Data Reduction Methods of Audio Signals for Embedded Sound Event Recognition

Master's thesis (Tech.)
University of Turku
Department of Information Technology
Electronic Systems
2019
Yan Junjie

Supervisors:
D.Sc. (Tech.) Tomi Westerlund

UNIVERSITY OF TURKU
Department of Information Technology

YAN JUNJIE: Data Reduction Methods of Audio Signals for

Embedded Sound Event Recognition

Master's thesis (Tech.), 63 p.
Electronic Systems
December 2019

Sound event detection is a typical Internet of Things (IoT) application task, which could be used in many scenarios like dedicated security application, where cameras might be unsuitable due to the environment variations like lights and movements. In realistic applications, usually models for this task are implemented on embedded devices with microphones. And the idea of edge computing is to process the data near the place where it happens, because reacting in real time is very important in some applications. Transmitting collected audio clips to cloud may cause huge delay and sometime results in serious consequence. But processing on local has another problem, heavy computation may beyond the load for embedded devices, which happens to be the weakness of embedded devices. Works on this problem have make a huge progress recent year, like model compression and hardware acceleration.

This thesis provides a new perspective on embedded deep learning for audio tasks, aimed at reducing data amount of audio signals for sound event recognition task. Instead of following the idea of compressing model or designing hardware accelerator, our methods focus on analog front-end signal acquisition side, reducing data amount of audio signal clips directly, using specific sampling methods. The state-of-the-art works for sound event detection are mainly based on deep learning models. For deep learning models, less input size means lower latency due to less time steps for recurrent neural network (RNN) or less convolutional computations for convolutional neural network (CNN). So, less data amount of input, audio signals gain less computation and parameters of neural network classifier, naturally, resulting less delay while interference. Our experiments implement three kind of data reduction methods on this sound event detection task, all of these three methods are based on reducing the sample points of an audio signal, using less sampling rate and sampling width, using sigma delta analog digital converter (ADC) and using level crossing (LC) ADC for audio signals. We simulated these three kinds of signals and feed them into the neural network to train the classifier

Finally, we derive the conclusion that there is still some redundancy of audio signals in traditional sampling ways for audio classification. And using specific ADC modules better performance on classification with the same data amount in original way.

Keywords: sound event detection; embedded learning; sigma-delta ADC; LC ADC

CONTENT

List of Abbreviation III
Chapter 1. Introduction 1
1.1 Background 1
1.2 Sound event detection 2
1.3 Efficient deep learning strategy 4
1.4 ADC techniques 5
1.6 Contributions and related works 7
Chapter 2. Feature Extraction of Audio Signal 10
2.1 Mel spectrogram 10
2.2 Cepstrum analysis 14
2.3 Mel Frequency Cepstral Coefficients 17
Chapter 3. Models for Sound Event Detection 21
3.1 Datasets 21
3.1.1 UrbanSound8K dataset 21
3.1.2 FSDKaggle2018 Dataset 21
3.2 Solutions for experiments 22
3.2.1 MFCCs with RNNs 23
3.2.2 Log Mel energy with CNNs 24
3.3 Solution for competition 29
3.3.1Multi model ensemble 29
3.3.2 Discussion 30
Chapter 4. Data Reduction Methods 31
4.1 Down Sample for audio signals 31
4.1.2 Analysis of experiment results 32
4.2 Sigma Delta ADC for audio signals 34
4.2.1 Principle and simulation 34
4.2.2 Analysis of experiment results 39
4.3 Level Crossing ADC for audio signals 42
4.3.1 Principle and simulation 42
4.3.2 Analysis of experiment results 47
4.4 Performance analysis 49

4.4.1 Theoretical analysis 49
4.4.2 Experiment analysis 52
Chapter 5. Conclusions 55
References 57
Acknowledgment 63

List of Abbreviation

Abbreviation	Full name
ADC	Analog Digital Converter
AIOT	Artificial Intelligence with Internet of Things
ASCI	Application-Specific Integrated Circuit
ASR	Automatic Speech Recognition
BN	Batch Normalization
CNN	Convolutional Neural Network
CQT	Constant Q Transform
DCASE	Detection and Classification of Acoustic Scenes and Events
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet
ECG	Electro Cardio Graphic
FFT	Fast Fourier Transform
FSD	Free Sound
GAN	Generative adversarial network
GMM	Mixture Gaussian Model
GMP	Global max pooling
GPU	Graphic Process Unit
HMM	Hidden Markov Model
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
LC	Level -Crossing
LSTM	Long Short-Term Memory
MFCC	Mel Frequency Cepstrum Coefficients
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
STFT	Short-Time Fourier Transform

Chapter 1. Introduction

1.1 Background

Recent years, Information explosion is growing far beyond the effective processing scope of human beings. Therefore, the processing of information by computers has become extremely important and will be widely used. Compared with computer vision, due to the convenience of audio, human and material resource efficiency, computer auditory also raised great research interest at home and abroad. In a broad sense, the auditory object of perception includes all the sounds of vibration, involved varieties categories [1]. For convenience, computer auditory research generally divides the sound to be researched into three categories: speech signals, music signals and general audio signals. Speech signal is a generalized communication way of human beings. It can be considered as a direct way of information exchange. Musical signals are more abstract and advanced sounds, often conveying emotions in the form of art, stories, thoughts, philosophies and other information. While general audio refers to other sounds that exist in the periphery of human beings, from all kinds of noise to the sound of water in the natural world, sound of birds, door of human activities, footsteps, street noise, restaurant background sound, and a variety of industrial machine sounds, etc. General audio signals contain huge information, but not all of them are valid for a specific human. Due to the variety of audio, sounds are different with speech and music in features. So, research on general audio signals are much difficult and have gained less results.

Sounds signals, usually refers to general audio signal (differ from speech and music signals), contains a lot of information about our environment and physical events which take place in there. We human beings could easily senses the sound scene (subway station, bar, etc.), and identify the various sound events (gun shot, people talking, etc.). Signal processing methods for extracting information from sound signals automatically could well solve the problem for lots of realistic applications, like, audio information retravel, context-aware devices, and intelligent security monitoring systems to monitor abnormal sound in environments. However, in realistic situation, sounds are usually polyphonic, which means multi sounds may occur simultaneously, making recognition

much difficult. Thus, related researches are still needed to be proposed for recognizing sound scenes and individual sound sources reliably in realistic soundscapes.

In real life application scenario, audio signals are often collected by microphone array embedded devices, put on public environment. So those are typical IoT application tasks.

In classic IoT scenarios embedded devices mostly play the role of sensor, detecting information from environment, with little computation ability. With the growth of IoT devices in edge, a significant amount of data is generated, usually processed with big data techniques. Complex data processing tasks are completed in cloud due to the computation limitation of edge devices. In last few years, with development of hardware side of embedded devices and efficient algorithms, edge computing, processing needs to be done near the location where the data is generated (the edge), has been proposed to be another feasible way in Artificial Intelligence with Internet of Things (AIOT) application scenario [60], which could reduce the heavy load of cloud and gain less latency in some tasks, and in some cases that's very important.

In this task, audio event detection, we can expect that audio event could be recognized in local instead of cloud, because of less detection latency could be important in some application like security monitoring.

Although we wish recognition decision could be done in local embedded devices, limited computation resource is the biggest challenge get in our way. As audio event detection task is a very challenge problem, which could be well solved by deep learning models, and better performance usually means much complex model. If we want to implement deep neural network on embedded devices which have limited computation resources, efficient strategies need to be implemented. There are a lot of works concern about this topic, most of them are on hardware or algorithms, developing specific hardware logic or compressing the model. Our methods are different from all of them, we compress audio signal on front analog acquis side, more details are discussed in chapter 4.

1.2 Sound event detection

Sound event detection (also known audio event recognition), is defined as recognition of individual sounds in audio, involving also estimation of onset and offset for distinct sound event instances (possibly for multiple sound classes). It assumes that

similar sounds can be represented as a single class, so the category is sufficiently different from other sound categories to allow recognition [1].

The purpose of audio event recognition is to distinguish between different types of audio events. The core problem can be divided into two interrelated problems. One is how to find out the difference between different events, and the other is how to find effective information to distinguish these events. Therefore, the fundamental problem in audio event recognition research is to find out discriminative information. However, that could be a challenge problem due to audio event signals' random distribution and variety categories.

The following two aspects leads to its randomness. First is the randomness of the distribution of the audio itself, which is between the same kind of audio. In different transform domains, representations of the same category audio signals are roughly similar, but there are still subtle differences, which is likely to cause misclassification. The second is the randomness of the distribution of audio categories. There are many kinds of audio. According to the different recognition details, some kinds may be divided into the same category or different fine level categories. Therefore, the audio category itself has randomness, it is necessary to define the category fineness clearly to eliminate the randomness.

Another major difficulty in audio event recognition is the diversity of events. Even if the audio category of the study is limited, the differences in these different audios are different, some are very similar, and some are significantly different. Whether in the time domain, frequency domain or other transform domains, different audio have similar or different characteristics. It is necessary to combine the characteristics of these fields to find the most obvious and effective distinguishing information, so that it is possible to distinguish different audio categories.

Despite these above difficulties, the current research on audio feature selection and classification models has been very deep and wide result in lots of solutions for these difficulties. Most of the current solutions use neural networks as classifiers, which generally feed the audio features into the feedforward neural network, convolutional neural network (CNN), and the long short-term memory model (LSTM) or a combination of different networks.

Over the last few years, audio event recognition task has raised increasing interest due to its potential realistic applications. And it is host in DCASE challenge and Kaggle

competition recent years, people do these tasks every year and tasks getting harder by years. This task we do in our experiment is based on DCASE 2018 task 2 [2]. However, challengers do this for accuracy, which is the most important performance in this competition, so lots of complicated models are proposed, which could not be run on embedded devices. So, we need to do extra works to trade off the accuracy and computation resource.

1.3 Efficient deep learning strategy

Deep learning has been widely applied in many fields, especially in the field of multimedia processing. Model training based on convolutional neural networks is significantly better than other traditional methods. However, complexity of the neural network model continues to increase with the increasing performance. For example, for ImageNet recognition, the size of winner model from 2012 to 2015 increased 16 times, and for Baidu's deep speech, the number of training operations increased 10 times for just one year. Such large model creates lots of problems. Limited memory, computing power and energy consumption are the bottlenecks for further training of large-scale deep learning tasks. To deal with the challenge of computation resource limitations on embedded devices, we need to make deep learning more efficient. Two keys points need to be considered in solution. One is to reduce the size of deep neural networks to get smaller memory footprint. Another is to reduce computation operations to get less latency. There has been a lot of works in different perspectives researching on these problems, mainly in two, hardware acceleration and model compression.

Hardware acceleration mainly includes structural optimization, replacing floating-point data with integer data, reducing model size, or using hardware Application-Specific Integrated Circuit (ASCI) for deep learning like Google's TPU [61], mainly in hardware circuit design, so it has minor changes to the algorithm.

And model compression algorithms mainly include six method, weight sharing, pruning, low rank approximation, quantization, binary net and Winograd transformation. The pruning means to remove some of the weights and still have the similar performance. This is first proposed by Professor Yann LeCun [30] in 1989. The Basic idea is to get rid of the redundancy of neural network, because not all the parameters are useful. We train a neural network first, and pruning some of the connections, then we retrain the remaining weights and through this process iteratively.

The basic idea of weight sharing is multiple network connections sharing one value. The method is to cluster weight matrix of each layer into several clusters by using K-means clustering algorithm, and calculate the center value of each cluster to represent the weight of each cluster. Since the weights of the same cluster share the same value, only the indexes of the weight clusters need to be saved. The weight corresponding to the index value is obtained by the lookup table. usually we compensate the weights by training the cluster and fine-tuning to reduce the loss of precision. The idea of quantization is very simple, reducing the resolution of parameters, use less byte of data type to represent a number. The weights and offsets in the network model are represented by single precision 4 bytes, 32 bits floating-point numbers or double precision 8 bytes, 64 bits floating point numbers, and operations between those high-resolution numbers are much slower and consume more memory. Those high-resolution operations can be converted to lower-order integer operations, usually 8 bits integer number operation or the 1-bit Boolean operation. this method can be very practical because the network model is robust to noise and small disturbances after training.

Those above methods are almost in two perspective, hardware acceleration and model compression. Here we propose a new perspective, combining the quantization method with hardware side. The basic idea of method we proposed is very similar to quantization, use less amount of data to represent an audio signal sequence, aiming at reducing the length of audio signal sequence directly. The basic assumption is that there might be some redundancy in audio signals for recognition tasks, so we propose some methods to reduce signal length directly while keep as much information as possible. To achieve this goal, we need the help of hardware circuit, the front-end acquis module. As analog audio signals are converted into digital bit streams by analog-to-digital converters (ADC) circuit, a proper assumption is that we can use ADC circuit to implement our proposal.

1.4 ADC techniques

Sampling is the process of converting a signal (such as a temporally or spatially continuous function) into a sequence of numbers (a discrete function in time or space). For most signal processing tasks, we use uniform sampling method to get uniform discrete sequence, following the Nyquist sampling theorem to prevent aliasing. There could redundancy in digital signals due to the information in the high-frequency

component is only a small part, but takes up part of the bandwidth. To reduce the need for anti-aliasing filters, some over sampling ADC like Sigma-Delta ADC, which generates high resolution signals, are developed and used widely. And for audio signals, in the past we need to recover the analog audio signals as we human beings can only hear analog signals. But now in this task, audio event recognition, we just need to extract features from original audio signals and feed them into classifier, it's not necessary to recover the original analog signal, so we consider to use non-uniform sampling method to get rid of the redundancy in uniform discrete audio signal. And there are also some new non-uniform ADC techniques developed for other signals like electrocardiographic (ECG), like level-crossing ADC (LC ADC) [40, 41, 42].

With the increasing requirements for signal acquisition accuracy in various fields such as biomedical, high-fidelity audio, and smart instruments, Sigma-Delta ADC has been widely used as a class of high-precision ADC. It is mainly composed of an analog modulator and a digital filter with decimator. The noise in band is moved out of the bandwidth due to the oversampling and noise shaping technology in sigma delta ADC, producing a higher signal-to-noise ratio. Digital filter mainly filters out the noise outside the signal bandwidth and down-samples the modulated signal to reduce the amount of data would be stored. The high precision and high signal to noise ratio are important properties for us to do the following experiments.

Now the state-of-art ECG signals modelling techniques almost use level crossing sampling to encode ECG signals. This encoding method gives a variable sampling rate for signal: high sample rate for fast-varying parts, while low for slowly-varying parts. As the name suggested, level crossing sampling means we do sampling while the amplitude cross the quantization level we defined. LC ADC could give a high amplitude resolution for signals because sampling is performed at the signal point where there is exactly the amplitude level value. It is easier for remote sensors generating signals and transmitting such signals compared to traditional technique. In this application, audio event detection, if we do not need to listen to the audio signal, just classifying what event it is, it might could be used in audio sound event classification task. We assume that frequency characteristics of audio event signals are similar with ECG signals, both are evet driven signals, so we tried to apply level crossing sampling on audio event signals for further classification task.

1.6 Contributions and related works

This work consists of several research area like sound event detection, efficient deep learning strategy and ADC techniques. Although there is no innovation on algorithm or hardware circuit, we proposal a new perspective for deep learning of audio field on embedded devices, which combines the works of these areas. So, the idea of this combination crossing these different areas is an innovation itself. In general, this thesis analysis the general audio features, and according to the properties of these features, designed two efficient neural network models for audio event recognition. Based on the models, we proposed three data reduction methods aimed at reducing the data amount the input signal. Audio signals are simulated for all of these three methods, and the simulated signals are feed into neural networks we designed to train a new classifier. Experiments results show an acceptable performance while we reduce the data amount of audio signals using the three methods. Thus, we derive the conclusion that for audio event recognition, there are still redundancy in audio signal sample by classic ADC method. The methods we proposed could reduce the redundancy efficiently while keep the performance well.

This thesis consists of 5 chapters, the first chapter introduced the background and related research areas of this thesis. Chapter 2 discussed the audio features we would use in this thesis, including Mel spectrogram, cepstrum analysis, and Mel Frequency Cepstrum Coefficients (MFCCs). Chapter 3 detailed the models we used in our experiments and also introduced the state-of-the-art work with the reason why it is not suitable in our experiment. Chapter 4 is the main part of this thesis, which talks about our three data reduction method for audio event recognition, with the performance analysis. Chapter 5 derive a conclusion for all our experiments. For each part, there many related works on the problem we concerned.

For sound event detection, there is a well-known challenge called detection and classification of acoustic scenes and events (DCASE), which belongs to the Institute of Electrical and Electronics Engineers (IEEE). This challenge contains several audio classification and detection tasks [2]. Sound event detection is a sub task of DCASE. Except for this competition, audio event recognition has also raised hot research interest in worldwide. In last decades, traditional feature engineering and machine learning methods are the main solution for this problem. Some manually selected high level features such as MFCCs [3], the constant Q transform (CQT) [4], and I-vectors [5] are

classic sound features for classification tasks. The well-known Mel spectrograms [6] is a general middle level feature which could be reprocessed by neural networks to extract further features, so, it has been used widely as input of neural network classifier. Traditional machine learning models like Mixture Gaussian model (GMM) [7] and hidden Markov model (HMM) [8] are both effective machine learning models. Recent years, deep learning techniques have been introduced to audio research area. For example, fully-connected neural networks with high level manually-selected features were used in DCASE 2016 [9] and DCASE 2017 [10] to replace the machine learning models. CNNs based models have archived the state-of-the-art performance for this challenge [11,13]. Recurrent neural networks (RNNs) [12, 14] are suitable to model the temporal information of sound events. Models with attention mechanism have the ability to focus on sound events information which are relative to the information we concern [15], especially for the data which are labelled weakly [16]. And the generative adversarial networks (GANs), proposed in last few years, which are usually applied in generation tasks, also have been used here for audio classifiers to improve their robustness [17]. In DCASE 2018, the main progress is based on specific tricks like data augmentation [18, 20], mixed features [20, 21], and multi model decision [19] with much deeper CNNs network. And the top team of leaderboard even has 55461000 parameters in their model [20]. However, our purpose is to implemented our model in embedded devices, so complex models like [20, 21] are not our best choice. Instead, we build a relative shallow model but performed a relative well result.

In on-side deep learning filed, efficient learning is researched on mainly two aspects, model compression and hardware acceleration. Adjusting the depth of the neural structure to achieve the best tradeoff between performance and complexity is an active research topic recently. Improvements on manually architecture searching performed by several teams in the following areas have led to major improvements in early design, like AlexNet, VGGNet, GoogLeNet and ResNet etc. [22, 23, 24, 25]. Recent advances in algorithm architecture have been made, including the exploration of hyperparameter optimization [26,27,28] and various network pruning methods [29, 30, 31, 32, 33, 34] and connectivity learning [35,36]. Many works committed to introduce sparsity [38] for structures or change the connections between the internal convolutional blocks, like ShuffleNet [37].

And for ADC techniques, the circuit design of level crossing ADC has been studied

for last several years [40, 41, 42]. The idea of LC-ADC is to sample a signal irregularly, that is to say, sampling moments are the points where the signal amplitude value is crossing the threshold levels. LC-ADC techniques have been used in several applications like, sampling of low-power speech [43], processing of ultra sound [44] and measurements of biological signal [45]. Compared to the conventional Nyquist ADC, LC-ADC achieves great performance improvement in power consumption, cost and silicon area [46,47]. The average sampling rate of LC-ADC could be much lower than conventional ADC with the same amplitude resolution [48]. For sigma delta ADC, the researches are mainly focused on circuit and product design [48, 49, 50, 51], and better signal noise ratio are obtained in these designs.

Chapter 2. Feature Extraction of Audio Signal

In this chapter, we will discuss the several widely used audio feature extraction methods which will be used in later experiments, such as Mel spectrum, Cepstrum analysis, and Mel Frequency Cepstral Coefficients.

Audio signals are the one-dimensional sequences in time domain, also known as waveforms, whose frequency variation cannot be visually observed. Although we could use Fourier transform to transform it into frequency domain, the frequency information we get is the overall distribution of all signal period, we still cannot observe the variation of frequency distribution with time scale. And if the signal in its whole time period is a non-stationary process, the frequency distribution for all periods makes non-sense for us. To deal with this problem, methods like short time Fourier transform (STFT), discrete wavelet (DWT) are applied for time frequency analysis. Based on these time frequency analysis methods, some further features for specific information are proposed to represent specific information of an audio signal, like Mel Frequency Cepstral Coefficients (MFCCs), is a well-known feature widely used in speaker recognition and automatic speech. For human speech signals, sounds are produced in sound channel, so sound channel is just like a system where the detail parts of original signal need to go through. The shape of the sound channel determines how a speech signal sounds like, and it is described by the envelope of short time power spectral. And the MFCC are a feature which could describes this envelope accurately, so MFCC is very suitable feature for speech signal. In our experiments, we assume that nature sound to be classified has some features in common with speech signal, so we consider to use MFCC as a feature engineering in audio event classification. Later we will discuss how MFCC could describe those specific speech information features and cepstrum analysis in detail.

2.1 Mel spectrogram

STFT is the most classical time-frequency domain analysis method for audio signals, which performs fast Fourier transform (FFT) on short-time signals. The short-time signal is derived from long-time signal framing. So, the principle of STFT is to frame and window a long signal, then we get the short-time frequency information of each frame by performing FFT on them. Finally, the spectrums of frames are stacked along the time scale, a two-dimensional matrix is generated, one dimension for time, another

for frequency, values represent the frequency amplitude. The two-dimensional signal, also known as sound spectrum map, is obtained by unrolling the original sound signal through the STFT. Figure 2-1 shows the processing progress of how to get spectrogram from original audio signal sequence.

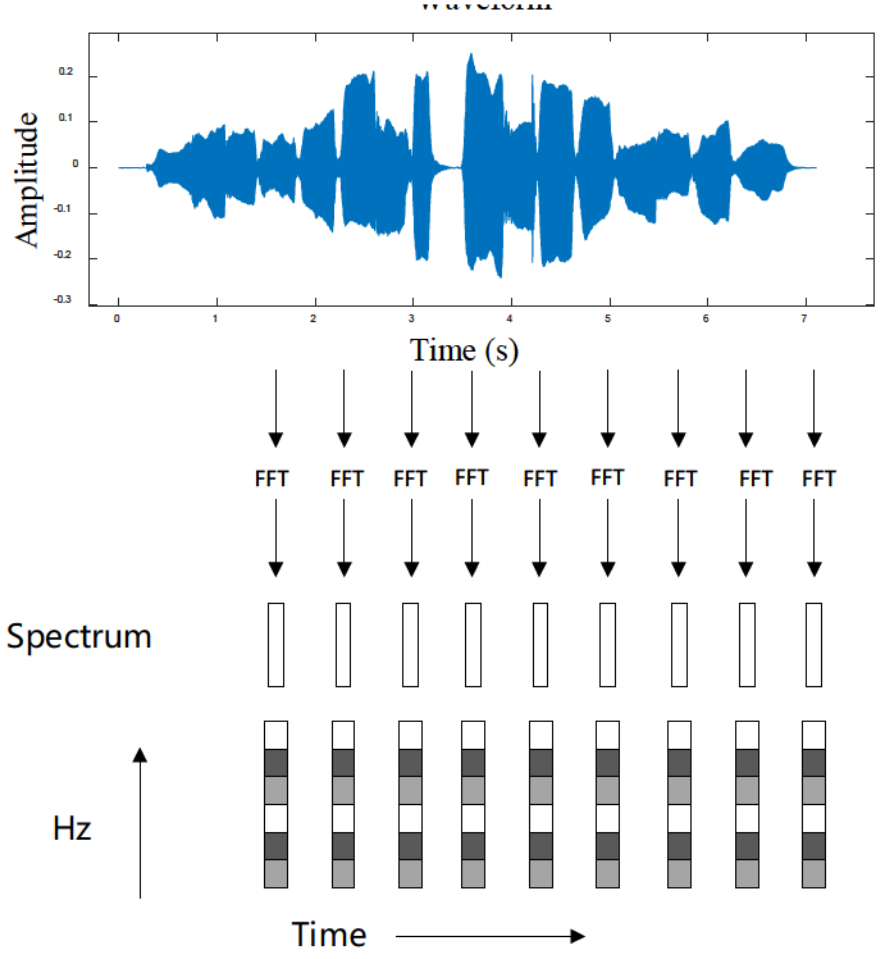


Fig.2-1 Spectrum of framed signal clips

Figure 2-2 shows the progress of how to stack the curve of spectrum of frames in time dimension. The spectrum of one frame is expressed through amplitude-frequency curve at first, as shown in the left of Figure 2-2. Then, rotates the spectrum in coordinate by 90 degrees to get the frequency-amplitude curve, as the mid of Figure 2-2. Then we map the values of amplitude to grayscale representation, where 0 represents black and 255 represents white, that is to say, larger amplitude value displays a darker area in gray graph. That's how we get the right of Figure 2-2.

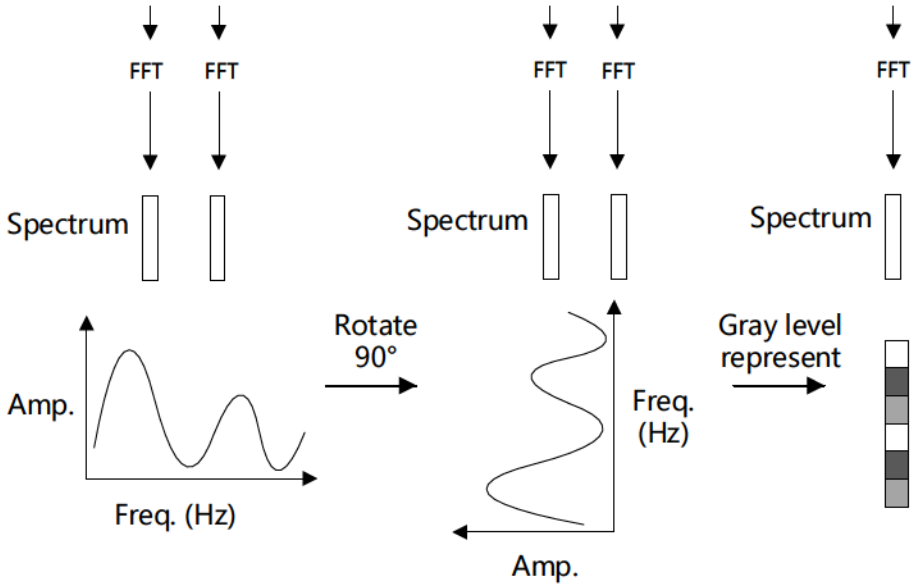


Fig. 2-2 Rotation and gray level representation for spectrum

This spectrogram map derived from STFT are usually called linear spectrum, because the amplitude varies linearly in Hertz, the unit of frequency. The frequency range that the human ear can hear is 20-20000 Hz, but human ear does not have a linear perception relationship to the scale unit of Hz, because our auditory system is a nonlinear system which has different response to different frequency, and it's good at extracting multilevel audio features. For example, when we adapt to a 1000Hz pitch, if the frequency of this pitch is increased to 2000Hz, our ears can only perceive that the frequency is increased a little, we will never notice the frequency is doubled. Experiments on human auditory shows that humans have a better ability at distinguishing subtle variations at low frequency than in high frequency.

The Mel measure associated with perceived frequency of pitch with its actual natural frequency. This nonlinear scale is more closed to what humans sensed. To convert linear frequency to Mel scale, we have formula (2.1):

$$M(f) = 1125\, ln(1 + f/700) \tag{2.1}$$

and from Mel frequency back to linear frequency, we have formula (2.2):

$$M^{-1}(m) = 700(exp\ (m/1125) - 1) \tag{2.2}$$

According to this formula, the linear spectrum can be mapped into the Mel-based

nonlinear spectrum based on auditory perception, as is shown in Figure 2-4.

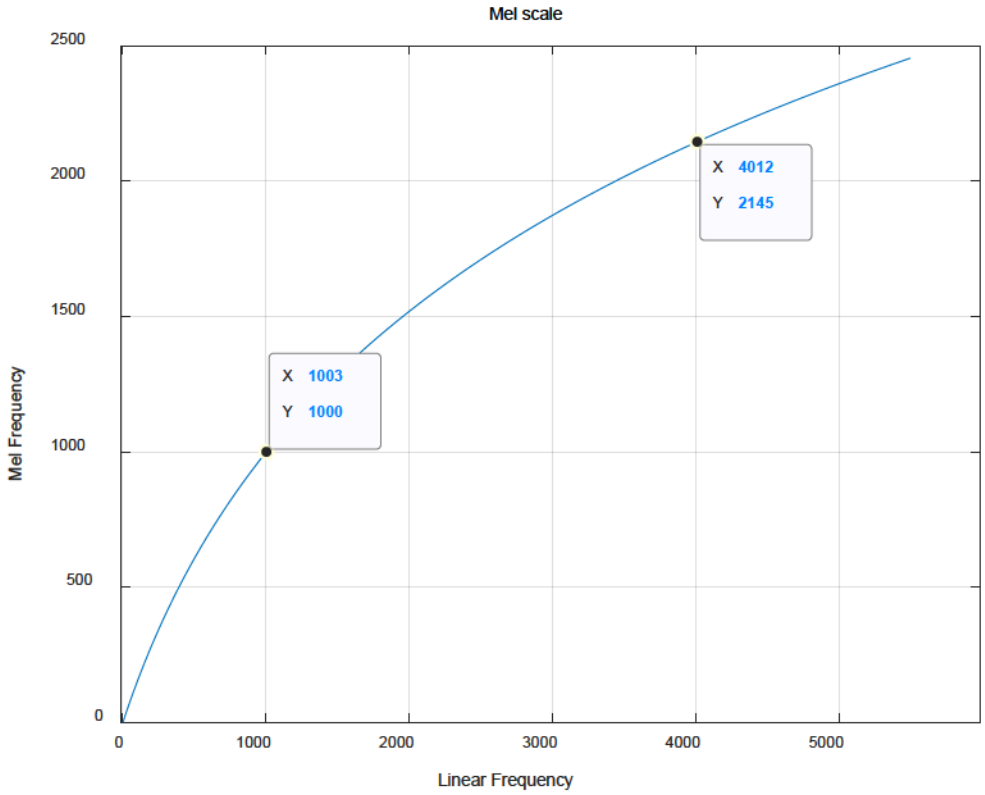


Fig. 2-4 Linear scale to Mel scale

Figure 2-5 shows the of Mel spectrum of an audio signal, which is the clip of human speech, this is derived in professional audio processing software (Adobe Audition).

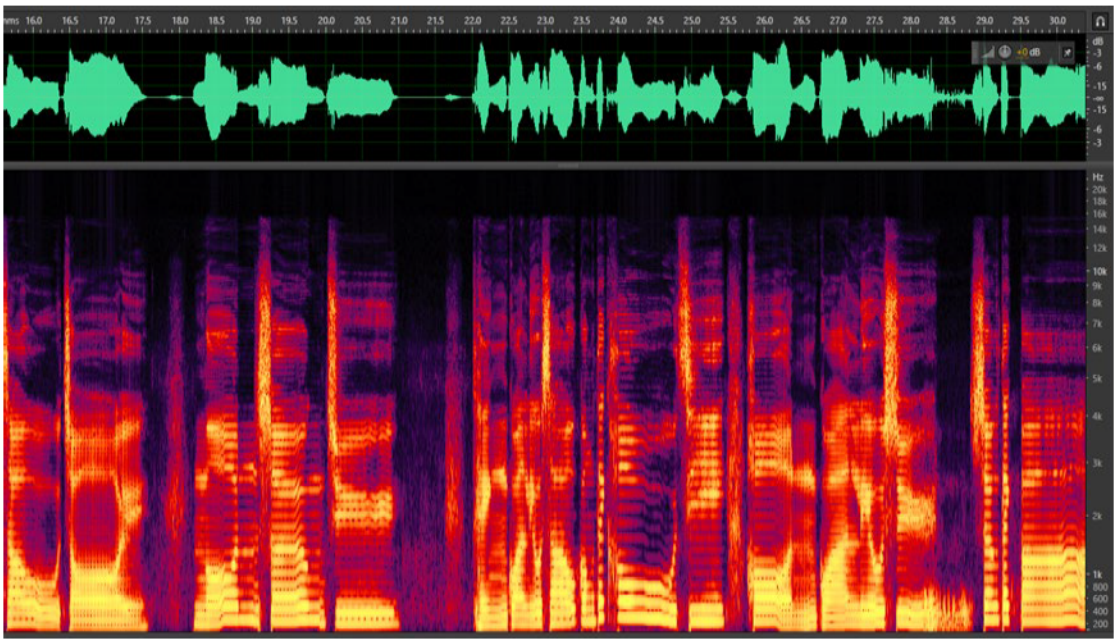


Fig. 2-5 Sound spectrum map of speech signal (from Adobe Audition)

2.2 Cepstrum analysis

To understand speech signal, vocal channel model needs to be discussed. Our vocal tract, whose shape are decided by throat, tongue, mouth and teeth are just like a filter or system response, filtering the sounds generated by vocal cord vibration. The shape of our vocal tract determines properties how voice sounds like. So, the accurately determined shape of vocal channel could represent features of phoneme being produced. The representation of shape could be represented by the envelope of power spectrum. To get the envelope, cepstrum analysis is necessary to discuss in detail.

Speech signal is composed of excitation source and vocal tract system components according to the above analysis. If we want to use these two components in different speech processing applications, they need to be analyzed independently. So, we have to separate excitation and vocal channel components from speech signal. Cepstral analysis gives method for extracting the separated excitation source and vocal tract components from speech signal without any priori information of each.

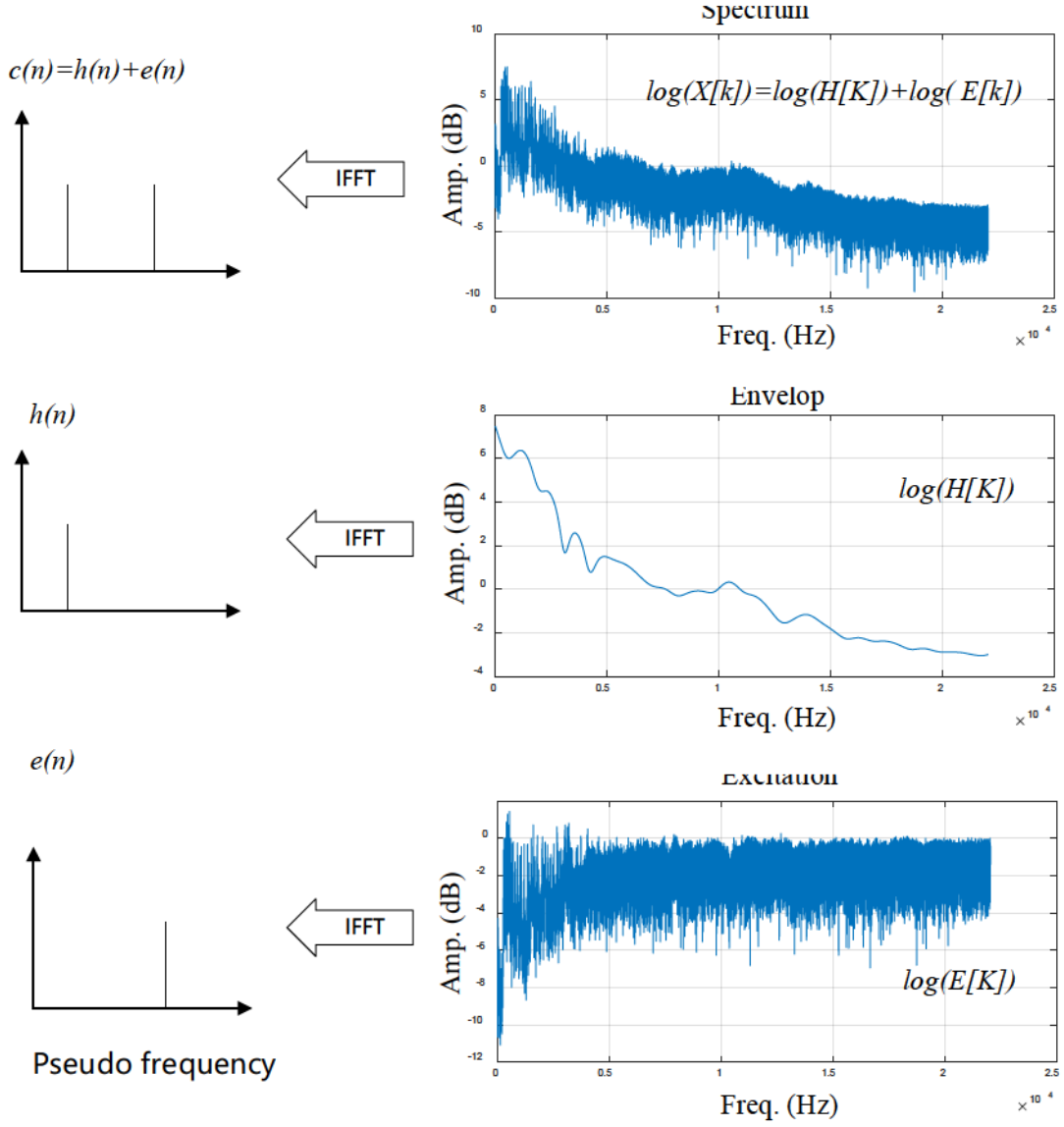


Fig. 2-6 Process of cepstrum analysis

As we can see from the curve of this spectrum in Figure 2-6, there are several peak values in this spectrum curve. These peak values, also known as formants, represent the main frequency components of a speech signal, so they carry the identification properties of a signal. A smooth curve which go through these formant points is called spectral envelope, which contains the information of sound channel. We can assume that the original spectral is combined with two main parts: envelop and details of the spectral. The details- of spectrum could be treated as excitation source part. so spectral envelop is the part for vocal tract. We can assume that speech signals are produced by source excitation with vocal tract system response through source filter theory. Thus,

the produced sounds can be thought of as a convolution of vocal channel filter and respective excitation source. We use $e(n)$ to represent the excitation sequence and $h(n)$ to represent the vocal channel filter, then the speech signal $x(\mathrm{n})$ could be expressed as follow:

$$x(n) = h(n) * e(n) \tag{2.3}$$

which in frequency domain is:

$$X[k] = H[K]\, E[k] \tag{2.4}$$

and the magnitude spectrum is,

$$|X[k]| = |H[K]|\, |E[k]| \tag{2.5}$$

Our purpose is to deconvolved the speech signal into excitation source and vocal tract system, which is not easy to do in time domain. However, it is not much difficult to convert the multiplication of the excitation and vocal channel to a linear combination in frequency domain. To get the individual components, cepstral analysis is the key to transform the linear combination into cepstral domain, then it's easy to get independent parts just using a filter liked stuff in cepstrum domain. We take logarithm to the spectrum $X[k]$ as well as the multiplication of the two components $E[k]$ and $H[K]$ to get the linear combination, which gives equation (2.6),

$$log(X[k]) = log(H[K]) + log(\, E[k]) \tag{2.6}$$

We can see from equation (2.6), logarithm operation turns multiplication operation into addition operation in the frequency domain. Then we take inverse Fourier transform to both sides, logarithm of signal spectrum $\log(\mathrm{X[k]})$ and the linear combination of these two components, $\log(\mathrm{H[K]}) + \log(\, \mathrm{E[k]})$ to separate them, just like Fourier transform on time domain to separate two signals with different frequency. The inverse Fourier transform of log frequency domain is called quefrency domain, which is similar to time domain. Relatively, the inverse Fourier transform of log spectral is called cepstral. This is progress can be described in equation (2.7), (2.8), (2.9),

$$c(n) = IDFT(log(X[k])) = IDFT(log(H[K]) + log(\, E[k])) \tag{2.7}$$

$$IDFT(log(H[K]) + log(\, E[k])) = IDFT(log(H[K])) + IDFT(log(\, E[k])) \tag{2.8}$$

$$IDFT(log(H[K])) + IDFT(log(\, E[k])) = \hat{h}(n) + \hat{e}(n) \tag{2.9}$$

where equation (2.8) is derived from the linear property of discrete Fourier transform.

In log frequency domain, the vocal channel components are the slowly varying parts, also known as envelop in Figure 2-6, and the excitation are the fast-varying part, also named details of spectrum. Relatively, in quefrency domain, the envelop part are located at low quefrency region as detail part are near the higher quefrency region. As we can see in Figure 2-6, the spectral envelope varies slowly so it determines the low quefrency part of speech spectral, while spectral details part varies fast thus determines the high quefrency part. Then these two parts could be separated easily by a low time liftering (correspond to low frequency pass filter). Liftering, as the name suggests, is corresponding operation of filtering in the frequency domain, which means we can get the interested quefrency region in quefrency domain by multiplying the cepstrum with a window where has values in the region we are interested. Similar to high frequency filtering and low frequency filtering in filters, there are also low-time liftering and high-time liftering for lifters. According to the properties of these two components in frequency domain, we could get the vocal channel parts by low-time liftering and extract the excitation parts in the quefrency domain by high-time liftering. Figure 2-7 illustrate the steps of converting a short-term signal waveform representation in time domain to its representation in cepstral domain.

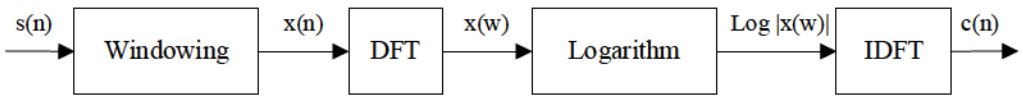


Fig. 2-7 Cepstrum analysis produce

2.3 Mel Frequency Cepstral Coefficients

We have discussed Mel spectrum and cepstrum analysis, now we will discuss how to get MFCCs based on these two progresses. We have already known that how important the vocal track characteristics is to speech signal and how to extract it from cepstral analysis. The mainly difference between MFCCs and vocal track extraction in cepstrum analysis is that MFCCs are the cepstrum analysis transformed in Mel spectrum, not linear spectrum. Generally, the implementation steps are listed below.

Step 1) is to frame the signal, which is also the first step of STFT. And the reason is the same, speech signal is a typical kind of time varying signal while short time speech

frames could be considered as stationary statically. Thus, we frame the signal into 20-50ms frames. Frame length is very important for STFT, the window length cannot be either too short nor too long. Short window length results in less frequency resolution while long window length generates low time resolution spectrogram, signals might has varied during the window period. We use $x(n)$ to represent the original speech signal in time domain, when we framed $x(n)$, we use $x_i(n)$ to denote the ith frame, where i ranges from 1 to the number of frames.

In some cases, there is also a pre-emphasis step to highlight the high frequency formants., The implementation method is as follows,

$$s_i(n) *' = s_i(n) - \beta * s_i(n-1) \tag{2.10}$$

$s_i(n)$ is the framed signal.

Step 2) is also very similar to the second step of STFT, but before apply FFT on each frame, we window each frame with analysis window $w(n)$, like hamming window, to smooth the edge of each frame and avoid the Gibbs Effect. Then power spectral of ith frame $S_i(k)$ are calculated through FFT:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)w(n)e^{-j2\pi kn/N}, \ 1 \le k \le K \tag{2.11}$$

where K is the lengths of the DFT. The purpose of windowing is to smooth the signal, where the sidelobe size and spectral leakage after the Fourier transform can also be attenuated. Hamming window function is as follows, α is usually taken as 0.46,

$$w(n,\alpha) = (1-\alpha) - \alpha\, cos(2 * \pi * n/(N-1)), \ 0 \le n \le N-1 \tag{2.12}$$

Signal is windowed by direct multiplication, and the implementation is as follow:

$$s_i(n) *' = w(n,\alpha) \times s_i(n), \ 0 \le n \le N-1 \tag{2.13}$$

$P_i(k)$ is the power spectrum of frame $s_i(n)$. And the power spectrum estimation based on periodogram is calculated by square the result of FFT, as follow:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \tag{2.14}$$

Step 3) is to compute the Mel-spaced filterbank. The obtained periodogram spectral has redundant information which is not desired for some specific audio application like Automatic Speech Recognition (ASR). Because our cochlea cannot distinguish two frequencies spaced closely, especially in high frequency region. Experimental

observations have shown that our ear is just like a filter bank, only specific frequency components could be sensed (our ear is selective to frequency). That is to say, our ear only passes signals of certain frequencies and filters out other frequency signals which are not expected to be perceived. What's more, these filters distribute on linear frequency scale non-uniformly. Many filters concentrate on the low frequency region while much less filters distributes sparsely in the high frequency region.

Therefore, we sum the periodogram bins of the cluster to see how much energy is present in each frequency region. Mel filterbank is used to simulate the human ear filter banks. We only interest in how much energy exist at each frequency range roughly. The range of first several filters are very narrow and indicates how much energy exists near low frequency range, for example, the first indicates the energy near 0 Hertz. As the frequency increase, Mel filters become wider as our attention to frequency variations get smaller. Mel scale tells us exactly how these filter banks are spaced in frequency axis and how wide to set for each filter bank. Figure 2-8 shows 10 Mel filter bank which imitates human auditory perception system about the variation of frequency.

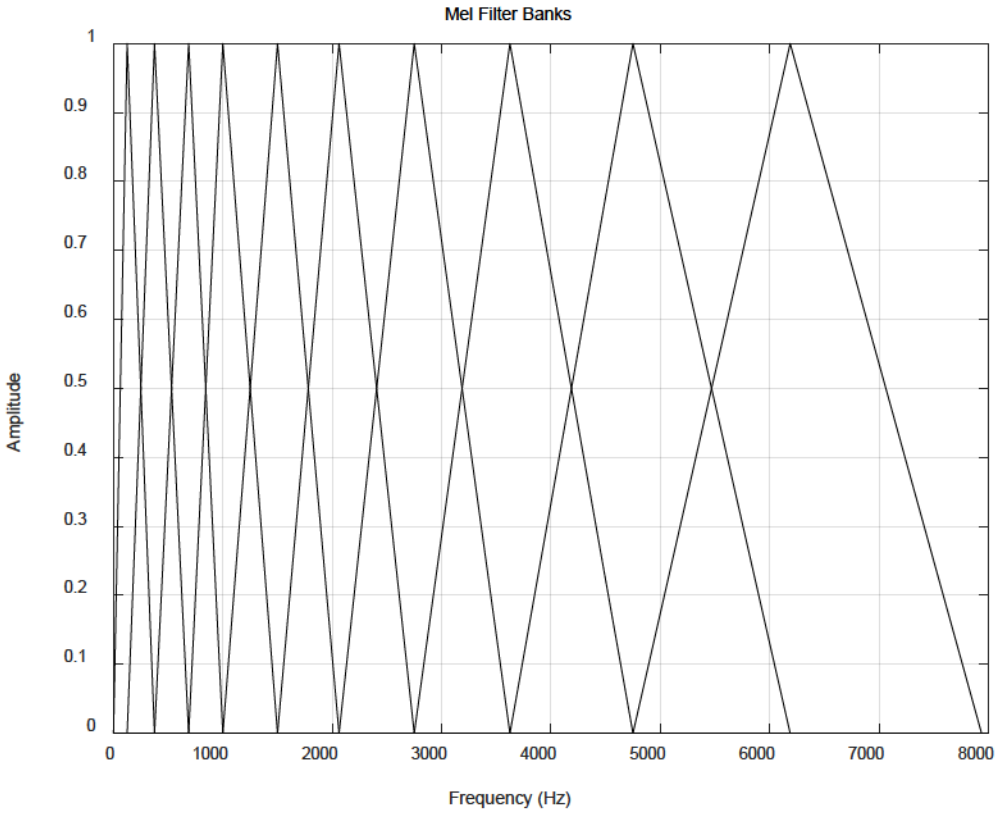


Fig. 2-8 The filter Mel Filter Banks

There are usually 20-40 Mel filter banks, we take 26 here. Each filter is a triangular filter in the form of a vector of the relative length, the length is half the FFT length in step 2) plus one, which is so called normalized frequency resolution. Most points of a vector are zeros, only non-zero for a certain frequency region. Then we calculate the Mel energies for each filter bank by calculate the inner product of each filter vector with the power spectrum vector. Finally, each filter bank has only one number left which indicates how much energy exists in each filterbank.

Step 4) is to take the logarithm of each Mel energies as we do in cepstrum analysis. Then we have 26 log Mel energies.

Finally, the step 5) is to use the discrete cosine transform (DCT) for the log Mel energies. The reasons for using DCT instead of IDFT are detailed below. The first reason is that the Mel filter banks are overlapped, so the Mel energies are correlated with each other. DCT could decorrelates the filter bank energies well so we just need to take the diagonal values of covariance matrices to extract features. We only keep the first half of the DCT coefficients, which is the power spectral envelop, as high DCT coefficients represent the fast-changing parts of Mel energy which is useless for recognition task according to experiments. So, the kept lower half coefficients after DCT are the MFCC features we need.

Besides the kept MFCCs, their differential and acceleration coefficients are often appended to the original MFCC vector. The MFCCs only describes the envelop of log Mel spectral for a frame, but the speech signal also has dynamic information, like the trajectories of MFCCs over time. According to experiments, appending the trajectories to the original MFCCs vector could increase the recognition performance. The length of feature vectors would double because one delta coefficient generated at the time steps of original coefficients. We use equation (2.15) to calculate the delta coefficients:

$$d_t = \sum_{n=1}^{N} \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.15}$$

where d_t represents delta coefficient derived from frame t, which is computed from the static coefficients c_{t+N} to c_{t-N}. N is usually set to 2. Delta-Delta (Acceleration) coefficients are also calculated with the equation, but the static coefficients is replaced with the deltas.

Chapter 3. Models for Sound Event Detection

3.1 Datasets

3.1.1 UrbanSound8K dataset

The first dataset used in our experiments is called UrbanSound8K [52] launched in 2014. This labeled urban sounds dataset including 10 classes: children playing, siren, car horn, drilling, street music, gun shot, jackhammer, dog bark, and engine idling. All the classes are chosen because they occur frequently in city noise complaints, except for “children playing” and “gunshots”, they are chosen for increasing the diversity of categories.

3.1.2 FSDKaggle2018 Dataset

The Second dataset used in our experiments is Free Sound Kaggle 2018 (FSDKaggle2018) [53]. As the name suggested, this is the dataset used in competition of Detection and Classification of Acoustic Scenes and Events 2018 (DCASE2018) task2, hosted in Kaggle. This dataset provides a total of 11,073 audio files, each file is uncompressed 16-bit PCM, 44.1 kHz, mono audio wav file. Clips files of each category is unequally chosen from the Audio Set Ontology [54] listed in table 3-1, which means this is an unbalanced data set. All audio clips have only one single ground truth label. Due to the preferences of users in Free Sound [55] when recording sounds and the variety of sound categories, the duration of audio clips ranges from 300ms to 30s.

Table 3-1 Categories composing FSDKaggle2018

Name	Clips	Time	Name	Clips	Time
Acoustic guitar	300	52	Electric piano	150	25
Applause	300	58	Fart	300	18
Bark	239	45	Finger snapping	117	6
Bass drum	300	13	Fireworks	300	48
Burping, eructation	210	12	Flute	300	46
Bus	109	28	Glockenspiel	94	8
Chime	115	24	Gong	292	42
Clarinet	300	35	Gunshot	147	11
Snare drum	300	18	Harmonica	165	19
Cough	243	22	Hi-hat	300	19
Cowbell	191	11	Keys jangling	139	19
Double bass	300	17	Knock	279	19
Drawer open, close	158	18	Computer keyboard	119	23
Cello	300	37	Meow	155	19
Microwave oven	146	25	Squeak	300	38
Oboe	299	15	Tambourine	221	10
Saxophone	300	34	Tearing	300	39
Scissors	95	16	Telephone	120	16
Shatter	300	26	Trumpet	300	28
Laughter	300	36	Violin, fiddle	300	27
Writing	270	48			

3.2 Solutions for experiments

There has been a lot of work in this area, recently, the performance of neural network is much better than other traditional machine learning methods. So, we only discuss the method of deep learning here.

3.2.1 MFCCs with RNNs

MFCCs (Mel Frequency Cepstral Coefficients) is a sequence of features of speech signals, describing the spectral envelope and energy information. As it's vector sequence in quefrency domain (similar to time domain, but not typically in time scale). An audio event clip could be described with such a sequence of MFCCs feature, so RNN is suitable classifier for these features to do classification tasks.

As MFCCs are very high-level features following determined human rules, so structure of neural network is unnecessary to be complicated, just need to capture the relations between these MFCC vectors. So here we use just to layers of bidirectional LSTM with 256 hidden cells to manipulate the input features, one fully connected layer to generate the output value of each class, followed by a softmax layer to generate the probabilities for each class.

For features extractions, we get rid of the audio clips less than 0.01s, which cannot even be recognized by human beings. And we skip the audio which are sampled using 24-bit ADC in dataset to keep consistent with most audio clips. For experiment hyper parameters, sampling rate are all set to 44100 Hz, window length for STFT is 50 milli second, and step length between each frame is 20ms, applying 1024 point-FFT on each frame, number of mel filter banks is 26, minimal frequency is 125 Hz, while maximum frequency is 7500 Hz. We take the lower 12 mel coefficients and their related delta coefficients, appending energy and delta energy as the whole feature for one frame, so one frame of audio can be represented by these 26-dimensional vectors. And for training, cross entropy are used as loss function while Adam as the optimizer, 10e-4 learning rate, and batch size is 128 to feed into neural network.

After 10 epochs for all chosen samples, we get a 91.65% accuracy for test set, which is 25% random chosen samples of our whole dataset, Figure 3-1 shows the confusion matrix of our test set.

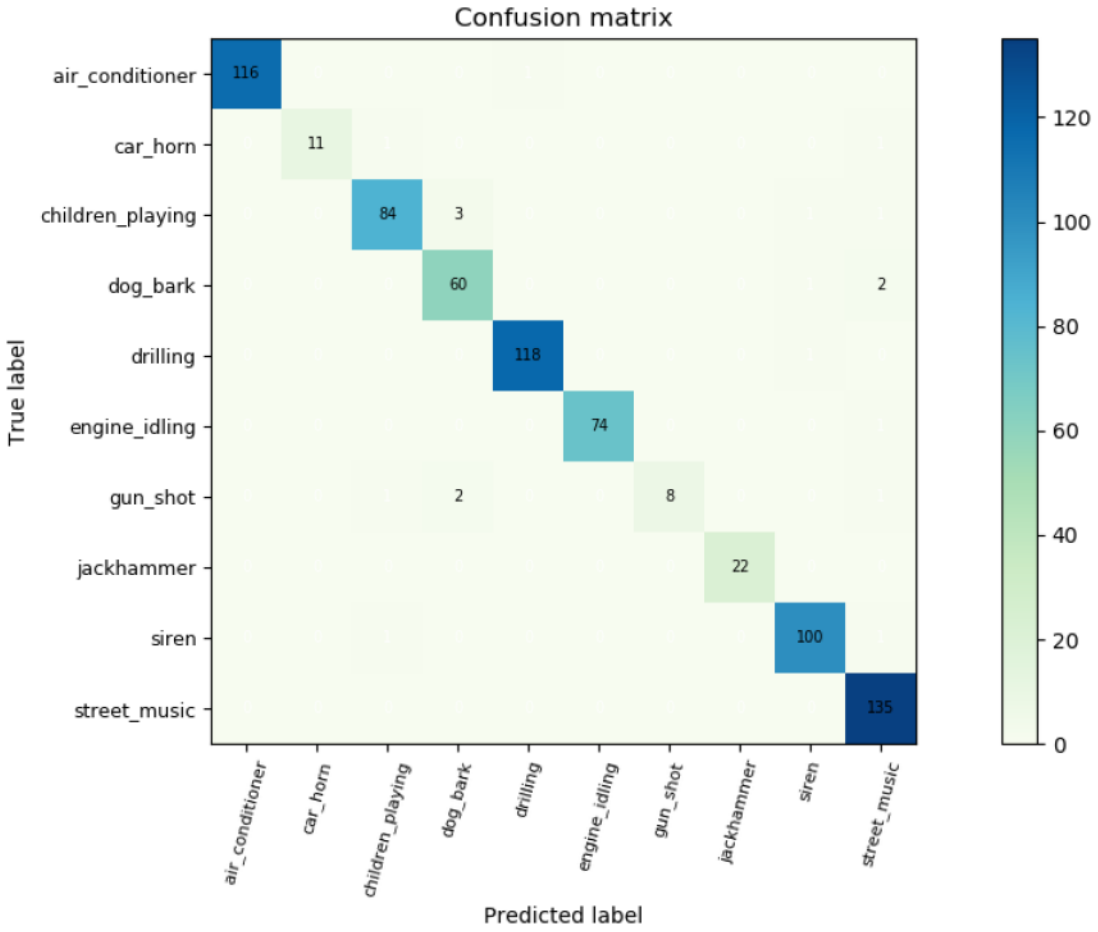


Fig. 3-1 Confusion matrix of test set

As we can see from the confusion matrix, our neural network classifier is doing well to distinguish most classes, except those short time event, like gun shot and dog bark, this might be related to the determined feature we chose, as we all know that MFCCs are not good at describe high frequency audio signals.

3.2.2 Log Mel energy with CNNs

Mel spectrogram is a set of frequency information about a short frame stacked in time domain, so it contains both time domain information and frequency domain information of an audio signal. As spectrogram could be processed as a picture, so an audio event clip could also be described with such a mel spectrogram, and CNN is more suitable for these features to do classification tasks. High level features which might be

useful for classification, can be extracted by convolutional blocks automatically.

For preprocessing, all stereo audio files are converted to mono wav files. We apply log Mel filter banks on the extracted spectrograms and perform logarithm operation on the result to get log Mel energy spectrum as our input feature. The number of the Mel bins is chosen to be 64 because it could be divided by 2 while max pooling and it is power of 2. The cut off frequency of Mel filter bank is 50 Hz. We subtract the mean and divide the standard deviation of Mel frequency bins to normalize the log-Mel spectrogram. We apply the same configuration for CNN4 and CNN8 for this task. Here Figure 3-2 is a spectrum sampled from our training set.

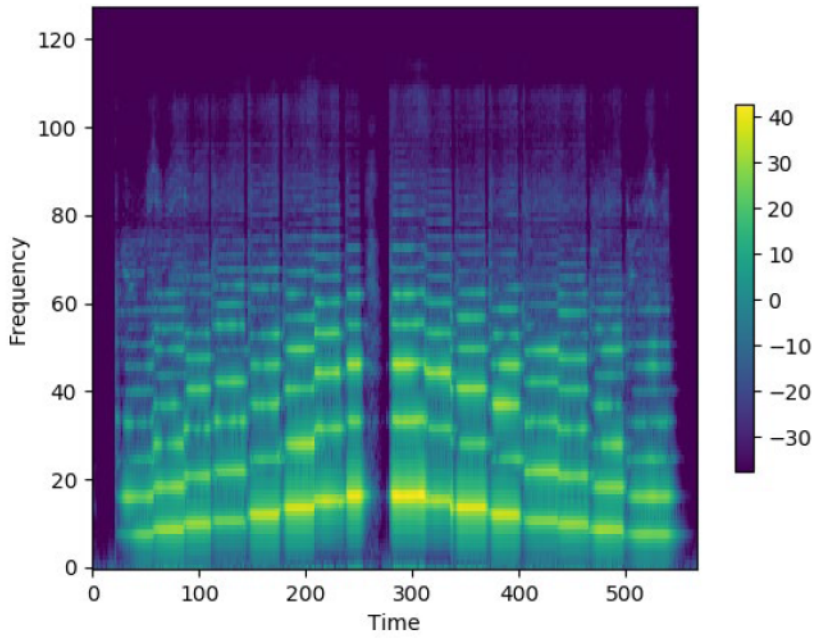


Fig. 3-2 Sampled Mel spectrum

For structure of neural network, we use CNNs style net. These CNN style net, such as ResNet and DenseNet, is the state-of-the-art models in image area. But these two kind nets are typically more than 100 layers to gain the better performance, with much more layers for little increasement of performance. The typical CNN structures usually include several convolutional blocks which consists of convolution layer with pooling layer. And fully-connected layers are connected with the last feature map. Each convolutional layer has several channels, one channel is a filter. And each filter connects with every feature map in each channel of the previous layer. The filters could capture local features of a pattern, such as lower layers might extract edges while complex contours may be extracted by higher layers' filters. In our experiments, we use 4

convolutional blocks and 8 convolutional blocks as models, as they are cost-effective for the trade of between performance and model complexity. We call the two models CNN4 and CNN8 in this thesis. The filter size of convolution layer in CNN4 is 5 × 5, while it is 3 × 3 in CNN8. to stabilize training, batch normalization (BN) is used after each convolutional layer, then rectified linear unit (ReLU) nonlinearity follows the BN layer. Then a global max pooling (GMP) layer follows the outputs of the last convolutional layer to summarize the last layer's feature maps. Finally, we use a fully-connected layer to connect the output vector of GMP, and softmax layer after fully-connected layer is used to calculate the final probabilities for each class. Architectures of CNN4 and CNN8 are detailed in Table 3-2.

Table 3-2. CNN model architecture

Feature map size	CNN4	CNN8
T × 64	Log mel spectrogram	
T/2 × 32	5 × 5, 64	$\begin{bmatrix} 3\times 3, & BN \\ 3\times 3, & BN \end{bmatrix}$, 64
	2 × 2, max pooling	
T/4 × 16	5 × 5, 128	$\begin{bmatrix} 3\times 3, & BN \\ 3\times 3, & BN \end{bmatrix}$, 128
	2 × 2, max pooling	
T/8 × 8	5 × 5, 256	$\begin{bmatrix} 3\times 3, & BN \\ 3\times 3, & BN \end{bmatrix}$, 256
	2 × 2, max pooling	
T/16 × 4	5 × 5, 512	$\begin{bmatrix} 3\times 3, & BN \\ 3\times 3, & BN \end{bmatrix}$, 512
	2 × 2, max pooling	
	Global max pooling	
	Classes num. fully connected, sigmoid or softmax	
Parameters	4309450	4691274

For training, the optimizer we use is Adam, the learning rate is 0.001 and reduce 0.1 times every 1000 iterations after step 10000. The batch size is 128 for one step. And Table 3-3and Table 3-4 shows the mean average precision of CNN4 and CNN8.

Table 3-3 Mean average precision (MAP) results of CNN4

Name	**MAP**	**Name**	**MAP**	**Name**	**MAP**
Acoustic guitar	0.8662	Finger snapping	0.6015	Microwave oven	0.8318
Applause	0.9754	Fart	0.7852	Oboe	0.852
Bark	0.8798	Electric piano	0.6734	Saxophone	0.7843
Bass drum	0.8221	Fireworks	0.8759	Scissors	0.6086
Burping, eructation	0.9048	Flute	0.9425	Shatter	0.8329
Bus	0.6172	Glockenspiel	0.6927	Snare drum	0.8031
Chime	0.9827	Gong	0.9511	Squeak	0.9704
Clarinet	0.8343	Gunshot, gunfire	0.527	Tambourine	0.9397
Computer keyboard	0.6745	Drawer open, close	0.8545	Tearing	0.6381
Cough	0.8762	Hi-hat	0.7857	Telephone	0.7494
Cowbell	0.8395	Keys jangling	0.8405	Trumpet	0.8072
Double bass	0.8661	Knock	0.6384	Violin, fiddle	0.8948
Harmonica	0.9361	Laughter	0.9762	Writing	0.7987
Cello	0.969	Meow	0.9028	Overall	0.8196

Table 3-4 Mean average precision (MAP) results of CNN8

Name	MAP	Name	MAP	Name	MAP
Acoustic guitar	0.8946	Electric piano	0.6582	Microwave oven	0.8287
Applause	0.9677	Fart	0.8549	Oboe	0.8641
Bark	0.882	Finger snapping	0.6399	Saxophone	0.7849
Bass drum	0.8495	Fireworks	0.9257	Scissors	0.7243
Burping, eructation	0.8826	Flute	0.9024	Shatter	0.8616
Bus	0.7356	Glockenspiel	0.7824	Snare drum	0.8382
Chime	0.9561	Gong	0.9781	Squeak	0.9872
Clarinet	0.9069	Gunshot, gunfire	0.6051	Tambourine	0.9518
Computer keyboard	0.742	Harmonica	0.9157	Tearing	0.734
Cough	0.8437	Hi-hat	0.7749	Telephone	0.7849
Cowbell	0.9375	Keys jangling	0.8084	Trumpet	0.8434
Double bass	0.8934	Knock	0.6893	Violin, fiddle	0.9107
Drawer open, close	0.8624	Laughter	0.9206	Writing	0.8894
Cello	0.9621	Meow	0.9164	Overall	0.8461

As shown in Table 3-3 and Table 3-4, CNN8 achieves a mean average precision of 0.8463 outperforming CNN4 net of 0.8210, respectively. Sounds classes such as “Cello” and “Meow” etc. have very high precision up to 0.9 event 1. but some classes such as “Glockenspiel” and “Finger snapping” have only 0.3-0.4 precision.

3.3 Solution for competition

We will talk about the state of art models and some useful tricks for sound this task in competition, and discuss the reason why we chose not to use these state of art models.

3.3.1Multi model ensemble

In competition of FSD Kaggle 2018, most of the teams in the top of leaderboard have applied voted decision strategy [18, 19, 20] by multi classifier with different features, like the two methods mentioned preciously. Because there are kinds of classes of events and differ from each other, so one classifier may only consider parts of these differences. For example, MFCCs has the ability to distinguish low frequency signals while Mel spectrogram focus on the whole frequency domain in Mel scale. If we have multi classifier and each take consider of some aspects of those varies features, the combination of their opinion may give the best decision.

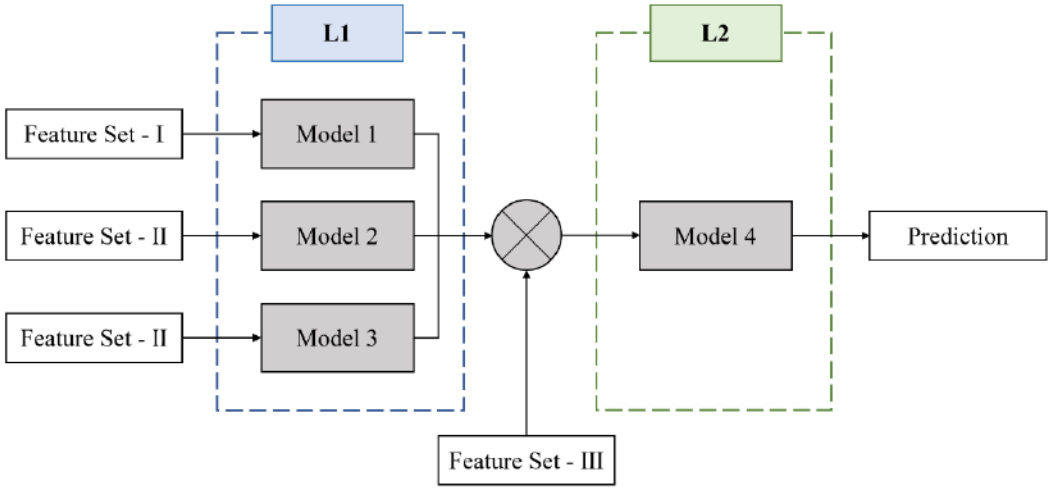


Fig.3-3 Multi model decision strategy

For example, Figure 3-3 shows a multi model decision strategy which are used in [19]. As we can see, there are two level models in their architecture, the three level 1 models take different feature sets as the input to predict the probabilities of all categories, then the predictions of level 1 models with another feature set are feed into level 2 model together to make the final decision. This multi model decision strategy is like the situation that one expert take the advices of other multi experts from different angles, then make the final decision. In this way, each model in level could achieve

nearly 0.9 average precision and the final precision could achieve 0.94 average precision. Of course, other competition tricks like classic data augmentation methods, time stretch, pitch shift and balanced mini dataset also contribute to the final result.

3.3.2 Discussion

In this section, we will discuss the role of these methods in real competition and our experiments. For real competition, good result is at the first place, so researchers would like to use multi and complex models to get a good grade, and kinds of tricks are appeared in competition. These are definitely the forward direction of the research on this sound event detection task, and we need to catch up these advanced methods. However, in this experiment, our purpose is to reduce the data amount of audio signals, these multi model strategy increase the model complexity hugely while increasing the performance. And the computation load might beyond the ability of embedded devices if compression methods are not used. Thus, we chose two relative lightweight but well performance models for our next experiments.

Chapter 4. Data Reduction Methods

Instead of sampling real audio signals, simulations are implemented for these three methods we proposed. Because datasets are precious resources for this kind of tasks and they are not easy to collected. It's a more proper way to do simulations from original dataset we already have than collecting real audio datasets using those sampling methods. These three methods and simulations are discussed in details.

4.1 Down Sample for audio signals

4.1.1 Principle and simulation

The first method is to down sampling for audio signals directly, in hardware side, we just need to use lower sample rate and lower quantization bit width. Recent years, with the development of AD convert circuit, the sampling rate could achieve up to 96KHz sampling rate, for high quality DVD and some application like that. Sampling rate for main acquisition cards in market is generally divided into three levels of 22.05KHz, 44.1KHz, 48KHz. 22.05KHz can only achieve the sound quality of FM broadcast, 44.1KHz is the theoretical limit quality of CD sound, 48KHz is for movie or professional audio. But in early years, 8KHz sampling rate is used in telephone, that's enough for human speech, so if we use 44.1KHz for speech sampling, there 5 times redundancy to acquis. In embedded applications, where resources are very critical in a system, 5 times redundancy could cause huge computation resource wasting, which inspired us to do the following experiments.

The first we tried is to down sample directly. The sampling rate of wav files in these two datasets are all 44.1KHz. We implemented the experiments with 32KHz, 22.05KHz, 11.25KHz and even 8KHz. Wav files are resampled and transformed to frequency domain to do feature extraction, the mel spectrum of resampled signal with original spectrum are shown in Figure 4-1. Then features are feed into neural network to train the classifier. Figure 4-1 shows the spectrum with 22050 Hz sample rate, we can see although the general shape is kept, but energy leakage starts to appear, it's even clearer with less sample rate.

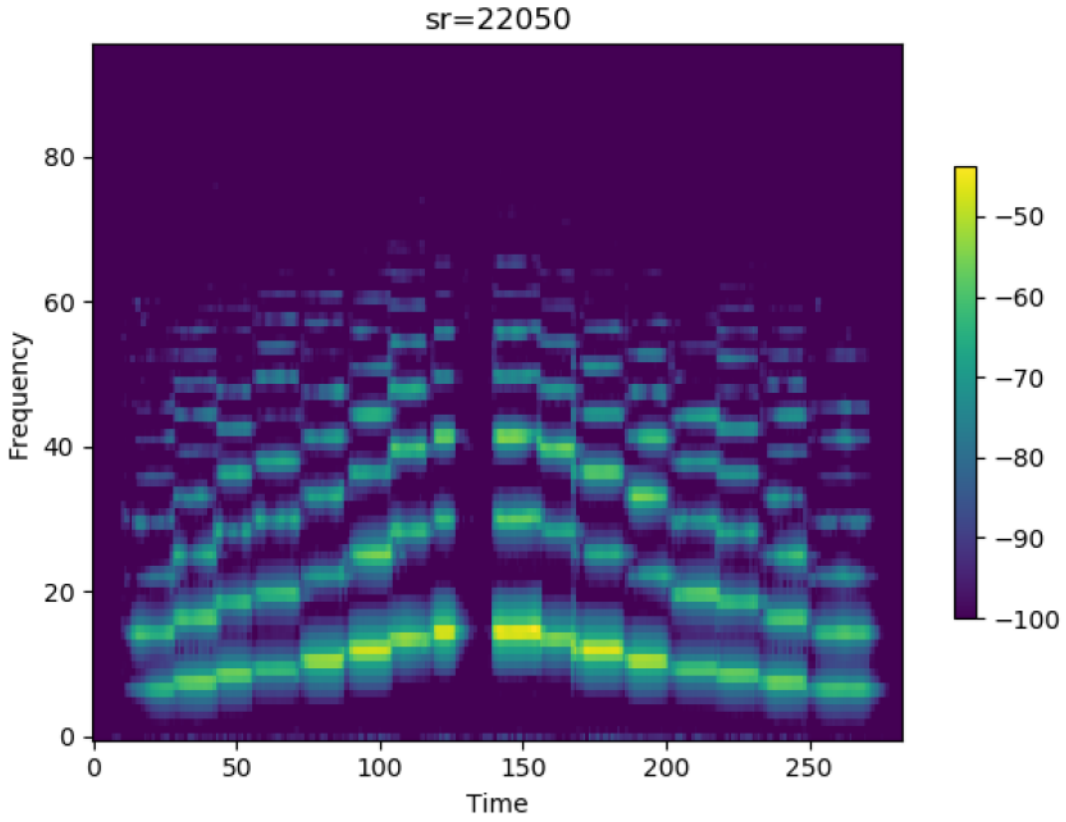


Fig. 4-1 Mel spectrum with sample rate 22050 Hz

Sample width of audio signals could be reduced with the same reason. As 16 bits width are used for most general audio application, 24 bits width are usually used in high quality application. We would like to see the influence of quantization error for this task, so sampling width are reduced to 12 bits, 8 bit and even 4 bits with different sampling rate to confirm the influence. New values after changing of bit width is described by the following equation:

$$x_{new} = x_{old} \times 2^{new\_width} / 2^{old\_width} \tag{4.1}$$

Also, the signals with new quantization values are feed into neural network with the same operation.

4.1.2 Analysis of experiment results

For experiments, we apply those above operations in both dataset and the results are shown in Table 4-1. As we can see, with the reduction of sampling rate, accuracy is not hurt much, with only 3 to 4 percent decline as 2 to 4 times reduction of sampling rate,

and even in 8KHz sampling rate, accuracy is still acceptable. In other words, the amount of signal data is reduced by 2 to 4 times.

Table 4-1 Result for changing sample rate (Urban8K dataset)

Sample rate	Accuracy	epochs	Sample width
44100 Hz	91.65%	10	16 bits
32000 Hz	91.37%	10	16 bits
22050 Hz	89.39%	10	16 bits
11025 Hz	86.73%	10	16 bits
8000 Hz	82.69%	10	16 bits

Table 4-2 shows the oppsite result with Table 4-1. Variation in quantization precision has much more loss in performance, results of 12 bits sampling width is in a reasonable range while 8 bits sampling width is unacceptable. But if we count for the data reduction rate, it's only 4/3 amount of data are reduced, while 2 times precision loss could cause bad result, so it's less effective compared with down sampling.

Table 4-2 Result for changing sample rate (Urban8K dataset)

Sample width	Sample rate	Accuracy	Epochs
16 bits	44100 Hz	91.65%	10
12 bits	44100 Hz	86.54%	16
8 bits	44100 Hz	72.78%	16
4 bits	44100 Hz	45.97%	16
16 bits	32000 Hz	91.37%	10
12 bits	32000 Hz	85.16%	16
8 bits	32000 Hz	72.41%	16
4 bits	32000 Hz	45.20%	16

With results of these two experiments, we could derive that down sampling is a more efficient method for reduce data amount of audio signals with little hurt of recognition performance. So, the following two experiments are mainly based on reducing sampling rate while keeping high quantization precision.

4.2 Sigma Delta ADC for audio signals

Signal delta ADC is a typical way to acquis high resolution signal with low data rate. So, we implemented experiments by signals with signal delta ADC.

4.2.1 Principle and simulation

Sigma delta ADCs are suitable for converting analog signals over a wide range of frequencies. Basically, sigma delta ADCs consist of a modulator which oversampling signals and convert them to 1-bit data, and a digital decimation filter follows the modulator which decimate oversampled 1-bit data and output high resolution data stream.

The input analog signal is oversampled in converter. And sampling rate to oversample the analog signal is much faster, usually hundreds of times larger than the data rate of digital output. The two main components of sigma delta ADC are the sigma delta modulator and digital filter with decimator. Sigma delta ADC's general structure is shown in Figure 4-2. As we can see, delta sigma modulator samples the analog input signal with a high sample rate and convert sampled signal into 1-bit data-stream. Then the digital filter with decimator takes the sampled 1-bit data-stream to converts it into digital output with high-resolution, low data rate. There is only one sample rate for most traditional converters, however, in sigma delta ADC, there are two, the first is the input oversampling rate f_S and the second is output data rate f_D.

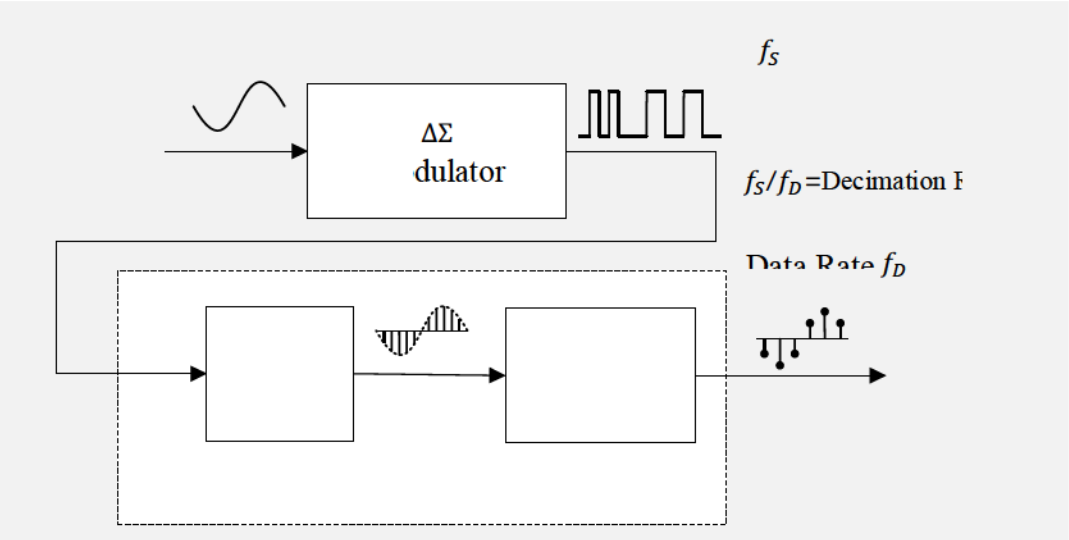


Fig. 4-2 General structure of sigma delta ADC

The core part of the sigma delta ADC is the modulator, which converts the analog input signal to digital 1-bit stream and reduce the low frequency noise, because the low frequency noise is pushed into high frequency due to over sampling. We call this function noise shaping, noise in low frequency is shaped to high frequency where it is outside the signal frequency band. This is one of the reasons why sigma delta ADCs are well suited for the measurements of low-frequency, high resolution signal.

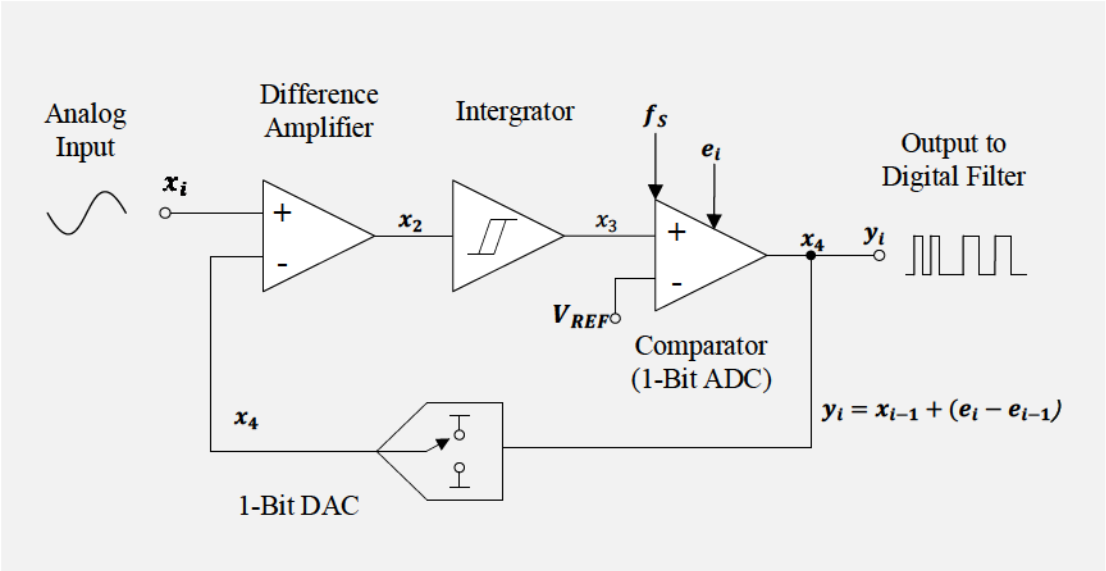


Fig. 4-3 Sigma delta modulator in time domain

Now the sigma delta modulator will be discussed in detail. We have two perspective to analysis the sigma delta modulator: time and frequency domain. Block diagram in time domain is shown in Figure 4-3, which is an example of first-order modulator. The analog input signal is converted to high data rate 1-bit modulated pulse wave by sigma delta modulator. From time domain analysis, sigma delta modulator takes input analog signal to generate 1-bit data stream. The system clock samples with the 1-bit comparator in modulator together. In this way, the quantization of the sigma delta modulator is implemented with the same sampling rate as system clock. Similar to all quantizers, the input voltages are represented by digital values, sigma delta modulator use a 1-bit digital stream here, so, the input analog voltage is represented by the ratio of the number of bits one to number of bits zero. And difference with most quantizers is that the sigma delta modulator has an integrator that shapes the quantization noise to higher frequency region. Therefore, the output of the modulator has non-flat the noise spectrum.

In time domain analysis, shown in Figure 4-3, the process of convert analog input to1-bit stream generates quantization noise of convert. So, the modulator output is the sum of the input signal and quantization noise, which is the Difference between the current quantization error and the previous quantization error.

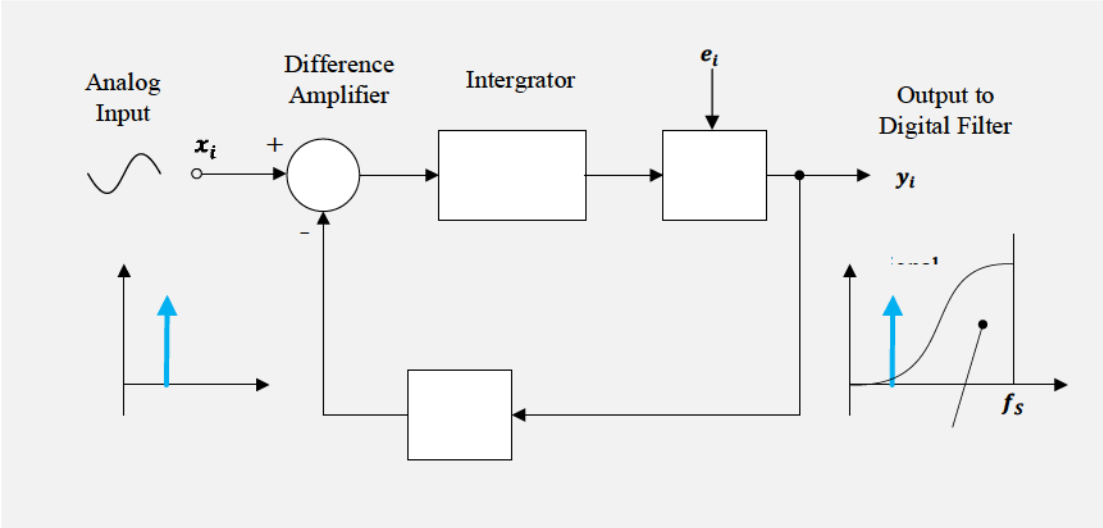


Fig. 4-4 Sigma delta modulator in frequency domain

Analysis in frequency domain is shown in Figure 4-4, we can see the location of quantization noise in output spectral. It also shows that how the modulator affects the noise and produces high resolution results. View on frequency domain, the output in time-domain is clearly separated as the input signal part and shaped noise part in high

frequency. It's clearly to see the noise properties in frequency domain, then we should understand the frequency operation of modulator as well as how the sigma delta ADC could achieve the ability of generating such high resolution for signals.

After sigma delta modulator, data stream come to the digital filter and decimator. The digital filter implements the function of low-pass filter by sampling the 1-bit stream of modulator. Fig. 4-5 shows the most common digital filter in sigma delta converters, which is a first-order low-pass averaging filter. As we can see from this figure, the digital filter is a weighted averaging filter. Sinc filters is one kind of the averaging filters used in almost all the sigma delta ADC. Many sigma delta ADCs use Sinc filters combined with other filters as part of the two-stage decimation process, especially for audio signal. There are only the Sinc filter used in sigma delta ADCs for Low-speed signals like in industrial application.

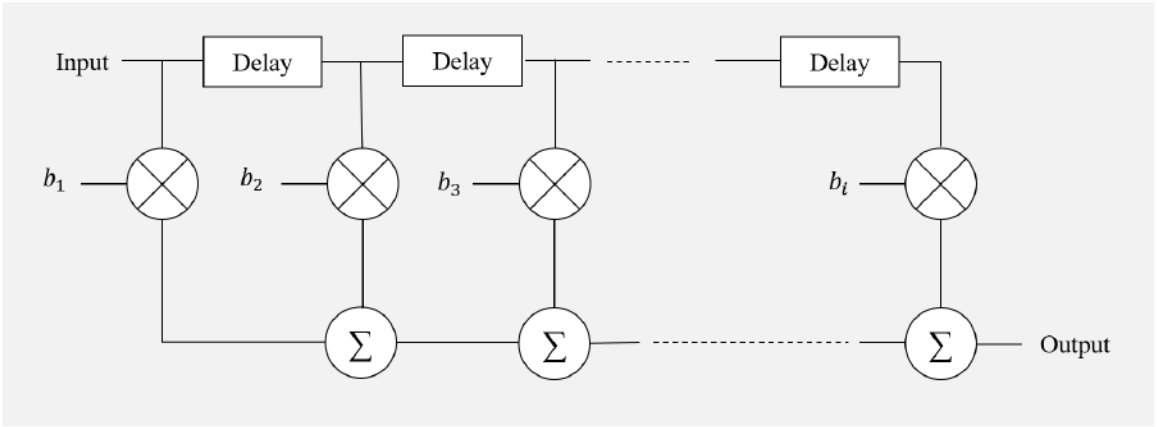


Fig. 4-5 First-order low-pass averaging filter

Output of digital filter as shown in Figure 4-6, has the same data rate with the oversampling rate. We can view the output in two perspective, frequency and time domain. In time domain, digital filter's output has a very high resolution, as shown in

(a), for example, 24 bits are used to represent a value of sampled signals. In frequency domain, the digital filter is just like low-pass filter applying to the signal. In this way, quantization error could be attenuated, at the same time, the frequency bandwidth is also reduced, just like any low-pass filter would do. As the quantization noise decreases, the signal reappears in the time domain.

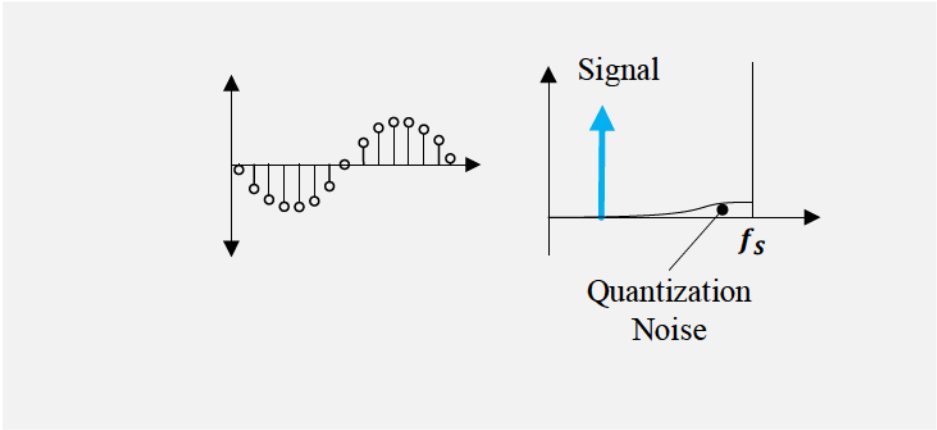


Fig. 4-6 Outputs of digital filter

The second part is the decimator. It is easy to understand the behavior of decimator to the samples of digital filter. In the circuit decimator, the output rate of digital signal is reduced by just discarding or ignoring parts of the output samples, which is also known as down sampling.

It seems to be unpleasant to throw away parts of the original complete signal with a lot of samples and only leaves the general shape of original signal. However, most of original samples are only the copies of nearby points produced by filters. In fact, according to the Nyquist theorem, the spectral information of the decimated waveform with only general shape is almost the same with the previous complete waveform, now we have a more efficient data rate. Decimating some samples does not result in any loss of information. The decimation process is shown in Figure 4-7.

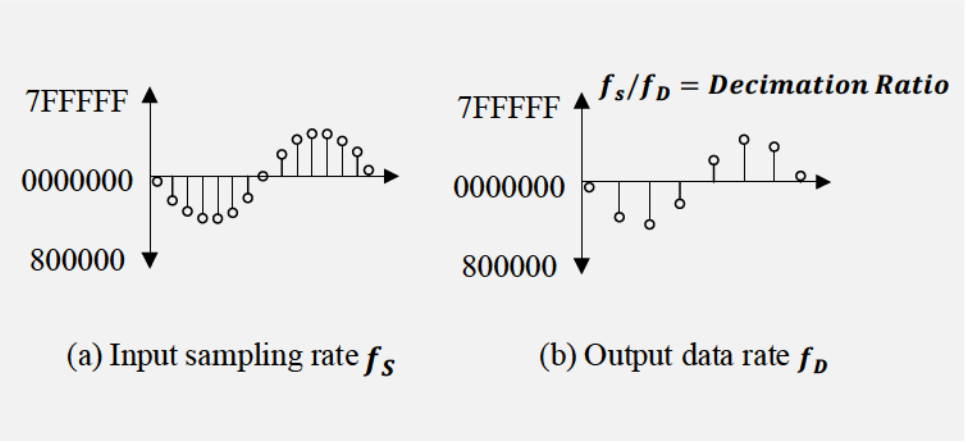


Fig. 4-7 Digital filter's output from decimation process

Figure 4-7(a) shows the output of digital filter's in time domain. Figure 4-7(b) shows the output signal after decimating process. This is the complete process of the data steam moving through the digital filter and decimator in sigma delta ADC.

4.2.2 Analysis of experiment results

According to the principle of sigma delta ADC, we have done the simulation for audio signal. Figure 4-8 is the raw spectrum of one audio sample in our dataset.

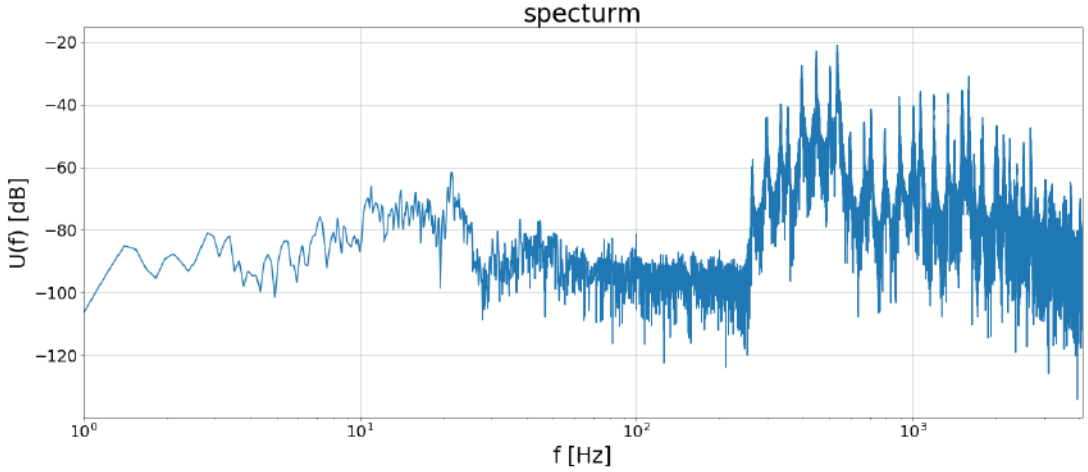


Fig. 4-8 Raw spectrum

And the input and output of sigma delta modulator are shown in Figure 4-9. Red line is input waveform and blue line is the discrete one bit output.

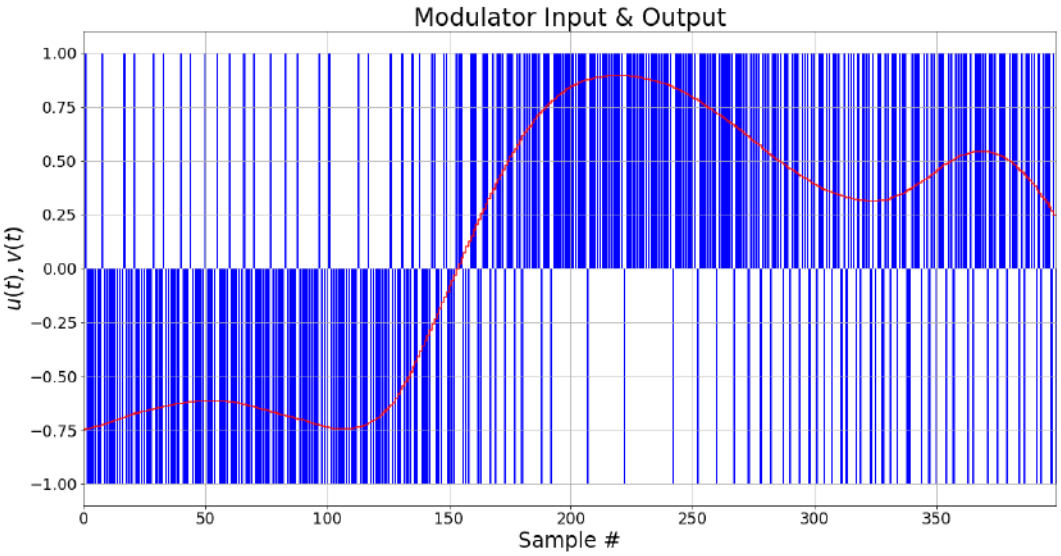


Fig. 4-9Input and output of sigma delta modulator

After sigma delta modulator, we could see the effect of noise shaping in Figure 4-10

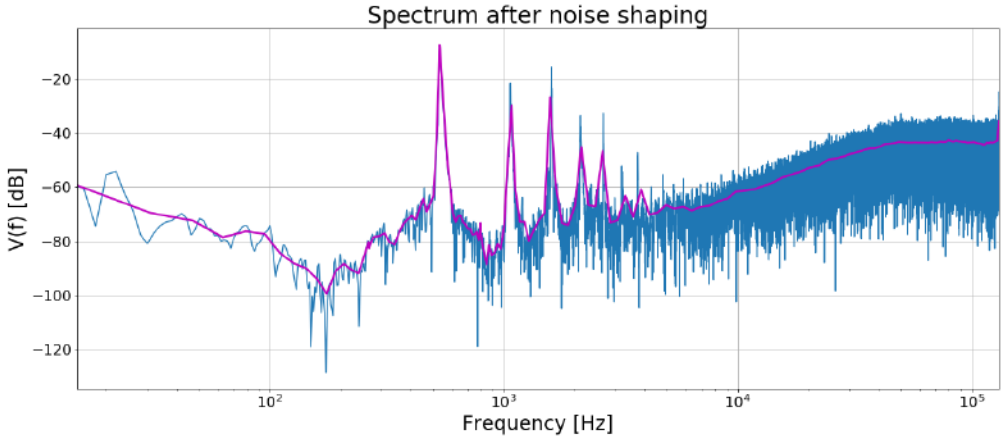


Fig. 4-10 Spectrum after nose shaping

Finally, signals are generated after digital filter and decimator. Figure 4-11 shows the raw waveform with the simulated waveform. As we can see the quantization error of this simulation is in green line, from -0.050 to 0.075, compared with the max normalized value 1.

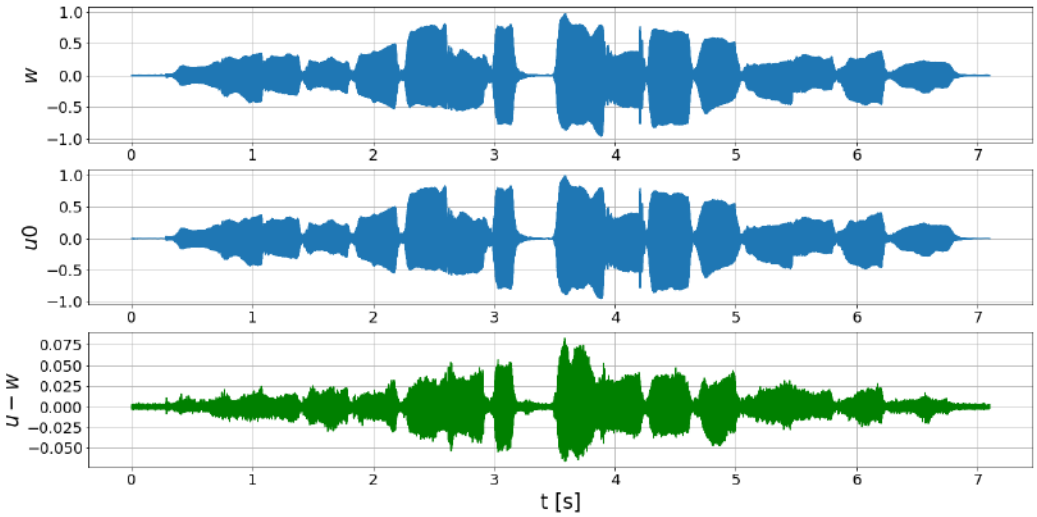


Fig. 4-11 Raw waveform with the simulated waveform and quantization error

And the spectrum of original signal, filtered noise and simulated signal are shown in Figure 4-12. As we can see spectrum of simulated signals are similar to the original and much more noise have been filtered by sigma delta modulator.

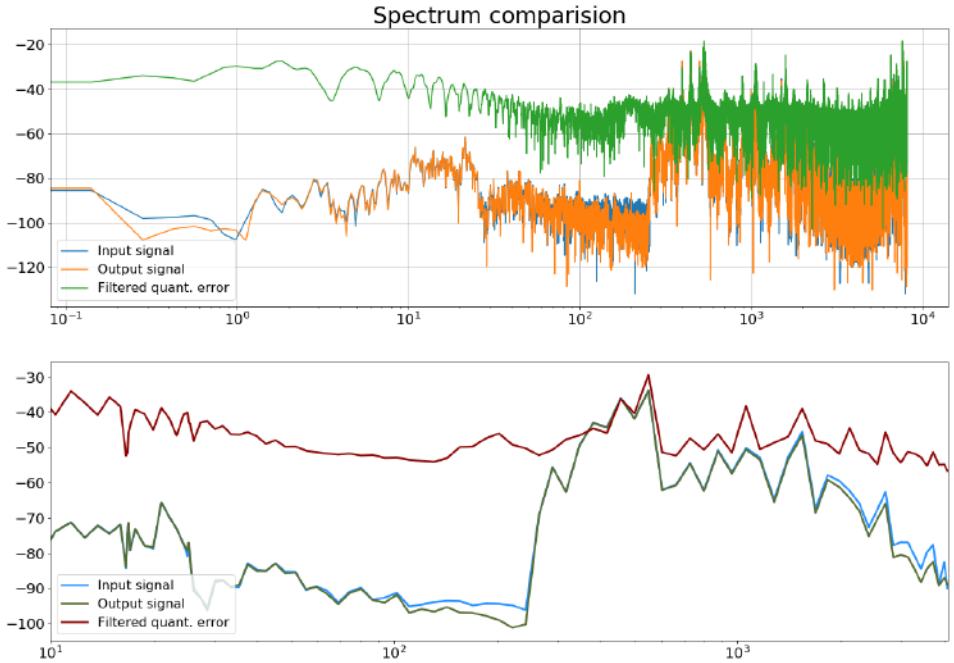
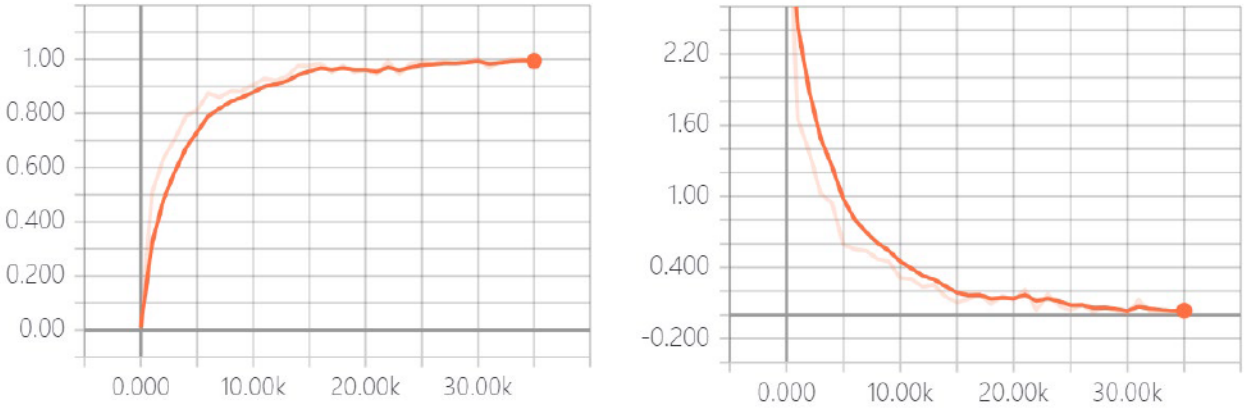


Fig. 4-12 Spectrum of original, simulated signal and filtered noise

The sampling rate of raw signal is 44.1KHz, and the output data rate after sigma delta simulation is only 8962Hz, about 4 times reduction. Also, simulated signals are trained with the RNNs classifier using Urban8K dataset, the accuracy we got is up to 0.8717. We haven't test the FSD2018 dataset, because simulation of sigma delta ADC can be a really slow process. Figure 4-12 shows the training process, as we can see the training loss is close to zeros at the last phase of training process.



(a) Training accuracy curve (b) Training loss curve

Fig. 4-13 Training accuracy and loss curve

4.3 Level Crossing ADC for audio signals

Level crossing ADC is nonuniform sampling way for event trigger signals such as ECGs, here we simulated these uneven signals in audio area and related experiments are implemented.

4.3.1 Principle and simulation

For traditional ADCs, an analog signal uniformly sampled and the digital value for the sample point of analog signal is quantized to the nearest predefined quantization level. As Figure 4-12 shows, the quantization levels belong to a quantization value set containing the value q_a, q_b… which are spaced uniformly on amplitude scales, in another word, the distance between quantization levels are all equal. With the discrete quantization strategy, the quantized amplitude levels of sampled signal usually are not the real values of sampled signal at the sampling instants. And the sampling rate should be enough to support the fast-varying parts of the signal being sampled with the sample frequency up to the twice of its highest frequency. However, the sampling rate is constant in classic ADC, the slow-varying parts and fast-varying parts have the same

sample rate, some redundancy in slow varying parts of signal, especially in sparse signals.

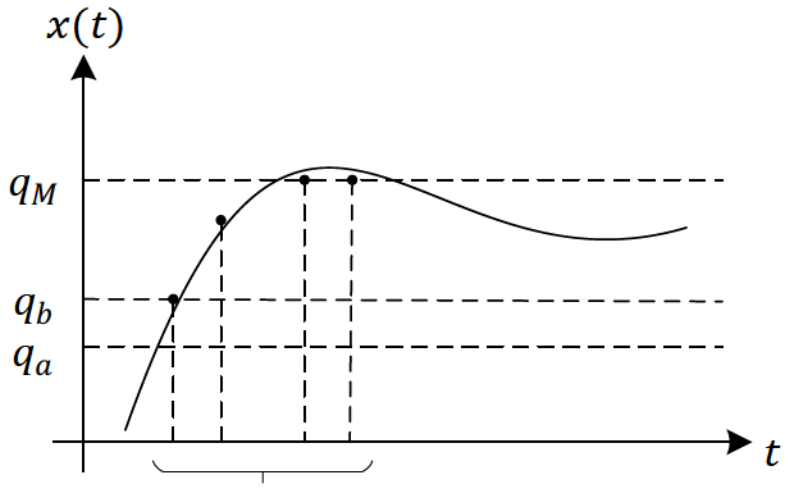


Fig. 4-12 Uniform sampling

Level crossing sampling is shown in Figure 4-13. The sampling for input signal only occurs at the signal points where it crosses the predefined discrete quantization levelsq_a, q_b, ... The corresponding time instants are denoted by t_a, t_b... So, the sampled signal by LC ADC can be represented by pairs of t_k and x_k, where the signal values $x(t_k) = x_k$ is one of the quantization levels. If we assume that the time axis is continuous, there are no quantization error in this signal representation from LC ADC, because the corresponding amplitude value of sampled time instant is exactly the value in original signal. Sampling times are not depended on global time clock, but decided by quantization levels with the waveform of signal. This LC ADC could generate high amplitude resolution signals, since there are no amplitude quantization errors. However, continues time axis only exist in mathematics, global clock in ADC circuit can never generate a continues time clock. Thus, high discrete time resolution is required in LC ADC. Fortunately, techniques which could generate high time resolution clocks is benefit from advanced development of VLSI

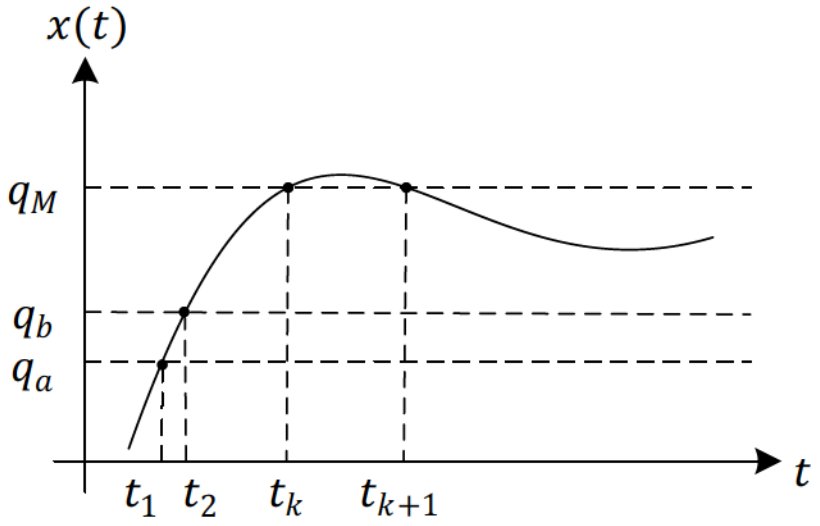


Fig.4-13Level-crossing sampling

According to Nyquist sampling theorem, to avoid the frequency aliasing while sampling a band limited signal, the uniform sampling frequency should be twice larger than the bandwidth of a signal. Thus, if we use uniform sampling with a Nyquist frequency for all the signal waveform including the slow varying parts, redundant data would be generated from slow varying parts. On the contrast, LC ADC gives a smaller sampling rate for the slow varying parts, as a result, unnecessary redundant data are avoided in this way.

According to the theory of level crossing ADC, we design an algorithm simulation method instead of Matlab Simulink. Because the data set we have is sampled in common Nyquist sampling way, which is not convenient to restore the no error values of signal amplitude. We can just only keep the quantized value as close to the original as possible during our reduction process. Keep the main idea of level crossing ADC in mind, it is not difficult to design the algorithm simulation without Matlab Simulink. We just need to give a higher sampling rate for fast-changing parts and lower sampling rate for slow-changing parts. In here, we can keep the fast-changing part with its original sampling rate, and do down sample for slow-changing parts. In this way, we kept the most information while down sampling the audio signals. The simulation signal waveforms are shown in Figure 4-14.

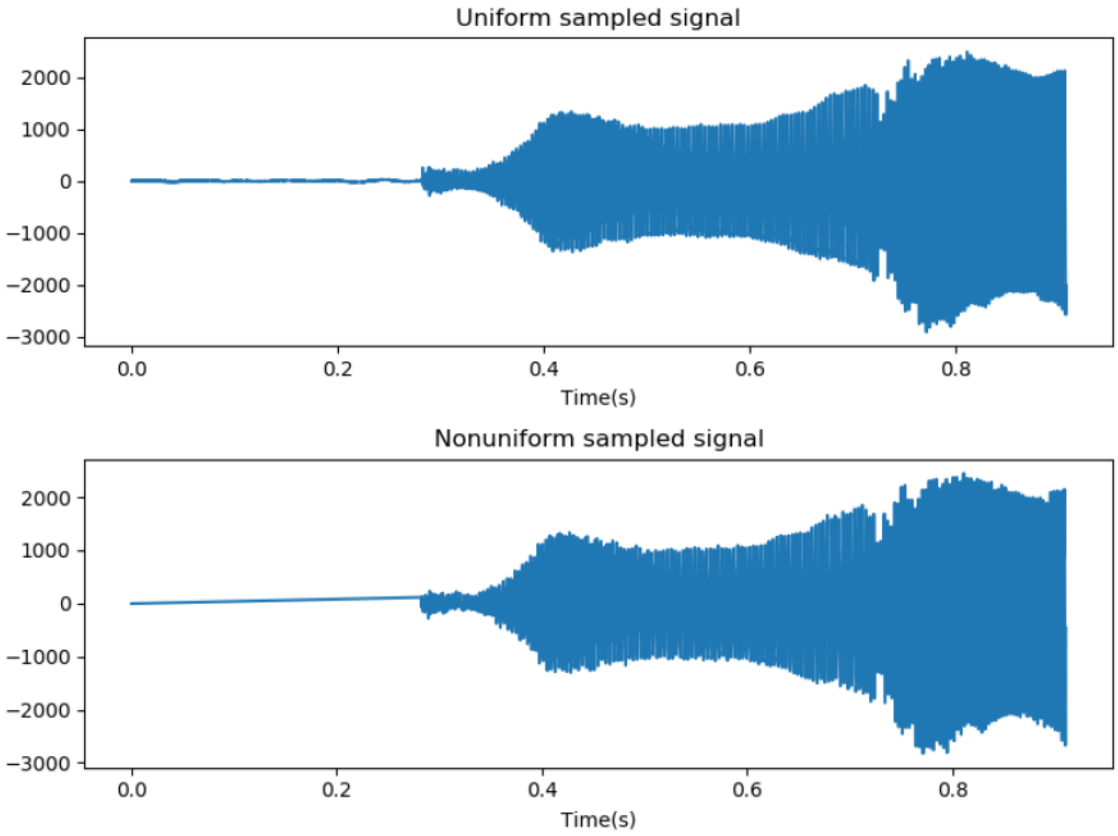


Fig. 4-14 Uniform sampled signal and nonuniform sampled signal

As we can see from Figure 4-14, nonuniform sampled signal has no difference with uniform sampled signal in fast-changing parts, the only difference is remained on slow-changing parts, which just has impact on low frequency component of the spectrum of signal. And the simulated nonuniform signal is almost half the length of original signal, which means we use only the half of the original average sampling rate on this signal, but most information is kept.

However, another problem is introduced at the same time. The signals we got through this LC ADC or our simulation method are nonuniform sampled data. Fast Fourier transform is not suitable anymore because time scale is not uniform, which means time interval between points are not equal. We can see from Figure 4-15, (a) is the original Mel spectrum, (b) is the spectrum obtained from direct STFT one nonuniform signal. As it shows, the spectrum is distorted. Obviously, we cannot use this distorted spectrum as the input of neural network classifier directly.

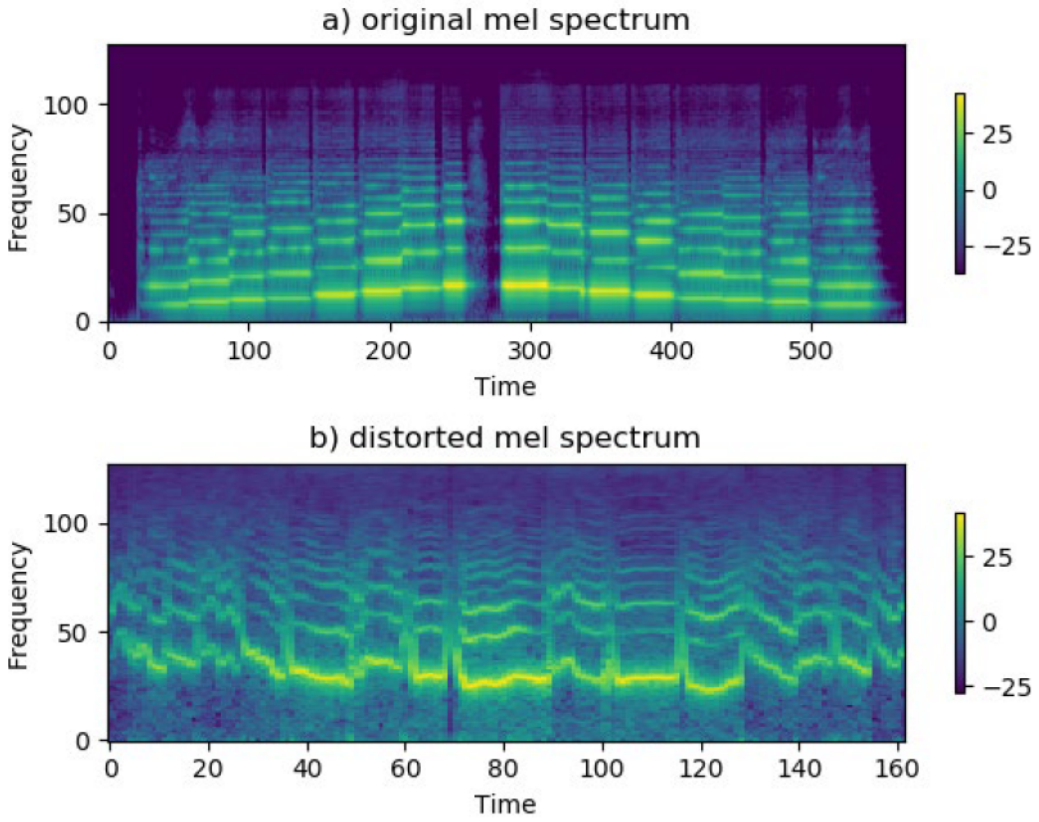


Fig. 4-15 Original and distorted Mel spectrum

For nonuniform sampled signals, nonuniform Fourier transform need to be applied as we would like to exploit the frequency information of nonuniform sampled data. There have been some researches about this, like Lomb-Scargle algorithm [56, 57, 58], which detects the significance of weak periodic signals with uneven temporal sampling. We framed the uneven temporal sampling signals manually based on time window shift, and do periodic analysis for framed nonuniform points using Lomb-Scargle algorithm, then we got power spectral of the framed signal. Like the short time Fourier transform, we stack the framed power spectral, finally, time-frequency power spectrum is obtained. Figure 4-16 shows the Mel spectrum of nonuniform sampled signal using nonuniform discrete Fourier transform mentioned before.

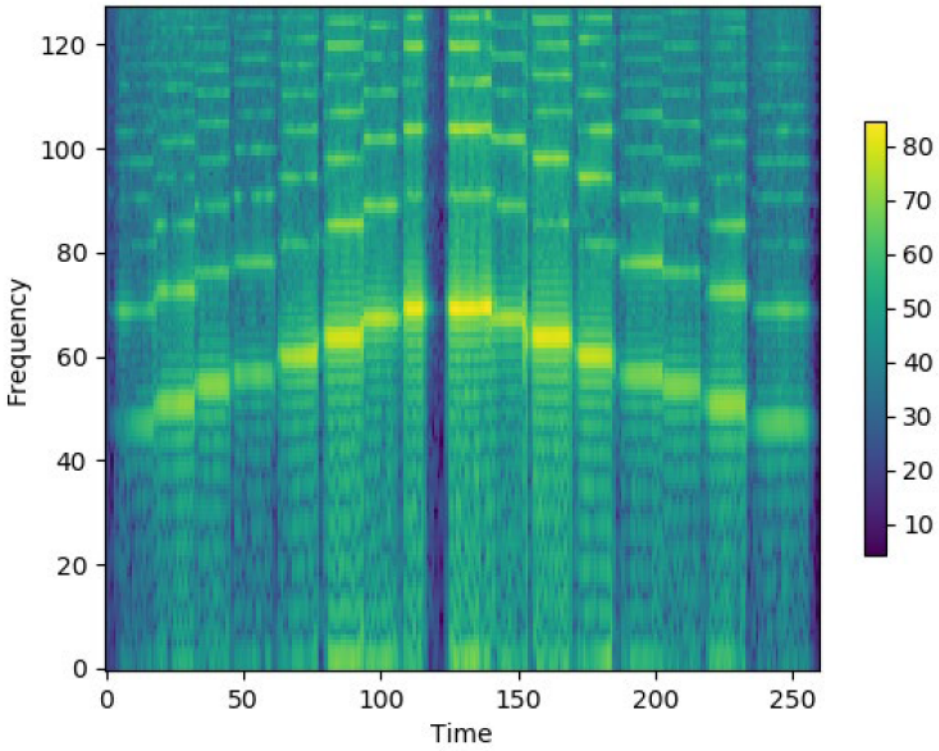


Fig. 4-16 Mel spectrum of nonuniform sampled signal

As we can see, this nonuniform Mel spectrum keeps the most related information of the original Mel spectrum like relative fundamental frequency and principle formants, with some frequency shift and basis of average noise. This linear transform problem could be well solved by neural networks, because neural network could learn a linear transform as long as it thinks this transform could benefit its minimization of loss function.

4.3.2 Analysis of experiment results

The nonuniform Mel spectrum is extracted for every audio clips and feed into neural network as inputs to train a classifier. For this experiment, we chose the CNNs model with FSD2018 audio dataset. As CNNs are better suited with this nonuniform Mel spectrum, it is much harder to extract MFCCs on nonuniform sampled signal. Thus, we give the nonuniform Mel spectrum as the input, expecting CNN blocks would filter and learn the information which it thinks meaningless or useful.

The same network structure of CNN4 in 3.2.2 is chosen because it is much smaller

than CNN8 with little accuracy loss. For hyperparameters of network, the only difference is the filter size, 5×5 filter in 3.2.2 is replaced by 4×4 here, because we get smaller Mel spectrum size after simulation and nonuniform short time Fourier transform. And training parameters is almost the same with 3.2.2, Adam optimizer is chosen, with a learning rate of 0.001 and the learning rate is reduced by multiplying 0.9 after every 1000 iterations training, and batch size is still 128. Table 4-3 shows the results on FSD2018 with our LC-ADC data reduction method.

Table 4-3 MAP results of CNN4 on FSD2018 with LC-ADC

Name	**MAP**	**Name**	**MAP**	**Name**	**MAP**
Acoustic guitar	0.7951	Electric piano	0.4175	Microwave oven	0.8150
Applause	0.9653	Fart	0.7683	Oboe	0.9065
Bark	0.8162	Finger snapping	0.8653	Saxophone	0.7829
Bass drum	0.8621	Fireworks	0.9374	Scissors	0.4308
Burping, eructation	0.7932	Flute	0.8953	Shatter	0.8535
Bus	0.5160	Glockenspiel	0.4378	Snare drum	0.8821
Chime	0.9692	Gong	0.9622	Squeak	0.9957
Clarinet	0.9175	Gunshot, gunfire	0.4953	Tambourine	0.9752
Computer keyboard	0.6207	Harmonica	0.8903	Tearing	0.6543
Cough	0.8732	Hi-hat	0.5573	Telephone	0.7866
Cowbell	0.8510	Keys jangling	0.8214	Trumpet	0.8764
Double bass	0.9153	Knock	0.6092	Violin, fiddle	0.9509
Drawer open, close	0.8014	Laughter	0.9388	Writing	0.8474
Cello	0.8391	Meow	0.9182	Overall	0.8017

We can derive that with LC-ADC, classification performance is well kept while reducing the average sample rate. At the same time, we also found that the level class

is related to the classification performance because it determines which points are skipped which are kept, in another words, the average sample rate.

4.4 Performance analysis

We have done three experiments, compare these three methods for sound event recognition. We have known the reducing of data amount of audio signals, which is benefited for reducing parameters, power consumptions and latency of classification model. We would like to see how much the parameters are reduced and latency are improved by using these methods. Thus, we designed another experiment to figure out the improvements, of course, except for classification accuracy.

4.4.1 Theoretical analysis

Model complexity, as known as parameters of model could be calculated by theoretical analysis, because we have the determined input size.

For the first method, down sample, because sample rate determines the bandwidth (the highest frequency component, $f \leq f_s/2$, f represents frequency of signal, f_s is the sample frequency) down sample reduce the max frequency limit. And frequency scale (normalized frequency resolution) is determined by the number of FFT points, number of frequency scale is equal to number of FFT points divided by 2 plus one (zero component), which gives equation (4.2),

$$n_{ferq\_bin} = \frac{n_{FFT}}{2} + 1 \tag{4.2}$$

where n_{ferq_bin} represent the number of linear frequency bins, n_{FFT} represent the number of FFT points. And FFT points is determined by framed window length, number of FFT points is equal to an integer power of 2 and must greater than the framed window points, as equation (4.3) described,

$$n_{FFT} = 2^{ceil(\log l_{win}/\log 2)} \tag{4.3}$$

where ceil function means rounded up, l_{win} means the points length of window. The window length equals to sample rate multiply window time, $t_{win} = l_{win} \times f_s$, t_{win} means the time of window. So, window time determines the frequency scales. At the same time, the number of Mel bins is usually determinate, often 80, 96 or 128 in speech tasks, so we need to keep the linear frequency bins consistent. If we want to keep the frequency scale consistent, window time need to increase to keep the window

length increase. Meanwhile, hop length of frame shift is usually half or quarter of window length, so hop length and hop time increase with window length, results in less frames of one audio clip after frame process. Less frames gives huge reduction for the audio features which would be feed into neural networks, whether it is Mel spectrogram or MFCCs. If we down sample n times, we get a new sample rate which is n times less than the original one, thus window time is n times larger than original, as well as hop time, eventually we get a nearly n times less frames.

For Mel spectrogram, although the number of Mel frequency bins is not changed, we have larger time shift in time scale, so the shape of Mel spectrum is reduced nearly n times in one dimension, which results in less convolutional computation, as well as shorter latency and less power consumption. Meanwhile, if the size of input is reduced much larger, we could even reduce the filter size of CNNs to extract the local information carefully. For example, we use 5×5 filter for CNN4 in 3.2.2, and the size of filter is reduced to 4×4 because Mel scale is reduced twice in time dimension when average sample rate is twice less than original in 4.3.2. The total parameters reduction of classification model could be calculated, for only the convolutional parameters, according to the structure of CNN4 described in 3.2.2, there are 4 convolutional blocks and the numbers of channels are 64, 128, 256, 512 respectively. So, the original number of parameters is $5 \times 5 \times (64 + 128 \times 64 + 256 \times 128 + 512 \times 256) = 4302400$. And the number of parameters after filter shrinking is $4 \times 4 \times (64 + 128 \times 64 + 256 \times 128 + 512 \times 256) = 2753536$, almost the one third convolutional parameters are reduced, which is 1548864, up to 1.5M parameters, if we use float64 to code one parameter, the saved memory can be 96M, which is a huge number for the memory size of embedded devices. Although the computation times can be increased slightly due to time filter shrinking, it's still a small number compared to the reduction of computation times due to input size reduction. In fully connection layer, the reduce of input size has a huge effect on reducing the weights connect flatten layer with fully connection layer. Assume that the final output of the last convolutional block is 10×4, which is a conservative estimate, the original fully connected weights is $10 \times 4 \times 41 \times 512 = 839680$, where 41 is the number of categories, 512 is the number of channels of last feature map. If we have reduced the input size by twice, thus, the reduction of fully connected weights can be $5 \times 4 \times 41 \times 512 = 419840$, which means another 0.4M parameters are

reduced only in the last fully connection layer, in another word, 25M space are saved in memory.

For MFCCs with LSTMs structure, also, down sample results in less frames of MFCCs. In LSTMs, one frame is a vector feed into LSTM cell for one time step, less frames means less time steps, so, computation times are reduced nearly n times if we down sample n times. As we tend to fix the number of Mel bins to keep the model basically unchanged, the number of LSTM units and layers are settled. Thus, the only reduction is on the last fully connection layer. Similar to CNNs, assume that we have 100 output time steps, now it is reduced to 50 by down sampling, relatively, the parameters of last fully connection layer is from $128 \times 100 \times 41 = 524800$ to $128 \times 50 \times 41 = 262400$, where 128 is the values we set for the hidden layers of LSTM cell in our experiments. Still, 0.2M parameters, 12.8M memory space are saved just for model size.

For sigma delta ADC method, the output data rate is almost 4 times less than original sample rate in our experiment (8962 VS. 44100), so MFCCs and Mel spectrum frames are both 4 times less, and the model size reduction is similar to the analysis above, just modifying the scaling factor of number of frames we can have the reduction size of model. The benefit of sigma delta ADC is that signal precision is kept while data rate is reduced, due to its noise shaping function when down sample from original signal. The only shortcoming of sigma delta ADC is that it could cause extra energy consumption and latency while signal acquisition, according to the principle of sigma delta ADC in 4.2.1, the process in sigma delta modulator could cause more energy consumption than traditional ADC due to its up-sample process. And in our simulation experiment, simulation for one audio clip could consume more than 30 seconds, which is a big challenge for our experiment. However, compared to the reduction of data amount and size of model while keeping the accuracy, this ADC energy consumption is worth to consume.

For LC ADC method, average sample rate for each audio clip is unstable, due to each audio clip has their own frequency property. What's more, even we got the average sample rate, it has no direct relation with final frames due to our framing method and nonuniform DFT algorithm. In another word, even the same average sample rate may give a different number of frames because of our nonuniform sample method. Thus, we need to count for all the signal to get the relationship of statistical

average sample rate and nonuniform Mel bins frames. Once we have got the statistical relationship between average sample rate and number of frames, it is easy to get to model size reduction according to above theoretical analysis. For specific case, like just one audio clip, we can analyze the average sample rate and number of frames to discuss the model complexity reduction in this case, it does have some representative for other case, but it's not in universal. In the case of 4.3.1, the picked one in Figure 30, the final data length is one third of the original, so we can assume that the average sample rate is also one third of the original sample rate. As for frames, according to experiment results, the original has 567 frames while the simulated has only 260 frames in Mel spectrum. So, it's more than twice the reduction, but not up to three times. Thus, the reduction of model complexity is twice the number of weights in fully connection layers and convolutional size shrinking factor if it shrinks.

Finally, for reduce the sample width, as sample rate is stay unchanged, the number of frames remains the same. The only difference is the space reduction of saved memory for signal number, because the parameters precision in our model is determined in model definition rather than ADC precision. Thus, in this experiment, the influence of bit length variation of audio signal can only reduce the working memory for input data access. And due to the unideal experiment result for reduce sample width (causes too much accuracy loss), we can derive that this is not a proper way to reduce data amount in audio detection task.

We have discussed the parameters of model complexity in this section, because it is possible to calculate in theoretical, we just need to figure out the relationship between the number of points in one signal and the model parameters. As for other dominant performance like latency and power consumption, it is hard to calculate in theoretical. Experiments and analysis for latency reduction is talk in next section.

4.4.2 Experiment analysis

As we discussed in previous section, it is hard to calculate latency of model inference and power consumption during inference in theoretical, we implement an experiment in this section to figure out the latency reduction, as power consumption is still not easy to implement in computer instead of real embedded devices with the exist data and resources.

Experiments for latency is relatively easy to implement because we just need to count

the average latency for different data reduction strategy with the same original signal in test data set. That is to say, we use the same audio test set to test the model related to different reduction strategy. For each strategy, we simulated for test set audio clips in the way of this strategy, then the simulated signals are feed into relatively trained classifier to get the inference latency. Table 4-4 shows the latency with related reduction strategies.

Table 4-4 latency vs. data reduction method

Reduction method	Model	Dataset	Sample rate	Average Frames	Total latency
No method	Bi LSTMs	Urban 8K	44100 Hz	81	74ms
Down sample	Bi LSTMs	Urban 8K	22050 Hz	40	51ms
ΔΣ ADC	Bi LSTMs	Urban 8K	8192 Hz	16	28ms
No method	CNN4	FSD2018	44100 Hz	81	43ms
Down sample	CNN4	FSD2018	22050 Hz	40	32ms
LC ADC	CNN4	FSD2018	7775 Hz	36	30ms
No method	CNN8	FSD2018	44100 Hz	81	68ms
Down sample	CNN8	FSD2018	22050 Hz	40	53ms
LC ADC	CNN8	FSD2018	7775 Hz	36	49ms

In table 4-4, the value of average frames is average frames per second signal for all audio signal, and average latency is also the average latency per audio file for all audio signal in test set, because audio signal in test set have much different duration, and the test audio files are the same in one data set, for statistic convenience, we use average latency per file as the metrices instead of average latency per file. Another point we have to state is that for each dataset, we use different test files, the reason is obvious, test set is better to be similar to the training set. So, comparison should be between the same dataset and the same model, only in this way we can see the effect of our method. And the number of frames and values of latency is only for reference, we use the statistic values to represent the actual different values here, because our purpose is to see the effect of our data reduction method.

From table 4-4, we can derive that average latency is closely related to average

frames, all the three method has the ability to reduce the average frames. It is worth to mention that when we compare the latency, we need to compare during the same model for different strategy, because the model structure is in dominance for inference latency. For LSTMs with Urban 8K, the reduction of frames reduced the latency hugely because frames determined the time step of LSTM, that is obvious to understand, one more frame means one more extra computation in LSTM cell. Both down sample and sigma delta ADC are for the same reason, so huge frames reduction could cause huge latency reduction. For CNNs with FSD2018, down sample and LC ADC can also reduce the latency in some extent, but the reduction degree is not as much as it in LSTMs. For this phenomenon, we think that's because of the structure of CNNs, although the input size is smaller using the data reduction method, computation is still in parallel due to the property of CNNs. Another important issue is that we implement this inference experiment on server machine with graphic process unit (GPU) acceleration, which is much suitable for matrix operations. So, the experiment environment is much different with realistic practical scene, the specific value only has reference value, we can see how much the latency are reduced with our data reduction method.

Chapter 5. Conclusions

For the typical IoT application in audio area, sound event detection, we proposed a new perspective on reducing the energy consumption and data amount for embedded devices. We had implemented several experiments for this task using our data reduction method.

At first, we used two model with two different audio features as the input, in two different datasets. The two models are LSTMs with MFCCs and Mel spectrogram with CNNs respectively. It turns out to be well for experiments, although there is still some distance from the best performance. However, state of art works in this task have huge model complexity, which is not suitable for embedded devices. So, the two simple but well performance models are chosen as our basic models for the next experiments. The first data reduction method is down sample and reduce the sample width directly, result of experiment shows that down sample could reduce model complexity and latency while keep the classification accuracy. For LSTMs, latency is reduced hugely with some complexity reduction. For CNNs, much more parameters are reduced when we feed less frames of Mel spectrogram for model and shrink the filter size. But the result also shows that reduce the sample width seems have huge impact on model performance, accuracy is losing sharply while reducing quantization precision. And for reduction of model complexity, it seems that there is no difference while using less sample width in this simulation experiment on computer.

Based on the results of our first experiment, we replaced the traditional ADC method with a new ADC technique called sigma delta ADC, which provides high resolution and low data rate signal. We simulated this kind of signals from the original dataset, of course, the data rate of simulated signal must be lower than the original sample rate, then we used those simulated signals to retrained a classification model. In the condition of the same data rate, compared to down sample directly, we found that signals from this sigma delta ADC way could better maintain the model accuracy, that is, more audio feature information. In terms of model complexity and latency reduction, it is similar to down sample directly, because the principle is to reduce the input size of model, which is the number of audio feature frames.

We further considered a nonuniform sample method, LC ADC, widely used in ECG signal acquisition because of its event triggered property. We would like to give more sample rate for signal segments where carried more information and vice versa to

reduce the data amount. Because the simulated signals are uneven in time scale, we used a nonuniform discrete Fourier transform method called Lomb-Scargle algorithm to approximate the short time frame energy. And Framing is still based on time window shift, but the signal points in one frame is not determinate anymore. Experiment result shows that, compared to down sample directly, signals from this LC ADC way has a better ability to keep the feature information, as well as the model precision. This comparation should be implemented with the same number of audio frames instead of average sample rate, because relations between number of frames and average sample rate is not a strictly linear relation. As for model complexity and latency, it is still similar to down sample directly with the same number of frames.

The above are all the conclusions of all experiments in this thesis. We found those data reduction method we tried could reduce the redundant data amount efficiently, however, there are still a lot of work to do to confirm the feasibility of this new perspective. We can implement experiments on real hardware side to test the result, after all, simulation environment is such different from realistic engineering. For LC ADC, we can do much more experiments to find the statistic relation between level classes, average sample rate and number of frames. At last, the task of sound event detection could be more realistic, we assume all the events are monophonic, it is easier to classify than polyphonic events (several events happening simultaneously) in experiment. However, most sound events are polyphonic in real word. Thus, to make this research have practical values, we need to research this polyphonic sound event detection topic further, where much more acoustic area algorithm involved. In a word, there are still a lot of work to do to make this research more valuable.

References

[1] D Stowell, D Giannoulis, E Benetos, et al. Detection and Classification of Acoustic Scenes and Events[J]. IEEE Transactions on Multimedia, 2015, 17(10):1-1.

[2] E. Fonseca, M. Plakal, F. Font, et al. General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline[C]// Detection and Classification of Acoustic Scenes and Events, 2018.

[3] D. Li, I. K. Sethi, N. Dimitrova, et al. Classification of general audio data for content-based retrieval[J]. Pattern Recognition Letters, 2001, 22(5):533–544.

[4] V. Bisot, R. Serizel, S. Essid, et al. Supervised nonnegative matrix factorization for acoustic scene classification[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2016.

[5] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, et al. A hybrid approach using binaural i-vectors and deep convolutional neural networks[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2016.

[6] K. Choi, G. Fazekas, and M. Sandler. Automatic tagging using deep convolutional neural networks[C]// International Society of Music Information Retrieval Conference, 2016.

[7] D. Stowell, D. Giannoulis, E. Benetos, et al. Detection and classification of acoustic scenes and events[J]. IEEE Transactions on Multimedia, 2015, 17(10):1733–1746.

[8] A. Mesaros, T. Heittola, A. Eronen, et al. Acoustic event detection in real life recordings[C]// 18th European Signal Processing Conference, 2014.

[9] Q. Kong, I. Sobieraj, W. Wang, et al. Deep neural network baseline for dcase challenge 2016[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2016.

[10] J. Li, W. Dai, F. Metze, S, et al. A comparison of deep learning methods for environmental sound[C]// IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

[11] S. Hershey, S. Chaudhuri, D. P. Ellis, et al. CNN architectures for large-scale audio classification[C]// IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

[12] Y. Aytar, C. Vondrick, A. Torralba. Soundnet: Learning sound representations from unlabeled video[C]// Advances in Neural Information Processing Systems (NIPS),

2016.

[13]E. Cakir, G. Parascandolo, T. Heittola, et al. Convolutional recurrent neural networks for polyphonic sound event detection[J]. IEEE ACM Transactions on Audio, Speech and Language Processing (TASLP), 2017, 25(6):1291–1303.

[14]H. Lim, J. Park, K. Lee, et al. Rare sound event detection using 1D convolutional recurrent neural networks[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2017.

[15]Y. Xu, Q. Kong, W. Wang, et al. Large-scale weakly supervised audio classification using gated convolutional neural network[C]// IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.

[16]A. Kumar, B. Raj. Audio event detection using weakly labeled data[C]// Proceedings of the 2016 ACM on Multimedia Conference, 2016.

[17]S. Mun, S. Park, D. K. Han, et al. Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2017.

[18]T. Nguyen, N. Khanh, Douglas L. Jones, et al. Iterative training, label smoothing, and background noise normalization for audio event tagging[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2018.

[19]H. Xueyu, L. Di, L. Qing, et al. Ciaic-gatfc system for dcase2018 challenge task2[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2018.

[20]J. Il-Young, L. Hyungui. Audio tagging system for dcase 2018: focusing on label noise, data augmentation and its efficient learning[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2018.

[21]D. Matthias, W. Gerhard. Training general-purpose audio tagging networks with noisy labels and iterative self-verification[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2018.

[22]K. Alex, S. Ilya, and E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in neural information processing systems, 2012, 25(2):1106–1114.

[23]S. Karen, Z. Andrew. Very deep convolutional networks for large-scale image

recognition[C]// International Conference on Learning Representations (ICLR), 2015.

[24] S. Christian, L. Wei, J. Yangqing, et al. Going deeper with convolutions[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[25] H. Kaiming, Z. Xiangyu, R. Shaoqing, et al. Deep residual learning for image recognition[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[26] B. James, B. Yoshua. Random search for hyper-parameter optimization[J]. Journal of Machine Learning Research, 2012, 13(1):281–305.

[27] S. Jasper, L. Hugo, P. Adams. Practical bayesian optimization of machine learning algorithms[J]. Advances in neural information processing systems, 2012, 4:2960–2968.

[28] S. Jasper, R. Oren, S. Kevin, et al. Scalable bayesian optimization using deep neural networks[C]// International Conference on Machine Learning (ICML), 2015.

[29] H. Babak, D. Stork. Second order derivatives for network pruning: Optimal brain surgeon[J]. Advances in Neural Information Processing Systems, 1992, 5:164–171.

[30] Y. LeCun, J. Denker, S. Solla. Optimal brain damage[J]. Advances in Neural Information Processing Systems, 1989, 2:598– 605.

[31] H. Song, P. Jeff, T. John, et al. Learning both weights and connections for efficient neural network[C]// Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, 2015.

[32] H. Song, P. Jeff, N. Shara, et al. DSD: regularizing deep neural networks with dense-sparse-dense training flow[C]// International Conference on Learning Representations (ICLR), 2017.

[33] G. Yiwen, Y. Anbang, C. Yurong. Dynamic network surgery for efficient dnns[C]// Advances in Neural Information Processing Systems (NIPS), 2016.

[34] L. Hao, K. Asim, D. Igor, et al. Pruning filters for efficient convnets[C]// International Conference on Learning Representations (ICLR), 2017.

[35] A. Karim, T. Lorenzo. Connectivity learning in multi-branch networks[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[36] V. Tom, D. Ludovic. Learning timeefficient deep architectures with budgeted super networks[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[37]Z. Xiangyu, Z. Xinyu, L. Mengxiao, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[38]C. Soravit, S. Mark, Z. Andrey. The power of sparsity in convolutional neural networks[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[39]W. Min, L. Baoyuan, F. Hassan. Design of efficient convolutional layers using single intra-channel convolution, topological subdivisioning and spatial "bottleneck" structure[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[40]E. Allier, G. Sicard, L. Fesquet, et al. A new class of asynchronous A/D converters based on time quantization[C]// Asynchronous Circuits and Systems, Proceedings. Ninth International Symposium on. IEEE, 2003.

[41]N. Sayiner, H. Sorensen, T. Viswanathan. A level-crossing sampling scheme for A/D conversion[J]. IEEE Transactions on Circuits & Systems II Analog & Digital Signal Processing, 1996, 43(4):339.

[42]P. Jungwirth, A. Poularikas. Improved Sayiner level crossing ADC[C]// the 36th Southeastern Symposium on System Theory, 2004.

[43]S. Kozat, K. Guan, A. Singer. Tracking the best level set in a level-crossing analog-to-digital converter[J]. Digital Signal Processing, 2013, 23(1):478-487.

[44]K. Kozmin, J. Johansson, J. Delsing. Level-crossing ADC performance evaluation toward ultrasound application[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2009, 56(8):1708-1719.

[45]Y. Li, D. Zhao, W. Serdijn. A sub-microwatt asynchronous level-crossing ADC for biomedical applications[J]. IEEE Transactions on Biomedical Circuits and Systems, 2013, 7(2):149-157.

[46]A. Mansano, Y. Li, S. Bagga, et al. An autonomous wireless sensor node with asynchronous ECG monitoring in 0.18 μm CMOS[J]. IEEE Transactions on Biomedical Circuits and Systems, 2016, 10(3):602-611.

[47]N. Ravanshad, H. Rezaee-Dehsorkh, R. Lotfi, et al. A level-crossing based QRS detection algorithm for wearable ECG sensors[J]. IEEE Journal of Biomedical and Health Informatics, 2014, 18(1):183–192.

[48]J. Custódio, J. Goes, N. Paulino. A 1.2-V 165uW 0.29-mm2 Multibit Sigma-Delta

ADC for Hearing Aids Using Nonlinear DACs and With Over 91 dB Dynamic-Range[J]. IEEE transactions on biomedical circuits and systems, 2013, 7(3):376-385.

[49]S. Zeller, C. Muenker, R. Weigel. A 0.039 mm2 inverter-based 1.82 mW 68.6 dB-SNDR 10MHz-BW CT-ΣΔ-ADC in 65nm CMOS[C]// Proceedings of the ESSCIRC (ESSCIRC), 2013.

[50]A. Donida, R. Cellier, A. Nagari. A 40-nm CMOS, 1.1-V, 101-dB Dynamic-Range, 1.7-mW Continuous-Time ΣΔ ADC for a Digital Closed-Loop Class-D Amplifier[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2015, 62(3): 645-653.

[51]S. Yan, E. Sánchez. A continuous-time sigma-delta modulator with 88-dB dynamic range and 1.1-MHz signal bandwidth[J]. IEEE Journal of Solid-State Circuits, 2004, 39(1):75-86.

[52]J. Salamon, C. Jacoby, J. P. Bello. A Dataset and Taxonomy for Urban Sound Research[C]// the 22nd ACM International Conference on Multimedia, 2014.

[53]F. Eduardo, P. Manoj, F. Frederic, et al. General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset and Baseline[C]// IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2018.

[54]J. Gemmeke, D. Ellis, D. Freedman, et al. Audio Set: An ontology and human-labeled dataset for audio events[C]// IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

[55]FreeSound[EB/OL].[2019.9.25].https://freesound.org

[56]N. Lomb. Least-squares frequency analysis of unequally spaced data[J]. Astrophysics and Space Science, 1976, 39(2):447-462.

[57]J. Scargle. Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data[J]. The Astrophysical Journal, 1982, 263:835-853.

[58]R. Townsend. Fast calculation of the Lomb-Scargle periodogram using graphics processing units[J]. The Astrophysical Journal Supplement Series, 2010, 191:247-253.

[59]S. Calo, M. Touna, D. Verma, et al. Edge computing architecture for applying AI to IoT[C]// IEEE International Conference on Big Data, 2017.

[60]N. Jouppi, A. Borchers, R. Boyle, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit[C]// the 44th International Symposium on Computer Architecture (ISCA), 2017.

Acknowledgment

I would like to thank my advisors Zou Zhuo and Zheng Lirong for the close and skillful guidance and support during my graduate career and the experimental research of this thesis. Their research experiences and academic passion have a huge impact on my study career. Also, I'm very grateful for my UTU supervisor Tomi Westerlund and Tuan in University of Turku, their the enthusiastic academic and life assistance help me entered a pleasant life and study journey in Finland. And I would also like to sincerely thank my colleagues who study with me these years, your excellence and enthusiasm have greatly inspired me while finishing this thesis. In addition, the supports of my roommates and classmate help me go through the tough time when I got into troubles, your encouragement give me strength to get on and keep moving. At last, I would like to sincerely thank my family. Thank you for your unconditional support for all my decision whether they are correct or not. And my great country warmed my heart during my downcast period, your greatness inspired me go ahead toward the direction of lights.