is a lot of noise and irrelevant events in the environment, which are hard to be filtered beforehand. However, they are likely to get low scores in the classification model, since these sounds carry very different features. On the other hand, it is acceptable if some events are discarded by mistake as knowing part of the events is enough for detecting human behaviours. Classification algorithms such as Naive-Bayes give both classification result and the probability, i.e. a degree of certainty about the result, which just corresponds to our needed score. However, SVM algorithm does not provide the prediction probability directly, it only gives the support

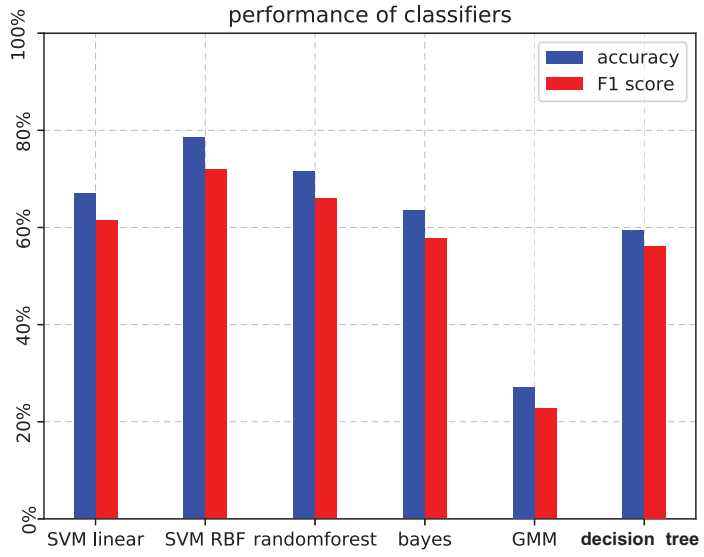


FIGURE 4.23: Performance of different classifiers using the same feature

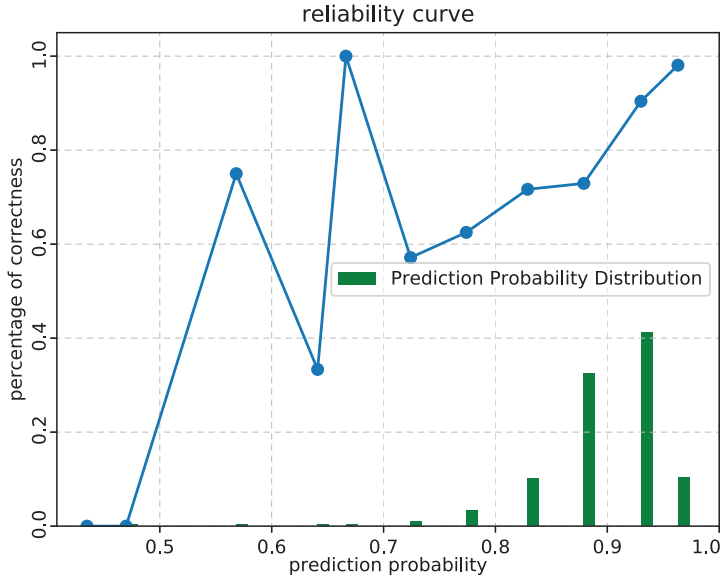


FIGURE 4.24: Confidence of prediction with SVM model

vectors, in which a larger margin means more confidence. Hence, we need to calibrate the support vectors into prediction probability, i.e. the score.

We use the algorithm from Wu [102] to calibrate the SVM support vectors into prediction

probability. This algorithm is the multi-class version of Platt Scaling: applying logistic regression on the SVM's scores, fit by an additional cross-validation on disjoint training data. However, normally this method needs a large dataset (1000 samples for each class) to work well, otherwise, the probability estimated may be inconsistent with the classification result. Figure 4.24 shows the real accuracy versus the calibrated probability of our classification results, where the green bars show the sample distribution. The real accuracy and prediction probability are roughly in positive proportion, which means the higher the score the more accurate the results. The violation of monotonic increasing of prediction accuracy could be from multiple reasons: our dataset is not big enough and the calibration method is not perfect. This result shows that nearly half the results are given 90% probability, whose real accuracy is at the same level.

4.4.3 Evaluation of deep learning based method

In this section, we will present the experimental results of the deep learning based method from different aspects. Deep learning models are very sophisticated models with quite extensive types of frameworks and layers. In contrast to classic models, deep learning models normally prefer raw and high dimensional features to engineered and short features. It is therefore the model framework and hyperparameters rather than the features that affect the performance most. As a result, our experiments only compare the Mel-bands spectrogram and its log form features. Regarding the model itself, we mainly measure the model complexity through its weights as it is the decisive factor of both the memory cost and the reasoning time.

4.4.3.1 Model Hyperparameters experiments

With the basic model framework settled, we also conducted comparative experiments with different hyperparameters to find the balance in three criteria: model-size, performance and robustness. Model-size refers to the number of weights, which directly affects the memory consumption of the application. Both CNN_0 and CNN_1 layer filter-numbers are good parameters to experiment with and contribute most to the model size. However as the CNN_1 layer size is the most decisive factor of the performance and is also more complex, we mainly present the results on the CNN_1 layer for gaining better insight.

For each experimental result, we use the average prediction accuracy of all event classes as the metric of performance. The same metric is also used for robustness evaluation using noisy inputs, since noises are inevitable in real environments and can affect accuracy to a great extent. In the robustness test, we added two different types of noises to the

original data to simulate the impact from different environments: Gaussian noise and occlusion noise. The Gaussian noise sound is quite pervasive which can be created by any surrounding objects while none of which is dominant. We use the -3db SNR Gaussian noise of all frequency bands for the first noise source. The occlusion noise refers to the impulse like noise, which is almost instantaneous, normally caused by unwanted activities such as mouse clicks, window vibrates or electromagnetic interference during microphone recording. To create the occlusion noise, we randomly wipe out a 200ms segment of the original sound and replace it with some random noise.

An example of the two types of noisy sound together with the clean sound is shown in Figure 4.25.

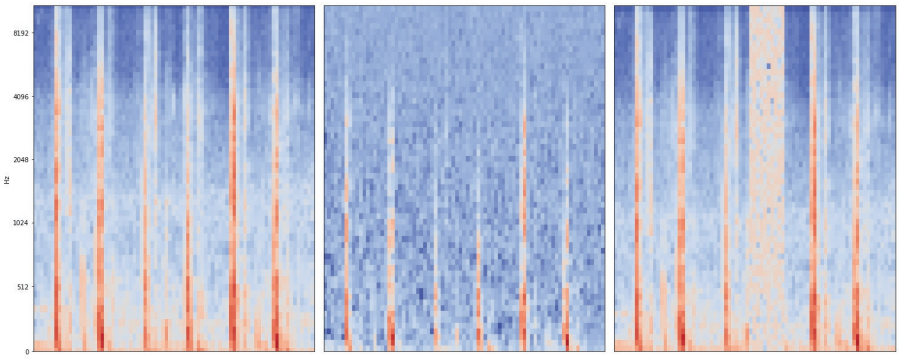


FIGURE 4.25: A sample of the walking sound with no-noise (left), Gaussian noise (middle) and occlusion noise (right)

For comparison, we also tested with the CNN-based model proposed in [41] as the baseline, which has two more hidden layers than ours, here referred to as CNN-L4. The overall result of the comparative experiments is shown in Table 4.2, all of which use the log-Mel bands as input features for uniformity. From the results, we could see that the prediction accuracy using clean sound is quite similar with $15 \sim 40$ CNN_1 filters and only drops dramatically when the filter number shrinks to 10. For the noisy inputs test, our model can resist the occlusion-noise very well but is error-prone to the -3db Gaussian noise. The occlusion-noise test also shows that although using 15 and 20 CNN_1 filters can both get high accuracy with the clean sound, there is a big gap in the robustness.

As compared to the baseline model, our model is much smaller in size and still reaches comparable performance. Whatsmore, our model seems to be better noise-resistant with proper hyperparameters.

TABLE 4.2: Comparative results of different model-hyperparameters with the Log-Mel feature

Models	Weights	Accuracy Clean	Accuracy Noise-Gaussian	Accuracy Noise-Occlusion
CNN1:40	8.3K	96.9%	69.4%	92.9%
CNN1:30	7.7K	96.6%	67.5%	92.2%
CNN1:20	7.0K	96.5%	67.3%	91.8%
CNN1:15	6.7K	95.8%	64.7%	86.9%
CNN1:10	6.3K	88.7%	50.1%	74.2%
Baseline*	8.0M	97.1%	63.1%	93.9%

Baseline is the adapted model from the 4-hidden layer CNN framework proposed in [41].

Table 4.3: Comparative results of different input features and shared-weights strategy

Input feature	Is share-weights ?	Accuracy	Accuracy Noise-Gaussian	Accuracy Noise-Occlusion
Log-Mel	Y	96.5%	67.3%	91.8%
Log-Mel	N	96.6%	64.3%	91.0%
Mel	Y	94.8%	94.0%	90.8%
Mel	N	94.0%	91.8%	90.1%

4.4.3.2 Model robustness evaluation

As the results in Table 4.2 shows, none of the models using Log-Mel bands is Gaussian-noise resistant, which could cause a problem in real-life applications. One of the solutions to address the problem is randomly adding Gaussian-noise in the training data, as proposed in [103]. However, this tricky solution is not perfect which only shifts to the noisy dataset and is vulnerable in classifying clean sound.

In contrast, we conducted comparative experiments with Log-Mel and Mel-bands features and found out using the Mel-bands could actually avoid the noise issue. Although the Log-Mel bands feature is suggested by many papers [41] since it smooths the feature space and reveals more details, it also makes the model easily overfitting to the less important information. As described in the methodology, our model framework also used the shared-weights strategy to prevent overfitting as well as to increase the robustness, we also justify it with the test on the non-shared weights model. In this test, we used 20 CNN_1 filters in all experiments for uniformity.

The results are shown in Table 4.3, which basically shows the shared-weights model with Mel bands feature performs the best on average. Our experiments show that using the Mel-bands feature only slightly decreases the performance with clean or occlusion-noise sounds, but can greatly improve the performance in the Gaussian-noise environment. For the environment that a lot of Gaussian noise exists, it is a good choice to use the model with the raw Mel-bands feature.

4.4.4 Comparison of the two methods

In this section, we compare the results of both classic models and deep learning models with the same dataset. Figure 4.26 and figure 4.27 show the confusion matrix of each method's best model, in terms of performance, to demonstrate the detailed results for each event class.

Figure 4.26 shows the confusion matrix of SVM-RBF model and grouped features, where the 'footsteps' and 'bell' are the best two classes in terms of accuracy. The 'speech' class performs the worst, which is acceptable because of the voice bands truncation. However, some 'cart' and 'clap' sound events are also likely to be misclassified as 'bell' although humans can clearly distinguish them. One of the downsides of using statistic features is that they can only portrait the briefs but not the details. For example, a transient signal with one single spike and another signal with several periodical waves can have similar mean and variances.

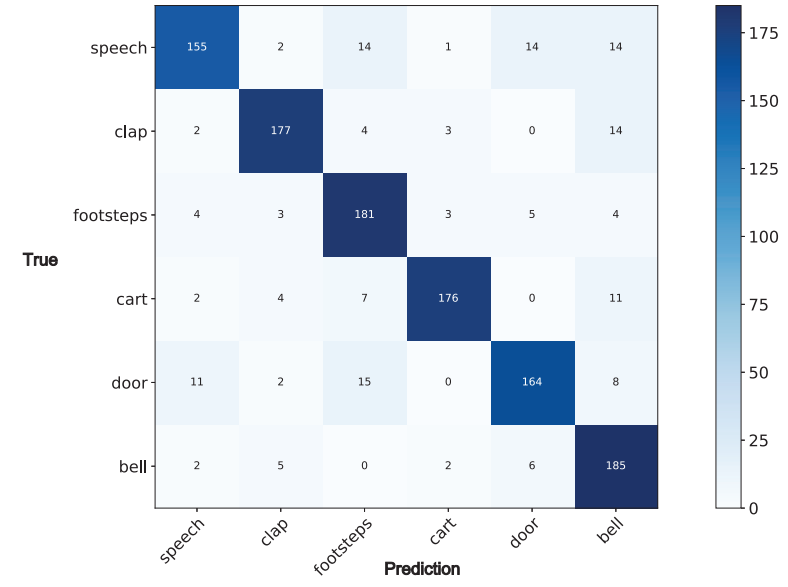


FIGURE 4.26: Confusion matrix of SVM model with the best feature group

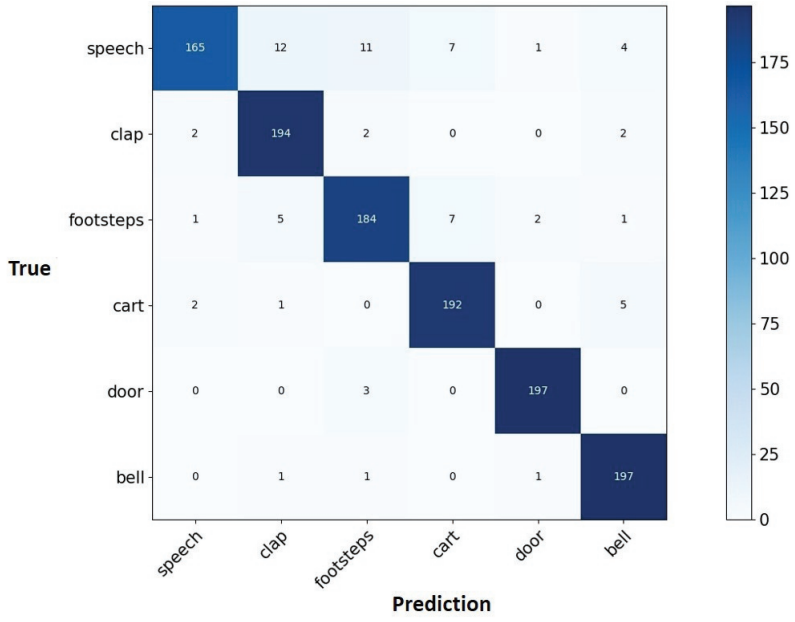


FIGURE 4.27: Confusion matrix of CNN_1 model with 20 filters

Figure 4.27 shows the confusion matrix of the CNN-L2 model with 20 CNN_1 filters. The overall prediction accuracy of this model is 94.0%, where the door sound is best classified and the footsteps sound is the most error-prone apart from speech sound. The overall

accuracy is significantly better than classic models and all classes apart from speech sound perform evenly good. The relatively poorer results on footsteps sound do meet our expectation, as our model framework basically scores each 500ms sound segment independently and uses the average score of all segments for the final classification. To tackle this specific problem, a possible solution is inserting one more full-connected layer between the output and CNN_1 layer, which basically adds non-linearity in the last classification step, which of course also needs much more weights and makes little help to other event classes.

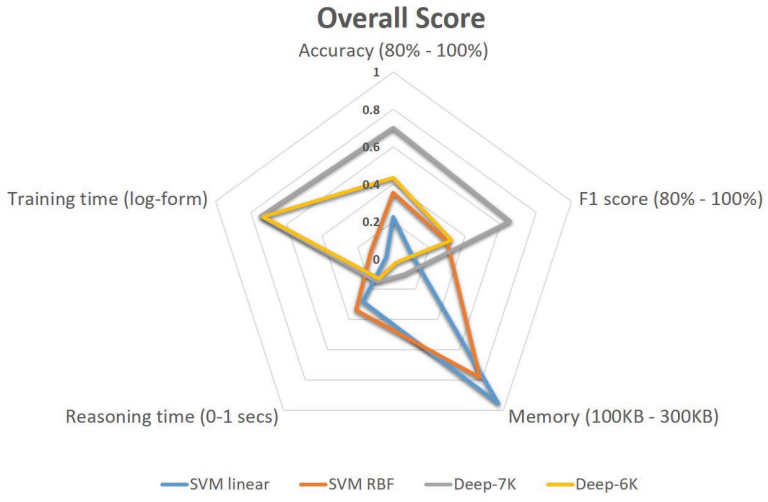


FIGURE 4.28: The overall score of the two methods

We next add costs into the evaluation metrics to provide insights from more perspectives. These cost-metrics consists of model training time, model size and reasoning time. Some new cost metrics are added in because it only makes sense to compare them between methods and not between models within one method. For example, the training takes very little time (several seconds) for all classic models and much longer (from hours to days) for deep learning models. We choose the top-2 best models from each method and plot their overall scores in Figure 4.28. All the values of the 5 metrics are properly stretched to between 0 and 1, with the center point being 0 and edge points being 1. Apart from training time, all the other 4 metrics are linearly stretched with the lower and upper bound labelled at the vertices. Take accuracy as an example, value 0 (the central point) refers to 80% while 0.5 refers to 90% accuracy, which makes the curves sparser and clearer. Training time however differs in order of magnitude so that we plot in a logarithmic form. SVM models only need a few minutes to train and deep learning models take more than 3 hours. The memory cost is calculated merely by the memory used to save the trained model, through the tool 'pickle' in python. One should also

consider the code or library that load and execute the model, in terms of designing a memory limited application.

4.5 Conclusion

In order to recognize indoor human activities using acoustic sensors, performance is not the only concern, efficiency and privacy are equally important considerations. To provide insights into building a resource-constrained application, we looked into classic models and deep learning based models separately. Our results show that with voice bands stripped off for the privacy concern, the system could still detect human activities with quite a high accuracy (86% for SVM, 94% for deep learning).

For the classic models based method, we first looked into the features selection. A comparative test for all single features show that LPCC performs best for non-voice, and MFCC performs best for full signal bands. These results match our expectation that MFCC is designed for speech recognition and voice bands details. We next tested combinations of features with the greedy searching experiment to further improve the accuracy. In this experiment, LPCC together with one frequency domain feature(flux) and two time-domain features(STE, entropy) perform best, since they portrait different aspects of signal: LPCC and 'flux' each portrait the static and dynamic characters of frequency domain, while STE and entropy do the similar job in time-domain. Another feature test is about the frame overlapping, the trade-off is between complexity and performance. In our experiment, half the frame length overlapping (0.02s frame length, 0.01s overlapping) is the best.

The tests on classification models show that SVM performs the best, and the prediction probability can be calibrated through Platt Scaling algorithm to filter out unreliable results. If we only keep half of the best predictions, the accuracy increases to more than 90%. However, the real accuracy is not monotonically increasing when the prediction probability increases. It may happen because our dataset is not sufficiently large or enough general for the Platt Scaling algorithm to work well. The most misclassified classes were 'door' and 'footsteps', this happens when people walk very slowly, our segmentation algorithm sometimes falsely split a series of footsteps into smaller events other than considering them as a whole. While in real applications, we could differentiate the two classes with the help of other methods, for example with sound localization algorithm since the door could not move.

For deep learning based method, we have proposed a small CNN-based model for environmental sound classification. Many different models proposed for environmental sound

classification show that the complex neural network models perform the best in general. However, more complexity also means more memory consumption and computation power, which can greatly limit the potentials of the neural network models in applications. Whatsmore, when the models become too large, they are difficult to interpret and are lack of debugging methods when errors happen. To tackle the aforementioned problems, we designed a very small CNN-based model with comparable classification accuracy to the large models. Compared to the similar model which normally has millions of weights, our model has only less than 10K weights. The model size is mainly cut by two tricks: 1. Remove the full-connected hidden layers after the CNN layers. 2. Use the shared-weights strategy in the output layer over the time axis. In order to justify and interpret our model, we not only explain the intuition behind each network layer, but also visualized the learned features of the CNN layers. We use the activation maximization technique to visualize the high-level features in CNN_1, which otherwise can hardly be interpreted in a straight forward way. With these interpretations especially the visualization on the CNN_1 features, our model parameters can be easily tuned and debugged.

The test on features shows that using Mel bands feature is only slightly less accurate than using Log-Mel bands, but is much more robust when the environment becomes noisy. Tests on the model show that the shared-weights framework performs just as we expected, which both cuts down the weights and increases the performance since it reduces overfitting. The confusion matrix also shows that footsteps sound is the most misclassified although the walking sound seems very easy to differentiate for humans. This downside is reasonable, as our model framework basically scores each short sound frame independently and uses the average score for the event classification, so that the information of the repeating pattern across frames is lost. To tackle this specific problem, either adding a hidden layer or increasing the time domain feature length could help, however with the price of more complexity.

Finally, we have given an overall score of the two methods using the best models from each. Results show that deep learning based model is significantly better than SVM-based model in performance. Surprisingly, our concise deep learning model also needs a smaller cost and less reasoning time than SVM based model. The only disadvantage of the concise deep learning model is the training time, which is undoubtedly larger. When the deep-learning model is very small, its execution time is much less than the feature extraction time, hence the one with simpler features is the winner in reasoning time. Regarding memory cost, when choosing a method suitable for its application, one also needs to consider the size of the library for running the model, because our study only calculated the model size. In conclusion, this research has shown the great potentials of using concise deep learning models in the resource-limited applications. To

further clarify the memory cost, we only compared the memory occupation of the trained model, where the size of SVM and deep learning library should also be considered in real applications. Even though our model is not perfect, we think this accuracy already makes it competent for a real indoor events recognition system. Also, our model is proved to be general since the dataset is from different contexts and different sound sources of high diversity. In real applications, the accuracy could be higher, since the microphone embedded devices are normally fixed installed, where the sound source should be more homogeneous and easier to classify.

Chapter 5

Sound-based activity recognition in crowded environments

In this chapter, we will further discuss sound-based human activities recognition, but in crowded environments. A crowded environment may refer to a restaurant, a supermarket or a party where many people make sounds simultaneously, thus a lot of sound signals overlap. It is widely agreed that overlapping sound events are much harder than a single sound event to recognize, for both computers and human hearings. To some extent, this problem can be solved by upgrading the single event model, for example, using a hierarchical model which identifies one event at a time. These models, of course, are inefficient and inaccurate, especially when there is a lot of events overlapping.
In our work, we will present two innovative solutions to tackle this problem from different aspects. In the first solution, we separate and classify the sound events with multiple microphones in a moderately crowded environment. When the environment becomes more crowded where individual sound events are barely separable, a second solution is proposed to predict the proportion of each sound event. The main content of this chapter is published in PETRA 2020 [104] and PERCOM 2020 [105].

5.1 Overlapping sound events classification with audio-sensor-network

Automatic localization and classification of environmental sound events can provide great aid to many human-centric IoT applications. However as many papers have mentioned, sound events in real life are complicated and very hard to classify especially when multiple sounds happen simultaneously. As a result, it has stimulated the interest of many researchers to classify overlapping events.

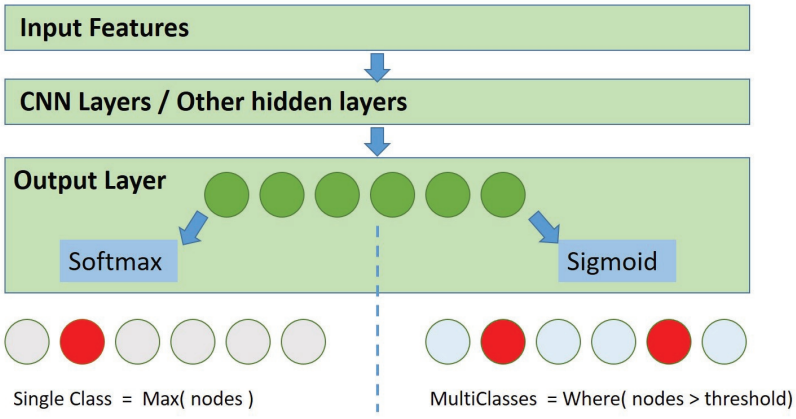


FIGURE 5.1: Modifying the output layer to support overlapping sound events

As is introduce in Section 2.1.2, there are mainly three types of solutions proposed for this problem. The first method extends the frame-based single-event models for overlapping events through extensive data training, which does not require the separation of signals. For example, with our deep-learning model proposed in the previous chapter, one can adjust it to overlapping sound by changing the output layer from softmax to sigmoid. A sigmoid output layer can output multiple 'positive' values, i.e. set multiple classes to true. This process is demonstrated in Figure 5.1. The second solution decomposes the signal by reducing a matrix into its constituent parts, which is mainly based on non-negative matrix factorization (NMF). The third method also decomposes the signal first but uses multiple microphones, usually microphone arrays, which is mainly based on the direction of arrival algorithms (DoA). In brief, single-channel models exhibit the inherent weakness in decomposing signals so that they could only be used in very limited scenarios for simple events. Additionally, despite the many advantages of leveraging multiple microphones, most algorithms are proposed for expensive microphone-arrays on robots with sufficient resources.

Unlike the aforementioned works, we propose in this section a lightweight and scalable solution for both localization and classification of overlapping indoor sound events with an audio-sensor-network. From the overlapping sound signals received from multiple microphones, we first decompose and locate each small keypoint in the spectrograms. These located and clustered spectrogram-keypoints are subsequently used for 'restoring' and classifying the original sound sources. To locate the position of each small keypoint, we built a global cost function to synchronize the time-difference-of-arrivals (TDOA) of each keypoint. With these clustered keypoints, 2 different classification models are used to classify the sound sources. Our experiments show that our solution is both accurate and low-cost in terms of computing cost.

In this chapter, Section 5.1.1 explains the TDOA algorithm which is the basis of the sound localization algorithm. Section 5.1.2 and Section 5.1.3 each describes our methodology of the localization and classification model for overlapping sound. Concrete details is given of using keypoints for sound event localization as it is the key contribution of our work. Section 5.1.4 describes the experimental and evaluation phase of this work with comparison to baseline models.

5.1.1 The basis of Sound localization: TDOA

Our sound event localization method is based on Time Difference Of Arrival (TDOA), which is a commonly used method in the area of localization and has many variations. The basic idea of TDOA is to locate the acoustic source from the time differences among all acoustic sensors(e.g. microphones) using trilateration [106]. This section briefly explains the principles and implementation of TDOA algorithm.

In a 2-D space topology, the basic TDOA algorithm can be explained as below:

(I) **Notation**

Let (x, y) be the unknown coordinate of the sound source being located. Let the known acoustic sensors coordinates be: $[(x_m, y_m)]_{m=1}^{M}$, where M be the sensors number. Let τ_m be the estimated relative time arrival to different sensors, where τ_1 is 0 for simplicity. Let r_m be the distance of the sound source to sensor m, and v be the sound travelling speed, so is the equation of the sound arrival time and the distance:

$$r_m - r_1 = v\tau_m - v\tau_1 = v\tau_m \tag{5.1}$$

for m = 2,3...M.

(II) **Calculation**

With some basic mathematical transformation on Equation (5.1), we can get:

$$v\tau_m + 2r_1 + \frac{r_1^2 - r_m^2}{v\tau_m} = 0 \tag{5.2}$$

We then subtract $v\tau_2 + 2r_1 + \frac{r_1^2 - r_2^2}{v\tau_2}$ from the left side for m = 3,4,...,M, to get:

$$v\tau_m - v\tau_2 + \frac{r_1^2 - r_m^2}{v\tau_m} - \frac{r_2^2 - r_m^2}{v\tau_m} = 0 \tag{5.3}$$

Substitute all the r with x, y coordinates, we could finally get:

$$\begin{bmatrix} A_3 & B_3 \\ A_4 & B_4 \\ ... & ... \\ A_M & B_M \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} C_3 \\ C_4 \\ ... \\ C_M \end{bmatrix} \tag{5.4}$$

Where:

$$A_m = \frac{1}{v\tau_m}(-2x_1 + 2x_m) - \frac{1}{v\tau_2}(2x_2 - 2x_1) \tag{5.5}$$

$$B_m = \frac{1}{v\tau_m}(-2y_1 + 2y_m) - \frac{1}{v\tau_2}(2y_2 - 2y_1) \tag{5.6}$$

and

$$C_m = v\tau_m - v\tau_2 + \frac{1}{v\tau_m}((x_1^2 + y_1^2 - x_m^2 - y_m^2) - \frac{1}{v\tau_2}(x_1^2 + y_1^2 - x_2^2 - y_2^2) \tag{5.7}$$

In order to get x, y coordinates of the sound source, we should first calculate the A, B, C vectors for all sensors, and then solve the linear algebra Equation (5.4). Notice that it yields a solution only when $M >= 4$, i.e. at least four sensors are needed. Although this written method is for 2-D space, one can easily scale to 3-D space, only the minimum M becomes 5.

Since τ_m is the only unknown variable in A, B, C, this localization problem equals the problem of finding τ_m, i.e. synchronize the start of signals in each sensor. Figure 5.2 shows the door slam sound received by 4 different acoustic sensors in the time domain. One typical method to synchronize these signals is time-smoothing together with cross-correlation [62]. Every two signals are synchronized when their cross-correlation value reaches the maximum. The red dashed lines in Figure 5.2 shows the calculated start point of each event with this method.

Cross-correlation finds the time lagging τ_m, of which two signals are most similar to each other. This τ_m is a single value that reflects the overall similarity of the entire signals, but not the details. To locate the overlapping sounds made by multiple sound sources from different locations(e.g. someone happens to be talking while the door slams), each source s should have a different τ_m^s instead of a shared τ_m. Obviously, this can not be achieved through the simple cross-correlation which only outputs a single value. As a result, a traditional cross-correlation based TDOA method can not locate overlapping sounds, nor can it decompose the overlapping sounds. An example is shown in Figure 5.3, when the short door slam sound is hidden in the longer sound event of talking.

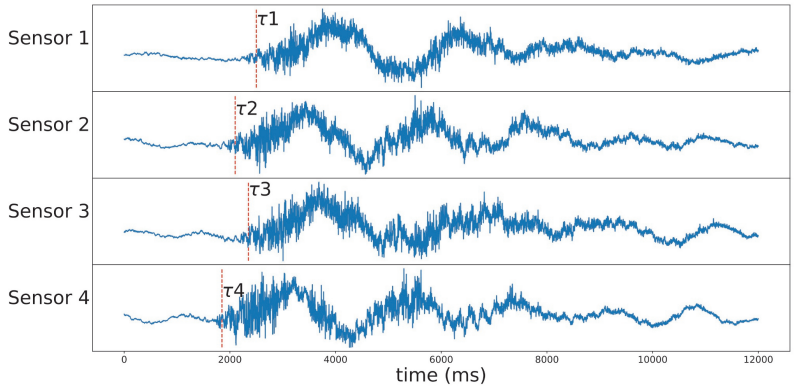


FIGURE 5.2: Cross-correlation gives the time lagging of signals, which can be used for sound localization

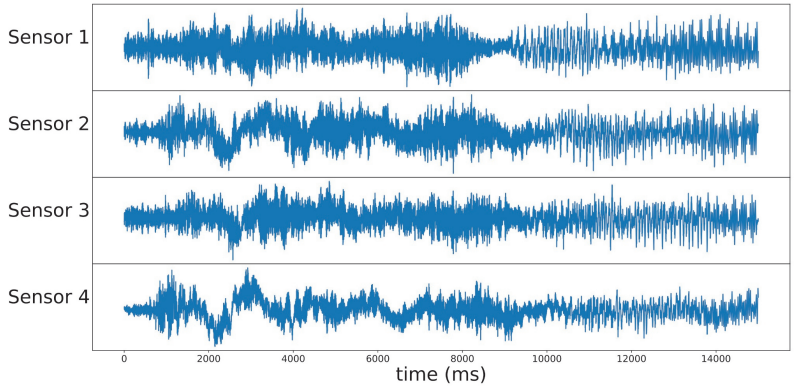


FIGURE 5.3: For overlapping sounds, cross-correlation can not synchronize the sub-components

5.1.2 Keypoint based sound sources decomposition

As described in Section 5.1.1, locating a sound source can be done through synchronizing the time difference τ of multiple sensors. A simple time-domain cross-correlation algorithm can only synchronize τ from single source sounds but not overlapping sounds. However, mixed signals can be distinguishable by applying STFT as illustrated in Figure 5.4, which shows two overlapping events received by 4 microphones. One can easily find the start of a door slamming event in each graph, a long (wide frequency) yet narrow (short time) column. The other signal concentrated in lower frequencies apparently corresponds to the voice bands. While splitting the two events seems easy in this particular

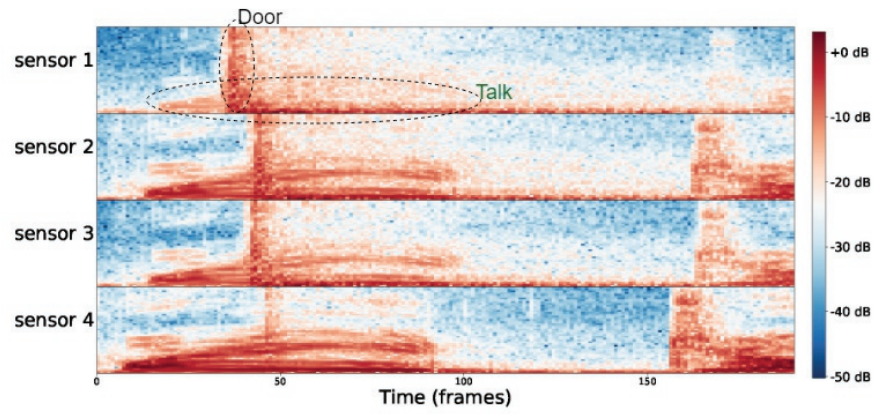


FIGURE 5.4: Spectrograms of sound received by different sensors

case, it is generally hard when these events in spectrograms overlap and are randomly shaped.

We aim to tackle this problem by looking into every small yet significant region in the spectrogram, from which the time-difference-of-arrival (i.e. τ) is easier to be synchronized. There are three steps in our sound source localization algorithm:

5.1.2.1 Keypoint detection

The idea behind here is to find important transients in both spectral and temporal domains, and cluster them to locate the sound events. Our intuition behind keypoints are:

I. The small and power condensed keypoints are easier to synchronize than the entire signal. Since keypoints are power peaks, they are noise-resistant and very likely to be preserved in all nearby sensors. Although some highly overlapping keypoints are difficult to be synchronized, the synchronization can rely on neighbouring keypoints by applying a global model.

II. Keypoints can be used for sound classification as many works have already proposed [107, 108]. Keypoints distribution indicates the major frequency bands of an event. The keypoints variation over time and frequency also describes the local Q-factor information of a small STFT region.

Mathematically, a keypoint is expressed as: $K_i = [s_i, f_i, t_i]$ where $s = 1, 2, ..M$ and M is the sensors number, f and t are the frequency and time. This three-element tuple

means one keypoint is detected at sensor s, time t and frequency f. We define G as the spectrograms of all sensors and K_i as the coordinate so that $G(K_i)$ represents the amplitude of a keypoint.

Keypoints are detected at locations that are local maximum across both frequency and time, which can be expressed as:

$$\begin{aligned} G(K_i) = G(s_i, f_i, t_i) = max(G(s, f, t), thr), \\ \forall s, f_i - h < f < f_i + h,\ t_i - w < t < t_i + w \end{aligned} \tag{5.8}$$

thr is a constant for filtering background and microphone noise, so that no keypoints would be detected in background sound. ***h*** and ***w*** define the local area, height and the width of a spectrogram. To clarify, in a STFT spectrogram of a 20 kbps sound with a window size 1024 of 3/4 overlap, $h = 10$ and $w = 6$ represent a region of 400 HZ and 15 ms. This region must be large enough to capture important shape information and small enough to have sufficient selected keypoints.

5.1.2.2 Keypoint synchronization and localization

We synchronize all keypoints through a two-step model. For each step, there is a different metric or cost function that reflects how good τ is. These two steps and corresponding cost-functions are described below:

Cross-correlation cost is a local cost that resembles the single sound event synchronization process. A keypoint should propagate to a nearby sensor at the same frequency with time-delay τ, at which point the signal cross-correlation should be the maximum.

Mathematically, we define the square-shaped region centered at keypoint K_i as $R(K_i)$:

$$\begin{aligned} R(K_i) = \{[s_i, f, t]\}, \\ f \in [f_i - h, f_i + h],\ t \in [t_i - w, t_i + w] \end{aligned} \tag{5.9}$$

and we define its spectrogram as:

$$GR(K_i) = \{G(p)\}, \forall p \in R(K_i) \tag{5.10}$$

Similar to single event TDOA, we search for the τ in each sensor that correlates $GR(K_i)$ with $GR([s, f_i, t_i + \tau])$ most. This results in the solution of τ using only local **cross-correlation cost**:

$$\begin{aligned} \tau_{s,i} = \arg\min_t(\rho(GR(K_i), GR([s, f_i, t_i + t]))), \\ \forall s \neq s_i, t \in \{0, 1, ...T_{max}\} \end{aligned} \tag{5.11}$$

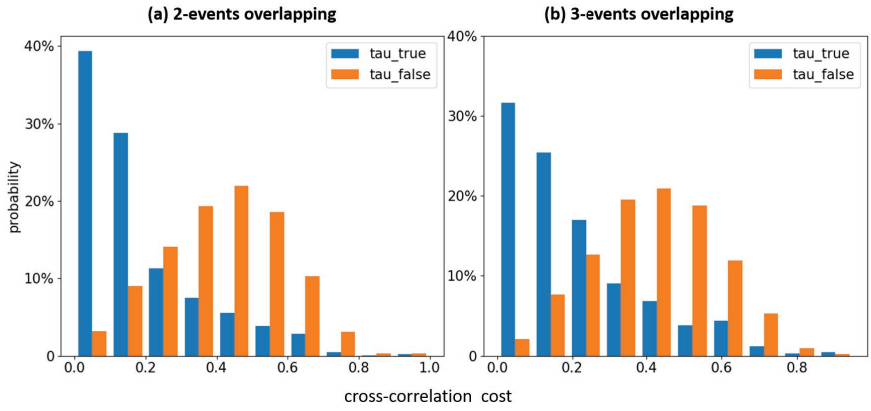


FIGURE 5.5: Joint probability of the cross-correlation cost and its corresponding τ correct rate

, where $\tau_{s,i}$ is the sound arrival time at sensor s for keypoint K_i, T_{max} is the maximum of possible time arrival difference, ρ is the Pearson correlation function [109] normalized to $(0, 1)$ so that smaller cost means more likelihood.

Synchronizing each keypoint independently based on Equation (5.11) can not be perfect because of noise, multi-path and highly-overlapped frequency bands. Through experimental data we can estimate the accuracy of using this method through the joint probability of: $P(\tau\_true, cost_\rho)$ and $P(\tau\_false, cost_\rho)$. $cost_\rho$ is the cross-correlation value at any t-delay while τ_true and τ_false means if t is the correct τ or not.

Figure 5.5 basically shows that if τ was to be 90% correct, the corresponding $cost_\rho$ value from Equation (5.11) should be smaller than 0.1. We next split all τs into 2 halves: the τ with $cost_\rho$ value smaller than the threshold C were fixed, the other τ still needed to be updated by the next step. C works as a split point and is an empirical value. More τ will be fixed by *cross-correlation cost* when C increases, while the error rate of τ also increases. In our experiments, the C was set to 0.1 so that the error rate of this step is estimated to be 10%.

Neighbour-likelihood cost is a global cost that additionally measures the similarity of τ between a keypoint and its neighbouring keypoints. The idea is that a sound event is normally a connected shape in the spectrogram which consists of many closely distributed keypoints with exactly the same τ.

Let KN^n be the n^{th} (i.e. n=1∼4) nearest neighbour of keypoint K, of which a neighbour is an already synchronized keypoint. Here the distance between two keypoints is the Euclidean distance of $[f, t]$ in the spectrogram. Let the τ of neighbour KN^n be τ^n and

we calculate the τ of K as:

$$\tau = \frac{1}{\sum_n S^n} \sum_n \tau^n S^n \tag{5.12}$$

Where S^n is our prediction of whether KN^n and K share the same τ:

$$S^n = \begin{cases} 1, & \text{if } KN^n \textit{ and } K \textit{ share the same } \tau \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

As τ^n is the already known value calculated in the previous step, the task boils down to predict the value of S^n. Next we assign the τ^n of KN^n to K and calculate the $cost_\rho$, which is denoted as $cost_\rho^n$ and its value stands for the local cost when $S^n = 1$. Let $cost_{dist}$ be the distance between KN^n and K, it is natural to expect that the smaller $cost_{dist}^n$ and $cost_\rho^n$, the more likely KN^n and K share the same τ and thus are from the same sound source. In another word, the binary S^n output is decided by the 2-dimension cost $[cost_{dist}^n, cost_\rho^n]$, which can be well solved by a classification model. We next train a linear regression model to solve this problem with our data.

One example of the keypoint synchronization result is shown in Figure 5.6. The different color of makers denotes the different sensors from which keypoints are detected. Round dots stand for the originally detected keypoints as spikes in spectrograms. '+' markers stand for the τ synchronized through the local cross-correlation cost and 'X' markers are the rest τ which are synchronized through the global cost.

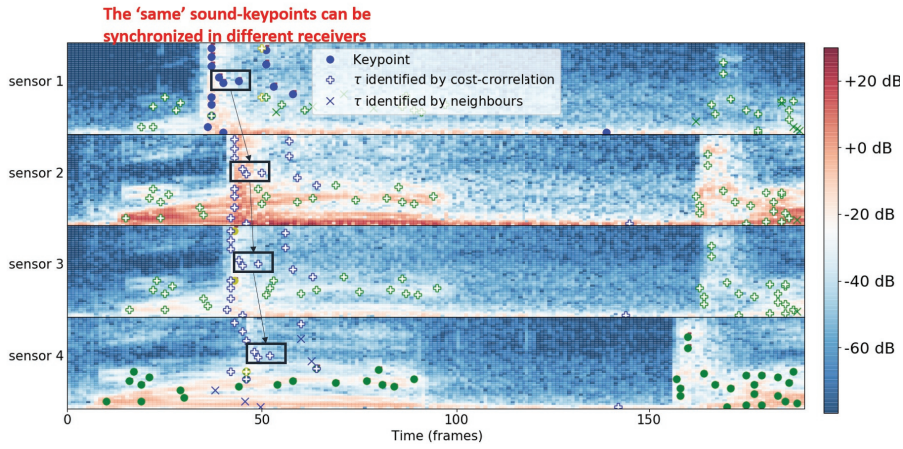


FIGURE 5.6: An example of τ synchronization using cross-correlation cost and neighbour-likelihood

5.1.2.3 Keypoints Clustering

After τ of all keypoints are synchronized, we then cluster the keypoints according to their physical locations to determine the original events, under the assumption that each event happens at a different location. Here we use k-means [110] to cluster the keypoints. In our case, the event number is the cluster number k whose center is the event location. Because the event number k is unknown, the algorithm has to find the best k.

Our keypoints clustering algorithm consists of 3 steps:

I. Find the best cluster-number k with silhouette [111] analysis from the k-means.

II. Filter out outlying keypoints and small clusters.

III. Re-calculating the cluster-centers as the final locations of events with the filtered k and keypoints.

Silhouette analysis in Step I is a method of interpretation and validation of consistency within clusters of data based on a so-called silhouette value. Silhouette value measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The similarity is measured based on the distances to cluster centers. The k with the highest silhouette value is chosen as the optimal cluster number, indicating samples are closer to their own center and are far away from other cluster centers.

Step II improves further keypoint location accuracy because the big errors are filtered. In practice, the τ synchronization errors are normally bipolar distributed instead of normally distributed. Most of them are *correct* and close to the event location. Some keypoints that cannot be correctly synchronized, we consider them *wrong*, are sparsely distributed elsewhere because of the significant error in τ synchronization which is in small portion but inevitable. This is because when multiple sounds sources deeply overlap in both time and frequency domain, τ is often hard to synchronize in every microphone. In this case, the errors in all microphones are accumulated, resulting in a big error in the final localization process. In the location graph we use the 'distance-to-center' criteria to filter them out. Let d_{ci} be the distance of point i to its corresponding cluster center c, we use a threshold D_{thr} to prune the outliers:

$$Discard\ K_i,\ if : d_{ci} < D_{thr}, \forall i \tag{5.14}$$

The threshold D_{thr} is a constant that denotes the maximum localization error to tolerate. This error can come from noise, background sound or even the movement of the sound

source. Another approach to improve the result is to remove the clusters which contain very few members, as they are likely to be made up of falsely synchronized keypoints. The cluster centers are re-calculated based on the remaining keypoints indicating the final locations of the sound events. One example of the clustering result is shown in Figure 5.7. It can be seen that there are two events in this overlapping sound which are decomposed and located (the red dots) beside the real event locations (the blue stars) and it is easy to filter out those outlying error-keypoints. Moreover, decomposing the overlapping signal is the prime goal so that we can actually tolerate the "not so accurate" localization results.

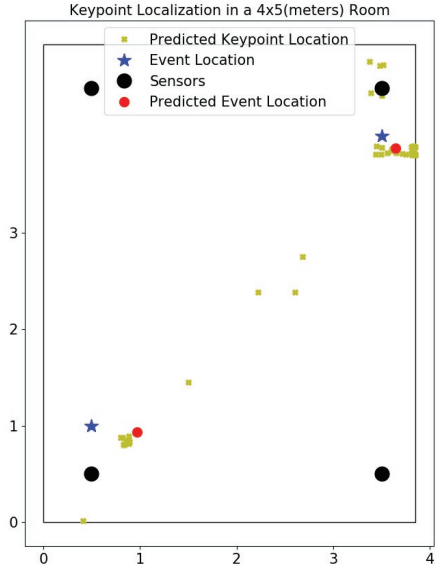


FIGURE 5.7: An example of keypoints localization result

5.1.3 Keypoint based sound event classification

After the event localization, we cluster the keypoints at each event location and classify them into pre-defined activity classes. As for the keypoints classification task, we just adapted the 2 single sound event based methods proposed in the previous chapter, which use framely features and SVM, CNN models. We call the adapted models KP-SVM and KP-CNN, of which the concrete steps are described below:

5.1.3.1 KP-SVM

Our frame-SVM model proposed in Section 4.2 first cuts an audio stream into short duration frames and extracts several time and frequency domain features from each frame. The statistics i.e. mean and variance is calculated from this feature array as the final features. Similarly, here we extract features by replacing the short duration frames with keypoints. A feature set, denoted as F_i, portraying the characteristics of the event both in time and frequency domain, is extracted from each keypoint:

$$F_i = \{f_i,\ Rolloff_i,\ STE_i\} \tag{5.15}$$

, where f_i is the center frequency of the keypoint and $Rolloff_i$ has two values in time and frequency domain respectively. Spectral-rolloff, used in music information retrieval [35], represents the point where N% power is concentrated below that frequency (normally N equals 90%). We define time-rolloff as the time duration where N% power is concentrated at the center. STE_i (Short-time energy) is a feature used in the audio analysis that describes the energy of signal [35].

The feature extraction schemes of the original frame-SVM and the adapted KP-SVM is shown in Figure 5.8.

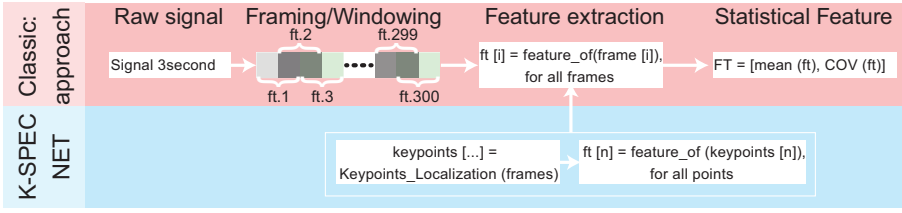


FIGURE 5.8: Feature extraction flow of frame-SVM and KP-SVM

After F_i of each keypoint is extracted, the mean and variance of all F_i are calculated as the event feature. This event feature is then fed to SVM classifier to predict the event type.

5.1.3.2 KP-CNN

Our CNN-L2 model introduced in Section 4.3 uses the log Mel-bands feature and use two stacked CNN layers. It requires very little pre-processing and resources compared to other works using neural network models [39], [41], [47]. The scattered keypoints, however, are not image-like features and do not fit in our CNN model automatically. In order to make CNN work, we use a fairly simple method to build an 'incomplete

spectrogram' out of the aggregated keypoints. The method is to pin every keypoint back to its original places (meaning: time and frequency) and leave the non-keypoints part of the spectrogram padded with zeros.

5.1.4 Experimental results

This section presents the experimental results with a labeled audio dataset for the entire localization and classification model. The dataset is split into 2 parts: training (70%) and test (30%), five folds cross-validation is applied on the training set.

5.1.4.1 Dataset

We here use the same sound events as the ones used for single event classification: speech, clapping, footsteps, cartwheels, door slam, bell. We first select 100 samples from each class from the single-event dataset as the sound pool. As aforementioned, each of these samples contains a clear sound clip of one single event, which is around 2 seconds long. We then used the indoor sound simulation tool [112] to create overlapping sound events from the samples of the sound pool. In total three data groups of different overlapping levels are created, with the level meaning the number of single events inside one overlapping sound event. We use different overlapping levels to evaluate the performance in a different noisy environment. As an example, a level-2 event which consists of one 'footstep' and one 'speech' sound is labelled as *'walk-talk.wav'*, where a typical level-3 event can have the name *'walk-talk-door.wav'*. For simplicity, only sound events from different classes are mixed, i.e. an overlapping event can not have two portions of door sound.

For each data group, we randomly pick from single events to create 1000 overlapping events. During blending, the amplitude and starting time of every single event is changed randomly. Apart from the file name that expresses the elements of an overlapping event, the location of each sound event is also labelled.

5.1.4.2 Sound event decomposition and localization results

Two metrics were used for the evaluation of our method, i.e. all-correct and partially correct. Partially correct means at least one of the events is located correctly. We tolerate a 0.5m error as there is no strict precision requirement in our application scenario since the major objective is to decompose overlapping events through the localization algorithm. Meanwhile, the baseline is only provided for signal decomposition (presented

TABLE 5.1: Localization and classification results for overlapping sound events

Sound localization accuracy using clustered keypoints									
	Level-1			Level-2			Level-3		
	n=1	n=3	n=5	n=1	n=3	n=5	n=1	n=3	n=5
All correct	95.5%	98.2%	96.8%	87.1%	90.3%	89.8%	51.7%	56.9%	57.7%
Partially correct	95.5%	98.2%	96.8%	91.2%	95.8%	94.6%	64.1%	67.8%	70.1%
n = number of neighbours used to synchronize τ									

Results of sound classification using clustered keypoints												
	Level-1				Level-2				Level-3			
	KP-SVM	KP-CNN	CNN2	CNN4	KP-SVM	KP-CNN	CNN2	CNN4	KP-SVM	KP-CNN	CNN2	CNN4
accuracy	89.5%	91.3%	94.7%	95.5%	76.4%	82.0%	54.5%	86.1%	46.1%	61.9%	35.1%	50.4%
F1	89.2%	91.3%	94.6%	94.9%	75.3%	81.5%	53.9%	83.8%	45.0%	61.5%	33.4%	48.8%
Reasoning time	0.61	0.58	0.27	1.23	0.71	0.60	0.27	1.23	0.79	0.62	0.27	1.23
Level-* means: * number of sound events overlap in one sound clip KP-CNN weights:8K, CNN-L2 model weights:8K, CNN-L4 (baseline) weights:8M												

together with the classification results) and not for localization as it is hard to find one for our scenario. Most related works such as [57] and [58] which propose to locate multiple human speaking sounds are designed for microphone-arrays of robotic applications. Apart from the hardware difference, their algorithms are basically based on the DOA(direction of arrivals) principle hence would not work in our case.

Table 5.1 shows the results for three different complexity levels. The level here refers to the maximum number of sound sources in an overlapping sound event. Results show that when only two events are overlapped(i.e. level-2), the accuracy jumps above 90% and drops significantly in all experiments when three events are overlapped. We generate the spectrogram by 20ms frame-length and 4ms hop-length. The threshold C is 0.13 where roughly half the keypoints are synchronized by local correlation-cost resulting in about 90% correct ratio.

Different neighbours-number n in keypoint-synchronization is tested to find the best value in different cases. Our experiments show that it is generally better to synchronize τ with more neighbours in more complicated cases (noisy environment). Moreover $n = 3$ is the best in general to synchronize τ and with too many neighbours also increases the algorithm complexity.

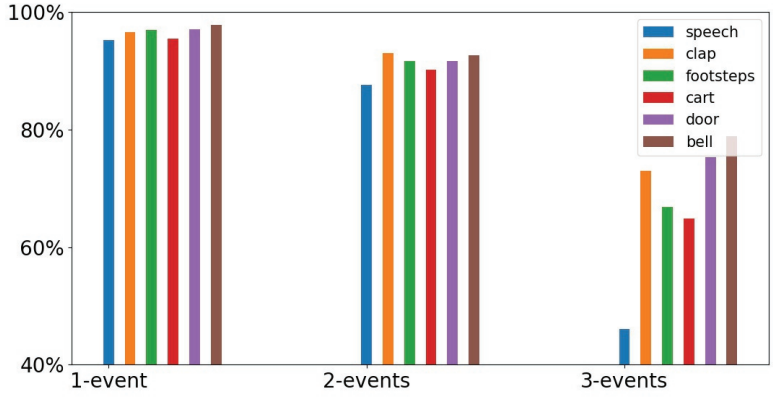


FIGURE 5.9: Sound event localization accuracy for different sound classes (*with 0.5m fault-toleration)

We also looked into the differences in performance between each sound class, as shown in Figure 5.9. These results are all from the attenuation-correlation model and show that bell sound is the best located while human voice is the worst, this difference becomes especially obvious in 3-events experiments.

5.1.4.3 Sound event classification results

In this section, we evaluate the performance of our KP-CNN and KP-SVM models with the aggregated keypoints. Apart from these two keypoint-based models, we also tested the same data on the CNN-based models introduced in Section 4.3. Upon changing the last layer from *softmax* to *sigmoid*, deep learning classifiers can be easily adapted to support multiple classes. We use both the adapted CNN-L2 and CNN-L4 models introduced in Section 4.3 as the baseline. The major difference between CNN-L4 and CNN-L2 is that CNN-L4 has two more full-connected layers and is with 1000 times more weights. While both these multi-label CNN models decompose and classify the overlapping sound in one complex model, the keypoint-based classification model is lighter in complexity to achieve comparable performances. This is because our keypoint based models use cross-correlation to decompose overlapping signals first so that the classifiers only need to classify single-event sounds.

The second half of Table 5.1 shows the results of overlapping event classification for all models. The metrics consist of Accuracy, F1 score and Complexity, as was used in Chapter 4.4.2. The definition of complexity is the same as in Section , which consists of the reasoning time and the memory cost. Reasoning time is the running time of classifying an audio sample divided by the audio length, reflecting the reasoning time of a unit length audio. Among the four models, CNN-L2 runs the fastest overall while CNN-L4 is the slowest (0.27 vs 1.23). The reasoning time of keypoint based models also differs for different inputs, since the keypoints number and processing time would rise with increasing sound events. The memory cost is mainly reflected by the weights of the CNN models which are given in the appendix of this table. One could notice that the baseline model CNN-L4 needs 1000 times more memory than the other two, which is obviously not suitable for IoT applications. In summary, keypoint-based classification models are largely simplified as they only handle single sound inputs, while the synchronization algorithm contains straightforward calculations at large thus costs merely a small overhead.

Among models with similar complexity, both of our keypoint-based models outperform the CNN2 model in classifying overlapped sound events, with KP-CNN being the best. KP-CNN is slightly worse than CNN-L4 in classifying level-1 and level-2 events and is better than KP-CNN in more complicated level-3 events (58.5% vs 47.1%). The major advantage for keypoint-based models comes from the previous localization step which already split overlapped signals into single events, thus, simplifying the problem. As a result, our classification accuracy is quite close to the localization accuracy. While CNN-L4 and CNN-L2 have similar performance over single sounds, CNN-L4 has shown much better results than CNN-L2 in more complicated sounds. This is reasonable since

it has almost 1000 times of weights from CNN-L2, making it more beneficial in higher complexity problems. In conclusion, it shows that a concise CNN model is able to differentiate single sound events, but would fall short with overlapping sounds when the pattern in data becomes hard to find.

The confusion matrix of the best model KP-CNN is shown in Figure 5.10. It shows that the performance of each class is equally good for single event classification and significantly different for overlapping events. Bell event is the most accurate, as its major frequency bands are much higher than the rest and can be easily identified. Speech sound is most likely to be misclassified, as its frequency bands all concentrate in fairly low-frequency bands and overlap with all other events. The good performance of clapping in 3-events group is likely from its non-stop duration across the whole event so that enough keypoints are able to be correctly located. The confusion matrix of other models are separately displayed in Figure 5.11 and 5.12.

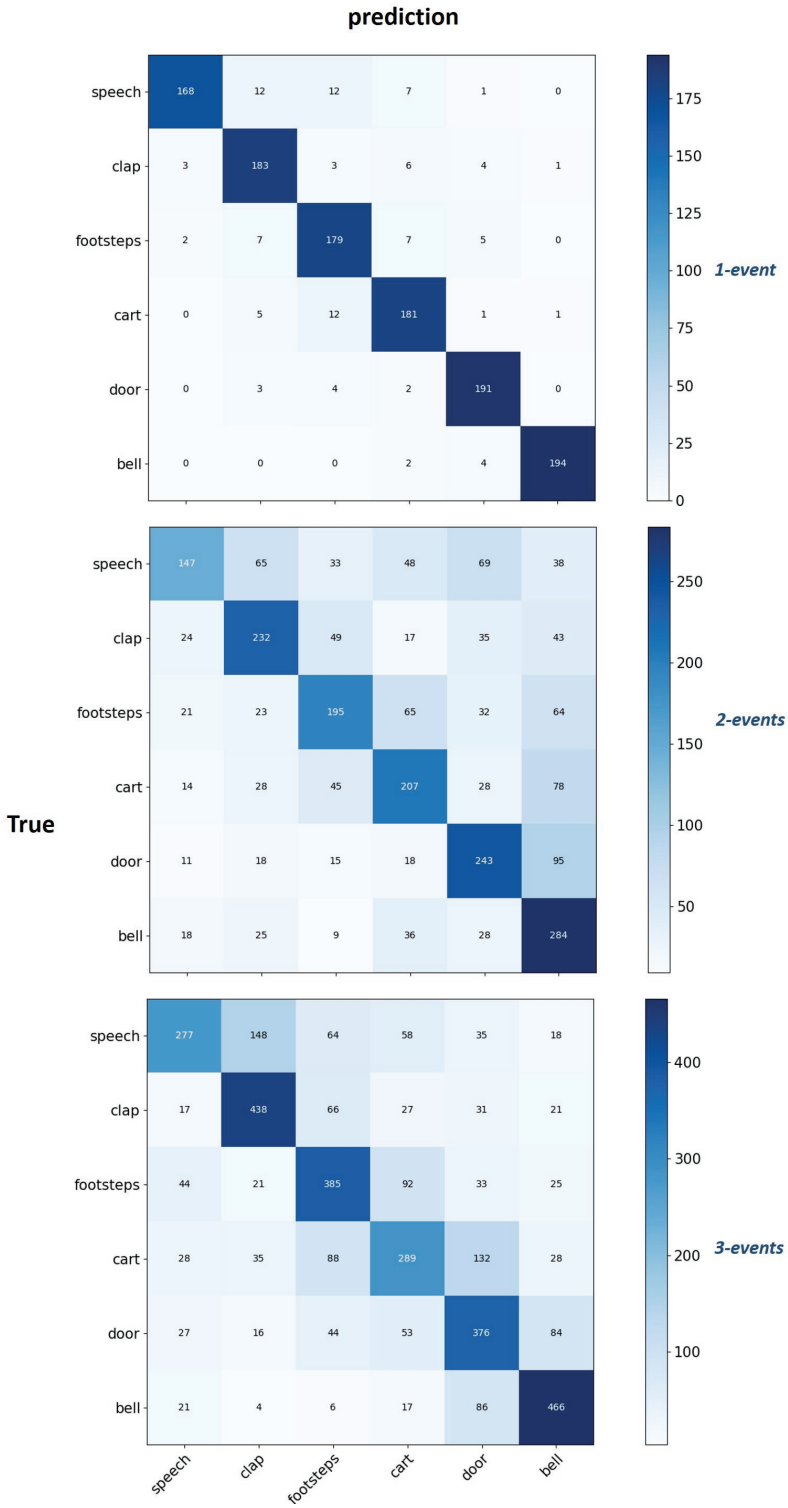


FIGURE 5.10: Confusion matrix of KP-CNN results from different groups (up)1-event (middle)2-events (down)3-events

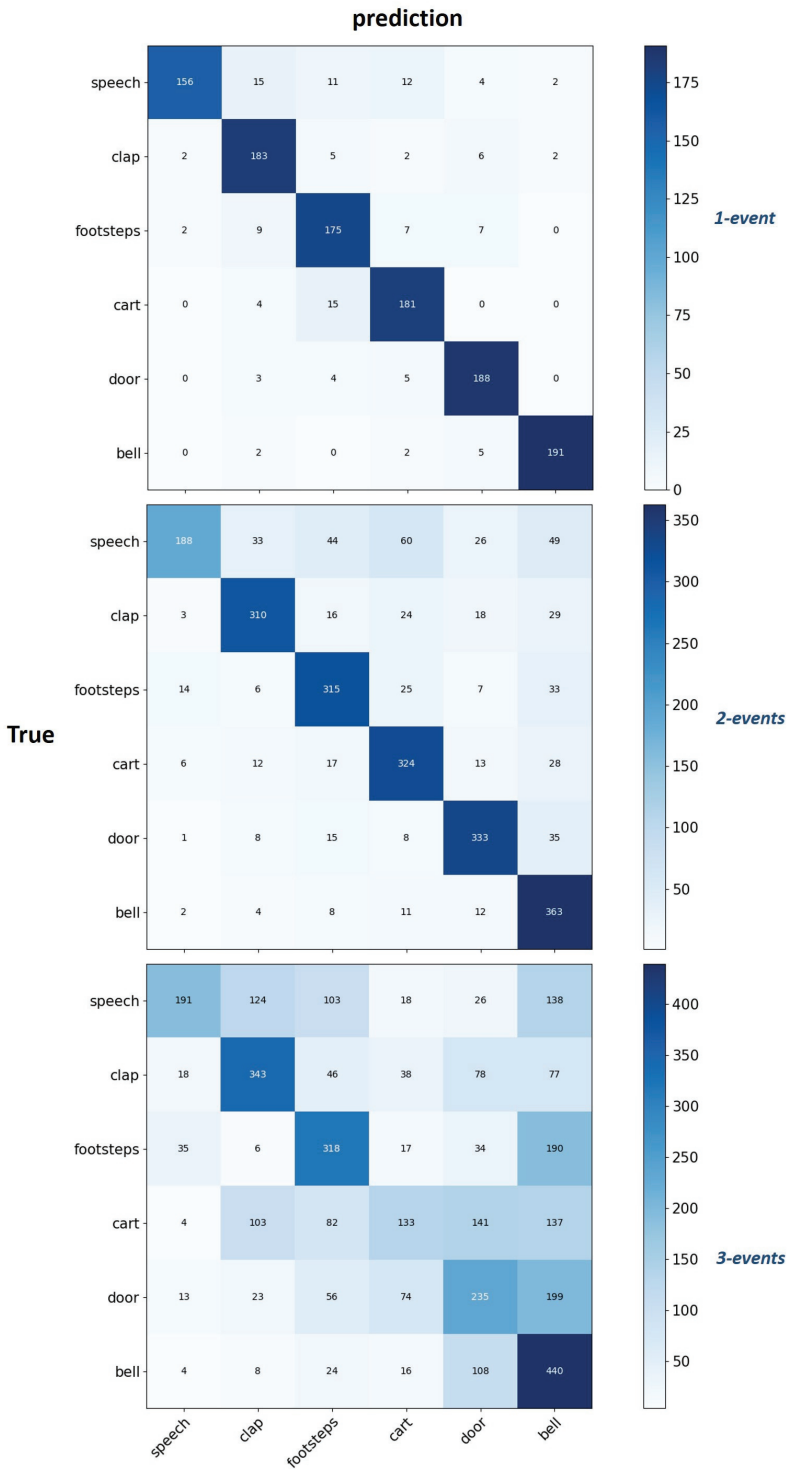


FIGURE 5.11: Confusion matrix of KP-SVM results from different groups (up)1-event (middle)2-events (down)3-events

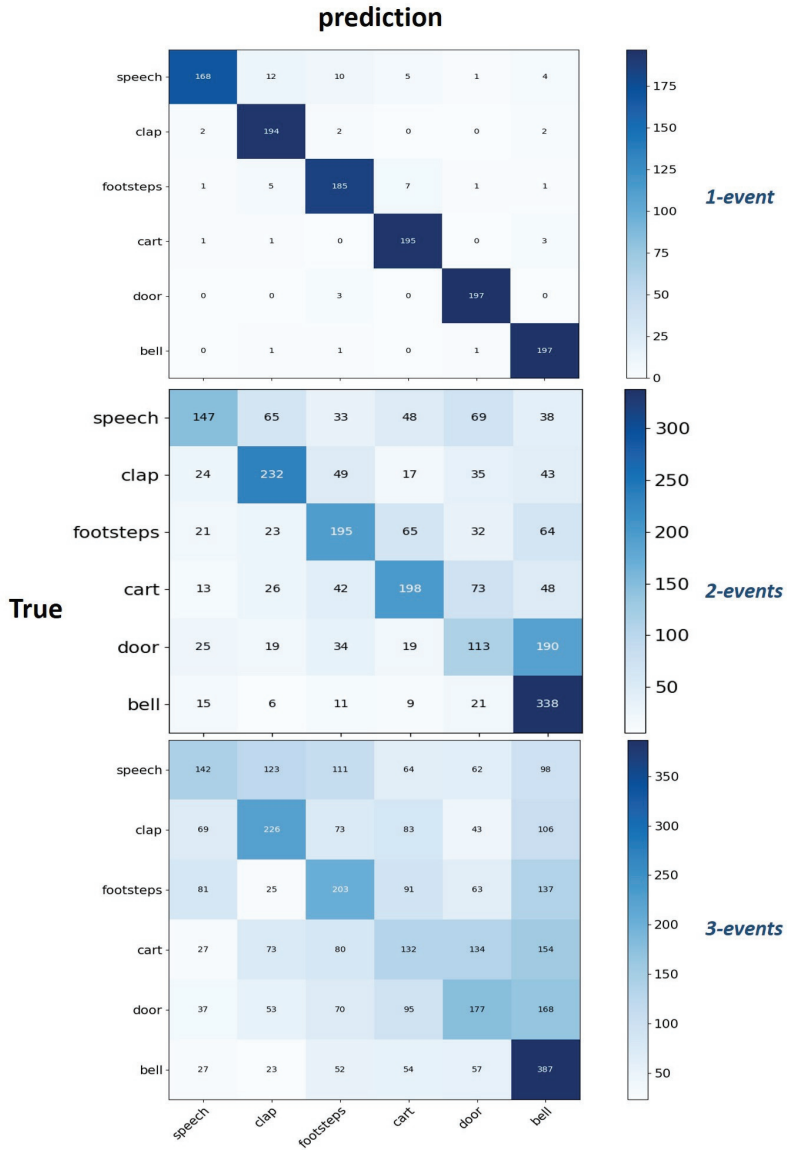


FIGURE 5.12: Confusion matrix of CNN2 results from different groups (up)1-event (middle)2-events (down)3-events

5.2 Crowd Activity Recognition

In the previous section, we have presented a method to decompose and classify overlapping sound events in a crowded environment. Like other proposed methods, our classification model can tackle the overlapping events problem to a certain extent, but would definitely fall short in a more crowded environment. However, it is unknown that if the existing models are applicable in the crowded environment since the audio dataset used by these researches have at most two or three sound events overlapping [47] [49] [51]. Moreover, a classification model only tells whether something is happening, but not how many events are happening simultaneously. In a crowded environment, these results barely improve our awareness of the environment since some events may happen all the time. Take cocktail parties as an example, knowing how many people are talking tells more about the atmosphere than knowing if someone is talking.

To fill in the gap, this section proposes to estimate the proportion of different activities from overlapping sound events. The rationale consists of telling what activities happened at a specific time, by building a regression model that outputs the ratio of each perceived activity in a relatively long duration. The model is more feasible than precisely detecting the activities of the individuals, still providing insights into what is most probably happening in that particular place and time.

The remainder of this work is organized as follows: Section 5.2.1 explains the methodology used for our crowd activity recognition and the intuition behind our model. Section 5.2.2 describes our dataset and the performance evaluation of our models and the baseline.

5.2.1 Methodology

Our overall aim is to build a model to estimate the proportion of different activities or sound events using the input audio stream. Apparently, what we need is a regression model which outputs the percentage of each event class. With this model, we can infer what are the main things happening around us and whether there are anomalies. Here the proportion of one sound event is defined as the accumulated duration of that event by the accumulated duration of all events. Figure 5.13 shows an example of how we count the proportion of different sound events in the crowd. In $t1$ there is only one person talking so that $r_{talking} = 1$ and $r_{walking} = 0$. In $t3$ two are talking and one is walking so that $r_{talking} = 0.66$ and $r_{walking} = 0.33$.

As this is a complex multi-class regression problem, we continue to use neural network models as our solution to this problem. We have designed two different neural network

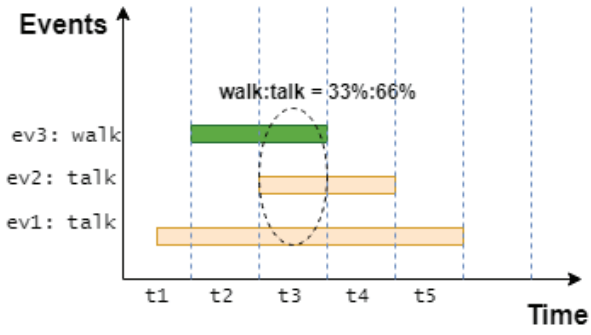


FIGURE 5.13: Scenario description: our model outputs the proportion of sound events

models, both of which are derived from the already introduced CNN model. The first model uses a 4 hidden layer framework, similar to the CNN-L4 model in Section 5.1 since it has shown much better results than CNN-L2 in complicated overlapping sounds classification. In general, a CNN layer is good at detecting the local characters of a sound spectrogram, but falls short for the global or statistic characters [113]. More specifically, a CNN-based model can learn and find repetitions of some short and sharp sound fragments all concentrated at 4KHZ-5KHZ as the 'footsteps', but can hardly learn the knowledge of 'more low-frequency sound (300HZ-3KHZ) means more people are speaking'.

In order to learn statistic characters better, we have designed the second model which additionally takes the global FFT-bands input on top of the first model. The details of these two models are described below.

5.2.1.1 The CNN regression model

This CNN-based model takes the frame-based Mel-spectrogram features as the input and outputs the proportion of each sound event. We here use the original Mel-spectrogram feature as the inputs, which is a two-dimensional feature with time and frequency axis. The setting of this Mel-spectrogram feature is also commonly used in other works [41]: the frame length is $100ms$ and half overlapped and the Mel-bands is 60, so that the Mel-spectrogram feature of a 10 seconds audio stream would is of size 60×200.

A CNN based model with 4 hidden layers is then used to learn and estimate the ratio of each sound event from this image-like feature set. This model is depicted in Figure 5.14 of which the input features from the top are processed to get the regression results by the neural network. This CNN-based model is much bigger in size than the neural

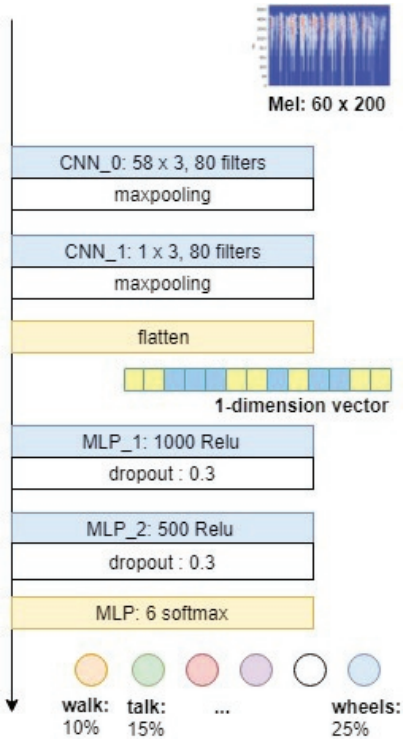


FIGURE 5.14: The CNN-based model for event-proportion prediction

network model for single-event classification used in Chapter 4, as it needs to retain much more information to represent the diverse features from overlapping sounds. In order to find good hyperparameters, we start with the similar CNN layer settings from [41] then gradually tune them through experiments. The details of each layer with one of our best hyperparameters is shown as follows:

1. *Convolutional neural network (CNN)*: Our model begins with two stacked CNN layers. A CNN-layer consists of multiple 2-dimensional filters of the same size(i.e. 5×5), which is also called the *filter-size* of a layer. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter and the input to produce a 2-dimensional output of that filter. As a result, the network learns filters that activate when it detects some specific shapes (or features) at any position in the input. In the output of a CNN-layer, each neuron only receives a subarea of the input (of the same size as the filter-size), which is also called the *receptive field* of the neuron. Therefore, the receptive area of a CNN-layer is much smaller than the entire input layer, which greatly reduces the calculation density than a full-connected layer.

We stacked two CNN-layers together so that the second layer can detect a larger shape or feature from the spectrogram. The layer's parameters mainly consist of: filter-number, filter-size and the strides of the convolution. For the first CNN-layer, a good setting is 80 filters and a filter-size of 58×6 with stride 1×1. The height of the filter-size (58) should be only slightly smaller than the entire frequency-bands (60) so that the learned feature is not much frequency-invariant. The second CNN-layer could have 80 filters and the filter-size of 1×3 so as to detect a larger size of the feature in time domain than the first CNN layer.

2. *Maxpooling*: In neural network, a CNN-layer very commonly accompanies a maxpooling-layer. A maxpooling layer applies a max filter to (usually) non-overlapping sub-areas of the input neurons. This layer can downscale the learned features from the CNN-layer and help with the overfitting by suppressing the influence of the smaller outputs.

   In our model, the maxpooling-layer size in time-axis is exactly the same as that of its previous CNN-layer so that no important features are lost.

3. *Flatten*: A flatten layer does not change the value but only reshapes the multi-dimensional matrix into a vector. This makes the data able to feed into a fully connected layer which is better suitable for the classification or regression task.

4. *Rectified Linear Units (ReLU)*: We use two stacked full-connected layers with *ReLU* activation functions for the regression of the flattened data. ReLU is a non-linear function which is widely used as the activation function in neural network [97]. The basic ReLU function has the form of:

$$ReLU(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise} \end{cases} \tag{5.16}$$

   Compared to the traditional sigmoid activation function, ReLU has several advantages such as faster computation, more efficient gradient propagation and sparse activation, etc. Many variants of the ReLU function have also been invented recently such as the Leaky ReLU, Noisy ReLU, or Parametric ReLU to solve potential problems [114, 115]. However as many papers have proposed, the actual differences in performance brought by these variants are very small and they do not grant better performance than the basic ReLU function [116]. For each of these two layers, the neuron number is always selected from between its input size and its output size.

5. *Dropout*: It is natural the neural network would likely overfit and learn the statistical noise from the training data, while performs poor with the new coming data.

Dropout is a simple yet efficient technique for reducing overfitting in neural networks. In the training process, neurons of the dropout layer would be randomly set to zero at a given ratio (i.e. 30%) so that it looks like some neurons are dropped. It has the effect of making the training process noisy and encourages neuron values learned within a layer to be more sparse. We attached one dropout layer to each ReLU layer and set the drop ratio from 30% to 50% in the experiments.

6. *Softmax*: Our output layer is the full-connected layer with the softmax activation function. Softmax function can normalize the input into a probability distribution like output where all values are positive and sum up to 1:

$$S(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{5.17}$$

Softmax is often used for classification but is also suitable for calculating proportions for its mathematical properties.

5.2.1.2 The Concatenate Regression Model

Our second model is a concatenate model of both frame-based STFT and the global frequency features, aiming to solve the problem by looking into both the local and holistic characters. As our problem is to calculate the proportion of each event and does not need revealing the details in each short time segment, it is possible that the global frequency bands may well correlate with the results. Based on the previous model, we add the Fast Fourier transform (FFT) results on the entire audio stream as the input. Only the magnitude of the FFT is used so that the data length is half of the audio stream.

The architecture of our model is shown in Figure 5.15, which is based on both the frame-based STFT and the global FFT features. In this model, each input feature is processed by a separate sub-routine in the upper (or lower-feature) layers until concatenated by the last hidden layer.

The architecture of the STFT sub-routine is exactly the same as the previous CNN-based model before the concatenate layer. For the FFT sub-routine, we first connect the input with the maxpooling layer to reduce the dimension of data for less overfitting. Next, we use two ReLU layers to extract the high-level representations for the final regression step. There are two layers specifically for the concatenation of the two sub-routines:

1. *Normalization*: Normalization is a family of methods that normalize the output value of the previous layer, i.e. applies a transformation that maintains the mean

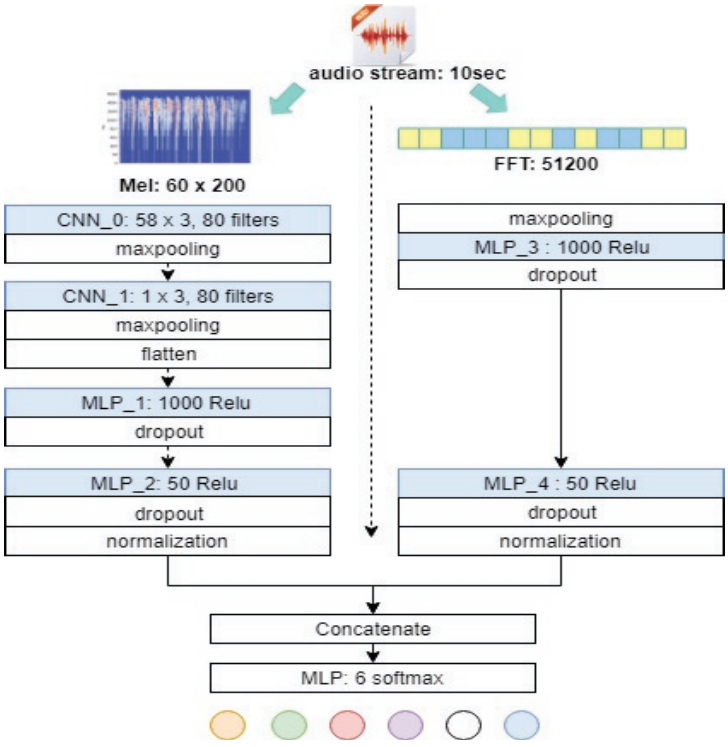


FIGURE 5.15: The concatenate model for event-proportion prediction

close to 0 and the standard deviation close to 1. This technique is normally used to improve the speed, performance, and stability of artificial neural networks. In the implementation, we use the BatchNormalization layer proposed by [117]. Before the concatenation of the two sub-routines, a *BatchNormalization* layer is attached to each side. The normalization layers force the neurons of both sub-routines subject to the same distribution thus are more balanced in their contribution to the subsequent concatenation layer.

2. *Concatenate*: A concatenate layer can take inputs from multiple layers and merge them into a new vector. For example, if each sub-routine has 50 neurons, the concatenate layer would output 100 neurons to the following layer. In neural network, this method makes the framework more flexible and comprehensive since it allows to handle multiple types of input. In one of the recent works, Ahmed et al. proposed a house price prediction model with both the photos and text-description inputs [118]. Their model achieved much better and stabler results than the models with either image or text input.

The final softmax layer then receives the concatenated neurons and calculate the final regression results for each event type.

5.2.2 Experimental Evaluation

In this section, we present the dataset and the experimental results of our two models from different aspects. In each experiment, we split the dataset into 80% training-set and 20% test-set and all results presented refer to the output of the test-set.

5.2.2.1 Dataset

As the crowd sound events are hard to collect and label, we here create the simulation dataset for the evaluations, using the same single sound events as the sound pool, i.e which include: speech, clapping, footsteps, cartwheels, door slam, bell. These sound events are selected since all of them are very commonly heard indoor and are related to different human activities.

With these single events, we next created the crowd sound samples by the simulation tool pyroomacoustics. Pyroomacoustics is a simulation tool for indoor acoustic environments, which can simulate the sound heard by people in a complex environment. Together we generated three groups of audio samples with different crowded-density, noted as crowd-4, crowd-6, crowd-8. The group name refers to the average overlapping rate, i.e. in crowd-4, 40 seconds of different sound events are mixed into 10 seconds so that there are 4 events overlapping in each moment in average.

We generated 20000 samples for each group and every sample is 10 seconds long. For the mixing part, the number or total length of each single sound event subjects to the same uniform distribution.

5.2.2.2 The hyperparameters

Before the experiments on different data, we need to choose a good set of hyperparameters, most of which are the filter numbers and size of each neural network layers. We tested different sets of hyperparameters and chose the best one for all our experiments, which is shown in Table 5.2. Basically, the concatenate model need to have fewer parameters in each layer compared to the CNN-model and also needs less dropout ratio since the global FFT features also contributes to the regression. Another important configuration is that ReLU_2 and ReLU_4 should have roughly equal neurons so that the concatenation is not too much biased to either sub-routine. For each experiment,

TABLE 5.2: An example of the hyperparameters tested in experiments

CNN-model		**Concatenate-model**	
CNN_1 filter number	80	CNN_1 filter number	60
CNN_1 filter size	57 x 6	CNN_1 filter size	57 x 4
CNN_2 filter number	80	CNN_2 filter number	60
CNN_2 filter size	1 x 3	CNN_2 filter size	1 x 3
ReLU_1 neuron number	1000	ReLU_1 neuron number	200
ReLU_1 dropout	0.3	ReLU_1 dropout	0.3
ReLU_2 neuron number	500	ReLU_2 neuron number	100
ReLU_2 dropout	0.2	ReLU_2 dropout	0.1
		ReLU_3 neuron number	1000
		ReLU_3 dropout	0.3
		ReLU_4 neuron number	100
		ReLU_4 dropout	0.1

TABLE 5.3: Overall results of crowd-activity prediction with different data-groups

(data-groups)	crowd-4		crowd-6		crowd-8	
	RMSE*	R2*	RMSE	R2	RMSE	R2
Concatenate-model	0.00094	0.7377	0.00143	0.6237	0.00204	0.4594
CNN-model	0.00112	0.7011	0.00175	0.4996	0.00263	0.303
Control-model*	0.0022	0.4191	0.00236	0.4095	0.00241	0.4029

*RMSE and R2 refer to the average value of all event class

*The control-model refers to the simple full-connected model with global-FFT feature inputs

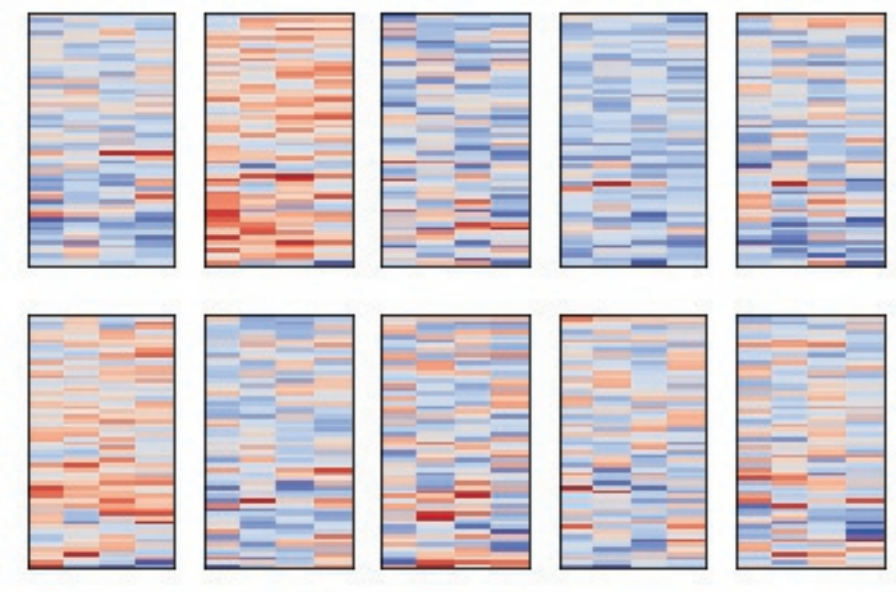


FIGURE 5.16: Examples of the features learned from the first CNN-layer

we set the maximum training epochs to 2000 with the learning rate of 0.001 and the early-stop is set to 100 epochs.

For the first CNN-layer (layer CNN_1), there are 80 filters learned, which are of the size 57×6. Part of these learned CNN-features in one of our experiments are shown in Figure 5.16, which are quite sparse in frequency bands.

5.2.2.3 The results

Table 5.3 shows the comparison results for each model in different data groups. For comparison, we also built a very simple model using the global FFT features and full-connected layers, i.e. the FFT sub-routine of the concatenate model. In the experiments, this controlled method performs significantly worse than our proposed models.

We use the root mean square error (RMSE) and R2 score as the metrics for evaluating the regression results. Both of which are commonly used indexes for evaluating regression results. R2 or R-squared score is known as explained-variation versus total-variation and is commonly used for measuring the differences between the real values and the predictions. R2 score can be explained as:

$$R2(r,p) = 1 - \frac{\sum_i (r_i - p_i)^2}{\sum_i (r_i - \overline{r})^2} \tag{5.18}$$

, where r and p refer to the real values and the predictions respectively. i is the index and $\overline{r}$ is the mean value of all r. The best possible R2 score is 1.0 when all the predictions and real values are equal and R2 can also be negative (because the model can be arbitrarily worse). A constant model that always predicts with the mean value of r, disregarding the input features, would get a R2 score of 0.

In all three groups, the concatenate model has shown remarkably better performance than the CNN-model. Compared to MSE, R2 score can present the regression results in a much more intuitive way as its output range is normalized so that is regardless of the input value range.

In order to reveal the impact of input audio length, we also tested the three models on shorter audio clips. In these experiments, we use the same audio content but cut each of the 10-seconds clips into 5-seconds and 3-seconds segments and tested with the same models and hyperparameters. Figure 5.17 shows that 10-seconds audio clips perform slightly better and more stable than 5-seconds and 3-seconds clips, proved that longer inputs can basically predict better. Furthermore, the performance of all three models can drop very dramatically when the audio-clips become too short (i.e. 3 seconds).

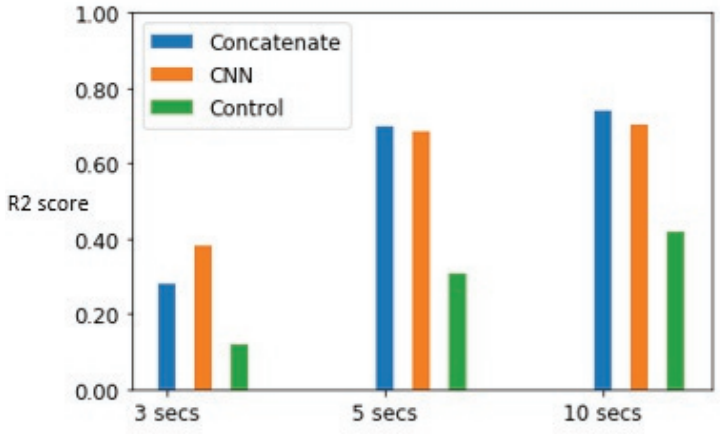


FIGURE 5.17: R2 score of crowd-4 with different input audio length

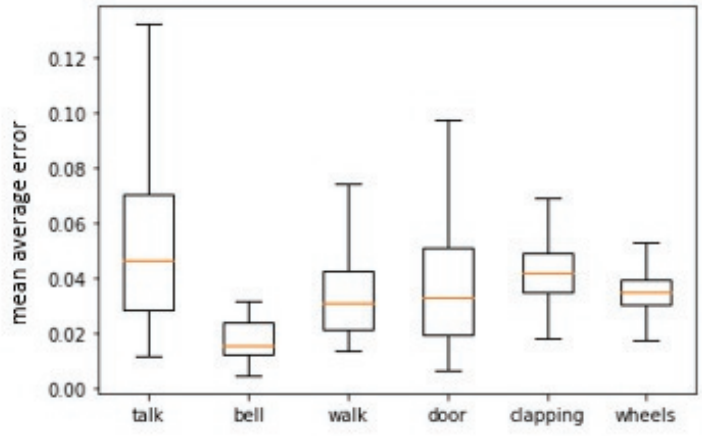


FIGURE 5.18: Per-event regression errors of crowd-4 with the concatenate-model

The per-event results of the concatenate-model is shown in Figure 5.18. In this figure, the per-class prediction errors of the crowd-4 data is shown with the boxplot. In our experimental results, the talking or the speech sound has the biggest error and the bell and door sound events are most accurately predicted. The inaccurate prediction of speech sound in the crowd is reasonable since human speech sound largely concentrates in the 300HZ - 3KHZ and overlap with the spectrogram of all other sound events. Furthermore, the noise in the environment also mainly concentrate on the low-frequency bands which further make the speech sound hard to distinguish.

5.3 Conclusion

Automatic audio signal processing is no more a new research topic in artificial intelligence and signal processing. However, to apply the sound events recognition techniques in real-life applications, one of the major challenges is the impact of high overlapping sounds. A lot of objects can make sounds in daily life and make the environment noisy, especially in a crowded environment. Therefore, it is inevitable for a sound-based activity recognition model to consider the overlapping sound. To tackle the sound overlapping issue, we have proposed two novel techniques where each aims at different scenarios.

In a less noisy environment where only two or three sound events occasionally overlap, we propose a multi-microphone system that decomposes the overlapping sound events into their sub-components. Compared to other popular models, this method is more scalable since it separates the mixed signal based on the sound propagation model, while most models depend on the extensive training of sound event mixtures.

In our experiments, this model has also shown significantly better performance and less complexity than the baseline methods. With the two-event overlapping sound, more than 80% of the events are correctly decomposed and located within 50 cm error. The advantage is more obvious with the three-event overlapping sound, where the KP-CNN model has a 12% higher accuracy than other models. Needless to say, this localization precision cannot be compared with microphone-arrays techniques that can reach millimetre level precision. One major reason is that the TDOA τ is calculated in discrete frames and not in real-time, where the bigger the frame-size the less localization precision would be. However, we chose not to further improve the precision since our major concern is the classification of overlapped sound events.

There are also limitations to using this multi-microphone based model in real-life applications. Firstly, it is best suitable for a two-event sound classification when enough keypoints can be correctly synchronized. One can already see a significant drop in performance in the three-event group. Additionally, this model could also suffer from challenges of all TDOA-based localization techniques, such as multi-path, room echo and system-time synchronization.

To explore sound recognition in a more crowded environment, we then propose the second technique: a regression model for crowd activity recognition. This model basically improves environmental awareness by understanding the statistics of sound events, rather than accurately identifying individual sounds. It is especially useful in a very crowded environment where even human hearings can not differentiate the sound sources. Instead of separating sound signals and recognize each individual event, this model directly estimates the proportion of each event. By common sense, the 'proportion' of sound events

should be better inferred by the statistic features rather than the transient features. This hypothesis is also demonstrated by our experiments, as the concatenate neural network model with both local STFT bands and global FFT bands works better than the CNN-based model with STFT only. Our experiments also show that it is better to predict this proportion with a long duration of sound, i.e. more than 5 seconds.

Although there are very few researches using sound for crowd activity recognition, we do think it has many potentials in real applications. Obviously, seeing is believing, sound is hardly as accurate as images in monitoring people's behaviours, especially in a noisy environment. On the negative side, sound has less and not as clean information than images. On the positive side, using sound is cheap and less intrusive. A sound based system can monitor the crowd activities without knowing any individual's information. With very little cost and privacy worries, businesses such as grocery stores and shopping malls are free to collect crowd information to improve their business strategies and floor plans.

Chapter 6

Sound-based people counting

In the previous chapters, we showed several sound events classification methods, which can distinguish speech sound from other environmental sound events. Compared to other environmental sound events, speech sound has a pervasive nature and posses a large amount of information. For privacy concerns, we are not going to investigate the content of the speech. However, on the other hand, these speech sound can be used to count the number of people using an appropriate model.
In this chapter, we will present a method to count people using the extracted human speech. In a smart building, this method can be used to count the number of people in a specific location, according to individual voices regardless of the speech content. More specifically, this work addresses the scenarios with an equal distribution of speaking persons and total occupants. The main content of this chapter is published in PERCOM 2020 [119].

6.1 Introduction to speech sound based people counting

The people counting technique is a popular and widely researched topic in substantial human-centric IoT applications [7]. Knowing the number and mobility pattern of customers can help retail businesses adjust their marketing plan [120]. In shopping malls, renters can decide the rent charges based on the areas of concentration and number of people [121]. In smart buildings, knowing the number of residents at specific time-frames can help save utility cost from a smart HVAC(heating, ventilation and air conditioning) controlling system [7]. People counting problems can be addressed using a single or a multitude of sensors, e.g. from cheap PIR sensors, WiFi receivers, to cameras with computer vision techniques [122].

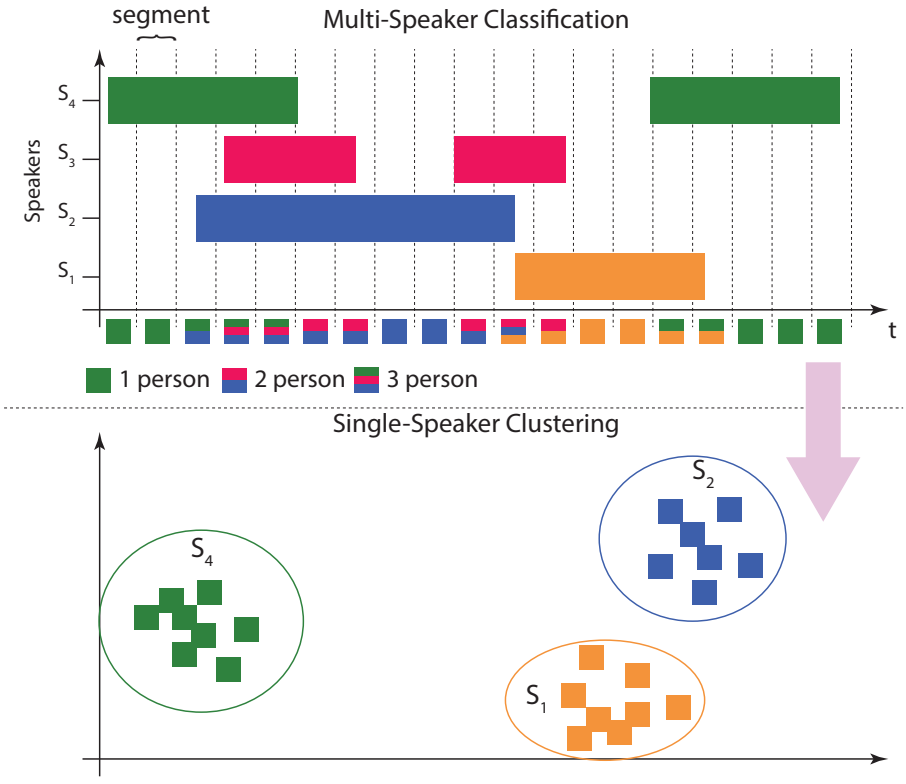


FIGURE 6.1: Scenario description for speaker counting

In this chapter, we propose a people counting system using text-independent human speech. We count the number of people in a specific location, according to individual voices regardless of the speech content. Speech sound has a pervasive nature and posses a large amount of information. Whatsmore, microphone is relatively cheap among all the sensors used for human counting. On the other hand, sound has disadvantages, especially in quiet environments, like auditoriums where the number of speakers is less than the total number of persons. To clarify, this work aims at the scenarios where everyone speaks.

To achieve this we address two related challenges: First, we estimate how many people are speaking at each time segment (*e.g. 2sec*). For this, we propose a classification model that yields the number of speakers even when voices overlap, (with a detection rate of maximum of 10 simultaneous speakers regardless of their identities). Second, all the classified single-speaker segments from the previous step are clustered into identity clusters. For this, an unsupervised single-speaker clustering model is proposed, based

on the identity-correlated feature(i.e. characteristics of the voice) so that the number of clusters corresponds to the number of speakers. Preceded by the overlapping speaker classification model, our approach should be more robust than previously proposed cluster-based counting models [73][72], which neglect the uncertainties inherited by crowded environments. Moreover, counting multi-speakers can provide additional information on the minimum number of speakers, which can further increase the counting accuracy. Figure 6.1 shows an overview of how these two combined approaches provide an estimate of people in that precise environment.

As stated in Chapter 2.2, it is the trend nowadays to use deep learning with raw signals as inputs instead of hard code audio features to address human voice related problems [73][71][76]. To get the good input features for our classification and clustering, we use the *Transfer Learning* paradigm which makes use of the training results from supervised learning models. With transfer learning, the features are not calculated by a rigid mathematical formula but are derived from the knowledge in a well-trained model that aims at a similar problem. More specifically, our features are based on the bottleneck layer neurons of the *SincNet* model [76], which is a supervised model that classifies speakers with their voices.

The rest of this chapter is organized as follows: Section 6.2 gives an overview of the Transfer Learning paradigm and explain what knowledge we transfer from *SincNet* into our model. Section 6.3 describes the methodology used to tackle the problem of distinguishing individual sounds in different contexts, as well as sub-models in each subsection. Section 6.4 describes the dataset used and the experimental results of the sub-models respectively. We conclude this paper with our open discussions in Section 6.5.

6.2 Transfer learning and *SincNetBN* feature

Representative learning paradigm allows us to reuse the functions and vectors learned by trained deep learning neural networks on other related problems, such as the case of this very problem. This methodology is called *Transfer learning*[113]. From a higher perspective *Transfer learning* can be described as the technique used to accelerate the training of new neural networks, either for better weight initialization or as a feature extraction method for another model. The latter characteristic is what we leverage in our work.

Regarding transfer learning in our work, the weights of a successful model in the field of supervised speaker identification could be a generalized feature that highly correlates to the voices of speakers, especially when the voices pool for training is large enough.

By projecting the voices into the learned weights, we would reasonably assume that similar voices would result in similar outputs. This way the new trained model benefits by decreasing the training time as well as having a smaller generalization error. The concept of *Transfer Learning* is successfully proved by many computer vision works when using the bottleneck-layer, normally fully connected as input features for new tasks alike [123] [124].

6.2.1 SincNet Speaker Identification Model

For this purpose we identified *SincNet* speaker recognition model [76] and use the weights extracted from output of the bottleneck-layer. Speaker recognition, or speaker identification, is the identification of persons from characteristics of voices, through the training of text-independent speeches [125]. It is usually a supervised learning problem since all speakers should be heard before they can be identified. Although clustering speakers without prior knowledge of the speakers is an unsupervised learning problem, we can make use of the knowledge learned from speaker recognition, as they both need human voice related features.

SincNet is trained on enormous and representative data and demonstrated a high accuracy. It uses a novel CNN-based architecture that accepts raw audio input and encourages the learning of meaningful audio-specific filters [76]. SincNet is designed on the idea of implementing multiple parametric sine filters with a deep neural network. In contrast to a standard CNN, which learns all elements of each filter, SincNet only learns low and high cutoff frequencies from the data. This offers a very compact and efficient way to derive a customized filter bank specifically tuned for the desired application. In experiments conducted on the speaker recognition task with very few training data($<$ 15 seconds per person), SincNet outperformed other models in both accuracy and efficiency [76]. The ability to accept raw data as input reduces the information loss associated with most classic audio processing models that require extracting engineered features from MFCC or STFT transformations. The detailed composition of the *SincNet* model for speaker recognition is shown in Figure 6.2 [76].

Figure 6.2 describes the speaker recognition model with associated *SincNet* layers, followed by conventional layers [76]. A *SincNet* layer is a modified CNN layer with parameter constraints on the learned filters. Compared to standard CNNs, where every parameter of the filterbank is directly learned, the *SincNet* only learns the low and high cutoff frequencies of band-pass filters.

In audio processing, a standard CNN layer performs as the impulse response filters (FIR) on the time domain waveform. One CNN layer consists of many filters of the same size

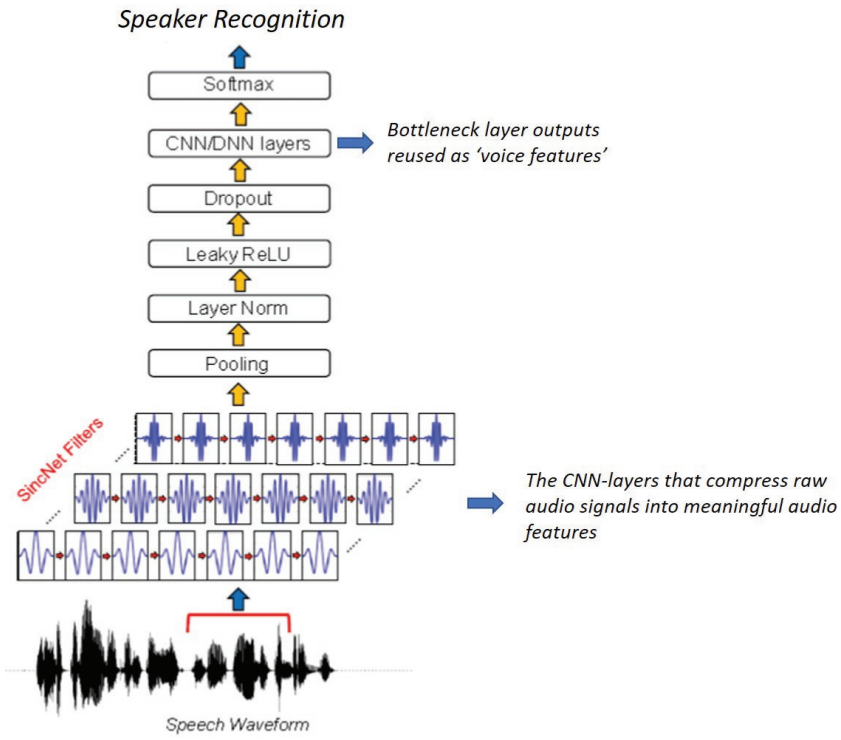


FIGURE 6.2: SincNet model of speaker classification

and for each filter in a CNN layer we have:

$$y[n] = x[n] * h[n] = \sum_{l=0}^{L-1} x[l] \cdot h[n-l] \tag{1}$$

, where x, y, h are the input, output and filter respectively and L is the filter size. During training, all parameters of the filter h need to be trained. Mathematically, the filters learned by CNN often take noisy and incongruous multi-band shapes which is not an issue for solving a general problem, but do not appear to be an efficient representation of the audio signal.

A *SincNet* however is based on parametrized Sine functions and a trained *SincNet* layer stands for a set of band-pass filters of which only the low and high cutoff frequencies are to be learned. For a *SincNet* layer, the input x is convolved with a predefined function g which has two learnable parameters:

$$y[n] = x[n] * g[n, f_1, f_2]$$

, where g has the form:

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n)$$

and when transformed to frequency domain (using Fourier transform) g has the shape of a rectangular band-pass filter:

$$G[f, f_1, f_2] = rect\left(\frac{f}{2f_2}\right) - rect\left(\frac{f}{2f_1}\right)$$

, where $sinc(x) = sin(x)/x$ and $rect$ is the rectangular function. The only parameters to be learned from a filter are f_1 and f_2, which are the low and high cut-off frequency of the band-pass filter g.

As the ideal band-pass filter requires an infinite number of elements L and g is with limited length, a windowing function is also applied to smooth out the abrupt discontinuities at the stop band. So eventually, g becomes:

$$g_w[n, f_1, f_2] = g[n, f_1, f_2] \cdot w[n]$$

, where w is the Hamming window function.

In the training process, the cut-off frequencies f_1, f_2 of each filter can be initialized randomly in the range [0, $f_s/2$], where f_s represents the sampling frequency of the input signal. These characters do not only save the training speed but also enables the model to learn much bigger filters than standard CNN.

With this scheme of predefined filter g, this *SincNet* layer could automatically learn a set of band-pass filters which are much more meaningful and interpretable than standard CNN for audio processing problems.

6.2.2 SincNetBN feature extraction

SincNet splits long audio segments into 200ms frames and use majority voting over the whole segment results after the soft-max layer to classify speakers. The weights of the bottleneck layer (previous to the soft-max layer) can be deemed as a speaker-identity associated feature.

Originally, its bottleneck-layer which is full connected has 2048 units, which means the feature-length would be 2048 if we use the bottleneck-layer weights as our feature. However, this feature would be too long and make the clustering algorithm un-robust. To keep the feature-length as short as possible, we retrained the *SincNet* model with

different bottleneck-layer units. Between 256/512/1024/2048, we chose 512 since it keeps the high accuracy in classifying speakers while reducing the complexity. The experimental results of choosing different nodes are shown in Section 6.4.

To use these weights as the feature for clustering an audio segment, we also need to adapt the length difference. The weights or feature extracted from $200ms$-frames can not be directly used for our clustering model since we need one feature per-segment only while a segment has multiple frames. Whatsmore, we can not simply concatenate these short-frame features not only because it's too long but because human speech segments could be quite diverse in duration, so that the length of the concatenated feature would not be unique thus is difficult for machine learning models to handle.

To both unify the length and compress the data, we propose the *SincNetBN* feature which uses the statistic (i.e. mean and variance) of all frame features as the global feature that represents each audio segment. Let ft_i be the bottleneck-layer weights of frame i, the unified feature of the entire audio segment is:

$$ft = [Mean(ft_i), Cov(ft_i)]$$

With the 512-length weights extracted from each $200ms$ frame, any audio segment would have the feature-length of 1024 regardless of the segment duration.

As the short-frame feature correlates with the speaker-identity, the statistic feature *SincNetBN* not only unifies the feature-length of audio segments, but also inherits the information about the number of speakers in an overlapping sound. Thus, it can be also used in the multi-speaker classification task.

6.3 Methodology

This section describes the methodology and integration of the aforementioned *SincNetBN* into our model. We first give a brief overview of the speaker-counting model and explain in detail all sub-components.

6.3.1 System overview

The flow-chart of the entire system is shown in Figure 6.3. Our system takes the audio stream as the input, and outputs the speaker number over time. Altogether it consists of 4 sub-functions as depicted:

1. segmentation,

2. feature extraction,
3. multi-speaker sound classification,
4. people clustering and counting.

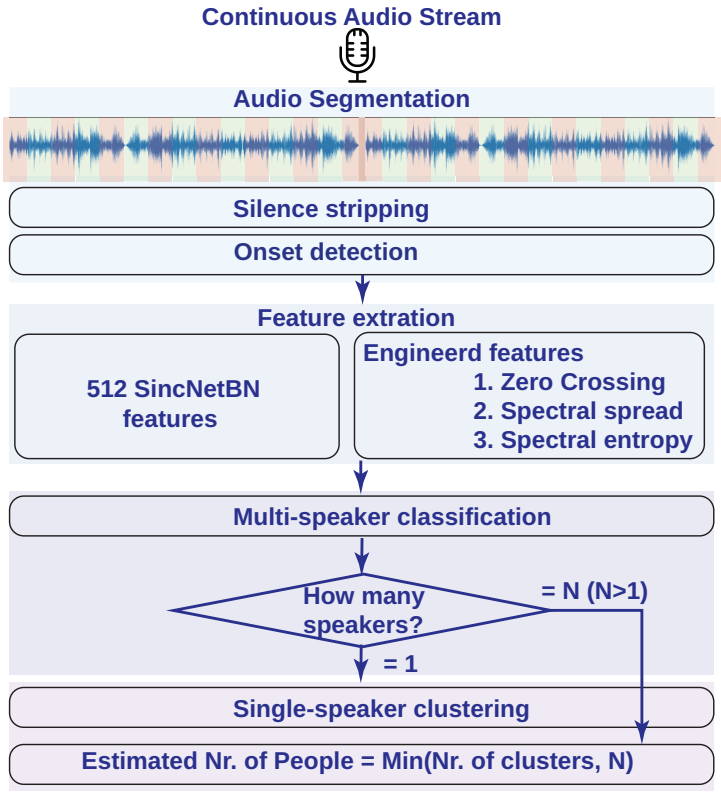


FIGURE 6.3: The flow of our speaker counting system

Audio segmentation refers to recognizing and extracting the duration of speech sound from background noise (or silence) in a continuous audio stream and cut it into small segments. Usually, the audio stream is cut into fixed-length segments using shifting windows and then filter out the unwanted ones [73] [72]. However, in a real-life environment, this 'fixed-length' segmentation scheme would miss a lot of 'useful' sound clips since people do not always speak at the beginning of a window and the length may be shorter than the window length.

To address this problem, we use a 'flexible-length' segmentation scheme which first detects the onset and the end of a speech sound. This method is similar to the one introduced in Section 4.1.2: (1) The audio stream is smoothed in time domain, and cut into fixed short frames (20ms) while the power is calculated for each frame. (2) The

frames with power higher than a threshold are labelled as 'active'. The threshold can be a preset static value or dynamic adjusted. (3) Adjacent 'active' frames are combined to form a segment. (4) Segments shorter than a given duration is dropped, while segments longer than a given threshold are split by needs.

Speech containing segments are then passed through *SincNet* to get the bottleneck weights as features both for multi-speaker classification and single-speaker clustering. The multi-speaker classification model labels each audio segment with the estimated number of speakers, from $1 \sim 10$. The audio segments with only a single speaker are then clustered based on the *SincNetBN* features where each cluster stands for a different person. The final estimated number of people is the minimal number of speakers from the multi-speaker classification model and the cluster number. The details of each function are depicted in the following subsections.

6.3.2 Multiple-speaker sound classification

This model inputs the features extracted from Section 6.2 and outputs the number of speakers. The novelty consists of using and combining *SincNetBN* features with other engineered audio features to increase the performance. These additional audio features are engineered from the frequency domain and listed as follows:

(i) **Zero crossing rate** (ZCR) is the rate at which the signal changes from positive to negative or vice versa. This feature is one of the simplest and widely used in speech recognition, which characterizes the dominant frequency of signal[91].

(ii) **Spectral spread** is the magnitude-weighted average of the differences between the spectral components and the spectral centroid, together they describe how to disperse and wide the frequency bands are [35].

(iii) **Spectral entropy** is calculated as the entropy of spectrum it reflects the flatness but spectrum [92].

As described in Section 6.2, *SincNetBN* contains the information of the number of speakers in an overlapping sound because of its statistic tributes. As a result, we used two lightweight classification models in this task: SVM and a 2-hidden layer full-connected neural network, both of these two classifiers are light-weighted compared to the RNN model used in [69]. We use the RBF (radial basis function) kernel for the SVM model as it is most commonly for non-linear problems. Our DNN model is shown in Figure 6.4, the output layer is a softmax layer that contains 10 nodes corresponding to $1 \sim 10$ persons.

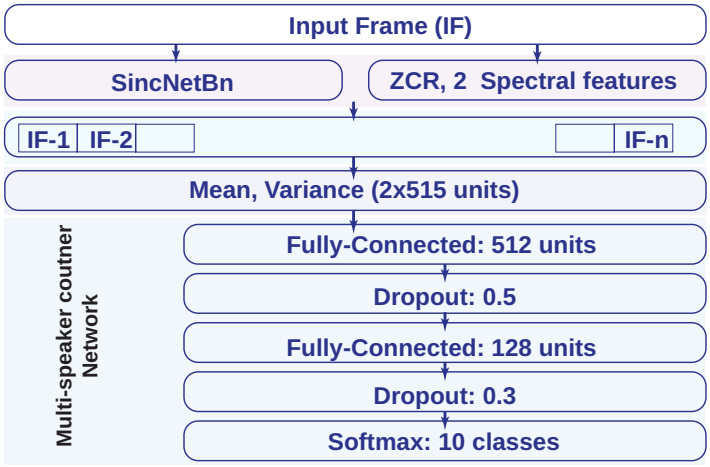


FIGURE 6.4: The multi-speaker classification model with 2-hidden-layers

6.3.3 Single-speaker sound clustering

As has been discussed previously, concurrent speaker-number in the multi-speaker sound is far from a perfect approximation of the total occupants number. The sheer multi-speaker counting has also oversimplified the problem that it keeps no memory of each individual's existence, models like this would make little help to most applications. A better and more appropriate method is therefore to identify and remember 'who is speaking'. With this speaker-identity 'remembered' model, the number of speakers can be approximated more accurately and viewed dynamically over time.

The single-speaker clustering model inputs the *SincNetBN* features and dynamically cluster them over time. Only single-speaker sound is clustered since it is a very difficult task to separate the multi-speaker signal into its sub-components which until now exhibits no perfect solution. A good feature and the corresponding distance (likelihood) function are the two most decisive factor that affects the clustering algorithm. With the *SincNetBN* feature extracted from Section 6.2, we mainly compared 2 different likelihood function: Euclidean distance and cosine likelihood.

In order to count people in real-time, we run the clustering algorithm whenever a new sample comes. Therefore our 'grouping' algorithm should be light-weighted to run on the fly. The intuition of our algorithm of counting people is inspired from [73], which is basically heuristic and straight-forward:

1. Let G_m be the set of samples(audio segments) that belong to speaker-group m, where $m = 1, 2...M$ where M being the group number. Let C_m be the center of the group G_m. The initial speaker-group number M is 0.

2. When a new sample s comes in, calculate the likelihood cost of s to all existing sample-group centers C_m, where we denote as d_m.

3. According to the value of d_m and two constants (θ_0, θ_1) from experimental results, there are three different cases:

   (a) If $min(d_m) > \theta_0, \forall m$, this sample is too far from every group, we should create a new group for this sample.

   (b) If $min(d_m) < \theta_1, \forall m$, this sample then belongs to the group with the minimal distance. We next put this sample to the corresponding group and update G_m, C_m.

   (c) If $\theta_1 < min(d_m) < \theta_0$, we are not confident about which group this sample belongs to, we should discard this sample.

With the calculated N from multi-speaker sound classification and the speaker-groups M, we estimate the speaker number as $Max(M, N)$. In real-time applications that require information about the number of persons in a certain time frame, e.g. the people in the last 5 minutes, this requirement can be simply fulfilled by aging the samples kept in the speaker-clusters G_n.

6.4 Experimental results

This section presents the experimental results which justify our methodology for the entire system and each sub-model. In order to evaluate the model precisely, we use not only statistical methods but also heuristic ones. The overall performance is measured through several public datasets and statistical results, while visualization tools are used to visualize the clustering of voice features.

6.4.1 Dataset

TABLE 6.1: Dataset used for evaluation

	SincNetBN training	**Multi-speaker model**	**Single-speaker model**	**Final evaluation**
Nr. Speakers	300	100	40	40
Nr. Minutes	200	1000	100	480
Source	TIMIT	TIMIT	TIMIT	Libricount

We mainly use two public speech dataset in our experiments: TIMIT [77] and Libricount [79]. The TIMIT dataset is mainly used for model training and Libricount is mainly used for the final evaluation [79].

The TIMIT corpus of reading speech has been designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. The dataset contains a total of 3696 sentences, 8 sentences spoken by each of 462 speakers of which the gender is nearly equal. We deliberately separate the 462 speakers into 2 parts, each is used for the training of SincNetBN features and the multi-speaker sound classification model. The Libricount dataset was released as a synthetic dataset for speaker count estimation. This dataset contains a simulated cocktail party environment of $0 \sim 10$ speakers, mixed with 0dB SNR from random utterances of different speakers from the LibriSpeech 'cleantest' dataset [78].

The concrete data partition is shown in in Table 6.1. 100 speakers data are randomly picked from TIMIT and shuffled for training the multi-speaker classification model. Doing this we ensure that the model training and final evaluation use completely different datasets and prevent data pollution.

6.4.2 SincNetBN feature

This section gives the comparative results of using different bottleneck-layer neurons in a SincNet speaker recognition model. The original bottleneck layer in [76] has 2048 units and achieved very good performance. We gradually reduce the units and retrain the model since less feature-length is better for the later clustering model. Figure 6.5 gives the speaker recognition accuracy with different bottleneck units (keeping other layers exactly the same). For comparison, we also put in the results of MFCC feature together with a GMM model as the benchmark, which is a classic speaker recognition model [126]. As one can see, units of 512 are the best choice as it performs equally good as of 1024 while units of 256 slide a big step in performance.

We also inspected the distribution of the bottleneck-layer weights. All weights of all speakers (512×300) are basically uniformly distributed centered at 0 (min -0.08 and max 0.08). As the output-layer predictor can be roughly seen as the linear combination of bottleneck-layer weights, this means a small altering of the bottleneck-layer (feature) would not change the speaker-identify (output). With this trained SincNet model we define *SincNetBN* feature as the $[mean, variance]$ of the bottleneck-layer weights on all frames, which has the length of 1024.

6.4.3 Multi-speaker sound classification

In this section, we classify the number of speakers in the multi-speaker sound. In our data, the signals of multi-speaker sound are all normalized otherwise the sheer amplitude

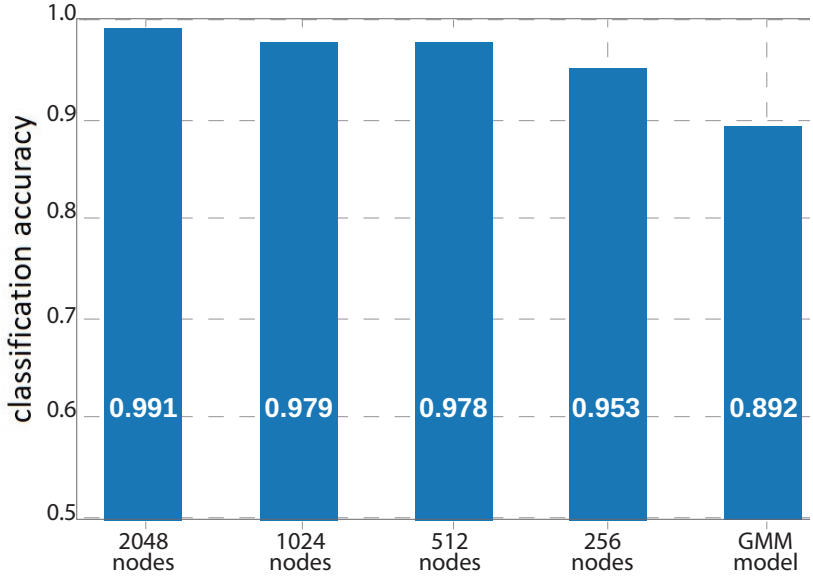


FIGURE 6.5: Speaker-identification accuracy of SincNet with different nodes, also compared with classic GMM model

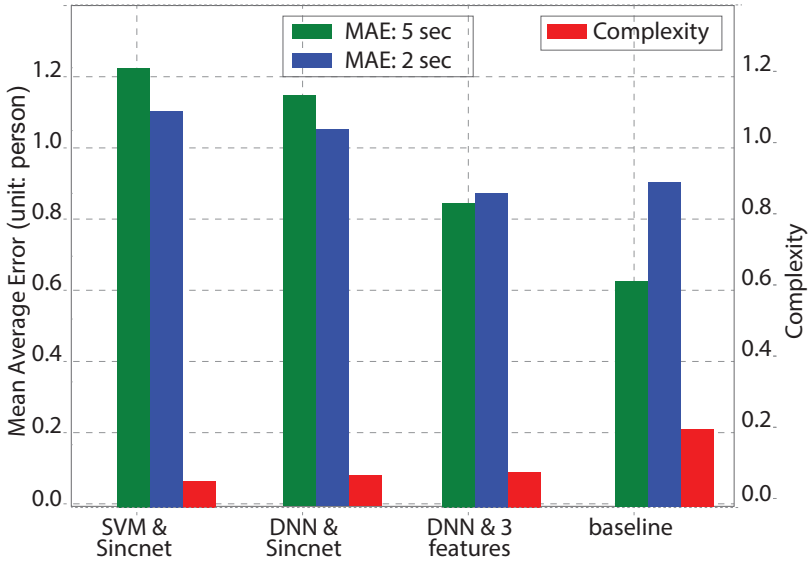


FIGURE 6.6: Multi-speaker sound classification results of different models

could easily correlate to speaker number which however would not be helpful in real applications. We experimented with two different classification models: SVM and a normal DNN model with two full-connected hidden layers. We used the RNN model used in [69] as the baseline model, which takes the STFT spectrogram as the input feature and a RNN based classification model. In order to test the model flexibility, we also prepared test audios with two different lengths: 5 seconds and 2 seconds. 5 seconds is used by the baseline RNN model and 2 seconds is the minimum duration that can identify a person well according to to [76].

At first, we only used SincNetBN feature as the input, afterwards three more simple features were added to achieve better performance. For each model and feature combination, we tested both the performance and the complexity. We used mean average error as the performance metric as it is often used in object counting tasks. Complexity equals the processing time over the audio duration, which is also important in real-time counting applications.

The comparative results of 4 different models are shown in Figure 6.6. With the sheer SincNet feature, DNN performs a little better than SVM model (0.083/0.062 smaller in errors of 5s/2s groups), but both are noticeably worse than the baseline method (¿0.5 error difference in 5s group). After adding two more features, DNN could perform equally good as the baseline in 2 seconds audios (0.878 vs 0.905). Our model is mainly better than RNN-based baseline model in the complexity (¿2 times faster), since RNN is significantly slower than a normal DNN model. Another advantage of our model is the flexibility with input audio length as it is trained with statistic features, unlike the baseline which always inputs the same length audio (5s).

More detailed results of the best model (DNN model and 3 features) are shown in Figure 6.7 and Figure 6.8. Figure 6.7 shows the mean average error(MAE) with different number of people, with 2-seconds and 5-seconds audio data. Figure 6.8 further shows the confusion matrix of the 5-seconds audio data. We could observe a sharp increase in errors when people number exceeds 5 persons in both data groups. Overall, the error increase as the number of people increases, but drops when this number reaches 8. The anomaly of being more accurate in a higher number of people is because of the limitation of the testing method. As all the models are trained with a maximum of 10 people, their output in the test phase would also be lower than 10, eliminating all high-prediction errors. In real applications with an arbitrary number of people, this model can only be less accurate when the number of people increases.

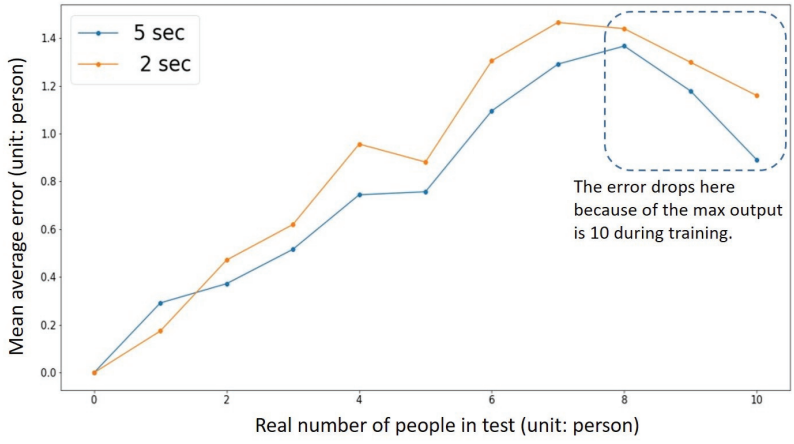


FIGURE 6.7: Results of multi-speaker classification with 2 and 5 second sounds

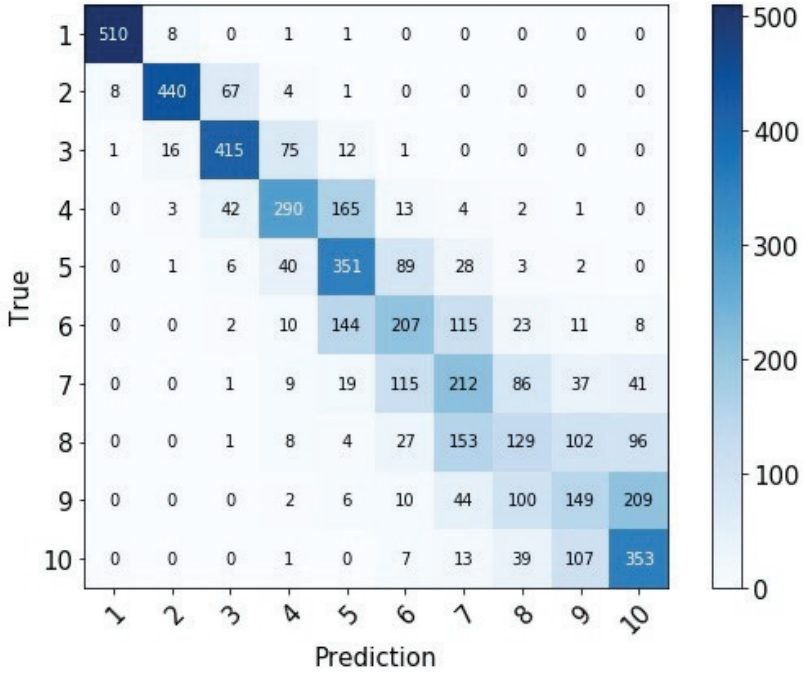


FIGURE 6.8: Confusion matrix of multi-speaker classification with 5 seconds clips

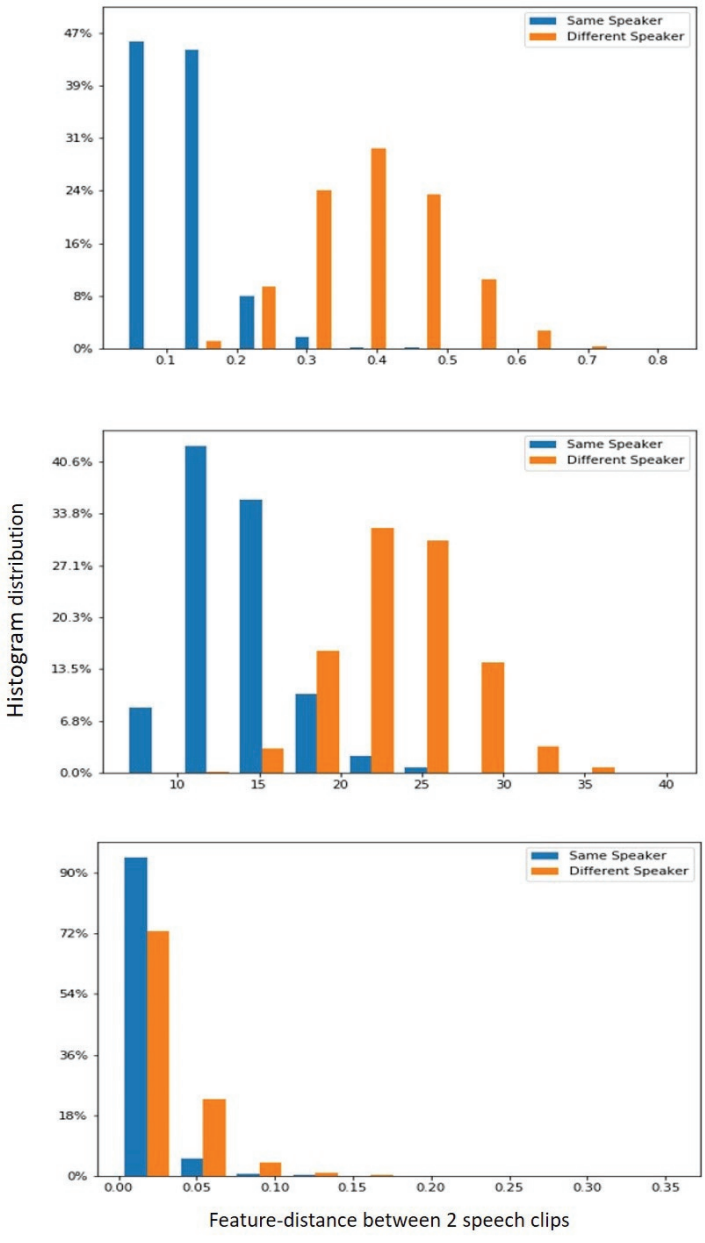


FIGURE 6.9: Feature-distances histogram with different features and cost-functions: (up) SincNet+Cosine (middle) SincNet+Euclidean (bottom) MFCC+Cosine

6.4.4 Single-Speaker clustering and people counting

In this section, we first conducted experiments to justify the ability of SincNetBN feature to differentiate speaker-identity from text independently utterances. As described in Section 6.3.3, a good speaker clustering model depends on both the used feature and the distance function between features from different segments. Speakers can be well separated if the feature-distances from the same speaker are statistically much smaller than those from the different speakers. With SincNetBN feature, we have chosen Euclidean and Cosine likelihood as the distance function in the comparative experiments. MFCC feature plus Cosine likelihood function is also used as the baseline [73]. The distribution of all feature-distances are shown in Figure 6.9. From this figure we could see that the MFCC is much less capable than SincNetBN feature in separating speakers as the distance distribution of two groups are mostly overlapping. With SincNetBN feature, the Cosine likelihood function works slightly better than Euclidean since it only calculates the angle difference between two vectors. This statistics shows that 2 audios must belong to the same speaker if their Cosine distance is less than 0.15 and must be from different speakers if the distance is more than 0.4.

Addition to the probabilistic distribution as a statistic proof, we also plotted the features to give a more intuitive view. We also use t-distributed stochastic neighbor(t-SNE) to visually view the feature-distances. t-SNE is used to to map the samples from high-dimension to 2-dimension for a clearer visualization. t-SNE is a machine learning based dimensionality reduction algorithm which is commonly used for embedding high-dimensional data for visualization in a low-dimensional space[127]. Specifically, it models each high-dimensional object by a low-dimension point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. Figure 6.10 shows this 2-dimensions t-SNE plot of our clustering model,with different distance functions, compared with the baseline function. From which we could see that 40 speakers can be separated quite well with SincNetBN feature while is hard with MFCC feature.

Figure 6.11 shows the counting result by the t-SNE plot of an example. The upper figure shows the ground truth, where 10 speeches are made by 4 speakers interactively which in total adds up to 30 seconds. As one can see, each speech segment has a random duration therefore contains different audio frames. In the lower figure, each valid audio frame is represented by a small dot whose location is calculated from t-SNE. Next, these dots can be clearly clustered into four groups, where each group is also given a different color. As a result, we can predict that there are four people speaking in this speech-data sample, just as the true case is. Figure 6.12 shows another example with a small counting error, where one of the seven speakers is not counted. This error is largely caused by the short

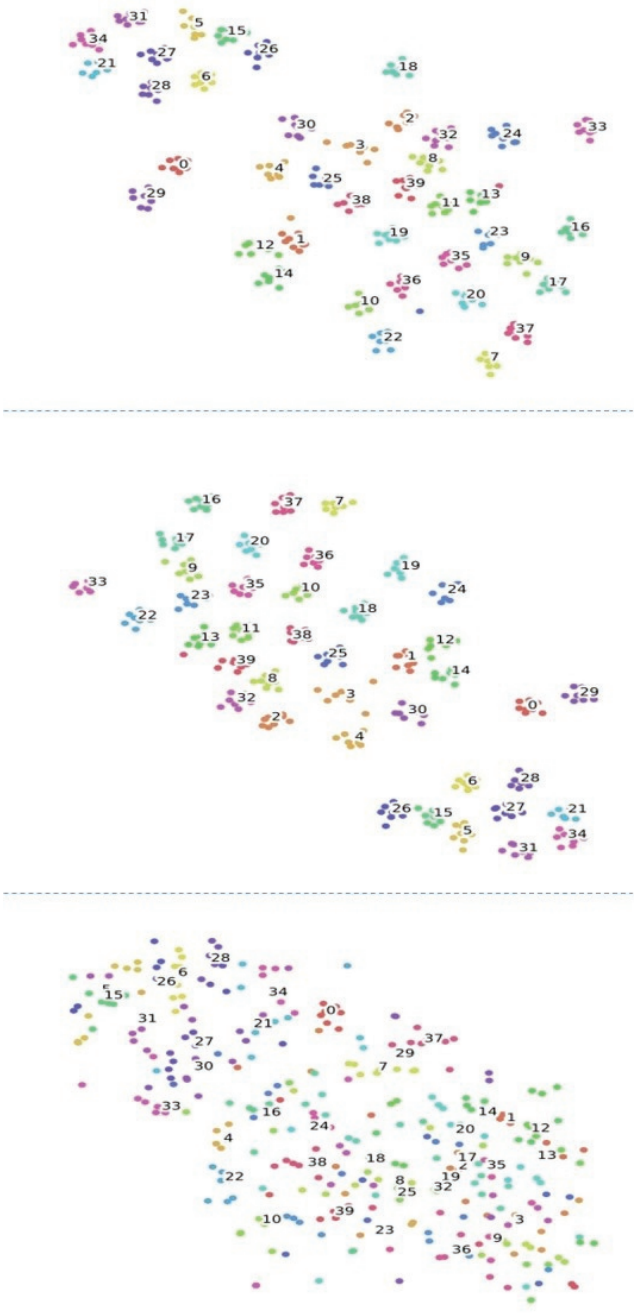


FIGURE 6.10: The t-SNE plot of different features and distance functions: SincNetBN+Cosine (up), SincNetBN+Euclidean (middle), MFCC+Cosine(bottom)

duration of the speech clip, which lasts merely 0.5 second and is falsely classified as a noise rather than a valid speech.

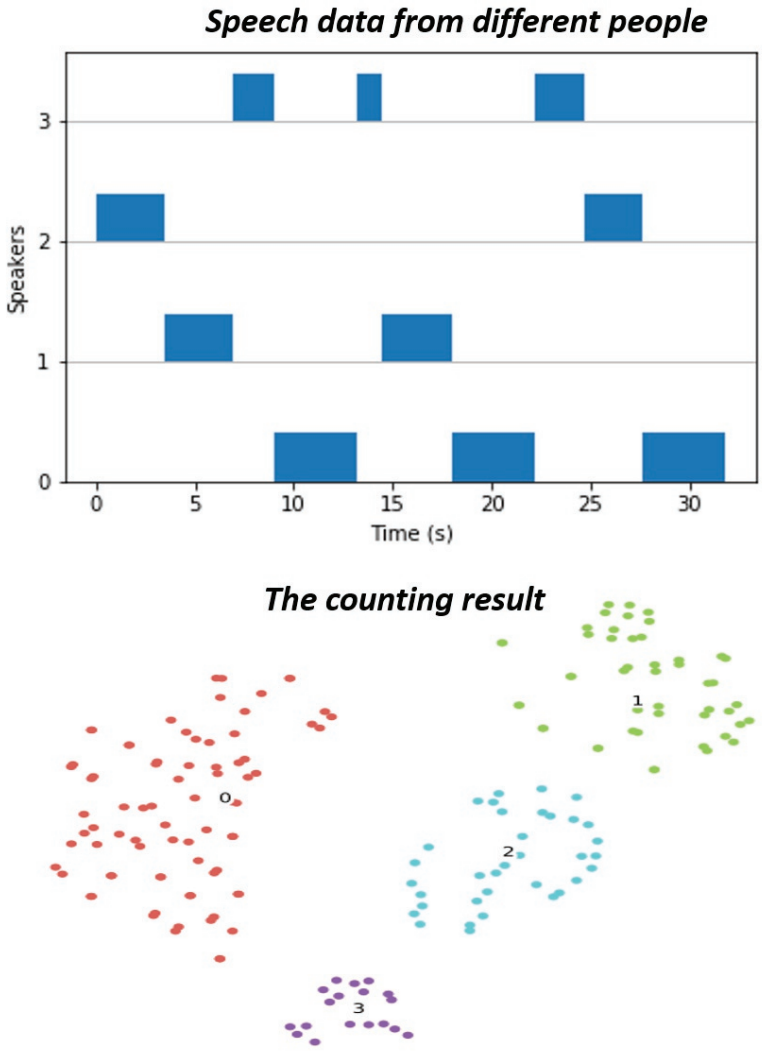


FIGURE 6.11: An example of people-counting result displayed through t-SNE plot

6.4.5 The final results

In the end, we put the two steps together and tested the MAE of the entire people counting system with the libricount dataset. To simulate the real crowded party scenario, we concatenated the discrete audio segments from the libricount dataset into longer audio streams. Each of the long audio stream contains speech sounds made by a different

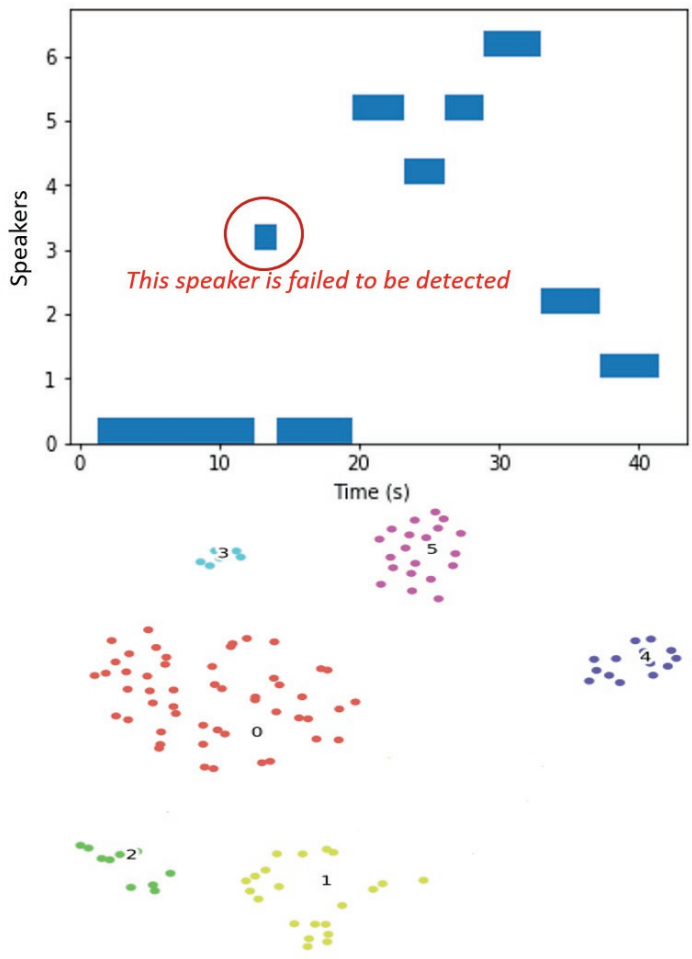


FIGURE 6.12: Another example of people-counting result with a small error

number of people in the range of 5 to 40. These concatenated audio streams contain clean and single voice sounds as well as overlapping speech sounds that made by up to ten voices. Our task is to estimate the number of speakers from each concatenated audio stream. Figure 6.13 shows the statistical results in different experiments from both our model and the baseline in [73]. With the minimal of 5 and maximum of 40 different persons, our method outperforms the baseline in each test. The gap between the baseline method and our method becomes bigger when the number of people increases, which is mainly due to the lack of overlapping voice filtering function in the baseline method.

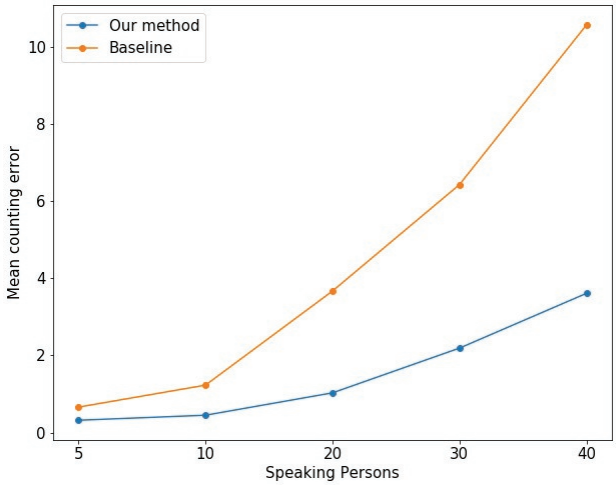


FIGURE 6.13: Final results of counting people with long audio streams

6.5 Conclusion

In this chapter, we proposed the research on counting people with speech sounds. In general, a sound-based people-counting model is not as easy and popular as other people-counting techniques such as WiFi signals sniffing or camera-based image-processing. First of all, sound can not be as easily mapped to the identities of people as WiFi signals, of which each Service Set Identifier (SSID) represents a unique device or a person. Secondly, sound is essentially a one-dimension signal, which is much harder to process than images when multiple sources and noises blend in one signal. However, since using sound is normally cheap and non-intrusive, this research can still provide an alternative solution where certain devices are not appropriate because of privacy or cost concerns.

In order to distinguish people by sounds, a quite natural idea is to use speech sound since people normally have very distinctive voices. Considering the complexity of a crowded environment, we employed a two-step method that handles both single and overlapping voices. The first method directly maps the overlapping voices into a number of speakers, which estimates how many people are talking at the same time. In our experiments, this method only works well when the number of people is very few (i.e, less than 5) and the less the better. Intuitively, this method directly maps the characters of time-varying frequency bands into the number of speakers. According to our common sense, the more complex and diverse the frequency-bands are, the more people must be speaking

at the same time. This method, however, would inevitably fall short when the number of people increases, because the complexity of the mixed signals may become too high to analyze. Using human hearings as an analogy, a human can easily tell if one or two persons are speaking, but is impossible to accurately tell from 10 or 12 speakers.

On the other hand, the clustering of single-speaker provides a more stable and practical estimation of speaker-number over time. It extracts the voice features from single voice sounds, and then cluster the similar features into different groups, i.e. persons. This method is more scalable than the previous one, since the complexity of signals would not increase with the number of people. By continuously monitoring the voice-features, this method can also present a dynamic view of how people might be coming and leaving in a certain time period. Before discussing the details of the methodology, it is worth mentioning that this method is vulnerable to overlapping sounds. An overlapping sound that is a mixture of two sounds is likely to have unique features that are similar to neither of those extracted from its two sub-components. To avoid errors made by overlapping sounds, we can use the method in step one to filter out the overlapping-voice sounds. In our experiments, this works well since the error rate of wrongly classifying overlapping voices into single voices is very low (9 out of 519).

The key core of voice-based people counting is the extracting of voice features. This voice feature needs to present very similar values when being extracted from the same speaker, while being different when being extracted from different speakers. Our intuition of the voice features comes from the paradigm of transfer learning. We reused the weights from the SincNet model, which is a supervised learning model that is able to classify the identity of people from text-independent speech sounds. As a result, we believe that the bottleneck-layer weights would highly correlate the characters of a voice, and is used in both multi-speaker classification and single-speaker clustering. In the multi-speaker classification experiments, a much simpler full-connected DNN model can achieve comparable results to the RNN baseline model. Moreover, this feature helps to dramatically increase the performance of singer-speaker sound clustering compared to the baseline which uses MFCC. In some examples displayed by t-SNE plot, we could also see vividly that the voice-features from different people stay much further than the ones from the same person.

After extracting the frame-based voice-features, we then need to cluster the features into groups where each stands for a person. In our experiments, the performance also depends on many parameters to an extent. An important setting that needs mentioning is the threshold of the likelihood that merges two features into one group, and the other and bigger threshold to split a new group. Between these two thresholds are the features that are neither too sure to be made by the same person nor too certain to be a

new person. In our evaluations, these two thresholds derive from experimental results, using the statistics from our training data, which all come from the same linguistic database. Therefore we can expect that these values need a re-calibration in the real-life applications to fit the context.

This research on sound-based people counting is mostly a proof of concept conducted in an experimental environment. More challenges would arise when applying this model to complicated real-world environments. First of all, although this voice-feature clustering model is an unsupervised learning model, the 'voice-feature' itself is trained from a supervised learning model. Therefore the quality of data, such as integrity and generality plays an important role. A model trained only from normal speed speeches may not be suitable for fast talkers, a model trained on English language speeches is also unsuitable for French speakers. A realistic solution to this issue may be using the incremental learning framework, which continuously absorbs input data into its knowledge base, so as to extend the existing model, i.e. to further train the model [128]. Secondly, using this people counting technique in a very noisy environment still faces many challenges. Basically, the overall accuracy of our method highly relies on the extraction of the single-voice sounds. In a noisy environment, it is quite difficult or impossible to get a clear voice from every person. In a very noisy environment, it may be very difficult or even impossible to get them clean voice from every person. Additionally, many sound events may also exist in crowded environments to contaminate the voice features, therefore reducing the accuracy.

Chapter 7

Conclusions and future work

In our thesis, we mainly aim to address two problems: sound-based activity recognition and people counting, both of which can promote energy saving in smart buildings. This research is mainly a proof of concept, which aims to fill the gap between the existing audio processing technologies and the ones required for intelligent building applications. Therefore, this thesis can be regarded as a theoretical guide to applications that are willing to use sound in smart buildings. Sound is hard to process by nature because of the highly time-varying characters and the ubiquitous noises. As a result, we have met many challenges during the research from which the concrete methodologies and corresponding evaluations are provided in this thesis. In this ending chapter, we would like to present the overall conceptual conclusions in a concise manner. By listing these conclusions into different categories, we hope that we can help those who are also interested in this research and want to apply these methods to applications.

About audio dataset collection

Before studying to research on the methods, one first needs to collect the audio datasets, be it environmental or speech sounds. Environmental sound events are for human activity recognition and the voice from speech can be used to count speakers. We devote most of the effort to collecting environmental sounds since they are more diverse in categories and harder to find than speech datasets. Apart from the common data-collecting routines, we have also found some experiences that are particularly important for environmental sound datasets:

1. *The data should come from multiple contexts or environments in order to be generic.*

   If all the audio events are collected from the recordings from the same scenario, the problem can be too simplified. These recorded sound events which are made

by exactly the same door, chair or coffee machine do not have much differences in each sample, thus does not represent a general class. A model trained on these data may simply learn from a small proportion of the data, i.e. a small range of sub-bands, rather than learning from the synthesis and pattern overall. In order to increase the data generality, our environmental sound events are collected from several different public databases and websites. For example, the category of 'door-sound' in our dataset includes the sounds made by many types of doors, such as wooden and metal, old and new, light and heavy ones.

2. *Use smart tools and automation to help with data collection.*

   Usually, when we first think of massive data collection or data labelling, it is a long and tedious task. However, I did not suffer from this although I collected a large amount of audio data.

   Firstly, we used a lot of automation scripts in collecting web data. There are unlimited data or audio resources distributed all over the internet. While a lot of these resources fit our researches, the problem is that they are not packaged and also require post-processing. Nevertheless, by writing an automation script or a crawling robot, I managed to download all the sounds I need in order to save time. It is also worth mentioning that not every website welcomes crawling, and also please be gentle when crawling the data, as our only purpose is not to attack the website but to download the data. For example, our download tasks in this script are scheduled to be very slow, with an extra two-minutes interval between each file download. In this way, we do not create bursts of traffic to the website, which also decreases the risk of being denied from the service.

   Secondly, which may sounds strange, the data used to evaluate our method - audio processing, is labelled automatically by another artificial intelligence technique - image processing. While doing audio processing in my research, I actually learned a lot of image processing knowledge. For the people counting task, we recorded some video and audio streams in our own office. The videos are used to count the ground truth, i.e. how many people are in the office. Looking at the videos and count manually is definitely not a good idea, which is exhausting and also error-prone when the labelling needs to be accurate to *0.1s*. It is therefore a good solution to use image-processing models to detect persons automatically from the captured videos. Running on a normal GPU, the YOLO-3 based object-detection model can easily detect people in more than 50 frame-per-second frequencies. Additionally, with an automatic labelling system, the captured videos from surveillance cameras do not need to be saved, which also protects the privacy of our colleagues during data collection.

3. *The way you store data matters.*

   For big data processing, a good way of storing data can save you hours or even days of time. By data storage, we refer not only the database, but also the data content and format. Initially, we saved all the audio files in the raw *.wav* format on the server for data training. However, this measure was later found to be very inefficient. In order to save the long training time of neural network models, we calculated the time spent on each code segment of the model through the training process. Surprisingly, the reading of the audio files in each epoch took up more than 2/3 of the entire time, even more than the time to update the neural network weights. This is mainly because that the dataset is too large to be pre-loaded into memory, therefore needs to be read frequently from the disk. To solve this problem, we then read the data array in each audio file and save them in the fast-access HDF5 database. In this way, the audio file parsing step is bypassed during the repeatedly training process where each audio is accessed just like a data array. In addition, the HDF5 database itself is optimized for the loading speed, which makes I/O speed faster than directly reading files from a disk. Another advantage of choosing HDF5 database is the ease of programming and data management. A HDF5 file is like a file-system that has virtual 'directories' and 'files', where the interface to access a data-array is the same as the interface to access a file on disk.

About sound events classification

1. *In quiet environments, sound is suitable for activity recognition applications.*

   In Chapter 4 we studied automatic sound event classification in quiet environments, with both classic and light-weight deep learning based models. Both models gave very good results with CNN-based model the better one, reaching around 95% accuracy with 6 classes of events. Moreover, these highly accurate results are based on very diverse sound inputs that are recorded in different environments, while in real applications the sounds should be less diversified. This means that even with resource-constraint devices, computers can differentiate different sound events very accurately in a quiet environment. Although this classification accuracy may decrease when the classes of sounds increases, we believe that the added complexity can be well addressed by a moderate increase of the neurons of our CNN-based model.

2. *There is no perfect solution for overlapping sounds classification.*

   For the classification of the overlapping sound, the solutions and their performance really depend on the scale of the problem. Many methods are proposed to address

this difficult problem, either with signal decomposition techniques, more microphones, or larger deep learning models. These models, including ours, have shown promising results when the sound signal is blended by only two or three sources. However, since sound is essentially a one-dimensional signal, its complexity can rise rapidly as the mixtures increase. In a similar problem - music notes recognition, a model usually tolerates more mixtures of sources. This is mainly because musical notes are short in time and narrow in frequency-bands, which can be easily separated through merely signal processing techniques. However, the sound events produced by ambient environments normally have a wide range of frequency bands and duration, making signal decomposition much harder. It can be expected that even humans can not recognize all the sound sources if the environment is too noisy. In this case, we can either try to recognize the dominant sound, or alternatively, to form a sense about the context as we proposed in Section 5.

3. *CNN-based model is highly recommended in resource-constraint devices.*

   In the field of speech recognition, RNN is the most popular method up to date. But it is not necessary to use RNN in sound events classification. The success of RNN in speech recognition is from its internal states that can represent context information or long memories. This character of RNN, on the downside, also largely increases the calculation cost. For speech recognition, it is the linguistic part rather than the audio part of a speech that needs context analysis. In order to know whether a bell is ringing, the context information that happens minutes ago is not needed. Instead, a CNN-based model with 'tall' filters in the first layer can efficiently represent the graph patterns in spectrograms.

4. *Better start deep learning from shallow.*

   To build a universal deep learning model from overwhelming data volume is inspiring, but it is more inspiring to truly understand why it works. During the time of my research, the most exciting moment was the first time that I feel like glimpsing the mysterious interior of the neural network model, which was totally a black box to me. That moment happens in Section 4.3, where I tested with a minimal CNN model with very few filters in each layer. The minimal model had a fairly good result and most importantly, it had very little if no redundancy in its neurons after training. With the help of visualization tools, I was then able to actually see and understand what features does each layer learn and how they cooperate to make the final prediction. Later I could slowly add in neurons or layers to observe the consequences and adjust the model accordingly. For example, once I picked out a wrongly classified door-slamming sound, which is clear and highly recognizable, to analyze what part was probably wrong. It turned out that there was a small

flaw in the 'strip' setting of the second CNN layer, which would cause the first 200ms data to be discarded. On the contrary, a very large model that well exceeds the complexity is not a good one to start with. It may or may not work, but the redundancy that is distributed everywhere makes it hard to interpret and to improve.

5. *Privacy is a big challenge in real applications.*

   Apart from the performance and other technical limitations, privacy risk is probably the major challenge of applying sound sensors indoor. In some sense, microphones expose even more privacy of humans than surveillance cameras. There is no easy way to dispel the doubts from inhabitants that are being spied on. Even if the device is programmed to only process audio without storing it, one may still worry about the device being maliciously hacked. The worries in software are further amplified in small IoT devices, which normally have very simple code and are not that safe. In contrast to fragile software stacks, hardware is generally considered much less vulnerable. In one of our researches, we stripped the vocal bands (from 300 to 300K HZ) from all the sounds so that speech content is hardly unrecognizable. After the voice truncation, we still manage to get a promising result in classifying different sound events. This experiment has proved that activity recognition tasks can still work well with 'non-voice' microphones. This intentionally distorted microphone could also be our future research work, where more types of distortion can be done to perfectly protect privacy.

About people counting

1. *Transfer learning provides an alternative option of features.*

   In order to count people by speech sound, we need to find the feature that best correlate the voice of speakers. In the earlier days, features are normally manually hand-crafted by researchers and domain experts, which takes time and is also too rigid. Fortunately, the concept of transfer learning can help to extract features automatically. The features of one problem can be extracted from the intermediate layers of a deep learning model that aims to solve another closely related problem. Regarding our problem, the voice feature is extracted from a speaker recognition model: SincNet. SincNet model uses an audio-data adapted one-dimension CNN layer and is trained on a large amount of speakers. Apart from the feature quality, transfer learning feature is also more flexible as new data can be used for training at any time.

2. *Visualize your data and features before hands-on.*

Seeing is important. Visualization of the dataset can always help us understand and problem better and find inspirations. However, human eyes can not visualize data in more than three dimensions. A typical solution is to use dimension reduction techniques, for example, PCA, to reduce the data dimensions by keeping only the most important information, or so called the principle components. However, this would arguably cause significant information loss when the dimension needs to shrink a lot. A better option is to use the t-SNE technique, which maps high-dimensional data into lower dimension according to the similarity or distance between data points. This would largely protect the non-linear and local structure of the high dimensional data. The t-SNE results not only can be visualized, but can also be directly used for the input of clustering. Another popular and alternative technique is UMAP (Uniform Manifold Approximation and Projection), which is not discussed in this thesis.

3. *Post-processing is essential in real applications.*

   This research on sound-based people counting is mainly carried out based on the speech dataset collected from a testbed context. However, human speeches can be much more complicated in a real environment. When we tried to apply it to real environment applications, many additional post-processing steps need to be considered and carefully optimized. One of the important optimizations is the minimal number of audio frames (or the dots in the t-SNE plot) that forms a cluster, i.e. a person. This setting is used to filter out the noise and the outliers. In real life, some sentences may be very short are can be hardly identified, which is better to just discard. Moreover, not all the frames in a speech have similar features, there may be high pitches, whispers or even silent frames that are the outliers from the representative frames. Therefore, it is reasonable that a new 'person' should be added when enough and most representative frames can be identified. In practice, this parameter depends on the frame length and the background noise level, where the value should increase with smaller frame size and more background noises. One problem in real applications that have not been solved in this thesis is how to accurately find the duration of multi-speaker sounds in a crowded environment. In our experiments, the speech clips are segmented from the input audio stream based on the silent time frames within. However, unlike environmental sounds which normally happen discretely, human conversations can go non-stop continuously, making this method error-prone in crowded environments.

Overall thoughts of the two tasks

1. *Sound-based activity recognition is closer to real applications than people counting.*

Between the two research objectives in our research, we think that sound-based activity recognition is more realistic to implement in practical applications than people counting. Firstly, sound event recognition has an overall better performance good performance. Our environmental sound classification model can reach 95% for 6 different event classes, even with very diversified sounds from completely different environments. In real applications such as offices, the installed microphone would be surrounded by similar sounds made by a few objects. On the other hand, speech-based people counting in the real environment can only be worse since people may not talk for a long time.

Secondly, a sound event recognition model can be much smaller and faster than the speaker-counting model, thus is more likely to be applied in IoT devices. For sound events classification, our 2 hidden-layer CNN model only needs several thousands of weights, while our CNN model for speaker counting model needs millions of weights.

Thirdly, a sound event recognition model also has less privacy concern. In our research, the voice bands can be completely truncated and have limited influences on the results. In contrast, a speaker counting model highly relies on human voices and can't easily address privacy issues.

Finally, there are many mature people counting methods such as WiFi sniffing and cameras, which makes sound-based counting only a less favourable alternative. By contrast, sound has information that cameras and wearable devices can not detect, which makes it is a competitive solution or a complement for detecting the behaviours of people in many scenarios.

2. *The combination can accomplish more.*

   In our work, we have conducted separate researches on activity recognition and people counting tasks. In the future, we believe more interesting tasks can be accomplished by combining the two methods or other sensors. For example, connecting multiple microphones scattered in buildings can turn the isolated people counting model into a mobility tracking model. Combing people counting and activity recognition models also give clearer clues about who is doing what, i.e. activity recognition together with labelled identities. Recognizing the sounds in environments may help camera-based activity recognition become more accurate and substantial. These combined research topics or applications definitely need more work and more elaborated designs, which is also an interesting direction for us in the future. In a word, there is still a lot to explore on sound so as to make our living environment smarter and better.

Bibliography

[1] Mohammed Arif, Martha Katafygiotou, Ahmed Mazroei, Amit Kaushik, Esam Elsarrag, et al. Impact of indoor environmental quality on occupant well-being and comfort: A review of the literature. *International Journal of Sustainable Built Environment*, 5(1):1–11, 2016.

[2] Paolo Bertoldi and Bogdan Atanasiu. Electricity consumption and efficiency trends in european union. *Renewable Energy Status Rep*, 2009.

[3] Renewable Energy. Energy efficiency trends in residential and commercial buildings. 2010.

[4] Jun Li and Michel Colombier. Managing carbon emissions in china through building energy efficiency. *Journal of environmental management*, 90(8):2436–2447, 2009.

[5] Vahid Vakiloroaya, Bijan Samali, Ahmad Fakhar, and Kambiz Pishghadam. A review of different strategies for hvac energy saving. *Energy conversion and management*, 77:738–754, 2014.

[6] Maria da Graça Carvalho. Eu energy and climate change strategy. *Energy*, 40(1): 19–22, 2012.

[7] Monica Drăgoicea, Laurenţiu Bucur, and Monica Pătraşcu. A service oriented simulation architecture for intelligent building management. In *International Conference on Exploring Services Science*, pages 14–28. Springer, 2013.

[8] Sanja Lazarova-Molnar, Hamid Reza Shaker, Nader Mohamed, et al. Fault detection and diagnosis for smart buildings: State of the art, trends and challenges. In *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pages 1–7. IEEE, 2016.

[9] Kurt W Roth, Detlef Westphalen, Michael Y Feng, Patricia Llana, and Louis Quartararo. Energy impact of commercial building controls and performance diagnostics: market characterization, energy impact of building faults and energy

savings potential. *Prepared by TAIX LLC for the US Department of Energy. November. 412pp (Table 2–1)*, 2005.

[10] Joshua Kneifel. Life-cycle carbon and cost analysis of energy efficiency measures in new commercial buildings. *Energy and Buildings*, 42(3):333–340, 2010.

[11] Soteris A Kalogirou. Artificial neural networks in energy applications in buildings. *International Journal of Low-Carbon Technologies*, 1(3):201–216, 2006.

[12] Xudong Ma, Ran Cui, Yu Sun, Changhai Peng, and Zhishen Wu. Supervisory and energy management system of large public buildings. In *2010 IEEE International Conference on Mechatronics and Automation*, pages 928–933. IEEE, 2010.

[13] Yudong Ma, Anthony Kelman, Allan Daly, and Francesco Borrelli. Predictive control for energy efficient buildings with thermal storage: Modeling, stimulation, and experiments. *IEEE control systems magazine*, 32(1):44–64, 2012.

[14] Nabil Nassif, Stanislaw Kajl, and Robert Sabourin. Optimization of hvac control system strategy using two-objective genetic algorithm. *HVAC&R Research*, 11(3): 459–486, 2005.

[15] Konstantinos Makantasis, Antonios Nikitakis, Anastasios D Doulamis, Nikolaos D Doulamis, and Ioannis Papaefstathiou. Data-driven background subtraction algorithm for in-camera acceleration in thermal imagery. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2090–2104, 2018.

[16] Eun Jeon, Jong-Suk Choi, Ji Lee, Kwang Shin, Yeong Kim, Toan Le, and Kang Park. Human detection based on the generation of a background image by using a far-infrared light camera. *Sensors*, 15(3):6763–6788, 2015.

[17] Rashmi Priyadarshini and RM Mehra. Quantitative review of occupancy detection technologies. *Int. J. Radio Freq*, 1:1–19, 2015.

[18] Tuan Anh Nguyen and Marco Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and buildings*, 56:244–257, 2013.

[19] William Kahl and Richard Settanni. Lighting control system with infrared occupancy detector, October 1987. US Patent 4,703,171.

[20] William R Tribe, David A Newnham, Philip F Taday, and Michael C Kemp. Hidden object detection: security applications of terahertz technology. In *Terahertz and Gigahertz Electronics and Photonics III*, volume 5354, pages 168–176. International Society for Optics and Photonics, 2004.

[21] Wei Xi, Jizhong Zhao, Xiang-Yang Li, Kun Zhao, Shaojie Tang, Xue Liu, and Zhiping Jiang. Electronic frog eye: Counting crowd using wifi. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 361–369. IEEE, 2014.

[22] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[24] Ash Tyndall, Rachel Cardell-Oliver, and Adrian Keating. Occupancy estimation using a low-pixel count thermal imager. *IEEE Sensors Journal*, 16(10):3784–3791, 2016.

[25] Yalong Ma, Xinkai Wu, Guizhen Yu, Yongzheng Xu, and Yunpeng Wang. Pedestrian detection and tracking from low-resolution unmanned aerial vehicle thermal imagery. *Sensors*, 16(4):446, 2016.

[26] Sarvesh Vishwakarma and Anupam Agrawal. A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer*, 29, 10 2012. doi: 10.1007/s00371-012-0752-6.

[27] Michalis Vrigkas, Christophoros Nikou, and Ioannis A Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:28, 2015.

[28] Bing Dong and Burton Andrews. Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings. In *Proceedings of building simulation*, pages 1444–1451. 6 - 9.

[29] Keith Dana Martin. Sound-source recognition: A theory and computational model.

[30] Frank I Klassner. *Data reprocessing in signal understanding systems*. PhD thesis, University of Massachusetts at Amherst, 1996.

[31] Michael Cowling and Renate Sitte. Comparison of techniques for environmental sound recognition. *Pattern Recognition Letters*, 24(15):2895–2907, 2003. ISSN 0167-8655. doi: https://doi.org/10.1016/S0167-8655(03)00147-8.

[32] Chien-Chang Lin et al. Audio classification and categorization based on wavelets and support vector machine. *IEEE TASLP*, 13(5):644–651, 2005.

[33] Y. Zigel, D. Litvak, and I. Gannot. A Method for Automatic Fall Detection of Elderly People Using Floor Vibrations and Sound—Proof of Concept on Human Mimicking Doll Falls. *IEEE Transactions on Biomedical Engineering*, 56(12): 2858–2867, December 2009. ISSN 0018-9294, 1558-2531. doi: 10.1109/TBME.2009.2030171.

[34] Huy Dat Tran and Haizhou Li. Sound event recognition with probabilistic distance SVMs. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6): 1556–1568, 2011. ISSN 1558-7916, 1558-7924. doi: 10.1109/TASL.2010.2093519.

[35] Dalibor Mitrović, Matthias Zeppelzauer, and Christian Breiteneder. Features for content-based audio retrieval. In *Advances in Computers*, volume 78, pages 71–150. Elsevier, 2010. ISBN 978-0-12-381019-9. 8 - 6 features.

[36] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen. Context-dependent sound event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(1):1, 2013. 9 -8 Add context-label to detect overlapping sound event by HMM.

[37] R. Cai, Lie Lu, A. Hanjalic, Hong-Jiang Zhang, and Lian-Hong Cai. A flexible framework for key audio effects detection and auditory context inference. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(3):1026–1039, 2006. ISSN 1558-7916. doi: 10.1109/TSA.2005.857575. 7 - 5.

[38] S. Chu, S. Narayanan, and C. C. J. Kuo. Environmental sound recognition with time #x2013;frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1142–1158, 2009. ISSN 1558-7916. doi: 10.1109/TASL.2009.2017438. 2-3.

[39] Peerapol Khunarsal, Chidchanok Lursinsap, and Thanapant Raicharoen. Very short time environmental sound classification based on spectrogram pattern matching. *Information Sciences*, 243:57–74, 2013.

[40] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6440–6444. IEEE, 2016.

[41] K. J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2015. doi: 10.1109/MLSP.2015.7324337.

[42] Haomin Zhang, Ian McLoughlin, and Yan Song. Robust sound event recognition using convolutional neural networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 559–563. IEEE, 2015.

[43] Xiaohu Zhang, Yuexian Zou, and Wei Shi. Dilated convolution neural network with leakyrelu for environmental sound classification. In *2017 22nd International Conference on Digital Signal Processing (DSP)*, pages 1–5. IEEE, 2017.

[44] Ilyas Ozer, Zeynep Ozer, and Oguz Findik. Noise robust sound event classification with convolutional neural network. *Neurocomputing*, 272:505–512, 2018.

[45] Bing Dong and Burton Andrews. Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings. In *Proceedings of building simulation*, pages 1444–1451, 2009. 1.

[46] Huy Dat Tran and Haizhou Li. Sound event recognition with probabilistic distance svms. *IEEE transactions on audio, speech, and language processing*, 19(6):1556–1568, 2011.

[47] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *2015 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2015.

[48] Sharath Adavanne, Giambattista Parascandolo, Pasi Pertilä, Toni Heittola, and Tuomas Virtanen. Sound event detection in multichannel audio using spatial and harmonic features. *arXiv:1706.02293 [cs]*, 2017.

[49] Andrey Temko et al. Acoustic event detection in meeting-room environments. *Pattern Recognition Letters*, 30(14):1281–1288, 2009. ISSN 1678655. doi: 10.1016/j.patrec.2009.06.009.

[50] Huy Dat Tran et al. Jump function kolmogorov for overlapping audio event classification. In *ICASSP*, pages 3696–3699, 2011.

[51] Toni Heittola et al. Sound event detection in multisource environments using source separation. In *CHiME*, 2011.

[52] Arnaud Dessein et al. Real-time detection of overlapping sound events with non-negative matrix factorization. In *Matrix Information Geometry*, pages 341–371. Springer, 2013.

[53] Annamaria Mesaros et al. Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations. In *ICASSP*, pages 151–155, 2015.

[54] Jean-Marc Valin et al. Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering. *Robotics and Autonomous Systems*, 55(3):216–228, 2007.

[55] Maher A Quraan et al. Detection and localization of hippocampal activity using beamformers with meg: a detailed investigation using simulations and empirical data. *Human brain mapping*, 32(5):812–827, 2011.

[56] Darren B Ward et al. Particle filter beamforming for acoustic source localization in a reverberant environment. In *ICASSP*, volume 2, pages II–1777, 2002.

[57] R. Takeda and K. Komatani. Discriminative multiple sound source localization based on deep neural networks using independent location model. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 603–609, 2016. doi: 10.1109/SLT.2016.7846325.

[58] W. He, P. Motlicek, and J. Odobez. Deep neural networks for multiple speaker detection and localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018. doi: 10.1109/ICRA.2018.8461267.

[59] Soumitro Chakrabarty and Emanuël AP Habets. Multi-speaker localization using convolutional neural network trained with noise. *arXiv preprint arXiv:1712.04276*, 2017.

[60] Sharath Adavanne, Archontis Politis, and Tuomas Virtanen. Multichannel sound event detection using 3d convolutional neural networks for learning inter-channel features. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018. doi: 10.1109/IJCNN.2018.8489542.

[61] James Scott et al. Audio location: Accurate low-cost location sensing. In *PerCom*, pages 1–18. Springer, 2005.

[62] Y. Guo et al. Localising speech, footsteps and other sounds using resource-constrained devices. In *IEEE IPSN*, pages 330–341, 2011.

[63] Achintha Maddumabandara et al. Experimental evaluation of indoor localization using wireless sensor networks. *IEEE Sensors Journal*, 15(9):5228–5237, 2015.

[64] Adelbert W Bronkhorst. The cocktail-party problem revisited: early processing and selection of multi-talker speech. *Attention, Perception, & Psychophysics*, 77 (5):1465–1487, 2015.

[65] Sayeh Mirzaei, Yaser Norouzi, et al. Blind audio source counting and separation of anechoic mixtures using the multichannel complex nmf framework. *Signal Processing*, 115:27–37, 2015.

[66] Oliver Walter, Lukas Drude, and Reinhold Haeb-Umbach. Source counting in speech mixtures by nonparametric bayesian estimation of an infinite gaussian mixture model. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 459–463. IEEE, 2015.

[67] Takayuki Arai. Estimating number of speakers by the modulation characteristics of speech. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 2, pages II–197. IEEE, 2003.

[68] Valentin Andrei, Horia Cucu, Andi Buzo, and Corneliu Burileanu. Counting competing speakers in a timeframe—human versus computer. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[69] F. Stöter, S. Chakrabarty, B. Edler, and E. A. P. Habets. Classification vs. regression in supervised learning for single channel speaker count estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 436–440, 2018. doi: 10.1109/ICASSP.2018.8462159.

[70] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals. Speaker diarization: A review of recent research. 20(2):356–370, 2012. ISSN 1558-7916. doi: 10.1109/TASL.2011.2125954. 5 - review speaker diarisation.

[71] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011. ISSN 1558-7916. doi: 10.1109/TASL.2010.2064307. 9 - ivector.

[72] Gerald Friedland, Hayley Hung, and Chuohao Yeo. Multi-modal speaker diarization of real-world meetings using compressed-domain video features. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4069–4072. IEEE, 2009.

[73] Chenren Xu, Sugang Li, Gang Liu, Yanyong Zhang, Emiliano Miluzzo, Yih-Farn Chen, Jun Li, and Bernhard Firner. Crowd++: Unsupervised speaker count with smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 43–52. ACM, 2013. ISBN 978-1-4503-1770-2. doi: 10.1145/2493432.2493435. event-place: Zurich, Switzerland.

[74] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In

Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1096–1104. Curran Associates, Inc.

[75] Z. Tan and M. Mak. Bottleneck features from SNR-adaptive denoising deep classifier for speaker identification. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1035–1040. doi: 10.1109/APSIPA.2015.7415429.

[76] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with SincNet. *arXiv*, 2018. 9 - sincNet.

[77] Victor Zue, Stephanie Seneff, and James Glass. Speech database development at mit: Timit and beyond. *Speech communication*, 9(4):351–356, 1990.

[78] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[79] Fabian-Robert Stöter, Soumitro Chakrabarty, Emanuël Habets, and Bernd Edler. Libricount, a dataset for speaker count estimation, April 2018.

[80] A. Mesaros, T. Heittola, and T. Virtanen. TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132, 2016. doi: 10.1109/EUSIPCO.2016.7760424.

[81] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *ACM International Conference on Multimedia (MM'13)*, pages 411–412, Barcelona, Spain, 21/10/2013 2013. ACM, ACM. ISBN 978-1-4503-2404-5. doi: 10.1145/2502081.2502245.

[82] freeSFX.co.uk - FREESFX.CO.UK CONTENT PUBLISHER LICENCE AGREEMENT, 2018.

[83] Wei Wang, Fatjon Seraj, Nirvana Meratnia, and Paul JM Havinga. Privacy-aware environmental sound classification for indoor human activity recognition. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 36–44, 2019.

[84] Michael Cowling and Renate Sitte. Comparison of techniques for environmental sound recognition. *Pattern recognition letters*, 24(15):2895–2907, 2003.

[85] Antti J Eronen, Vesa T Peltonen, Juha T Tuomi, Anssi P Klapuri, Seppo Fagerlund, Timo Sorsa, Gaëtan Lorho, and Jyri Huopaniemi. Audio-based context recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14 (1):321–329, 2006.

[86] Jan Paul Kolter. *User-centric Privacy: A Usable and Provider-independent Privacy Infrastructure*, volume 41. BoD–Books on Demand, 2010.

[87] Ranya Aloufi, Hamed Haddadi, and David Boyle. Privacy-preserving voice analysis via disentangled representations. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 1–14, 2020.

[88] Wikipedia contributors. Voice frequency — Wikipedia, the free encyclopedia, 2018. [Online; accessed 30-May-2018].

[89] Richard C Sprinthall and Stephen T Fisk. *Basic statistical analysis*. Prentice Hall Englewood Cliffs, NJ, 1990.

[90] Donald Michie, David J Spiegelhalter, CC Taylor, et al. Machine learning. *Neural and Statistical Classification*, 13(1994):1–298, 1994.

[91] Benjamin Kedem. Spectral analysis and discrimination by zero-crossings. *Proceedings of the IEEE*, 74(11):1477–1493, 1986.

[92] Jérôme Sueur, Sandrine Pavoine, Olivier Hamerlynck, and Stéphanie Duvail. Rapid acoustic survey for biodiversity appraisal. *PloS one*, 3(12):e4065, 2008.

[93] Md Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech communication*, 54(4):543–565, 2012.

[94] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.

[95] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[96] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[97] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[98] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[99] Giambattista Parascandolo et al. Recurrent neural networks for polyphonic sound event detection in real life recordings. *ICASSP*, pages 6440–6444, 2016. doi: 10.1109/ICASSP.2016.7472917.

[100] Jerome H. Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, March 1997. ISSN 1573-756X. doi: 10.1023/A:1009778005914.

[101] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[102] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5 (Aug):975–1005, 2004.

[103] Guodong Guo et al. Content-based audio classification and retrieval by support vector machines. *IEEE Transactions on Neural Networks*, 14(1):209–215, 2003. ISSN 1045-9227. doi: 10.1109/TNN.2002.806626.

[104] Wei Wang, Fatjon Seraj, and Paul JM Havinga. A sound-based crowd activity recognition with neural network based regression models. In *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 1–8, 2020.

[105] Wei Wang, Fatjon Seraj, and Paul JM Havinga. k-specnet: Localization and classification of indoor superimposed sound for acoustic sensor networks. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2020.

[106] Tian He et al. Range-free localization schemes for large scale sensor networks. In *MobiCom 2003*, pages 81–95. ACM, 2003.

[107] Jonathan Dennis et al. Image feature representation of the subband power distribution for robust sound event classification. *IEEE TASLP*, 21(2), 2013.

[108] Ian McLoughlin et al. Robust sound event classification using deep neural networks. *IEEE TASLP*, 23(3), 2015.

[109] I Lawrence et al. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, pages 255–268, 1989.

[110] John A Hartigan et al. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[111] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20: 53–65, 1987.

[112] Robin Scheibler et al. Pyroomacoustics: A python package for audio room simulation and array processing algorithms. In *ICASSP*, pages 351–355, 2018.

[113] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN 9780262035613.

[114] Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions. In *International conference on machine learning*, pages 3059–3068, 2016.

[115] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[116] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[117] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[118] Eman Ahmed and Mohamed Moustafa. House price estimation from visual and textual features. *arXiv preprint arXiv:1609.08399*, 2016.

[119] Wei Wang, Fatjon Seraj, Nirvana Meratnia, and Paul JM Havinga. Speaker counting model based on transfer learning from sincnet bottleneck layer. In *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–8. IEEE, 2020.

[120] Jeremy York, Paul Dixon, Thomas C Opdycke, and Wan-Chung William Wu. Measuring effectiveness of marketing campaigns presented on media devices in public places using audience exposure data, January 2009. US Patent App. 12/233,872.

[121] Ron Kohavi, Neal J Rothleder, and Evangelos Simoudis. Emerging trends in business analytics. *Communications of the ACM*, 45(8):45–48, 2002.

[122] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2008.

[123] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35 (5):1285–1298, 2016.

[124] Joseph J Lim, Ruslan R Salakhutdinov, and Antonio Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Advances in neural information processing systems*, pages 118–126, 2011.

[125] Homayoon Beigi. Speaker recognition. In *Fundamentals of Speaker Recognition*, pages 543–559. Springer, 2011.

[126] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83, 1995. ISSN 1063-6676. doi: 10.1109/89.365379. clear & phone sound - supervised - GMM -.

[127] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[128] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 31(4):497–508, 2001.