Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2020-4786610

M.Sc. Thesis

Deep Learning Based Sound Identification

Shaoqing Chen

Abstract

Environmental sound identification and recognition aim to detect sound events within an audio clip. This technology is useful in many real-world applications such as security systems, smart vehicle navigation and surveillance of noise pollution, etc. Research on this topic has received increased attention in recent years. Performance is increasing rapidly as a result of deep learning methods. In this project, our goal is to realize urban sound classification using several neural network models. We select log-Mel spectrogram as the audio representation and use two types of neural networks to perform the classification task. The first is the convolutional neural network (CNN), which is the most straightforward and widely used method for a classification problem. The second type of network is autoencoder based models. This type of model includes the variational autoencoder (VAE), β-VAE and bounded information rate variational autoencoder (BIR-VAE). The encoders of these systems extract a low dimensionality representation. The classification is then performed on this so-called latent representation. Our experiments assess the performances of different models by evaluation metrics. The results show that CNN is the most promising classifier in our case, autoencoder-based models can successfully reconstruct the log-Mel spectrogram and the latent features learned by encoders are meaningful as classification can be achieved.

Deep Learning Based Sound Identification

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

SIGNALS AND SYSTEMS

by

Shaoqing Chen
born in Jinan, China

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF
MICROELECTRONICS & COMPUTER ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Deep Learning Based Sound Identification"** by **Shaoqing Chen** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 30-October-2020

Chairman: _____
prof.dr.W.B. Kleijn

Advisor: _____

Committee Members: _____
dr.ir. R. Heusdens

_____
dr. F. Fioranelli

Abstract

Environmental sound identification and recognition aim to detect sound events within an audio clip. This technology is useful in many real-world applications such as security systems, smart vehicle navigation and surveillance of noise pollution, etc. Research on this topic has received increased attention in recent years. Performance is increasing rapidly as a result of deep learning methods. In this project, our goal is to realize urban sound classification using several neural network models. We select log-Mel spectrogram as the audio representation and use two types of neural networks to perform the classification task. The first is the convolutional neural network (CNN), which is the most straightforward and widely used method for a classification problem. The second type of network is autoencoder based models. This type of model includes the variational autoencoder (VAE), β-VAE and bounded information rate variational autoencoder (BIR-VAE). The encoders of these systems extract a low dimensionality representation. The classification is then performed on this so-called latent representation. Our experiments assess the performances of different models by evaluation metrics. The results show that CNN is the most promising classifier in our case, autoencoder-based models can successfully reconstruct the log-Mel spectrogram and the latent features learned by encoders are meaningful as classification can be achieved.

Acknowledgments

I hereby express my sincere gratitude to all the people who have supported me and accompanied me during my study in the master program. I would like to first thank professor Bastiaan Kleijn, who is my supervisor and guided me through my thesis. In our weekly meeting, he gave me insightful feedback and constructive suggestions, leading me to improve my methods and obtain new findings. Also, I'm grateful to the Ph.D. student Wangyang Yu, who has helped me with various practical problems in programming and shared her experience in research. I would also like to express my deep gratefulness to George Boersma and Rinus Boone and the company Munisense, who offered me the chance to do this project. Last but not the least, I would like to thank my family for constant support and unconditional love.

Shaoqing Chen
Delft, The Netherlands
30-October-2020

Contents

List of Figures

List of Tables

1 Introduction

In this chapter, we will give an introduction to our work, which is urban sound identification using deep learning method. We first point out the motivation and usefulness of our project. Then, we conduct literature research on the related work, discussing the existing methods and their results. Afterward, the objective of our thesis is given. The last section lists the organization of the thesis.

1.1 Motivations

Sound identification and recognition is a technology that is based on audio signal processing methods, combined with traditional machine learning algorithms or deep neural networks. The related topics in sound identification tasks include the data signal analysis techniques, feature extraction approaches and classification algorithms. The goal of this technology is to have knowledge of the sound events in the environment and detect certain types of sound that are interesting. It is beneficial in many different applications, for example, in health care, the sound features of human organs can be a helpful assistance to monitor body conditions [1–3]; in security and surveillance systems, the sound classification system gives information on the events taking place in the surroundings [4, 5], such that decisions and actions can be made; speech recognition is changing the way people interact with electronic devices, allowing human to talk to a device that interprets the requests and responds to commands [6]; for artificial intelligence system like smart vehicle, recognizing urban environments is crucial for navigation safety [7].

Modern methods to perform classification commonly are based on large scale neural network models that use deep learning algorithms [8]. In 1997, Tom M. Mitchell, computer scientist and professor at Carnegie Mellon University (CMU), gave the following definition of machine learning: ”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.” The machine learning or deep learning algorithms perform learning tasks, such as classification in our case, using a training data set, gaining experience and improvement during the process of training. After the training process, the model has sufficiently learned as indicated by the performance measure, then it can be used to execute a similar task on new data. Based on this idea, deep learning methods are becoming more and more popular as the available data set is growing larger and larger, together with the development of computational power such as graphics processing unit (GPU) and cloud computing. With a huge dataset and powerful computational support, deep neural network models can be efficiently trained and computed, leading to state-of-the-art performance.

In this project, we consider the application of deep learning-based methods to the classification and identification of sound in a municipality. The goal of this project is to achieve the classification of urban sounds using different neural network models. The practical application and usefulness of this work are that it can be used by Munisense, a company that specializes in sensor networks, especially those for environmental sound and urban noise measurements, to provide information that can be used to set policies such as speed limits, call-out of emergency services or law enforcement.

1.2 Related work

Motivated by various artificial intelligence applications related to urban information processing, the topic of urban sound classification and identification becomes a research field that has received more and more attention in recent years [9]. There are numerous works on this topic, where the implementations of different methods were conducted, and their results are compared. Algorithms with their advantages and disadvantages together with the performances are discussed in the literature. In this section, we will provide a brief literature review on the related work of sound identification.

The task of urban sound identification and classification is challenging as this type of audio sound is less structured and full of interfering noise. To ensure the effectiveness of the classifier model, it's important to find discriminating and informative audio representation as feature and apply the classification task with robust algorithm and model. Therefore, researchers focused on finding powerful machine learning and deep learning methods on one hand and, on the other finding high dimensional and distinguishing audio features. To facilitate the classification performance, the choice of audio features input to the classifiers is crucial. The commonly used audio representations are raw waveform, spectrogram, Mel spectrogram and Mel-frequency cepstral coefficients (MFCCs). Compared to audio waveform, the time-frequency representations retain more information and lower in dimension. Although a large and complex model can learn features directly from the raw waveform, huge computational expense is expected [10]. A comparison among different time-frequency representations was made in [11], where the convolutional neural network (CNN) was used as classifier. The results showed that Mel spectrograms outperformed the spectrograms and MFCC and gave good performances across different datasets. The MFCC applies the discrete cosine transform (DCT) to the Mel spectrograms, this operation decorrelates the spectral energies and loses the local pattern in time-frequency representation [12]. Mel spectrogram contains more distinguishable details than spectrogram because of the Mel filter banks.

In recent years, many works have proven that deep neural network-based models are more promising than traditional classifiers in solving complex classification problems. Support vector machines (SVM), Gaussian mixture models (GMM) and k-means clustering are widely used conventional machine learning algorithms for classification tasks. However, in sound classification applications, these classifiers are vulnerable to the presence of noises and sensitive to the temporal dynamics of the audio, resulting in a lack of robustness [13–16]. With the popularity of deep learning-based models, an increasing number of investigations

have exploited such methods in urban sound identification tasks. The most popular and straightforward deep learning model for classification tasks is the CNN, which is commonly used in computer vision and image classification applications. It is also a promising model for our task as a sound can be interpreted as a 2-D time-frequency representation, where localized spectrum patterns can be learned.

In environmental sound classification research, [17] was the very first work to evaluate the performance of urban sound classification task using CNN. Its model consists of two convolutional layers with max-pooling and followed by two fully connected layers. Log-Mel spectrogram and its delta information were used as audio representation feature to be the input for CNN. The experiment was based on three publicly available datasets, ESC-50, ESC-10 [18], and UrbanSound8K [19]. For each dataset, accuracies of 80.5%, 64.9% and 72.7% were obtained respectively. [20] proposed a CNN architecture with eight convolutional layers, every two convolutional layers were followed by a max-pooling layer, the performances of this proposed CNN were compared with VGG [21]. The results showed that the proposed CNN performs better than VGG, evaluated on three datasets ESC-50, ESC-10 and UrbanSound8K using spectrogram as the input, the accuracies of the proposed CNN were 76.8%, 88.7% and 74.7% respectively. Large and deep CNN models used in the image classification are also applied to sound identification, good performance can be achieved as well. In [22], researchers applied AlexNet [23] and GoogLeNet [24] to the spectrograms of audio, and evaluated on the datasets ESC-50, ESC-10 and UrbanSound8K. The best accuracies were given by GoogLeNet, which were 73%, 91%, and 93% respectively on each dataset. For the same setups, GoogLeNet achieved higher classification accuracy than AlexNet, the reason for this is that GoogLeNet is considerably deeper and has many more layers than AlexNet.

Some works directly used raw waveforms in time-domain as input to the classification model. [10] first used CNN to classify raw waveforms of environmental sounds. The model consists of 34 layers, where convolutional operations were 1-D convolution. The result on UrbanSound8K was 71.8%, which was comparable to [17] using log-Mel spectrogram inputs and 2-D convolution. However, the neural network model was much larger compared to two convolutional layers in [17]. [25] divided the waveform into overlapped frames by sliding window and used 1-D CNN that directly learned features from waveforms. The model achieved 89% of accuracy on UrbanSound8K, competitive to other results from methods using spectrogram representations and 2-D CNN. However, the input was a long sequence of vector and the computational cost was huge.

1.3 Objectives

In this project, to achieve the classification of urban sounds, we will explore several deep learning methods, including the CNN and autoencoder-based models. We define our models using different parameters and architectures, and investigate how the performances can be influenced. Based on the evaluation metrics, comparison and analysis of the results will be discussed.

Most of the urban sound classification and identification tasks are based on CNN as classifier, with spectrograms as input. However, autoencoder-based models such as the variational autoencoder (VAE) are rarely used in the tasks of audio classification based on spectrograms. VAE [26] is a model that jointly trains an inference network and a generative network. During the training process, the inference network learns the probability distribution of the input data and outputs a vector whose data is sampled from this distribution, those samples are called latent variables. Then the generative network reconstructs the input based on latent variables provided by the inference network. After the training, the first network has learned the distribution of the data, which can be the feature extractor for a classification task. In our work, we will first build a CNN as our baseline model to achieve the urban sound classification. Then, autoencoder-based models with different architecture, hyper-parameters and optimization functions will be built and perform feature learning and classification on spectorgrams. To investigate the performance of neural network models, we will use different evaluation metrics.

1.4 Outline

The rest of the thesis is organized as follows.

- Chapter 2 introduces the dataset that we will use in this project. The relevant feature representations of audio are studied, their definitions and derivations are given, advantages and disadvantages are discussed. Also, the signal processing pipeline is listed.
- Chapter 3 gives a thorough study of the neural network models that are involved in this project. We describe each model's architectures and derive their loss functions.
- Chapter 4 defines the experimental setups, including signal processing to obtain audio representations and parameters of neural network models' architecture. Also, evaluation metrics to assess model performances are introduced.
- Chapter 5 presents the results of the experiments and gives discussions on the performances of the different models.
- Chapter 6 ends the thesis by giving the conclusion and future work.

2 Data and Audio Feature

In this chapter, we will first introduce the dataset that will be used in this project, which is the Urbansound 8k. Then different representations of sound signals that can be used as input to deep neural networks are explained and motivated, at the same time, visualizations are provided. In the end, we give the procedures and operations of the audio signal processing.

2.1 Audio Datasets

For the research of detection and classification of acoustic scenes and events (DCASE), there are numerous open-source datasets available. The most popular and the most widely used datasets for this research topic are ESC-50, ESC-10 [18] and UrbanSound8K [19, 27], all of those are taken from [28], which is a website that collects field recordings uploaded by different contributors.

In this work, as our interest is the urban sound environment, we will choose UrbanSound8K dataset to be the main focus as our work. [27] has proposed and motivated this dataset and produced a detailed and taxonomy of urban sounds, the dataset can be found and downloaded from [19]. There are 8732 sound clips in this dataset, each of the clip is no longer than 4 seconds and separated in 10 folders. The number of audio clips in each folder is listed in table 2.1.

Fold Number	Number of Samples
fold 1	873
fold 2	888
fold 3	925
fold 4	990
fold 5	936
fold 6	823
fold 7	838
fold 8	806
fold 9	816
fold 10	837

Table 2.1: Number of audio samples in each fold.

An important property of this dataset is that the audio clips are labeled, which allows us to perform supervised learning. Moreover, the types of audio class included in this dataset are relevant to the sound that we are interested in, which are the common urban noises. The 10 classes of labeled urban sounds are: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music, each of them has a

unique numeric class identity (ID) from 0 to 9. The class names and the corresponding numeric class IDs, together with the number of samples for each class are listed in table 2.2. These 10 classes of sounds were evenly pre-sorted into the 10 folds, which allows us to perform cross-validation or choose the training set and testing set base on the fold number.

Class Name	Numeric Class ID	Number of Samples
Air conditioner	0	1000
Car horn	1	429
Children playing	2	1000
Dog bark	3	1000
Drilling	4	1000
Engine idling	5	1000
Gun shot	6	374
Jackhammer	7	1000
Siren	8	929
Street music	9	1000

Table 2.2: Class name, class IDs and number of samples.

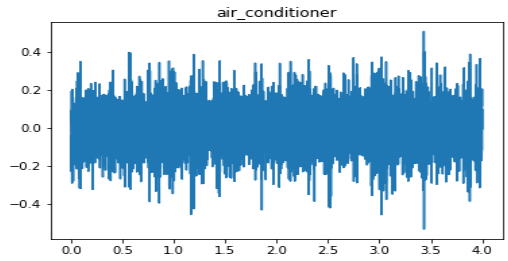
2.2 Audio Feature

In this section, some common representations of audio signals are explained. The study of these terms follows the order of audio waveform, spectrogram, Mel spectrogram, the delta feature. Each subsection studies the definition of these features and visualizes them, the advantage and disadvantages of choosing each feature as input to the neural network are motivated.

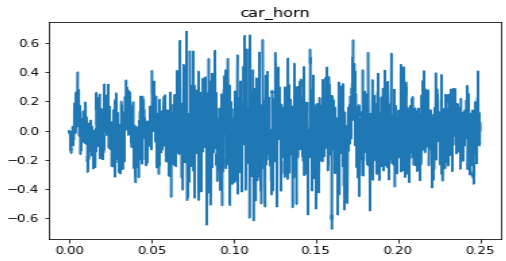
2.2.1 Waveform

The waveform of a signal is a graph that displays its shape as a function of time, showing how the signal changes in amplitude over time. From the perspective of an audio signal, the audio waveform shows how the loudness of the sound changes with respect to time. Visualization of 10 different classes in waveform is shown in Figure 2.1, where each audio is sampled from a sampling frequency of 22.05kHz.

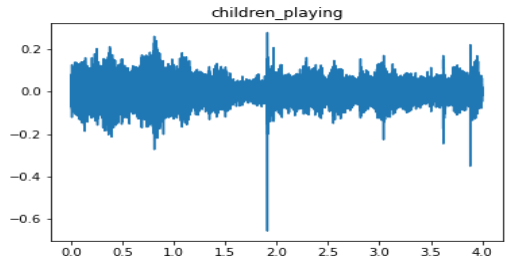
The waveform only contains the information of how audio changes with respect to time, but the information in frequency is lacking, thus it is a less discriminating representation compared to the spectrogram. Consider a deep neural network that is complex enough so that it is capable of directly learning the underlying features from the waveform and the classification task could be performed [10, 25], however, the required complexity for the model and the computational cost is tremendous, and a large amount of input samples are expected. Thus, to efficiently achieve classification, time-frequency spectral features are more promising, which will be discussed in the following subsections.



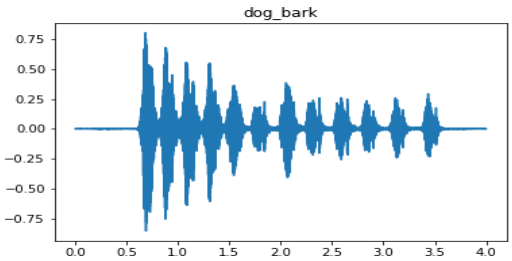
(a) Waveform of air conditioner



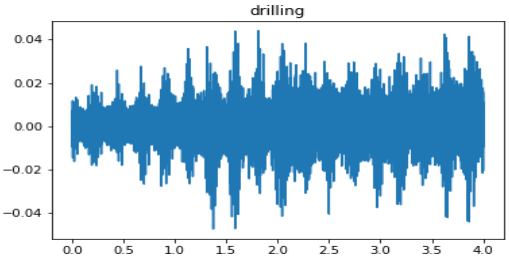
(b) Waveform of car horn



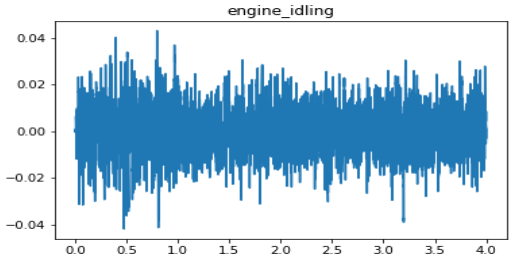
(c) Waveform of children playing



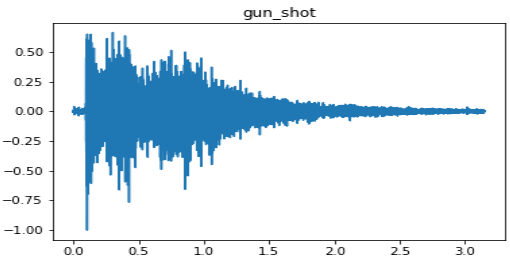
(d) Waveform of dog bark



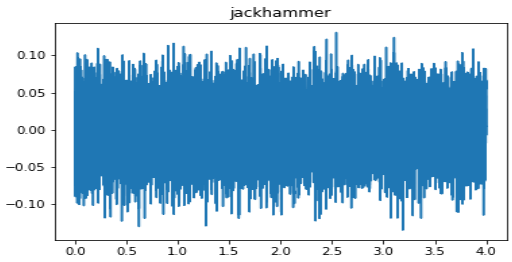
(e) Waveform of drilling



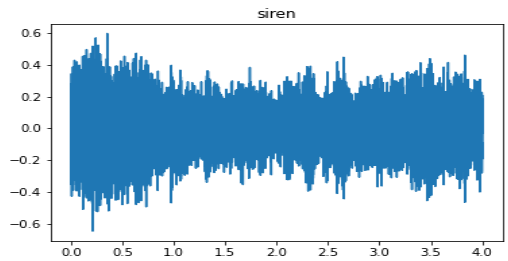
(f) Waveform of engine idling



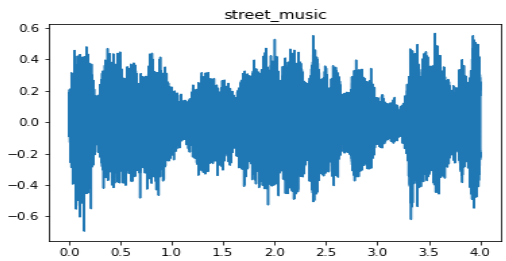
(g) Waveform of gun shot



(h) Waveform of jackhammer



(i) Waveform of siren



(j) Waveform of street music

Figure 2.1: Waveform visualizations

2.2.2 Spectrogram

The spectrogram is a representation of a signal that displays signal strength over time at various frequencies. Spectrograms can be two-dimensional graphs with a third variable represented by color. One dimension represents the time which is on the horizontal axis, and the other dimension indicates the frequency at the vertical axis. The third variable shows the amplitude or the power of a certain frequency at a particular time, represented by color intensity or brightness of each pixel point in the two-dimensional graphs. Thus, a spectrogram visualizes sound's frequency spectrum as a function of time. In a spectrogram, not only can we identify where the energy is distributed over frequency, but also see how energy levels vary across time.

Given an audio signal, which is a set of sampled points denoted as $x(n)$, to generate a spectrogram, the steps and calculations are as follows:

1. Split the original signal $x(n)$ into overlapping short-time frames with equal length. Each contains N samples, which is denoted as the frame length. The overlap length between two frames is usually 50% of the frame length N. The samples in each frame are represented as

$$x_\alpha(n) = \begin{cases} x\left(n + (\alpha - 1) * \frac{N}{2}\right), & n \in [0, N-1] \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

   where α is the index number of one short-time frame, and $x_\alpha(n)$ denotes samples in the α^{th} frame taken from original signal $x(n)$.

2. Each frame is applied with a window function $w(n)$, such as Hann window, in order to overcome spectral leaks through sidelobes [29]. Then, the Fourier Transform is applied to each short-time frame.

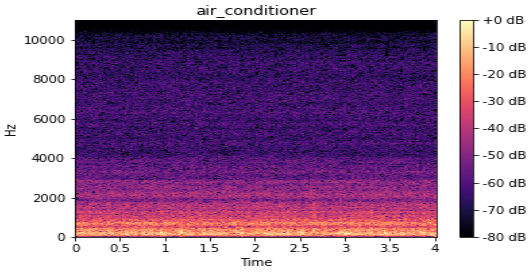
$$X_\alpha(k) = \sum_{n=0}^{N-1} x_\alpha(n) w(n) e^{-j2\pi kn/N} \qquad k = 1, 2 \ldots \frac{N}{2} \tag{2.2}$$

3. The power spectrum density is obtain by taking the square of the magnitude of the frequency spectrum. In each short-time frame, its power spectrum density is denoted as $P_\alpha(k)$, which is given as follows

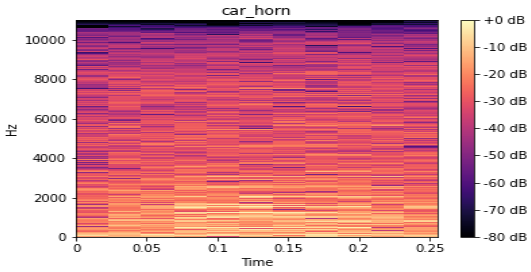
$$P_\alpha(k) = \|X_\alpha(k)\|^2 \tag{2.3}$$

4. Finally, the spectrogram is obtained by transferring the power into the decibel scale, which is a unit measuring the intensity of a sound. This decibel scale is inspired by human's perception of loudness. The human ear has a large dynamic range in sound reception, the ratio between the sound intensity that leads to permanent damage and the limit of smallest audible sound is close to 1 trillion (10^{12}) [30], which is an enormous measurement range. To easily express this range, and conveniently measure an exponentially changing sound loudness over time, the base 10 logarithmic scale is motivated.

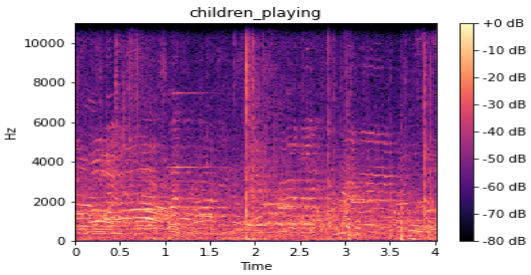
$$P_\alpha^{dB}(k) = 10 log_{10}(P_\alpha(k)) \tag{2.4}$$



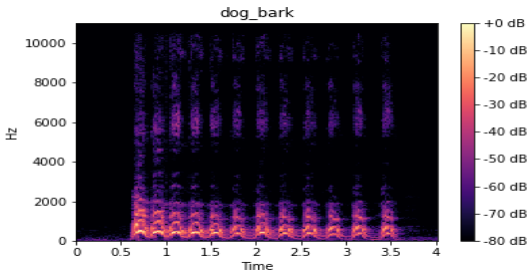
(a) Spectrogram of air conditioner



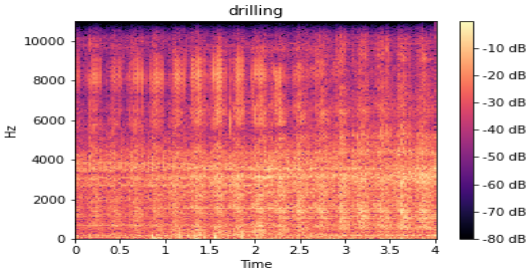
(b) Spectrogram of car horn



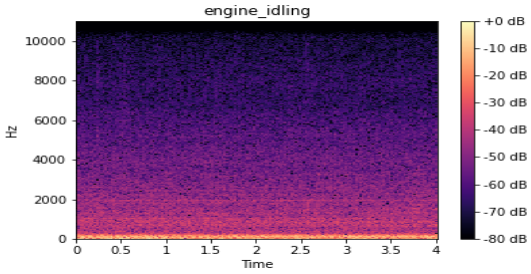
(c) Spectrogram of children playing



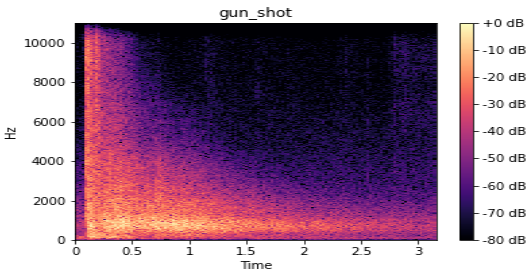
(d) Spectrogram of dog bark



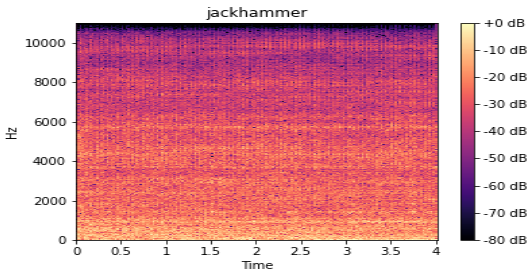
(e) Spectrogram of drilling



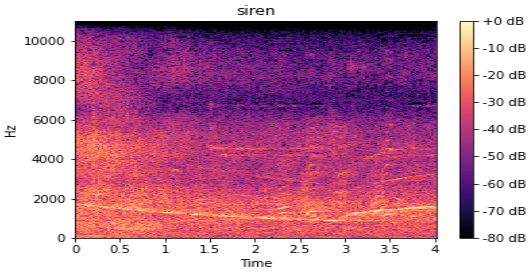
(f) Spectrogram of engine idling



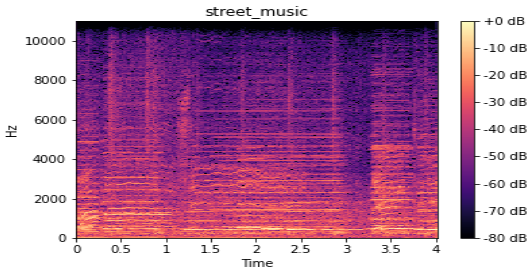
(g) Spectrogram of gun shot



(h) Spectrogram of jackhammer



(i) Spectrogram of siren



(j) Spectrogram of street music

Figure 2.2: Spectrogram visualizations

A spectrogram visualization of the same set of samples in Figure 2.1 is shown in Figure 2.2. Compared to waveform, the spectrogram is a more discriminating representation of an audio as it extracts time-frequency spectral features of audio and displays the energy distribution. However, the horizontal axis displays frequency in a linear scale, while human perception of pitch is not equally sensitive to all sound frequencies. Thus, this can be further modified into more compact feature.

2.2.3 Mel spectrogram

The Mel spectrogram involves applying the Mel filter banks to a spectrogram, converting the frequency (Hz) scale into Mel scale. This is a non-linear transformation that is motivated by human's auditory system, where frequencies are not perceived in a linear scale. It is more noticeable for a human to distinguish the sound in lower frequencies, ranging from 500 and 1000 Hz, than in higher frequencies ranging from 7500 and 8000 Hz. The range of frequencies that can be heard by humans is generally from 20 to 20,000 Hz [31], and the sensitivity is gradually lost as the frequency increases. Thus, applying the Mel filter bank is beneficial as it highlights the variations in lower frequencies and gives more discriminating and informative audio representation.

The Mel scale is transformed from the frequency scale by (2.5)

$$m(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{2.5}$$

the Mel scale can be transformed back to the frequency by (2.6)

$$f(m) = 700\left(10^{m/2595} - 1\right). \tag{2.6}$$

Based on the above transformation formulas, next we can generate and calculate the Mel filter bank.

Mel filter bank is a set of triangular filters, each of the filter is centered and reaches the peak at its center frequency and decreases to zero linearly at the two neighboring filters' center frequency. Those center frequencies of triangular filters are spaced non-linearly as they are obtained by the linearly placed Mel scale and calculated by (2.6). The linearly placed Mel scale is decided by the frequency limits and the number of Mel filters defined.

We specify the parameters that are needed in order to build the Mel filter bank. Assume a spectrogram is given, the sampling rate f_s and the short-time frame length N are known. We first define the number of Mel filters, denoted as N_{Mel}. The highest frequency component in a sampled signal is half of its sampling rate according to Nyquist sampling theorem, $f_{max} = f_s/2$ and the lowest frequency is $f_{min} = 0$. Given all the parameters above, the Mel filter banks are built by the following steps.

1. Based on (2.5), given the maximum and minimum frequency f_{max} and f_{min}, we can calculate the corresponding maximum and minimum Mel value, denoted as m_{max} and m_{min} respectively.

$$m_{max} = 2595 \log_{10}\left(1 + \frac{f_{max}}{700}\right), \tag{2.7}$$

$$m_{min} = 2595 \log_{10}\left(1 + \frac{f_{min}}{700}\right). \tag{2.8}$$

2. Specify the number of Mel filters N_{Mel}, generate number of $N_{Mel}+2$ linearly spaced points between m_{max} and m_{min}, where each point $m(i)$ is

$$m(i) = m_{min} + (i-1) * \frac{m_{max} - m_{min}}{N_{Mel}+2}, \qquad i = 1,2 \ldots N_{Mel}+2 \tag{2.9}$$

3. Based on (2.6), convert each linearly spaced $m(i)$ point back to frequency scale, denoted as $h(i)$.

$$h(i) = 700\left(10^{m(i)/2595} - 1\right), \qquad i = 1,2 \ldots N_{Mel}+2 \tag{2.10}$$

4. Calculate the non-linearly spaced center frequencies for the triangular filters, denoted as $f(i)$

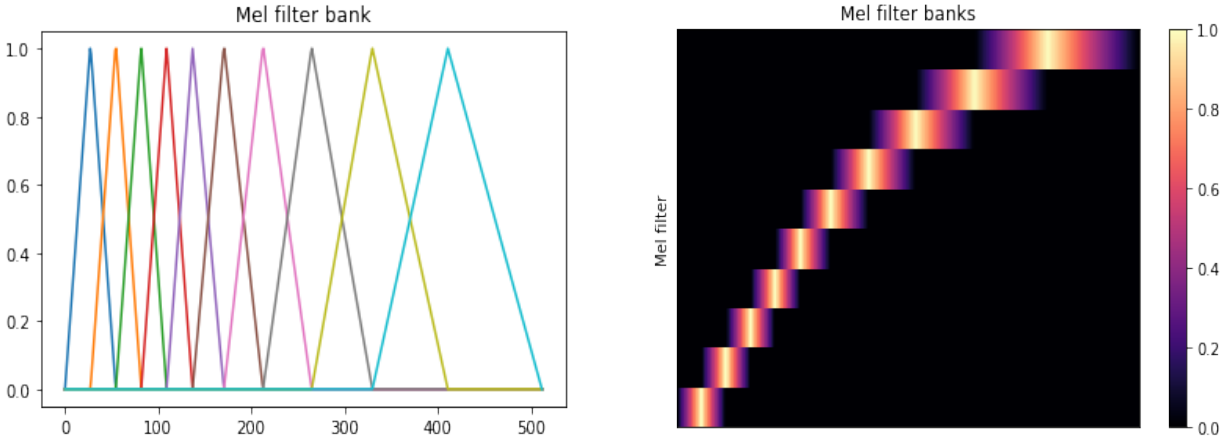
$$f(i) = \left\lfloor \frac{(N_{FFT}+1) * h(i)}{f_s} \right\rfloor, \qquad i = 1,2 \ldots N_{Mel}+2 \tag{2.11}$$

5. Finally, given all the center frequency points f(i), the Mel filters, denoted as $H_m(k)$, are generated and calculated by the following formula

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases} \qquad m = 1,2 \ldots N_{Mel} \tag{2.12}$$

where m denotes the index number of a Mel filter and k is the point in that filter.

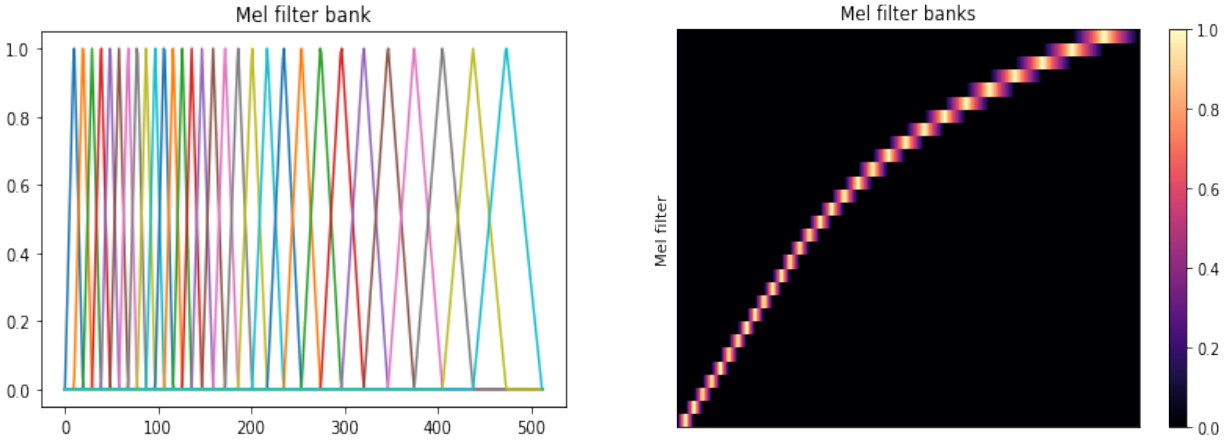
Here we visualize two examples of Mel filter banks. The parameters are set as follows. Assume the sampling rate f_s is 8000 Hz, the length of short-time frame N is 1024. The first example has 10 Mel filters, and the second has 30. The visualization of Mel filters' triangular windows and intensity are both shown in Figure 2.3 and 2.4 respectively.



(a) Triangular overlapping window

(b) Intensity of Mel filter banks

Figure 2.3: 10 Mel Filter banks visualization.

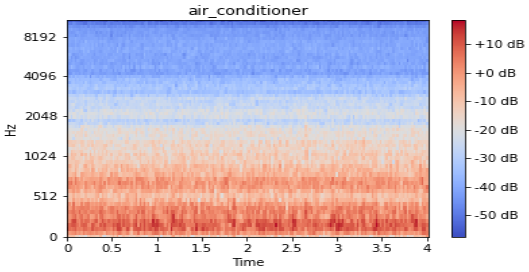


(a) Triangular overlapping window

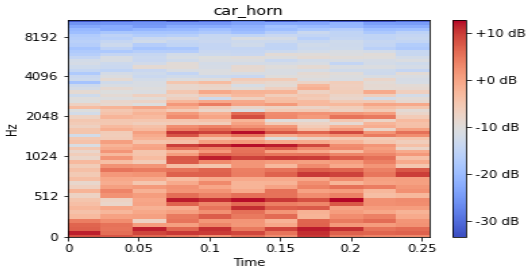
(b) Intensity of Mel filter banks

Figure 2.4: 30 Mel Filter banks visualization.

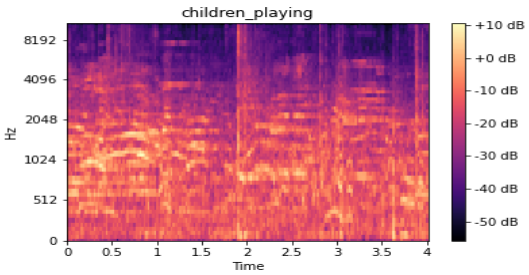
The Figure 2.3a and Figure 2.4a shows 10 and 30 overlapping triangular curves respectively, Figure 2.3b and Figure 2.4b shows their intensity. From the figures, we can observe that in lower frequencies, the center frequency points are more tightly spaced, and their filters are narrower. When applying this Mel filter bank to the spectrogram, the variation of spectrum in lower frequencies are more concerned than that in higher frequencies.



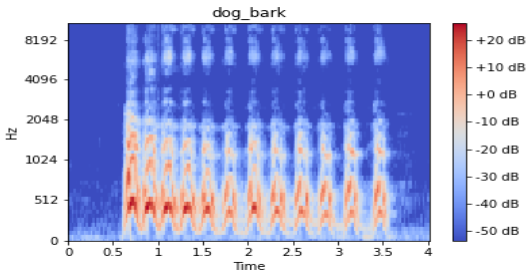
(a) Mel spectrogram of air conditioner



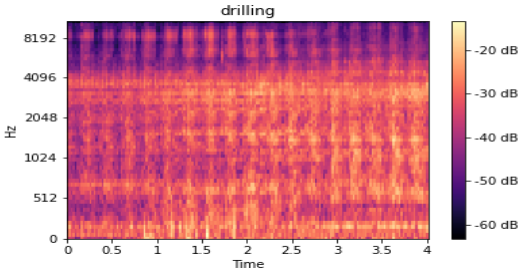
(b) Mel spectrogram of car horn



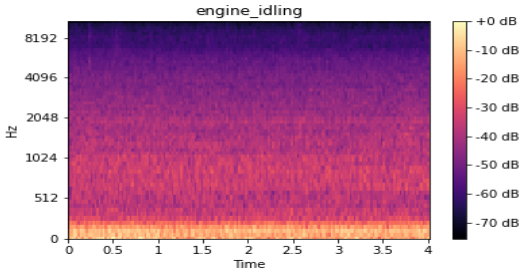
(c) Mel spectrogram of children playing



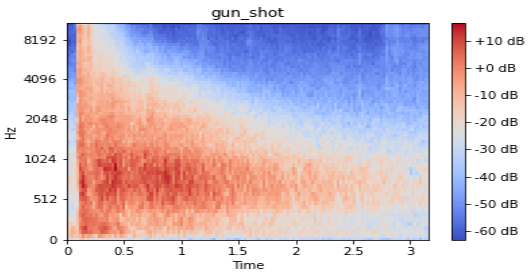
(d) Mel spectrogram of dog bark



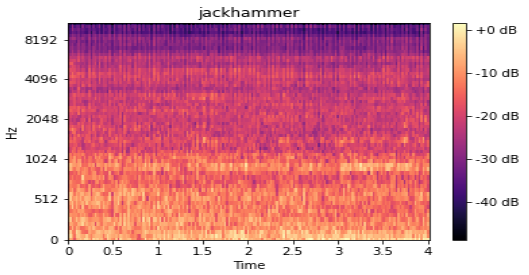
(e) Mel spectrogram of drilling



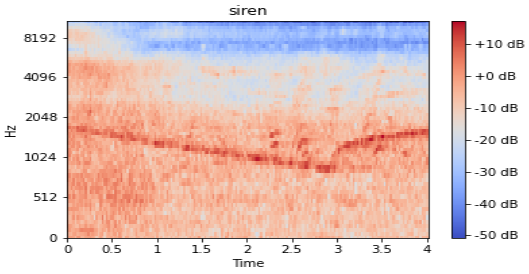
(f) Mel spectrogram of engine idling



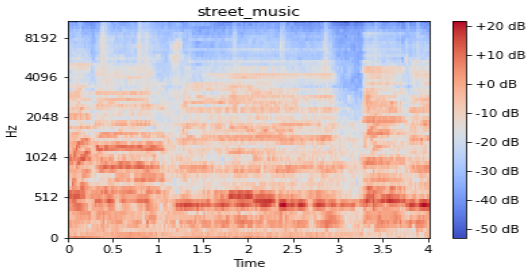
(g) Mel spectrogram of gun shot



(h) Mel spectrogram of jackhammer



(i) Mel spectrogram of siren



(j) Mel spectrogram of street music

Figure 2.5: Log-Mel spectrogram visualizations

After defining the Mel filter bank, we apply it to the power spectrum obtained by (2.3) with the matrix multiplication, where each point in the Mel spectrogram $MEL(m,\alpha)$ is calculated as

$$MEL(m,\alpha) = H_m(k) \cdot P_\alpha(k), \tag{2.13}$$

then take the logarithmic scale to transform the power into decibel unite, resulting in the log-Mel spectrogram,

$$MEL^{dB}(m,\alpha) = 10log_{10}(MEL(m,\alpha)). \tag{2.14}$$

Visualization of log-Mel spectrogram is shown in Figure 2.5, which is obtained by applying 64 Mel filter banks to the spectrogram in 2.2. Benefited from the Mel scale, which highlights and emphasizes variations on components in lower frequencies, the characteristics of power distribution with respect to time and frequency are more distinctive compared to the spectrogram in 2.2.

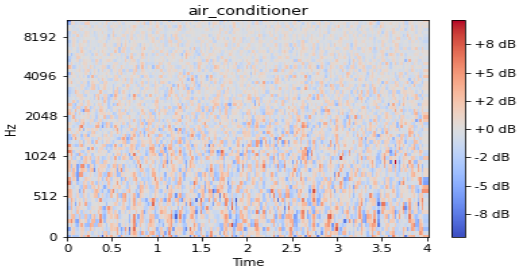
2.2.4 Delta feature

Delta feature of the audio is also known as differential and acceleration coefficients because it describes the dynamical changes in the spectrogram with respect to time. This feature can be calculated by taking the difference between a time frame $MEL^{dB}(\alpha)$ in the log-Mel spectrogram and a previous time frame that is s steps ahead of it, denoted by $MEL^{dB}(\alpha - s)$, where s is usually chosen to be 3. The calculation is defined by the following formula,

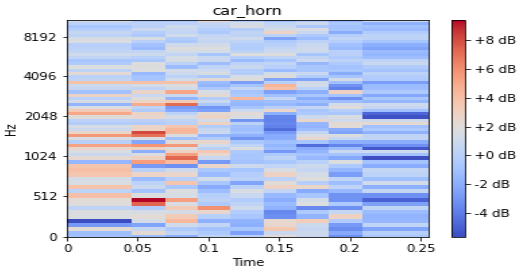
$$\Delta_\alpha = MEL^{dB}(\alpha) - MEL^{dB}(\alpha - s). \tag{2.15}$$

A visualization of the delta feature is shown in Figure 2.6.

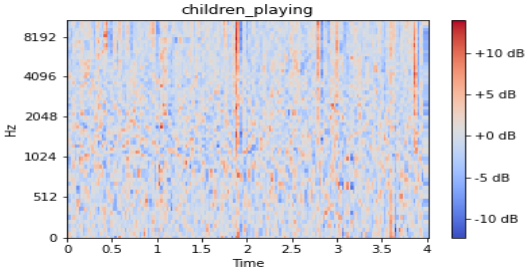
From the figures we can see trajectories of how the log-Mel spectrogram changes over time. As spectrograms only carry the information of power spectral envelope of one single frame, which are static features, while the delta feature contains dynamic information. Combining those two may carry more information of the sound, and the neural networks' performance can be improved [32, 33]. However, this will introduce a bigger input and requires more parameters of the model, resulting in an increase of computational complexity.



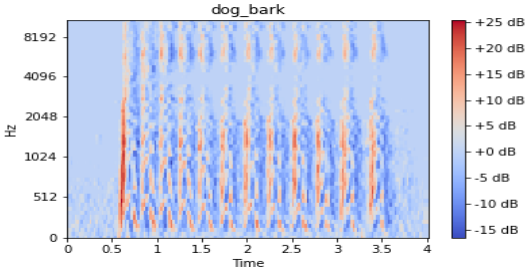
(a) Delta of air conditioner



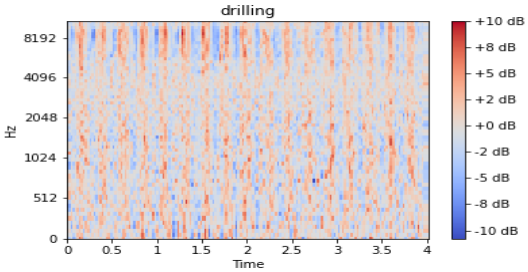
(b) Delta of car horn



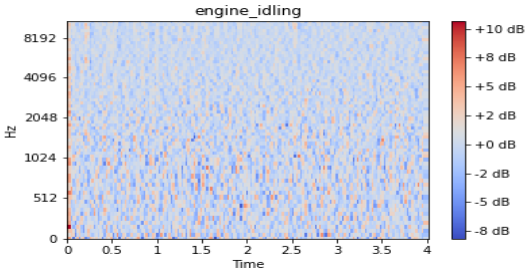
(c) Delta of children playing



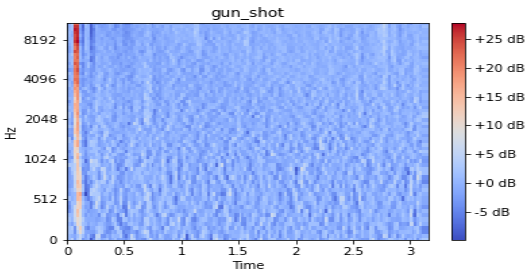
(d) Delta of dog bark



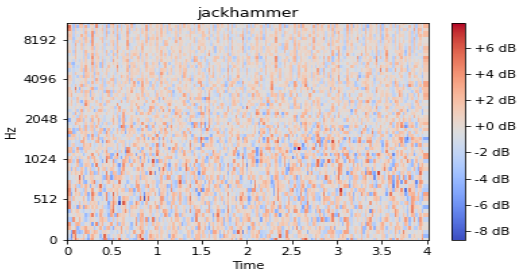
(e) Delta of drilling



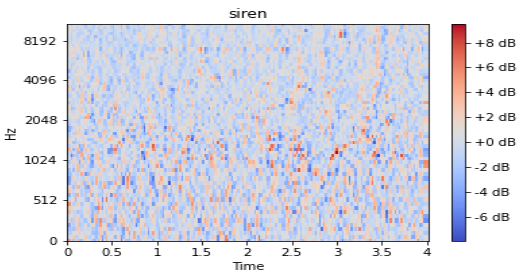
(f) Delta of engine idling



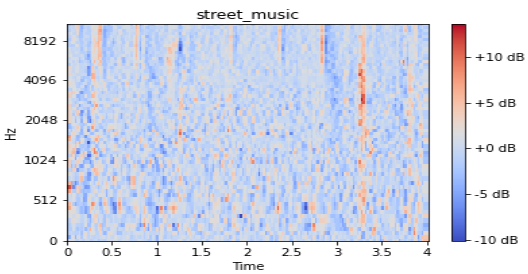
(g) Delta of gun shot



(h) Delta of jackhammer



(i) Delta of siren



(j) Delta of street music

Figure 2.6: Delta feature visualizations

2.2.5 Discussion

In this section, various audio features that can be extracted from a sound sample have been studied. Among all the audio features we have discussed, the log scaled Mel spectrogram is the most promising one as it aligns with the mechanism of the human's auditory system. The human's' perception of pitch is not in a linear extent, instead, more variations in lower frequencies are captured than in higher frequencies. Thus, Mel spectrogram puts more attention on lower frequencies, giving a more discriminating representation than the spectrogram whose frequency axis is linearly scaled. The use of logarithm operation is to convert the power in Mel spectrogram into decibel scale, allowing convenient expression of a large range and the exponentially changing power in a signal. Moreover, the delta feature can also be used together with log-Mel spectrogram, as it illustrates the dynamical variations in the static spectrogram.

2.3 Pre-processing

In this section, we will give the procedures and operations of our pre-processing system for audio files in Urbansound8k dataset in order to prepare the input for neural networks. We introduce the procedures that are taken in the signal processing task, converting data from raw audio into log-Mel spectrogram, at the same time the involved parameters are specified. Also, we consider data augmentation strategies to expand our dataset.

2.3.1 Procedures

The steps and operations involved in our pre-processing task will be discussed. The goal of pre-processing is to transform all the audio samples into log-Mel spectrogram with consistent dimension. The steps are explained as follows and the notations are specified.

1. Sample all of the audio files at a sampling frequency denoted as f_s, obtaining the audio waveform $x(n)$.
2. We want to obtain audio representations consistent in dimension, but durations of audio samples in the dataset are not the same. We define a duration of L seconds, which is the duration that all the clips are trimmed into. For samples that are originally shorter than L, they are repeated until exceed L.
3. Define the short-time frame length N and the overlap length as half of N, obtain the power spectrum by (2.1), (2.2) and (2.3).
4. Decide the number of Mel filters N_{Mel} and apply the Mel filters to the power spectrum, obtaining the Mel spectrogram.
5. Extract a subset segment of length L from the audio to make the dimensions consistent, this segment includes the loudest part of the original audio. It is achieved by a sliding window that selects $\left\lfloor \frac{2L}{f_s * N} \right\rfloor$ short-time frames with the maximum power in the Mel spectrogram.
6. Convert the Mel spectrogram into decibel scale, resulting in log-Mel spectrogram.

A diagram of this procedure is shown in Figure 2.7. In each step, the notations of parameters are specified, and the expected audio representations after each operation are given.

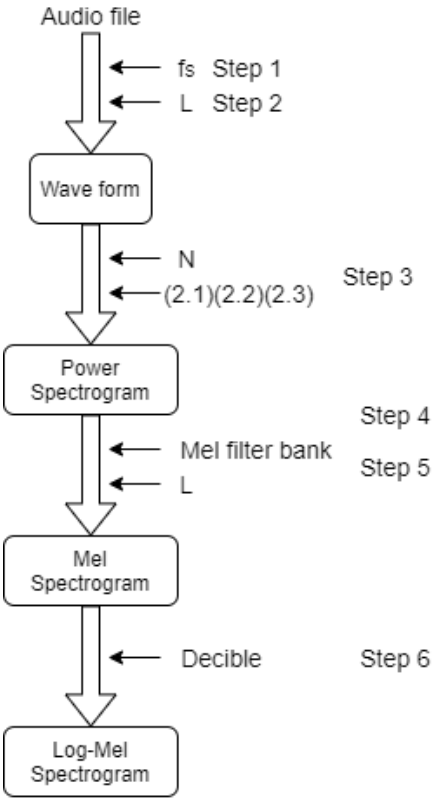


Figure 2.7: Diagram of pre-processing procedure

2.3.2 Data augmentation

Data augmentation is an approach that is used to expand the dataset without collecting new data, which helps to reduce overfitting and gives better generalization. This is achieved by applying slight modifications to the existing data, resulting in an increase of the diversity in the dataset. To expand the existing Urbansound8k dataset, we apply some deformations to the waveform of the original audio samples, which are namely time stretching and pitch shifting.

Time stretching is a time scale modification that stretches or compresses the duration of a given audio signal, resulting in the slow down or speed up of the sound, while preserving the pitch of the original signal. Pitch scaling is the opposite, which is a process of raising or lowering the pitch of the audio sample without changing the duration or the speed. To note that the deformation parameters must be properly chosen such that the augmentations do not lose the semantic validity of the label. Thus the time stretching factors that we choose are 0.81, 1.23, and the pitch scaling factors are -2, 2, which are suggested by [34]. The visualizations of the resulting log-Mel spectrogram are shown in Appendix A. From the vi-

sualizations, we can discover that the fundamental shape and pattern of the spectrogram was not changed. However, the horizontal axis indicating the time was stretched or compressed by the time stretch effect, and the frequency distribution was shifted upwards or downwards by the pitch shift effect.

2.4 Summary

The dataset and the audio features were investigated in this chapter, and pre-processing procedures were given. In this project, we base our work on the dataset of Urbansound8k as its contents are most relevant. The audio samples are processed into the representation of the log-Mel spectrogram, which will be used as the input to our neural network models. Log-Mel spectrogram was chosen because it aligns with the human auditory system with respect to pitch and gives the most discriminating representations in our urban noise case. The pre-processing procedures that transform the audio samples into the expected feature in a consistent dimension were detailed step by step. Data augmentation was considered to help expand the dataset by applying distortions to the original samples, pitch shifting and time stretching are applied to the waveform.

3 Neural Network Models

In this chapter, we will study the neural network models that are involved in this project, including convolutional neural network (CNN) and autoencoder-based models. CNN is the most straightforward model to perform classification and identification tasks and its operations can efficiently learn the structure and pattern of an image. The autoencoder-based models include variational autoencoder (VAE), β-VAE and bounded information rate variational autoencoder (BIR-VAE). These models consist of two networks, which are encoder and decoder. The autoencoder-based models can reconstruct the input data and learn its underlying feature.

3.1 Convolutional Neural Network

In this section, we will introduce components and operations in a CNN model, which mainly include convolution operation, non-linear activation, and pooling operation together with its optimization function. Based on these components, we will build and define a CNN model to perform classification task in the subsequent experimental part.

3.1.1 Architecture

The architecture of CNN can be separated into two parts, one is the feature learning part and the other is the classification part. The first part mainly consists of convolutional and pooling operation, and the second part is a fully connected network, which only consists of fully connected layers and their activation functions.

The convolution operation is the most important building blocks of CNN. This process learns and extracts the underlying patterns of the image and identifies boundaries. The operation uses a matrix called filter or kernel to slide over the input across its width and height, and produces a feature map, which is the response of that kernel. The feature map is calculated by a dot product, or the element-wise multiplication, between the filter matrix and the patch in the image that a kernel covers, and then all the elements are summed into a scalar value, comprising one pixel of the feature map. A simple example of this operation is visualized by Figure 3.1

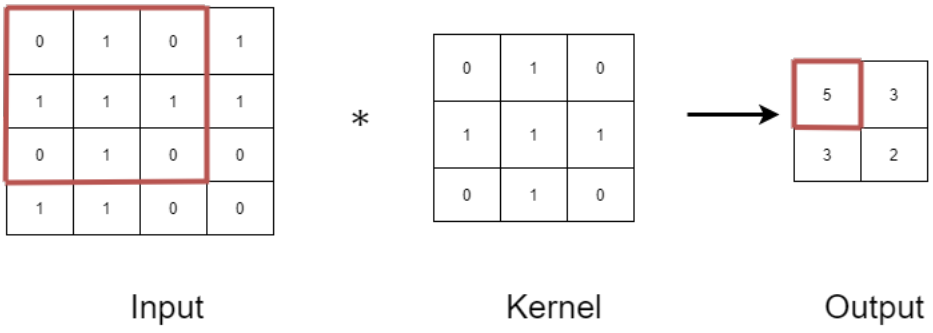


Figure 3.1: Convolution operation.

The hyper-parameters in this opertion are the kernel size, denoted as $\boldsymbol{K}$, which is the dimension of the filter matrix; the number of kernels C_{out}, which decides how many feature maps that an operation learns; the stride size $\boldsymbol{S}$, indicating the number of pixel that the filter slides each time. The padding operation is used to control the dimension of the output size by padding the input with zeros around the border, the size of which denoted as padding size $\boldsymbol{P}$. Now given an input image or feature map with size (C_{in}, H_{in}, W_{in}), the resulting output size $(C_{out}, H_{out}, W_{out})$ after a convolutional operation is calculated by

$$\begin{aligned} H_{out} &= \left\lfloor \frac{H_{in}+2\times P[0]-K[0]}{S[0]} + 1 \right\rfloor, \\ W_{out} &= \left\lfloor \frac{W_{in}+2\times P[1]-K[1]}{S[1]} + 1 \right\rfloor. \end{aligned} \tag{3.1}$$

After each convolution operation, an activation function is applied, which is usually Rectified Linear Units (ReLU) activation expressed as $f(x) = max(0,x)$. This brings non-linear property that makes the features more intense and cuts off unnecessary details by setting a threshold at zero.

The pooling layer usually follows a convolution layer. The operation of pooling is to down-sample and reduce the dimensionality of the feature map, such that the computational efficiency can be increased and prevent over-fitting, which is the phenomenon where the model performs well on the training set but poorly on the test set. The pooling operation results in a compressed feature map, but still contains the essential features and significant information of the image so that the detection and classification can be realized. The most common type of pooling is max-pooling, which takes the maximum value in a portion of the image covered by the pooling window. These pooling window sizes are hyper-parameters that need to be specified beforehand, which decides the reduction size of the feature maps.

The classification part of a CNN is the fully connected network, this network learns a set of weight based on the features learned by the previous feature learning network and outputs the predicted label. The convolutional layers output a set of feature maps, before passing them through the fully connected layers they should be flattened into one vector. The fully connected layers are simply composed of neuron unites and ReLU activation functions. To prevent over-fitting, dropout operation is used, which is an operation that randomly skips

some number of neurons in a layer during training [35]. As in our classification case with 10 classes, the last layer's activation function should be softmax, which is expressed as

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}. \tag{3.2}$$

The softmax activation is used to predict the probabilities of a class label. The $\boldsymbol{x}$ is the input vector to the last fully connected layer and n is its dimension. i is a class label, the value output by the above softmax function is the probability of the i_{th} class.

3.1.2 Loss function

As there are 10 different classes of audio to identify in the data set, which leads to a multi-class classification problem. Thus, as we discussed previously, the last layer's activation function should be softmax, which outputs the probability of the predicted class. We denote a vector indicating the predicted probability of each class as $\hat{\boldsymbol{y}}$, where each element is $\hat{y}_i$. The dimension of this vector corresponds to the number of classes of the data set. This $\hat{\boldsymbol{y}}$ is calculated by softmax and compared with the one-hot encoded class label of the ground truth denote as $\boldsymbol{y}$, where each element is y_i and $y_i = 1$ if the class label index is i and $y_{j \neq i} = 0$.

To measure the difference between the output prediction and ground truth, we use the cross-entropy as loss function. Assuming the number of a class is n, the cross-entropy loss is express as follows

$$Loss = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i. \tag{3.3}$$

This process can be simply understood as returning the probability of the estimated class label because only one value in the one-hot encoded vector is one while the others are zero. The cross-entropy loss is a good measure of how different two discrete samples are distributed from each other. The negative sign ensures that the loss decreases when distributions get closer to each other. Thus, minimizing this loss encourages the estimated class label to be close to the ground truth label, which is expected to ideally predict the probability of the ground truth label as one while the other class is zero.

3.1.3 Discussion

CNN is generally composed of two parts, one is the feature learning part and the other is the classification part. The feature learning part is capable of capturing patterns of images by its convolution operation and reduce the number of parameters by pooling operations. The classification part is a fully connected network, which learns a set of weights of the features. CNN learns the mapping from input image data to the output label, which is a discriminative feed forward model. We start with CNN in our experiment as it is the most straightforward model to perform classification task.

3.2 Variational Autoencoders

In this section, we first introduce the variational autoencoder's functionality and motivate the usefulness of this model in classification tasks. Then, we explain its architecture and derive the optimization formula. On top of the fundamental theories, its variations and new models are introduced.

3.2.1 Background

Variational autoencoder (VAE) [26] is a framework that jointly trains two models, which are encoder and decoder. Encoder of VAE is an inference model that learns the latent distribution of the data and outputs latent variables that are sampled from this distribution. Decoder is a generative model that reconstructs the input data during training, where the reconstruction is based on latent variables given by encoder. After the training, the decoder can also generate new samples that are similar to the input data. The process of training this framework involves two terms, the first is reconstructing the input data at the output, the second is a constraint that enforces the latent distribution to be close to the prior distribution of latent variables [36, 37]. After training the VAE, the encoder has learned latent distribution of the data, and it can represent meaningful features about the inputs in a lower dimension. In classification task, the encoder can be used as the feature extractor.

3.2.2 Architecture

The architecture of VAE [26] is motivated by the approximation of a posterior probability density and the architecture of autoencoder [8, 38]. We first introduce the architecture of the autoencoder, and then derive the approximation of this posterior probability function that motivates the architecture VAE.

An autoencoder is an unsupervised neural network that is trained to reconstruct its input at the output. However, this is not an exact copy of input because some of its details are lost. This process is done by first represent the input data using a lower-dimensional vector by a deterministic mapping, then reconstruct the original input from this vector. The architecture of the autoencoder is shown in Figure 3.2

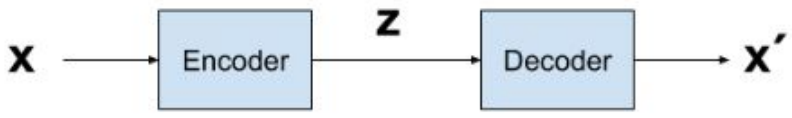


Figure 3.2: Architecture of autoencoders,

where x is the input and $x^{'}$ is input's reconstruction at the output, z is the latent vector. The cost function of the autoencoder is referred to as the reconstruction loss, which is generally selected to be the mean-squared error between the output and input, measuring

how close the reconstructed input is to the original input.

Consider the case where the data $\boldsymbol{x}$ is generated by a random process, involving an unobserved continuous latent variable $\boldsymbol{z}$, whose distribution is the prior probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, and $\boldsymbol{x}$ is generated from the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, both of those distributions are assumed to be differentiable. We are interested to obtain this generative process's parameter $\boldsymbol{\theta}$ and the posterior probability density of $\boldsymbol{z}$ given $\boldsymbol{x}$, which is expressed as

$$p_{\theta}(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_{\theta}(\boldsymbol{z})p_{\theta}(\boldsymbol{x}|\boldsymbol{z})}{p_{\theta}(\boldsymbol{x})}. \tag{3.4}$$

However, the denominator $p_{\theta}(\boldsymbol{x})$, which is called evidence, is intractable because the integral $p_{\theta}(\boldsymbol{x}) = \int p_{\theta}(\boldsymbol{x}|\boldsymbol{z})p_{\theta}(\boldsymbol{z})d\boldsymbol{z}$ is unavailable and $\boldsymbol{z}$ is unknown. Hence the true posterior probability density $p_{\theta}(\boldsymbol{z}|\boldsymbol{x})$ is intractable. To tackle this problem, an approximation of $p_{\theta}(\boldsymbol{z}|\boldsymbol{x})$ is introduced, which is an inference model, denoted as $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$, where ϕ are the parameters that define this model. In the perspective of autoencoder model, the $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ and $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$ are interpreted as the probabilistic encoder and decoder respectively, data $\boldsymbol{z}$ are the latent variables. The encoder $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$, parameterized by ϕ, outputs a set of parameters that describe the probability distribution of $\boldsymbol{z}$ given the input data $\boldsymbol{x}$, for example, mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of a Gaussian distribution. The decoder $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$, parameterized by θ, describes the distribution over $\boldsymbol{x}$ given $\boldsymbol{z}$, reconstructing the input $\boldsymbol{x}$ at the output given the latent variable $\boldsymbol{z}$. In the scenario where the latent variable $\boldsymbol{z}$ is under Gaussian distribution, the architecture of a VAE is shown in Figure 3.3

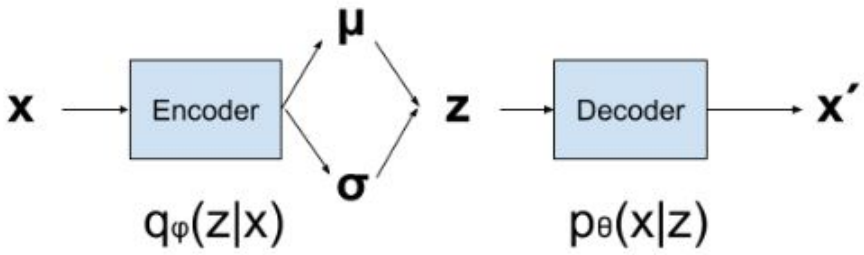


Figure 3.3: Architecture of variational autoencoders

3.2.3 Loss Function

In this section, we are going to derive and explain the loss function of an VAE. In the previous section, we discussed that the architecture of VAE was motivated by an approximation of an intractable posterior probability distribution $p_{\theta}(\boldsymbol{z}|\boldsymbol{x})$ using variational inference. The approximation of this true $p_{\theta}(\boldsymbol{z}|\boldsymbol{x})$ is denoted as $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$. To measure how close our approximated distribution $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ is to the true distribution $p_{\theta}(\boldsymbol{z}|\boldsymbol{x})$, and evaluate this approximation, we introduce the Kullback-Leibler divergence, in short, Kl divergence. The Kl divergence is the expectation of the logarithmic difference between the probabilities, and it quantitatively measures the distance between two probability distributions and how different they are. The Kl divergence is always positive and the closer the two distributions are, the

smaller its value is. Now expressing two distribution as $p_1(x)$ and $p_2(x)$, the KL divergence is

$$KL(p_1(x)||p_2(x)) = \mathbb{E}_{\boldsymbol{x}\sim p_1(x)} \log \frac{p_1(x)}{p_2(x)}. \tag{3.5}$$

We propose our variational inference problem as minimizing the KL divergence between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, and the optimum $q^*_\phi(\boldsymbol{z}|\boldsymbol{x})$ leads to the minimum value of the KL divergence, the problem can be started as

$$q^*_\phi(\boldsymbol{z}|\boldsymbol{x}) = \arg\min \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x})). \tag{3.6}$$

As previously discussed, the $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is not available because it requires the calculation of the evidence $p_\theta(\boldsymbol{x})$, which is intractable. Thus, the KL divergence in (3.6) cannot be calculated or optimized directly. Instead, we further derive the loss function into a term called the evidence lower bound (ELBO) and the natural logarithm of evidence term $\log p_\theta(\boldsymbol{x})$. The derivation is as follows

$$\begin{aligned}
\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x})) &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})} \\
&= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{z}|\boldsymbol{x}) \\
&= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log \frac{p_\theta(\boldsymbol{z},\boldsymbol{x})}{p_\theta(\boldsymbol{x})} \\
&= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{z},\boldsymbol{x}) + \log p_\theta(\boldsymbol{x}) \\
&= -\mathrm{ELBO} + \log p_\theta(\boldsymbol{x}).
\end{aligned} \tag{3.7}$$

In (3.7), the ELBO is

$$\mathrm{ELBO} = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{z},\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log q_\phi(\boldsymbol{z}|\boldsymbol{x}), \tag{3.8}$$

which is the only term in the KL divergence of (3.6) that depends on $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and skips the intractable term $\log p_\theta(\boldsymbol{x})$. We can rewrite (3.7) as

$$\log p_\theta(\boldsymbol{x}) = \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x})) + \mathrm{ELBO}. \tag{3.9}$$

Since the KL divergence is always non-negative, the ELBO is a lower bound of the evidence,

$$\log p_\theta(\boldsymbol{x}) \geq \mathrm{ELBO} = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{z},\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log q_\phi(\boldsymbol{z}|\boldsymbol{x}). \tag{3.10}$$

The optimization problem in (3.6) can then be equivalently interpreted from the minimization of KL divergence that contains intractable evidence term into the maximization of the ELBO, which is tractable in this case. And we further derive the loss function by rewriting the ELBO in the (3.9) into

$$\begin{aligned}
\text{ELBO} &= -\text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x})) + \log p_\theta(\boldsymbol{x}) \\
&= -\text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|\frac{p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})}{p_\theta(\boldsymbol{x})}) + \log p_\theta(\boldsymbol{x}) \\
&= -\text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) + \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\log p_\theta(\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})}{p_\theta(\boldsymbol{x})}) + \log p_\theta(\boldsymbol{x}) \\
&= -\text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) + \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) - \log p_\theta(\boldsymbol{x}) + \log p_\theta(\boldsymbol{x}) \\
&= -\text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) + \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\log p_\theta(\boldsymbol{x}|\boldsymbol{z}),
\end{aligned} \tag{3.11}$$

Maximizing the ELBO involves two terms. The first term is maximizing the negative KL divergence between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the prior probability distribution of the latent variable $p_\theta(\boldsymbol{z})$, which is equivalent to minimizing the positive of it. The second term is to maximize the expected log-likelihood, indicating the probability that the input can be reconstructed at the output. The expectation is taken over the decoder's distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with respect to the encoder's distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. In order to perform gradient descent optimization, the lost function of VAE is to minimize the negative of the ELBO. The final loss function of VAE is formulated as (3.12),

$$L_{VAE}(\phi,\theta) = -\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) + \text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})). \tag{3.12}$$

In optimizing the loss function 3.12, the gradient is taken with respect to the variational parameters ϕ and generative parameters $\boldsymbol{\theta}$, then the two parameters are updated jointly. Hence it is important to make sure the ELBO is differentiable with respect to these two parameters. However, since $\boldsymbol{z}$ are stochastic variables randomly sampled from distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, we need to express $\boldsymbol{z}$ in a deterministic way to make sure the expectation with respect to $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ in (3.12) is differentiable. The reparameterization trick introduces an auxiliary variable ε, with $\varepsilon \sim p(\varepsilon)$, to express the random variable $\boldsymbol{z}$ deterministically. In Gaussian distributed case, the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ outputs the parameters mean μ and standard deviation σ that describes this Gaussian distribution $N(\mu,\sigma^2)$

$$\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = N\left(\boldsymbol{z};\boldsymbol{\mu},\boldsymbol{\sigma}^2\boldsymbol{I}\right). \tag{3.13}$$

The latent variable $\boldsymbol{z}$ is sampled from this distribution, thus $\boldsymbol{z} \sim N(\mu,\sigma^2)$. Expressing in unite Gaussian as

$$\frac{\boldsymbol{z}-\mu}{\boldsymbol{\sigma}} \sim N(0,1), \tag{3.14}$$

let $p(\varepsilon) = N(0,1)$, then $\varepsilon \sim N(0,1)$. To obtain latent variable $\boldsymbol{z}$, instead of directly sample from $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, we use the ε, which is a set of data randomly sampled from the unit Gaussian $N(0,1)$. Then shift ε by the latent distribution's mean μ, and scale it by the latent distribution's standard deviation σ. The latent variable $\boldsymbol{z}$ can be obtain as

$$\boldsymbol{z} = \mu + \sigma \odot \varepsilon \qquad\qquad \varepsilon \sim N(0,1), \tag{3.15}$$

where $\odot$ represents the element-wise product. With this reparameterization trick interpreting the random variable $\boldsymbol{z}$ in a deterministic way, the expectation term in (3.12) can be written in an expression that is differentiable with respect to parameter ϕ, which is described by μ and σ, as they are learned by the encoder network parameterized by this ϕ. The expectation term can now be expressed as

$$\begin{aligned} \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x}|\boldsymbol{z}) &= \mathbb{E}_{\varepsilon\sim N(0,1)} \log p_\theta(\boldsymbol{x}|\mu + \sigma\varepsilon^{(l)}) \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\boldsymbol{x}|\mu + \sigma\varepsilon^{(l)}) \qquad\qquad \varepsilon^{(l)} \sim N(0,1), \end{aligned} \tag{3.16}$$

where L is the mini-batch. With this expression the gradients with respect to ϕ and θ can be taken and the two parameters can be optimized jointly.

3.2.4 Disentanglement and β-variational autoencoder

Disentanglement is a notation that refers to a representation of data where each variable in this representation is only sensitive to one factor and at the same time, invariant to other factors [39, 40]. This requires the neural network models to recognize the independent factors of variations underlying in the data. In generative modeling and disentangled representation learning, such as the case of VAE model, a set of observable high dimensional data, denoted as $\boldsymbol{x}$, is generated through a generative process, from a set of lower dimensional data denoted as $\boldsymbol{z}$. This $\boldsymbol{z}$ is often not observable but describes the factors of variations of $\boldsymbol{x}$. Successful disentangled representation requires $\boldsymbol{z}$ to capture the variations and underlying factors of $\boldsymbol{x}$, such that $\boldsymbol{z}$ contains the essential information that describes the $\boldsymbol{x}$, at the same time, in a lower dimension and compact interpretable structure.

A modification of VAE that encourages disentanglement is the β-VAE [41, 42]. The derivation of its loss function is similar to the incentive of VAE. The idea is to reconstruct the original input data at the output, at the same time, keep the distribution describing the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the prior probability distribution of the latent variable $p_\theta(\boldsymbol{z})$ as close as possible, which is measured by the KL divergence. This KL divergence is under a small constraint δ. The initial optimization function is formulated as follows

$$\begin{aligned} &\max_{\phi,\theta} \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \\ &\text{subject to } \mathrm{KL}\left(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z})\right) < \delta \end{aligned} \tag{3.17}$$

Re-write the (3.17) as a Lagrangian with a Lagrangian multiplier β under the KKT condition.

The optimization problem with constraint can be formulated into

$$F(\phi,\theta,\beta,\boldsymbol{x},\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) - \beta \left(\mathrm{KL}\left(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z})\right) - \delta\right). \tag{3.18}$$

As β and δ are positive, maximizing the (3.18) is equivalent to maximizing the following equation

$$\begin{aligned} F(\phi,\theta,\beta,\boldsymbol{x},\boldsymbol{z}) &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) - \beta \left(\mathrm{KL}\left(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z})\right) - \delta\right) \\ &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) - \beta \mathrm{KL}\left(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z})\right) + \beta\delta \\ &\geq \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) - \beta \mathrm{KL}\left(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z})\right) \end{aligned} \tag{3.19}$$

Maximizing the lower bound of the Lagrangian in (3.19) is equivalent to minimizing the negative of it, which gives our loss function for the β-VAE

$$L_{\beta-VAE}(\phi,\theta) = -\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \beta \mathrm{KL}\left(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z})\right), \tag{3.20}$$

where the β is considered as a hyper-parameter. If the $\beta = 1$, it corresponds to the original VAE; in the set-up of β-VAE, it's usually $\beta \geq 1$. Varying the value of β changes the degree of stress and constraint on the latent space during training. A greater β encourages the disentanglement, and trade-off between the reconstruction quality and the extent of disentanglement. In our experimental part, we will investigate how different β value would affect the reconstruction and classification performances.

3.2.5 Bounded information rate variational autoencoder

It was pointed out that the objective function of variational autoencoder, with the goal of maximizing the ELBO 3.11, may lead to two problems, amortized inference failures and information preference property [43–45]. Amortized inference failures, or exploding latent space problem is the case where the approximated distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is very different from the true posterior density $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. This occurs because that the ELBO can be maximized by optimizing the likelihood of the reconstruction, regardless of the KL divergence. Even though $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is inaccurate and the KL divergence is not updated, a large ELBO can still be achieved. The second problem is information preference property, which refers to the case where the KL divergence is close to zero and latent variables $\boldsymbol{z}$ and data $\boldsymbol{x}$ are independent. Thus, the latent variable $\boldsymbol{z}$ is uninformative and does not represent any information about the data $\boldsymbol{x}$. This is indicated by the cost function of VAE, where the term $\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x}))$ and $\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}))$ should be zero when ELBO is maximal, which occurs only when $\boldsymbol{z}$ is independent from $\boldsymbol{x}$.

The mutual information maximization based VAEs are proposed to alleviate the two problems pointed out above. In information theory, mutual information is a measure of the amount of information that one random variable contains about another, which quantitatively

indicates the dependency between two variables [46]. Mutual information is applied to the VAE in order to evaluate the dependency between the latent variable $\boldsymbol{z}$ and the input data $\boldsymbol{x}$. The bounded information rate VAE (BIR-VAE) [45] incorporates the maximization of mutual information between $\boldsymbol{z}$ and $\boldsymbol{x}$, intending to enforce the latent variable to learn meaningful information of the input data. This model interprets the latent space as a communication channel, where the information rate in the latent space is constrained by a pre-defined signal-to-noise ratio. In the following content, we are going to illustrate the architecture of BIR-VAE and derive its optimization function.

In VAE, encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ outputs two sets of variables describing Gaussian distribution, one is the mean and the other is the standard deviation, both of which are learned by the network. Where the architecture of BIR-VAE differs is that the encoder network of BIR-VAE only learns the mean of the distribution. While the standard deviation, instead of learned by the network, is fixed and pre-determined, denoted as ϵ, which is defined as Gaussian distributed noise, $\epsilon \sim N\left(0, \sigma_\epsilon^2\right)$. We denote the network that learns the mean $\boldsymbol{\mu}$ as $\mu_\phi(\cdot)$, the output of which is $\boldsymbol{y} = \mu_\phi(\boldsymbol{x})$. The noise ϵ is added to $\boldsymbol{y}$, giving the latent variable $\boldsymbol{z} = \boldsymbol{y} + \epsilon$. The architecture is illustrated in Figure 3.4

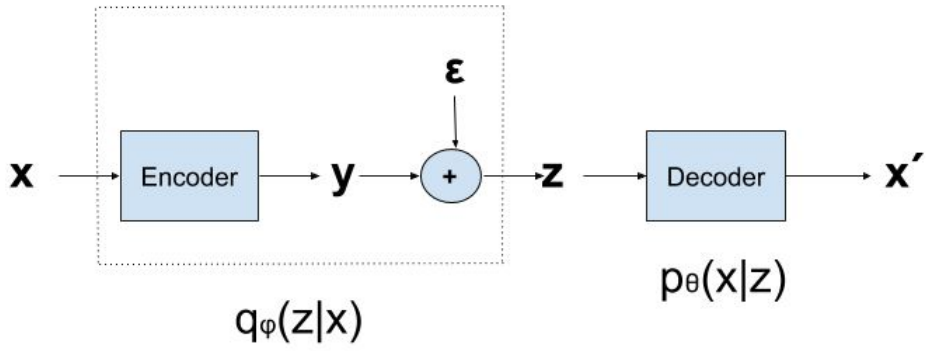


Figure 3.4: Architecture of bounded information rate variational autoencoder.

The variance of the noise ϵ satisfies the condition $\sigma_\epsilon^2 < 1$. The information rate constraint I in the latent space is determined by this variance and the dimension of the latent layer d,

$$I = \frac{d}{2}\log_2\left(\frac{1}{\sigma_\epsilon^2}\right). \tag{3.21}$$

Based on (3.21) we can set the variance of to the noise $\sigma_\epsilon^2 = 1/4^{\frac{I}{d}}$

The objective function of BIR-VAE maximizes two terms. First is the likelihood that the input can be reconstructed at the output, denoted as $\mathbb{E}_{p_D(\boldsymbol{x})}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right]$, where $p_D(\boldsymbol{x})$ is the data distribution over the input data $\boldsymbol{x}$, and $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ are the encoder and decoder respectively. The second is the mutual information between the latent variable and the input data under joint distribution $q_\phi(\boldsymbol{x},\boldsymbol{z})$ with weighing ω denoted as $\omega I_{q_\phi}(\boldsymbol{z};\boldsymbol{x})$. This objective is subjected to two constraints, one is that the distribution of the latent variable $q_\phi(\boldsymbol{z})$ is unite Gaussian $N(0,\boldsymbol{I})$, the other is that $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is Gaussian

distributed with arbitrary mean and a variance σ_ε^2 that is fixed and pre-determined. The objective function and constraints are given as

$$\begin{aligned} \max_{\phi,\theta} \quad & \mathbb{E}_{p_D(\boldsymbol{x})}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] + \omega I_{q_\phi}(\boldsymbol{x};\boldsymbol{z}) \\ \text{subject to} \quad & q_\phi(\boldsymbol{z}) = N(0,\boldsymbol{I}) \\ & \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\left(\boldsymbol{z} - \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}]\right)^2\right] = \sigma_\varepsilon^2 \boldsymbol{I} \end{aligned} \quad . \tag{3.22}$$

To transform and simplify 3.22, we introduce Maximum Mean Discrepancy (MMD) [47], which is a term that measures how different two distributions are. Given two sets of data, $\boldsymbol{x}$ and $\boldsymbol{y}$, both are sampled from two different distributions $P(x)$ and $Q(y)$ respectively, the MMD is expressed as

$$\begin{aligned} \text{MMD}(P\|Q) &= \mathbb{E}_{P(x),P(x')}\left[k\left(x,x'\right)\right] + \mathbb{E}_{Q(y),Q(y')}\left[k\left(y,y'\right)\right] - 2\mathbb{E}_{P(x),Q(y)}\left[k\left(x,y\right)\right] \\ &= \frac{1}{N_x\left(N_x-1\right)}k\left(x,x'\right) + \frac{1}{N_y\left(N_y-1\right)}k\left(y,y'\right) - \frac{2}{N_xN_y}k\left(x,y\right) \end{aligned} \tag{3.23}$$

where N_x and N_y are the number of samples in $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, the $k\left(x,x'\right)$ is denoted as kernel. The common option is a Gaussian kernel denoted as

$$k\left(x,x'\right) = e^{-\frac{\|x-x'\|}{2\sigma^2}}. \tag{3.24}$$

The MMD will be equal to zero if and only if the two distributions are the same. The equality constraint in 3.22, $q_\phi(\boldsymbol{z}) = N(0,\boldsymbol{I}) = p_\theta(\boldsymbol{z})$ is satisfied when the MMD between $q_\phi(\boldsymbol{z})$ and $N(0,\boldsymbol{I})$ is zero. Thus, transforming the equality constraint into MMD term and the optimization function 3.22 can be written into Lagrangian form

$$\begin{aligned} \max_{\phi,\theta} \quad & \mathbb{E}_{p_D(\boldsymbol{x})}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] + \omega I_{q_\phi}(\boldsymbol{x};\boldsymbol{z}) - \lambda MMD\left[q_\phi(\boldsymbol{x})\|N(0,\boldsymbol{I})\right] \\ \text{subject to} \quad & \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\left(\boldsymbol{z} - \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}]\right)^2\right] = \sigma_\varepsilon^2 \boldsymbol{I} \end{aligned} \quad . \tag{3.25}$$

Mutual information term $I_q(\boldsymbol{x};\boldsymbol{z})$ can be written into the summation of two differential entropy terms,

$$I_{q_\phi}(\boldsymbol{x};\boldsymbol{z}) = h_{q_\phi(\boldsymbol{z})}(\boldsymbol{z}) + \mathbb{E}_{p_D(\boldsymbol{x})}\left[h_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}(\boldsymbol{z})\right]. \tag{3.26}$$

The first term $h_{q_\phi(\boldsymbol{z})}(\boldsymbol{z})$ is the differential entropy of $\boldsymbol{z}$, which is fixed when the condition $q_\phi(z) = N(0,I)$ is satisfied. The second term involves the differential entropy of $\boldsymbol{z}$ given $\boldsymbol{x}$, denoted as $h_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}(\boldsymbol{z})$, which is also fixed because the distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is Gaussian

distributed with pre-determined variance σ_ε^2. Thus the mutual information $I_q(x;z)$ can be omitted from the optimization problem. The calculation of the mutual information is not necessary if the constraints are satisfied, making it computationally more efficient.

The objective function can be written as

$$\begin{aligned} \max_{\phi,\theta} \quad & \mathbb{E}_{p_D(\boldsymbol{x})}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \lambda MMD\left[q_\phi(\boldsymbol{z})\|N(0,\boldsymbol{I})\right] \\ \text{subject to} \quad & \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\left(\boldsymbol{z}-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}]\right)^2\right] = \sigma_\varepsilon^2\boldsymbol{I} \end{aligned}, \tag{3.27}$$

where we jointly optimize two terms, one is the reconstruction likelihood, which is the same as the original VAE, and the other is MMD between the latent variable distribution and the unite Gaussian. A hyper-parameter λ is introduced to make values of the first term and the second term on the same scale, under the condition that the variance of the encoder is pre-determined and fixed.

In the model of BIR-VAE, the architecture is set up to satisfy the constraint in the (3.27) as shown in Figure 3.4, where the variance of $\boldsymbol{z}$ is pre-defined. The optimization function of BIR-VAE can be finally formulated as minimizing (3.28)

$$L_{BIR-VAE}(\phi,\theta) = -\mathbb{E}_{p_D(x)}\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \lambda MMD\left[q_\phi(z)\|N(0,I)\right]. \tag{3.28}$$

3.2.6 Discussion

The architectures of VAE and its variants generally consist of two parts, one is the encoder and the other is the decoder. The encoder is an inference model that learns the latent distribution of the input and outputs latent variables at the latent space. During training, the decoder reconstructs the input based on the latent variables. The cost functions involve two terms, the first is related to the reconstruction of the input data, the second is a regularization term that enforces the latent distribution to be close to a prior distribution. VAE and β-VAE use KL divergence to measure the distance between the distribution of latent variables and prior distribution. By increasing the hyper-parameter β, more weights can be put onto the KL divergence term. BIR-VAE measures this distance by MMD and interprets latent space as a communication channel where information rate is constrained.

3.3 Summary

In this chapter, we gave a study for the theory of the neural network models that are involved in this project, including CNN and autoencoder-based models. CNN is a discriminative model that performs classification in a most straightforward manner. The autoencoder-based models, including VAE, β-VAE and BIR-VAE, are investigated regarding to their architectures and cost functions. In the classification task, the encoders can be used as the feature extractor.

4 Experimental Setups

Having studied all the theoretical backgrounds of the audio representations and the neural network models, we will set up our experiment in this chapter. We first specify the parameters and operations involved in signal processing. Then, define the architecture and hyper-parameters of our neural network models. In the end, we introduce evaluation metrics that are used to compare the networks' results.

4.1 Pre-processing

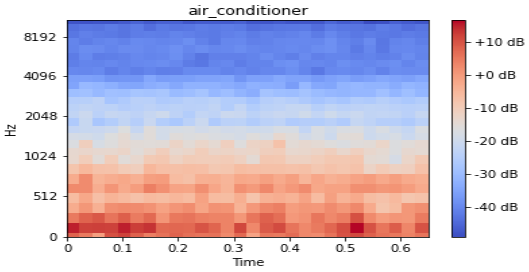
In this section, we will give parameters that are required to transform the audio sample into the desired audio representation. We use two sets of pre-processing parameters to obtain two different dimensions of the log-Mel spectrogram. One is the dimension of 28-by-28 and the other is 64-by-64. In the Table 4.1, all parameters involved in the pre-processing are listed.

Parameter	28-by-28 setup	64-by-64 setup
Sampling frequency (f_s)	22.05kHz	22.05kHz
Short-time length (N)	1024	1024
Overlap length ($N/2$)	512	512
Extract clip length (L)	0.65s	1.48s
Number of Mel filters(N_{Mel})	28	64

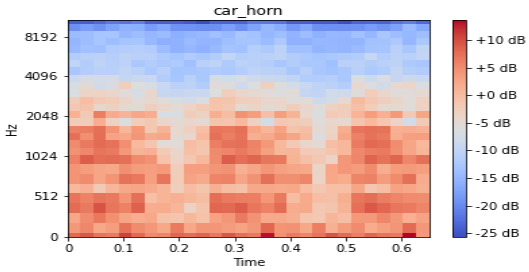
Table 4.1: Parameters in pre-processing.

The sampling frequency f_s is 22.05kHz, the number of samples in a short-time frame N is 1024 and the overlap is half of it. We extract a sub-segment from the original audio clips, the two lengths are considered, which are 0.65 seconds and 1.48 seconds. The number of short-time frame is calculated by $\left\lfloor \frac{2L}{f_s * N} \right\rfloor$. The number of Mel filters is 28 and 64 respectively.

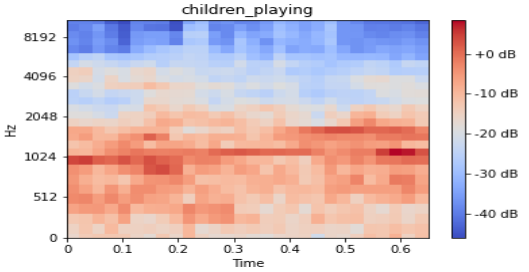
The visualization of 28-by-28-dimensional data is shown in Figure 4.1 and 64-by-64 is shown in Figure 4.2.



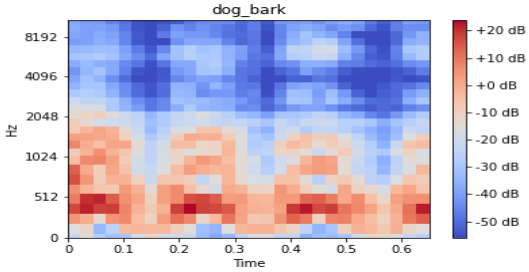
(a) Air conditioner in 28-by28 dimension



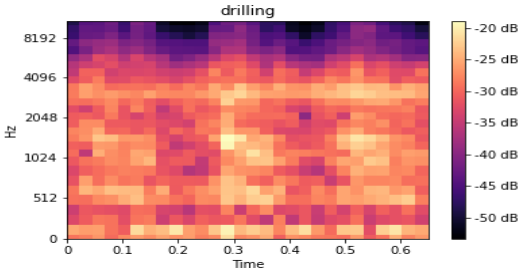
(b) Car horn in 28-by28 dimension



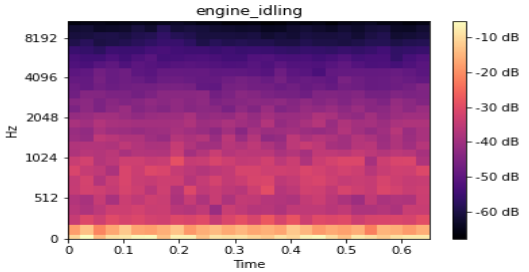
(c) Children playing in 28-by28 dimension



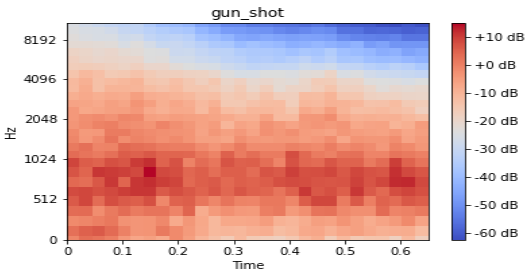
(d) Dog bark in 28-by28 dimension



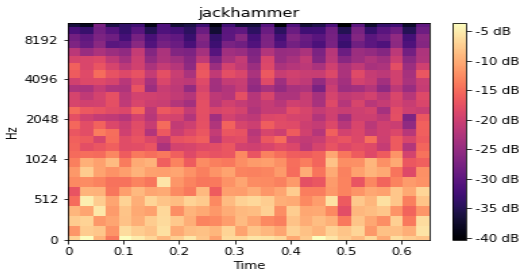
(e) Drilling in 28-by28 dimension



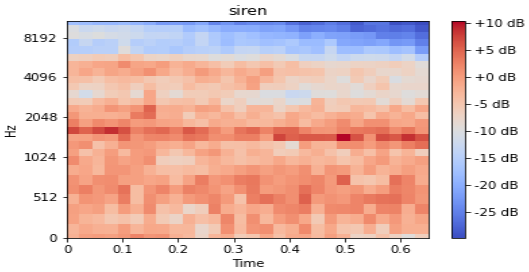
(f) Engine idling in 28-by28 dimension



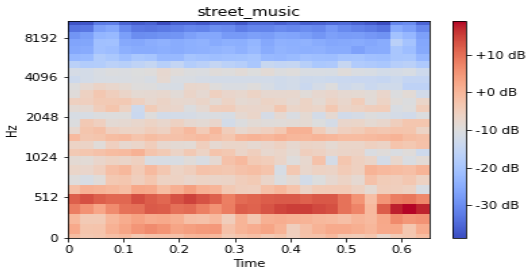
(g) Gun shot in 28-by28 dimension



(h) Jackhammer in 28-by28 dimension

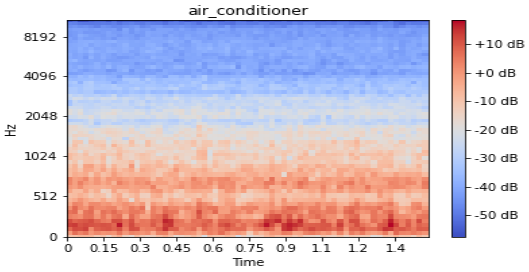


(i) Siren in 28-by28 dimension

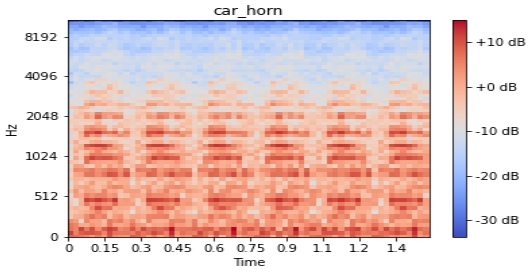


(j) Street music in 28-by28 dimension

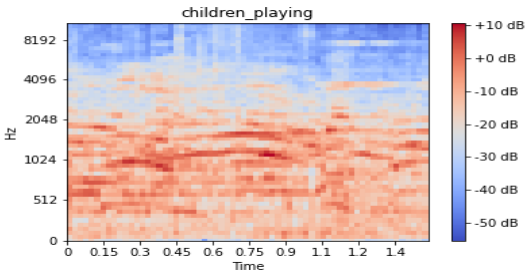
Figure 4.1: Log-Mel spectrograms in 28-by28 dimension



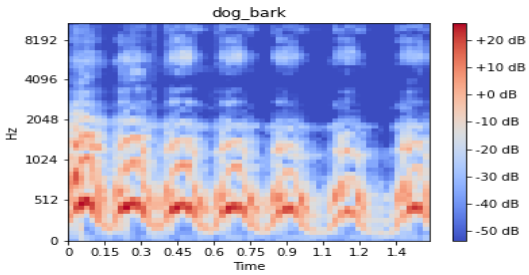
(a) Air conditioner in 64-by-64 dimension



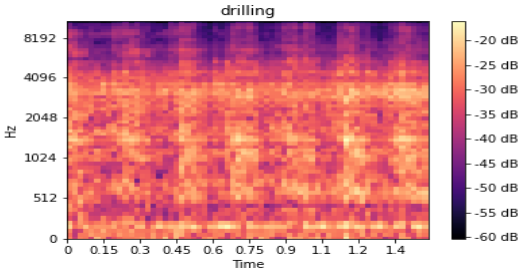
(b) Car horn in 64-by-64 dimension



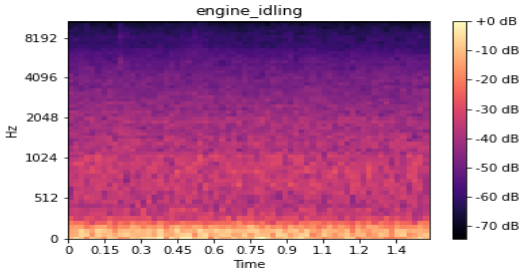
(c) Children playing in 64-by-64 dimension



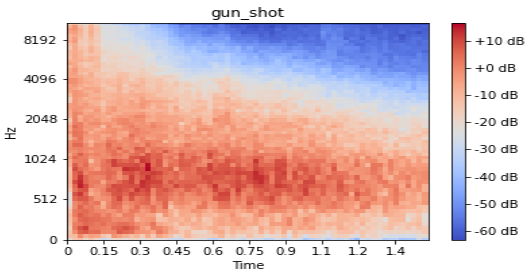
(d) Dog bark in 64-by-64 dimension



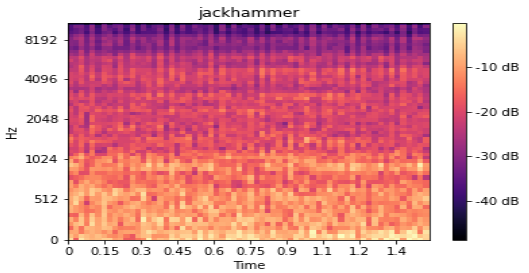
(e) Drilling in 64-by-64 dimension



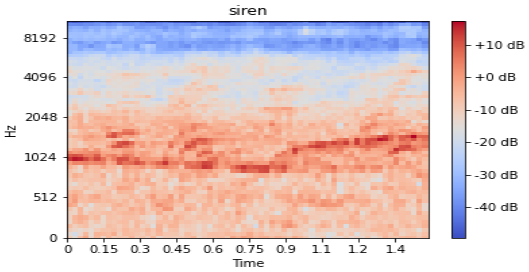
(f) Engine idling in 64-by-64 dimension



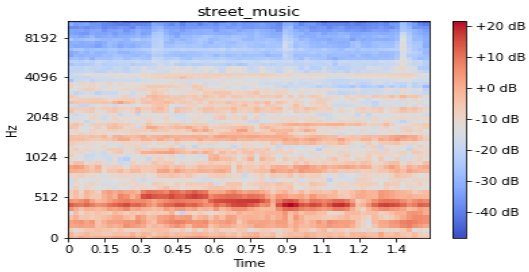
(g) Gun shot in 64-by-64 dimension



(h) Jackhammer in 64-by-64 dimension



(i) Siren in 64-by-64 dimension



(j) Street music in 64-by-64 dimension

Figure 4.2: Log-Mel spectrograms in 64-by-64 dimension

The delta feature's step is set to three. For data augmentation, the time stretching factors are 0.81, 1.23, and the pitch scaling factors are -2, 2. Our practical process on audio data is done in python, using the package Librosa [48], which is a popular tool in audio related research and practical applications. We choose fold 9 and fold 10 as our test set and the other 8 folds as training set. An overview of the number of samples in training set and test set of each class is shown in Table 4.2.

Class Name	Training set	Testing set
Air conditioner	800	200
Car horn	364	65
Children playing	800	200
Dog bark	800	200
Drilling	800	200
Engine idling	818	182
Gun shot	311	63
Jackhammer	822	178
Siren	764	165
Street music	800	200
In total	7079	1653

Table 4.2: Samples in training set and testing set.

4.2 Convolutional neural network model

Our CNN model consists of 3 convolutional layers and 2 fully connected layers. The input to the CNN model is a three-dimensional tensor, each dimension corresponds to the number of channels, height and width of the log-Mel spectrogram respectively. At the input, the number of channels is one, heights and widths are 28-by-28 or 64-by-64. As the spectrogram is a 2-dimensional representation, the convolutional and max pooling operation in the program will be 2-D. These two operations will result in a reduction of feature map, its output dimension in height and width are calculated by (3.1). The number of channels changes with the defined number of filters in each convolutional layer. The structure and parameters of our defined CNN are listed in Table 4.3.

Layer type	Parameters	28-by28 setup	64-by-64 setup
Input		(1, 28, 28)	(1, 64, 64)
Conv 2d	filter number = 64, kernel size = (3, 3), stride = 1	(64, 26, 26)	(64, 62, 62)
Activation	ReLU	(64, 26, 26)	(64, 62, 62)
Maxpooing	pooling size = (2, 2), stride = 1	(64, 13, 13)	(64, 31, 31)
Dropout	rate = 0.5	(64, 13, 13)	(64, 31, 31)
Conv 2d	filter number = 128, kernel size = (3, 3), stride = 1	(128, 11, 11)	(128, 29, 29)
Activation	ReLU	(128, 11, 11)	(128, 29, 29)
Maxpooing	pooling size = (2, 2), stride = 1	(128, 5, 5)	(128, 14, 14)
Dropout	rate = 0.5	(128, 5, 5)	(128, 14, 14)
Conv 2d	filter number = 256, kernel size = (3, 3), stride = 1	(256, 3, 3)	(256, 12, 12)
Activation	ReLU	(256, 3, 3)	(256, 12, 12)
Maxpooing	pooling size = (2, 2), stride = 1	(256, 1, 1)	(256, 6, 6)
Dropout	rate = 0.5	(256, 1, 1)	(256, 6, 6)
Flatten		256	9216
Dense	unites = 512	512	512
Activation	ReLU	512	512
Dropout	rate = 0.5	512	512
Dense	unites = 128	128	128
Activation	ReLU	128	128
Dropout	rate = 0.5	128	128
Dense	unites = 10	10	10
Activation	Softmax	10	10

Table 4.3: Parameters in convolutional neural network.

The loss function of our multi-class classification problem is the cross-entropy loss given in (3.3). The optimizer we use is ADAM [49] with a learning rate of 0.001. The batch size is 512. We initially set the number of training epochs to be 300, and add early stopping to monitor the validation loss and set the patience to be 64, which means if the validation loss doesn't improve for 64 epochs, the training process will be terminated even though the 300 epoch is not completed.

4.3 Autoencoder based model

In this section, we are going to define the architecture of the autoencoder-based models including encoder network decoder network. The encoder consists of 3 convolutional layers followed by 2 fully connected layers. The last layer's dimension is the latent dimension, which is a hyper-parameter that we will investigate in next chapter. For the two input setups, we use different parameters in the network. For larger input, we use bigger kernel size and step size such that the dimension could be efficiently reduced. The architectures and parameters of our defined encoders are listed in Table 4.4 and Table 4.5 respectively.

Layer type	Parameters	28-by28 setup
Input		(1, 28, 28)
Conv 2d	filter number = 32, kernel size = (5, 5), stride = 2	(32, 12, 12)
Batch Norm 2d	feature number = 32	(32, 12, 12)
Activation	ReLU	(32, 12, 12)
Conv 2d	filter number = 64, kernel size = (5, 5), stride = 1	(64, 8, 8)
Batch Norm 2d	feature number = 64	(64, 8, 8)
Activation	ReLU	(64, 8, 8)
Conv 2d	filter number = 128, kernel size = (3, 3), stride = 1	(128, 6, 6)
Batch Norm 2d	feature number = 128	(128, 6, 6)
Activation	ReLU	(128, 6, 6)
Flatten		4608
Dense	unites = 1024	1024
Dense	unites = latent dimension	latent dimension

Table 4.4: Parameters of encoder for 28-by-28 input.

Layer type	Parameters	64-by-64 setup
Input		(1, 64, 64)
Conv 2d	filter number = 32, kernel size = (7, 7), stride = 2	(32, 29, 29)
Batch Norm 2d	feature number = 32	(32, 29, 29)
Activation	ReLU	(32, 29, 29)
Conv 2d	filter number = 64, kernel size = (5, 5), stride = 2	(64, 13, 13)
Batch Norm 2d	feature number = 64	(64, 13, 13)
Activation	ReLU	(64, 13, 13)
Conv 2d	filter number = 128, kernel size = (5, 5), stride = 1	(128, 9, 9)
Batch Norm 2d	feature number = 128	(128, 9, 9)
Activation	ReLU	(128, 9, 9)
Flatten		10368
Dense	unites = 1024	1024
Dense	unites = latent dimension	latent dimension

Table 4.5: Parameters of encoder for 64-by-64 input.

The decoder is a reverse of the encoder, as it reconstructs the encoder's input based on encoder's output. The dimension of the latent variables input to the decoder corresponds to the latent dimension, which matches the output dimension of the encoder. The first two layers of the decoder are the fully connected layer, then the vector is reshaped to a tensor with the dimension that was flattened at encoder's fully connected layer. To reconstruct the input, the up-sampling operation called transposed convolutional operation is used. Comparing with (3.1), output dimension is calculated as (4.1)

$$
\begin{aligned}
H_{\text{out}} &= (H_{in} - 1) \times S[0] - 2 \times P[0] + K[0], \\
W_{out} &= (W_{in} - 1) \times S[1] - 2 \times P[1] + K[1].
\end{aligned}
\tag{4.1}
$$

In order to match the dimension of the input, 4 transposed convolutional layers are needed in the decoder. The operations and the resulting outputs are given in Table 4.6 and Table 4.7 respectively for two data setups.

Layer type	Parameters	28-by28 setup
Dense	unites = 1024	1024
Dense	unites = 4608	4608
Reshape		(128, 6, 6)
Transpose Conv 2d	filter number = 128, kernel size = (3, 3), stride = 1	(128, 8, 8)
Batch Norm 2d	feature number = 128	(128, 8, 8)
Activation	ReLU	(128, 8, 8)
Transpose Conv 2d	filter number = 64, kernel size = (5, 5), stride = 1	(64, 12, 12)
Batch Norm 2d	feature number = 64	(64, 12, 12)
Activation	ReLU	(64, 12, 12)
Transpose Conv 2d	filter number = 32, kernel size = (5, 5), stride = 2	(32, 27, 27)
Batch Norm 2d	feature number = 32	(32, 27, 27)
Transpose Conv 2d	filter number = 1, kernel size = (2, 2), stride = 1	(1, 28, 28)

Table 4.6: Parameters of decoder for 28-by-28 input.

Layer type	Parameters	64-by-64 setup
Dense	unites = 1024	1024
Dense	unites = 10368	10368
Reshape		(128, 9, 9)
Transpose Conv 2d	filter number = 128, kernel size = (5, 5), stride = 1	(128, 13, 13)
Batch Norm 2d	feature number = 128	(128, 13, 13)
Activation	ReLU	(128, 13, 13)
Transpose Conv 2d	filter number = 64, kernel size = (5, 5), stride = 2	(64, 29, 29)
Batch Norm 2d	feature number = 64	(64, 29, 29)
Activation	ReLU	(64, 29, 29)
Transpose Conv 2d	filter number = 32, kernel size = (7, 7), stride = 2	(32, 63, 63)
Batch Norm 2d	feature number = 32	(32, 63, 63)
Transpose Conv 2d	filter number = 1, kernel size = (2, 2), stride = 1	(1, 64, 64)

Table 4.7: Parameters of decoder for 64-by-64 input.

The optimization functions of β-VAE and BIR-VAE are expressed as (3.12) and (3.28), both functions involve minimizing the reconstruction error between the input and output of the model. In our practical implementation, this term is expressed as the mean square

error between the input and output. The regularization term of β-VAE is the KL divergence between the encoder's distribution and the prior distribution, where the prior distribution $p_\theta(\boldsymbol{z})$ is assumed to be unite Gaussian. This KL divergence term is derived as follows,

$$\begin{aligned}
\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) &= \int q_\phi(\boldsymbol{z}|\boldsymbol{x})\left(\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p_\theta(\boldsymbol{z})\right) d\boldsymbol{z} \\
&= \int q_\phi(\boldsymbol{z}|\boldsymbol{x})\left(\log q_\phi(\boldsymbol{z}|\boldsymbol{x})\right) d\boldsymbol{z} - \int q_\phi(\boldsymbol{z}|\boldsymbol{x})\left(\log p_\theta(\boldsymbol{z})\right) d\boldsymbol{z} \\
&= \int N\left(\boldsymbol{\mu},\boldsymbol{\sigma}^2\right)\log N\left(\boldsymbol{\mu},\boldsymbol{\sigma}^2\right) d\boldsymbol{z} - \int N\left(\boldsymbol{\mu},\boldsymbol{\sigma}^2\right)\log N(0,\boldsymbol{I}) d\boldsymbol{z} \\
&= -\frac{1}{2}\left(1+\log\boldsymbol{\sigma}^2\right) + \frac{1}{2}\left(\boldsymbol{\mu}^2+\boldsymbol{\sigma}^2\right) \\
&= -\frac{1}{2}\left(1+\log\boldsymbol{\sigma}^2-\boldsymbol{\mu}^2-\boldsymbol{\sigma}^2\right),
\end{aligned} \tag{4.2}$$

and implement as $\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) = -\frac{1}{2}\left(1+\log\boldsymbol{\sigma}^2-\boldsymbol{\mu}^2-\boldsymbol{\sigma}^2\right)$. The MMD is expressed as (3.23) and its kernel is (3.24), where the $\boldsymbol{\sigma}$ in the kernel is 1.

In our experiments, we will first implement β-VAE and BIR-VAE using various hyperparameter values, evaluating the their reconstruction performance by the signal-to-noise ratio (SNR), which is explained in the next subsection. To train the autoencoder-based models, the optimizer is ADAM with a learning rate of 0.0003. The batch size is 512 and we will train for 300 epochs.

After training the autoencoders, we use the trained encoder as feature extractor to perform classification. The mean learned by the encoders is used as the input to the classifier. We will use the fully connected layers as classifier. During the update, the parameters in the encoder are fixed and only the fully connected layers are updated. The classifier is trained for 100 epochs using the stochastic gradient descent (SGD) with a learning rate of 0.1. Parameters in the fully connected classifier are given in Table 4.8.

Layer type	Parameters	output shape
Dense	unites = 1024	1024
Activation	ReLU	1024
Dropout	rate = 0.5	1024
Dense	unites = 256	256
Activation	ReLU	256
Dropout	rate = 0.5	256
Dense	unites = 64	64
Activation	ReLU	64
Dropout	rate = 0.5	64
Dense	unites = 10	10
Activation	Softmax	10

Table 4.8: Parameters of fully connected classifier.

4.4 Evaluation metrics

In this section, we are going to introduce the evaluation metrics that are involved in assessing the models' performance. The classification performance is measure by accuracy, precision, recall and F1-score, the confusion matrix gives a visualization of where the misclassification occurs. The reconstruction performance is evaluated by the average SNR across the dataset. Evaluation metrics give mathematical and numerical representations of the models' performance, allowing us to compare the results straightforwardly.

4.4.1 Classification evaluation

The classification performance is usually decided by the following criterion.

- Accuracy: Accuracy is the ratio between the correct prediction and the total number of samples in the dataset, indicating how often the classifier makes a correct prediction. When the dataset has more than 2 classes, and each class has an unequal number of samples, accuracy alone is not sufficient.
- Confusion matrix: The confusion matrix visualizes the correct and incorrect predictions in each class. Each element in the matrix is denoted as c_{ij} which represents the number of ground truth in class i that were predicted to be in the class j. In our case where there are 10 classes involved in the classification, the index i and j range from 1 to 10.
- Precision: The precision of one class is calculated by the ratio between the number of correct prediction and the total number of instances that are predicted to be this class. The precision of class i is calculated by

$$\text{Precision } = \frac{c_{ii}}{\sum_{i=1}^{10} c_{ij}} \tag{4.3}$$

  A low precision indicates that other classes of data are more likely to be misclassified as this class i, the confidence of the prediction is relatively low.

- Recall: Recall is the number of correct predictions divided by the total number of the samples in that class. The recall of class i is calculated by

$$\text{Recall} = \frac{c_{ii}}{\sum_{j=1}^{10} c_{ij}} \tag{4.4}$$

  A low recall value suggests that samples in this class i usually fail to be correctly classified.

- F1-score: F1-score is the harmonic mean of the precision and recall, where an F1-score reaches its best value at 1 when the relative contribution of precision and recall to the F1-score are equal, and its worst score is at 0 when either precision or recall is 0. The F1-score is given as

$$F1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.5}$$

4.4.2 Reconstruction evaluation

To evaluate the reconstruction performance of the autoencoders, we use the signal-to-noise ratio to compare the reconstructed log-Mel spectrogram with the original one. In signal processing and communication, SNR is used to compare the level of a desired signal with the level of noise, which is defined as the ratio of signal power to the noise power and often expressed in decibels.

We will quantitatively evaluate the reconstruction performance by the autoencoder-based models in a perspective of communication and signal processing. Assume we have a dataset $\mathbb{X}$ containing a number of N spectrograms, each denoted as $\boldsymbol{X}_i$. One spectrogram passes through the autoencoder and its reconstruction is denoted as $\hat{\boldsymbol{X}}_i$. The difference between the original and reconstruction is consider as noise denoted as $\boldsymbol{N}_i$, this can be expressed as Equation 4.6,

$$\hat{\boldsymbol{X}}_i = \boldsymbol{X}_i + \boldsymbol{N}_i. \tag{4.6}$$

Both the signal and noise are 2-dimensional matrices, we will use the square of Frobenius norm to represent the power. A matrix $\boldsymbol{M}$, with a dimension of (m,n) and each element is denoted as m_{ij}, its Frobenius norm is calculated as

$$\|M\|_{\text{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \left|m_{ij}\right|^2} \tag{4.7}$$

with the given definition, we obtain the power of the original spectrogram and the noise by,

$$\begin{aligned} P_{\boldsymbol{X}_i} &= \|\boldsymbol{X}_i\|_{\text{F}}^2, \\ P_{\boldsymbol{N}_i} &= \|\boldsymbol{N}_i\|_{\text{F}}^2. \end{aligned} \tag{4.8}$$

and represent the SNR in decibel form

$$SNR_i^{dB} = 10\log_{10}\frac{P_{\boldsymbol{X}_i}}{P_{\boldsymbol{N}_i}}. \tag{4.9}$$

To evaluate the reconstruction performance throughout the entire dataset $\mathbb{X}$, we take the average SNR

$$SNR^{dB} = \frac{1}{N}\sum_{i=1}^{N} SNR_i^{dB}. \tag{4.10}$$

4.5 Summary

In this chapter, we pre-processed the audio data and obtained the desired feature representation in two different dimensions. Then, we built up the neural network and define the training process. The evaluation metrics to quantitatively compare classification and reconstruction results were studied.

5 Results

In previous chapters, we have introduced the basic concepts and gave the experimental setup. In this chapter, the outcomes of the experiments will be reported and illustrated. We first train the convolutional neural network model using the log-Mel spectrogram and delta features. Then, autoencoder-based models are implemented with different hyper-parameters. The results include the values of evaluation metrics, and comparative visualizations between the original spectrograms and reconstructions are shown.

5.1 Convolutional Neural Networks models

In this section, we will describe the performance of the CNN model defined in 4.3. We first use the log-Mel spectrogram. Then, we further investigate how the delta features and the data augmentation affect classification. To train the CNN using the log-Mel spectrogram and its delta features, we add an extra input channel and combine the two features, making the CNN's input into a 2-channel image.

5.1.1 Results

The classification results of CNN using 28-by-28-dimensional and 64-by-64-dimensional inputs are listed in Table 5.1 and Table 5.2 respectively.

	Original data	Original data + delta	Original data + aug
Train accuracy	92.64%	88.13%	84.44%
Test accuracy	68.24%	68.91%	67.17%

Table 5.1: Classification accuracy of CNN on 28-by-28 input.

	Original data	Original data + delta	Original data + aug
Train accuracy	99.10%	98.46%	91.80%
Test accuracy	75.20%	75.26%	67.80%

Table 5.2: Classification accuracy of CNN on 64-by-64 input.

The accuracy of using a 64-by-64-dimensional log-Mel spectrogram is 75.20%. We give the evaluation metrics based on this result. The confusion matrix is shown in Figure 5.1 and the classification reports containing precision, recall and F1-score values are given in Table 5.3

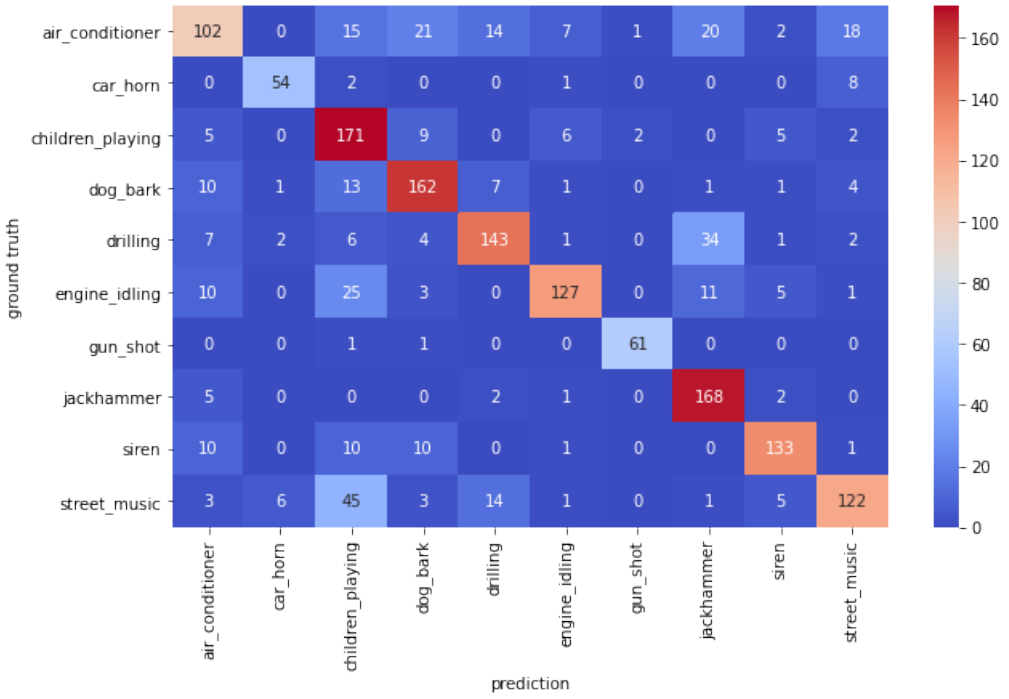


Figure 5.1: Confusion matrix of CNN.

Class name	precision	recall	F1-score	support
air conditioner	0.67	0.51	0.58	200
car horn	0.86	0.83	0.84	65
children playing	0.59	0.85	0.70	200
dog bark	0.76	0.81	0.78	200
drilling	0.79	0.71	0.75	200
engine idling	0.87	0.70	0.77	182
gun shot	0.95	0.97	0.96	63
jackhammer	0.71	0.94	0.81	178
siren	0.86	0.81	0.83	165
street music	0.77	0.61	0.68	200

Table 5.3: Classification report of CNN.

5.1.2 Discussion

By comparing the overall accuracy, we find that the larger dimensional input produces a higher accuracy. For the same CNN model, a test accuracy of 68.24% was obtained using 28-by-28-dimensional input and 75.20% using 64-by-64-dimensional. Hence the improvement was 7% higher. The results obtained from 2-channel input show a slight improvement, while the data augmentation did not improve the accuracy. The best classification accuracy obtained in this experiment is 75.26% by CNN using the combination of the 64-by-64-dimensional log-Mel spectrogram and its delta features.

From the classification report, we can find that the lowest precision comes from the children playing: the confusion matrix shows that street music has the most samples misclassified as children playing. The lowest recall values are from the classes of air conditioner and street music: samples of air conditioner are mostly misclassified as dog bark, jackhammer and street music. The air conditioner has the worst F1-score, which is the most difficult class to identify.

5.2 Autoencoder-based models

In this section, we we will discuss the experiment of autoencoder-based models. We investigate the reconstruction and classification performances by implementing a set of different hyper-parameters. The values of evaluation metrics are given, and the comparison between the original spectrograms and their reconstructions are visualized.

5.2.1 Variational Autoencoder and β-Variational Autoencoder

We now discuss different latent dimensions based on the given architectures of encoders and decoders, and set different β values in optimization function (3.20). The results of reconstruction and classification on 28-by-28-dimensional input are given in Table 5.4.

The best reconstruction performance with the highest SNR is 17.36 dB, achieved by the VAE model with latent dimension 128. The model with 256 latent dimension and β value of 10 gives the best performance in classification, achieving an accuracy of 64.85% on the test set.

Latent dimension	β	Train SNR (dB)	Test SNR (dB)	Train accuracy	Test accuracy
16	1	15.88	14.51	80.94%	56.99%
32	1	17.68	16.08	88.56%	60.19%
64	1	18.41	16.96	95.23%	63.40%
128	1	18.50	**17.36**	97.46%	63.46%
256	1	18.30	16.84	98.30%	61.58%
512	1	17.96	16.83	98.62%	61.95%
64	10	17.94	16.75	95.56%	62.86%
128	10	18.47	16.85	97.84%	64.73%
256	10	17.74	16.32	99.01%	**64.85%**
64	100	16.73	15.45	92.98%	63.10%
128	100	16.80	15.48	94.35%	62.25%
256	100	16.01	14.85	95.82%	61.34%

Table 5.4: Results of VAE and β-VAE on 28-by-28 input.

Next, we investigate the performance of VAE and β-VAE using 64-by-64-dimensional log-Mel spectrogram. The result are shown in Table 5.5. The model with $\beta = 10$ and a latent dimension of 256 gives the best SNR of 15.85 dB and an accuracy of 65.34%.

Latent dimension	β	Train SNR (dB)	Test SNR (dB)	Train accuracy	Test accuracy
128	1	16.52	15.18	97.20%	59.89%
128	10	16.87	15.53	98.41%	62.55%
256	1	16.11	14.82	98.88%	62.67%
256	10	17.12	**15.85**	99.43%	**65.34%**
512	10	16.36	14.76	99.01%	62.61%

Table 5.5: Results of VAE and β-VAE on 64-by-64 input.

5.2.2 Bounded Information Rate Variational Autoencoder

The optimization function of BIR-VAE is given as (3.28). We fix the hyper-parameter $\lambda = 10^6$ to balance the values of reconstruction loss and regularization term, and investigate the effect of latent dimension and information rate, which are denoted as d and I respectively in the table and σ is calculated by (3.21). We first implement the BIR-VAE using 28-by-28-dimensional input, the SNR and accuracy are listed in Table 5.6.

The best reconstruction performance is given by the BIR-VAE with latent dimension of 128, an information rate of 256 with $\sigma = 0.25$, the SNR is 17.44 dB. The highest accuracy is 62.61%, given by the BIR-VAE with a latent dimension of 256 and information rate of 1024 with $\sigma = 0.06$.

d	I	σ	Train SNR (dB)	Test SNR (dB)	Train accuracy	Test accuracy
64	32	0.7	18.23	16.73	90.62%	59.71%
64	64	0.5	17.51	16.33	91.35%	61.34%
64	128	0.25	17.60	16.27	92.26%	60.62%
64	256	0.06	17.86	16.57	91.72%	60.74%
128	64	0.7	17.13	15.84	92.97%	58.98%
128	128	0.5	17.66	16.27	93.49%	60.56%
128	256	0.25	19.12	**17.44**	94.87%	62.13%
128	512	0.06	18.42	17.11	95.34%	61.95%
256	128	0.7	17.68	16.51	94.05%	59.77%
256	256	0.5	17.01	15.54	94.14%	60.38%
256	512	0.25	18.68	17.17	95.55%	62.19%
256	1024	0.06	17.52	16.47	96.45%	**62.61%**

Table 5.6: Results of BIR-VAE on 28-by-28 input.

We also implemented the the BIR-VAE on 64-by-64-dimensional input. The classification accuracy and the SNR are given in Table 5.7. We can find that the model with a latent dimensionality of 128 and information rate of 256 achieved the best accuracy by 62.19% and highest SNR by 15.74 dB.

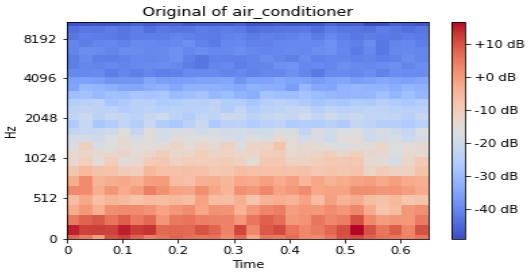
d	I	σ	Train SNR (dB)	Test SNR (dB)	Train accuracy	Test accuracy
128	128	0.5	16.72	15.55	97.10%	61.40%
128	256	0.25	17.11	**15.74**	97.39%	**62.19%**
128	512	0.06	16.56	15.35	98.26%	61.96%
256	1024	0.06	16.81	15.61	98.26%	61.52%

Table 5.7: Results of BIR-VAE on 64-by-64 input.

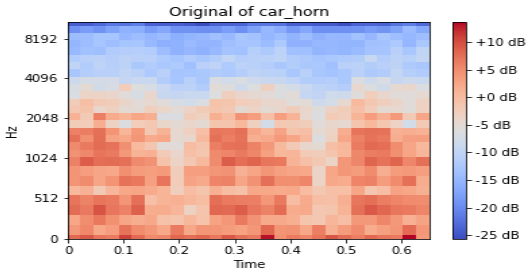
5.2.3 Visualizations of results

In this part, based on the best reconstruction performance, we provide a visual comparison of the reconstructed log-Mel spectrograms and the original ones. Also, we will give the values of evaluation metrics of the result with the highest accuracy.

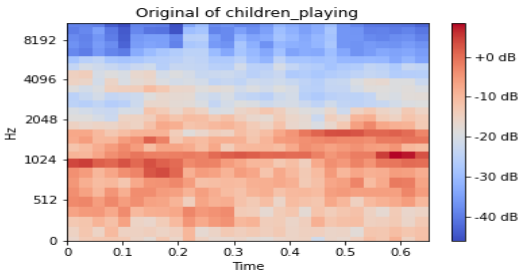
The best reconstruction performance on 28-by-28-dimensional log-Mel spectrograms is obtained by BIR-VAE, with latent dimension of 128 and information rate of 256. An SNR of 19.12 dB is achieved on the training set, the visualizations comparing them are Figure 5.2, which is the original spectrograms, and the reconstructed ones are shown in Figure 5.3. The test set is reconstructed with an SNR of 17.44 dB. Original samples in test set are illustrated in Figure 5.4, their reconstructions are shown in Figure 5.5. We also use the decoder of this model to generate new samples, which are shown as Figure B.1 in Appendix B.



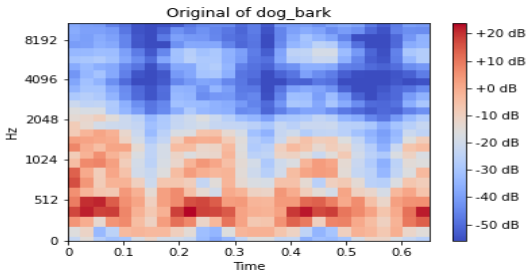
(a) Original of air conditioner



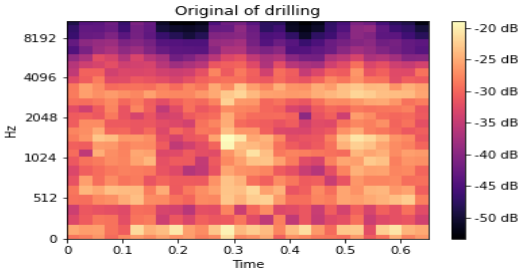
(b) Original of car horn



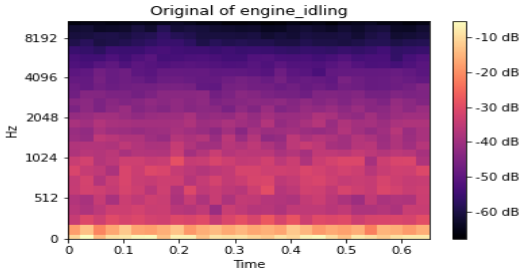
(c) Original of children playing



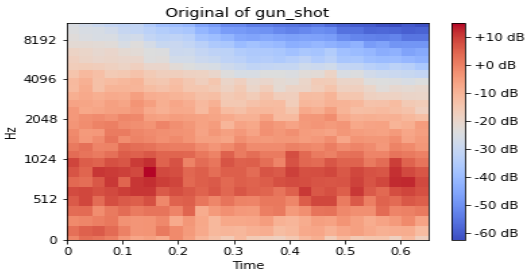
(d) Original of dog bark



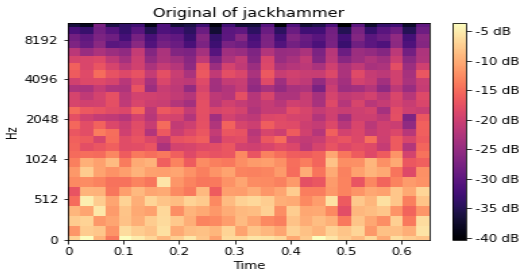
(e) Original of drilling



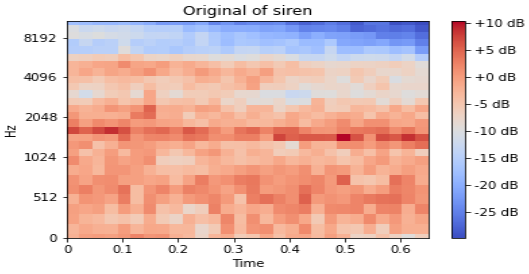
(f) Original of engine idling



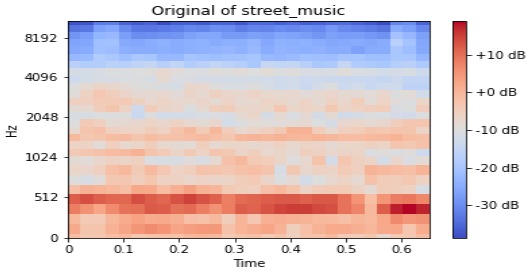
(g) Original of gun shot



(h) Original of jackhammer

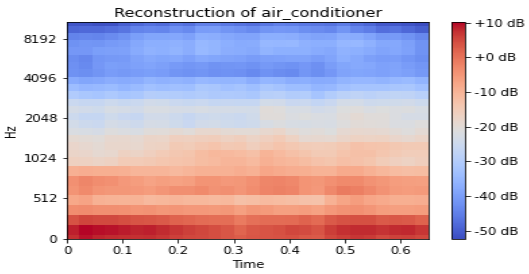


(i) Original of siren

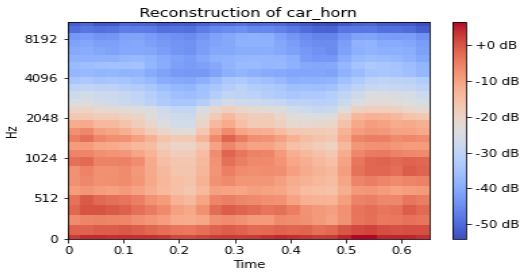


(j) Original of street music

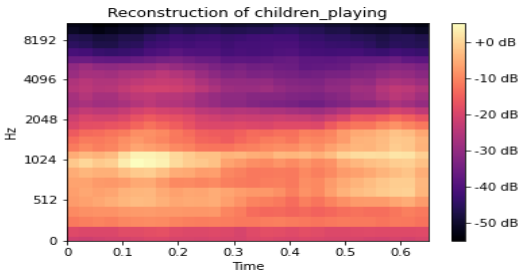
Figure 5.2: Original 28-by-28 log-Mel spectrograms in training set



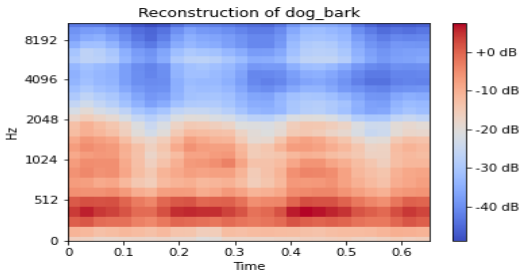
(a) Reconstruction of air conditioner



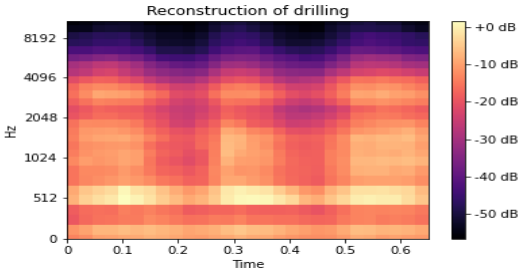
(b) Reconstruction of car horn



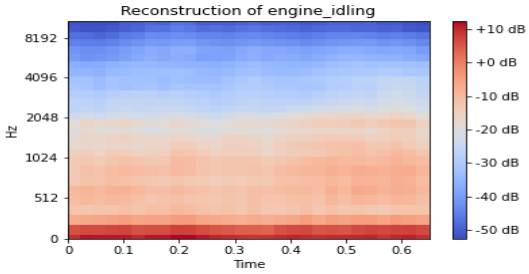
(c) Reconstruction of children playing



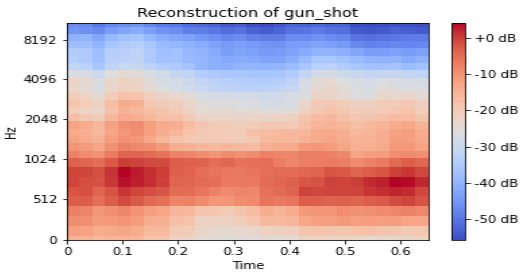
(d) Reconstruction of dog bark



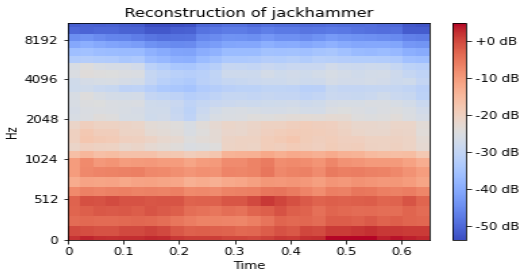
(e) Reconstruction of drilling



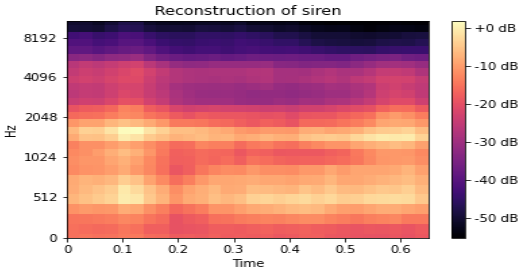
(f) Reconstruction of engine idling



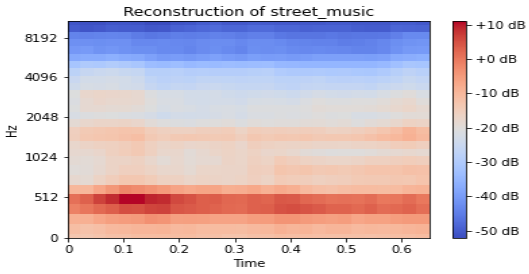
(g) Reconstruction of gun shot



(h) Reconstruction of jackhammer

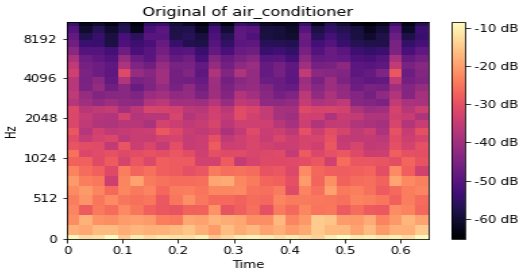


(i) Reconstruction of siren

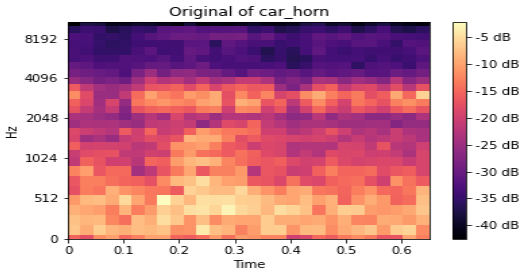


(j) Reconstruction of street music

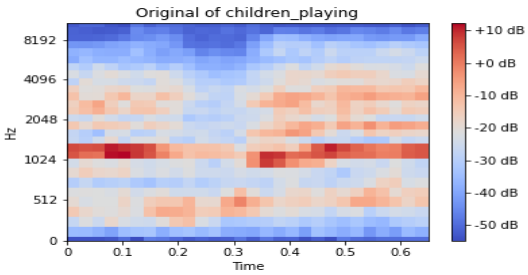
Figure 5.3: Reconstruction of 28-by-28 log-Mel spectrograms in training set with SNR = 19.12 dB



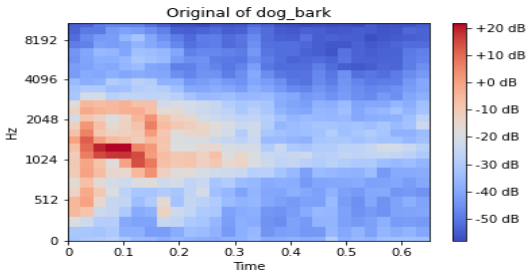
(a) Original of air conditioner



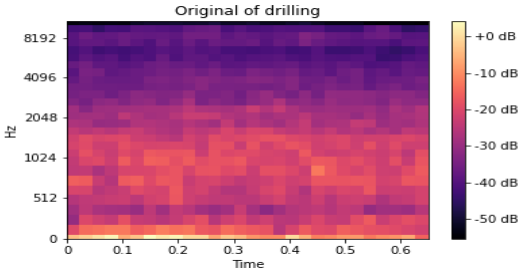
(b) Original of car horn



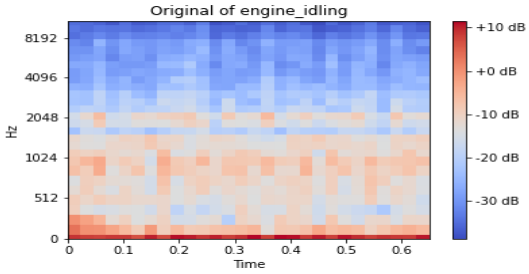
(c) Original of children playing



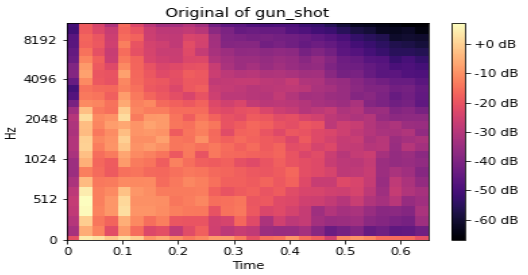
(d) Original of dog bark



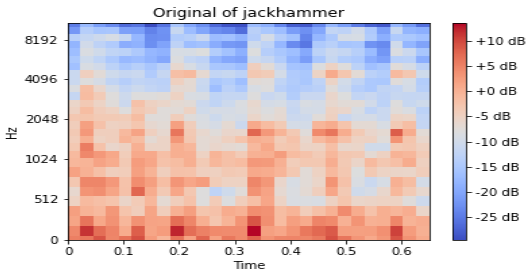
(e) Original of drilling



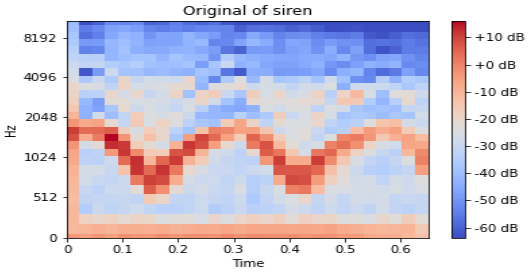
(f) Original of engine idling



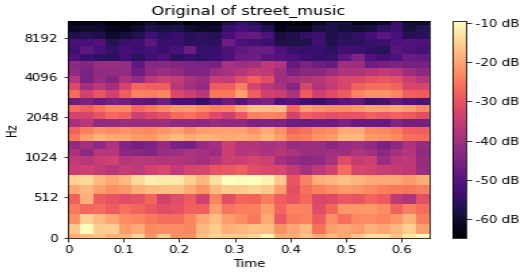
(g) Original of gun shot



(h) Original of jackhammer

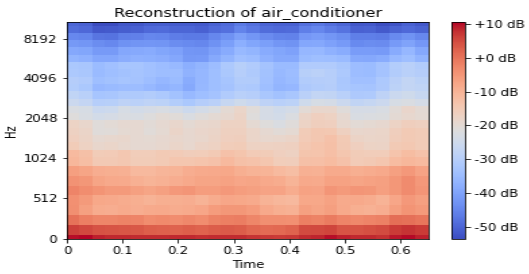


(i) Original of siren

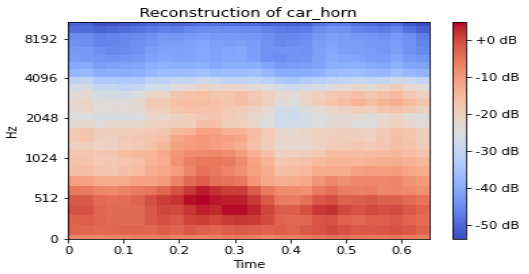


(j) Original of street music

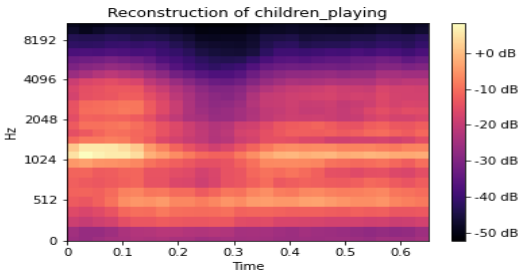
Figure 5.4: Original 28-by-28 log-Mel spectrograms in test set



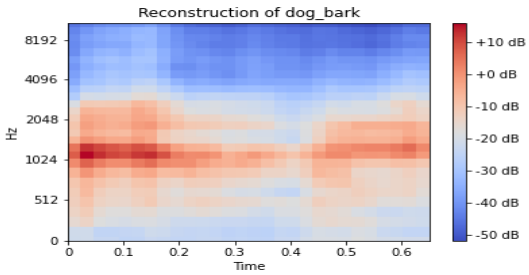
(a) Reconstruction of air conditioner



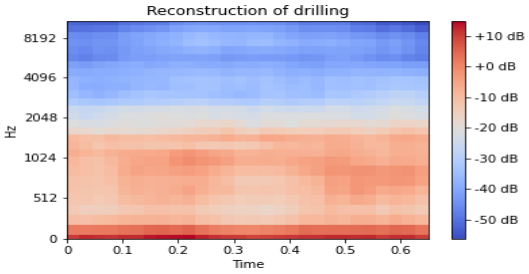
(b) Reconstruction of car horn



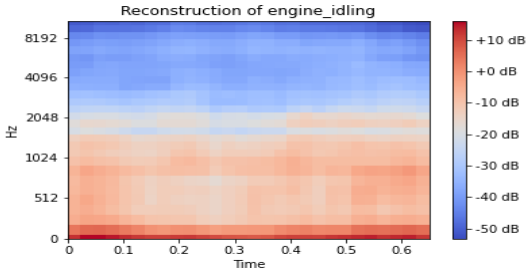
(c) Reconstruction of children playing



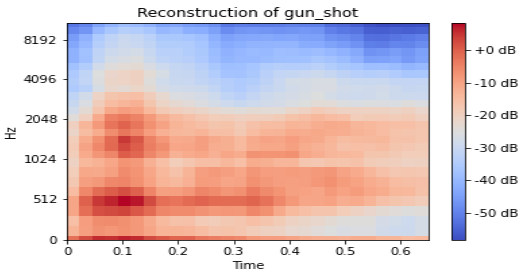
(d) Reconstruction of dog bark



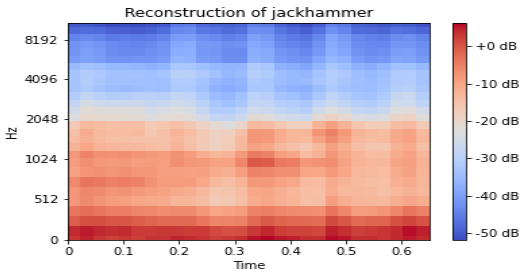
(e) Reconstruction of drilling



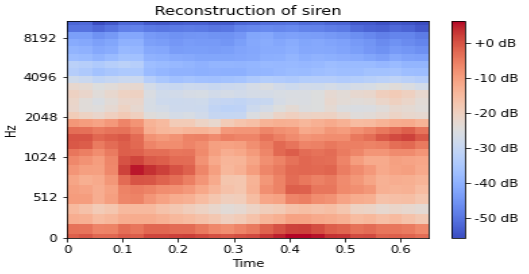
(f) Reconstruction of engine idling



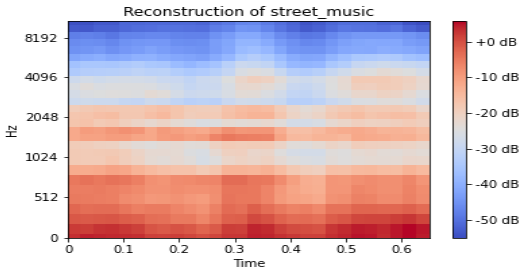
(g) Reconstruction of gun shot



(h) Reconstruction of jackhammer



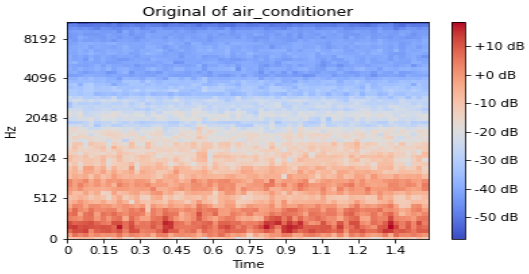
(i) Reconstruction of siren



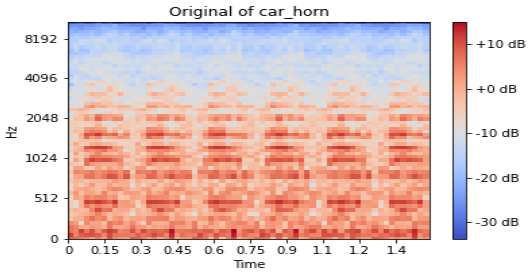
(j) Reconstruction of street music

Figure 5.5: Reconstruction of 28-by-28 log-Mel spectrograms in test set with SNR = 17.44 dB

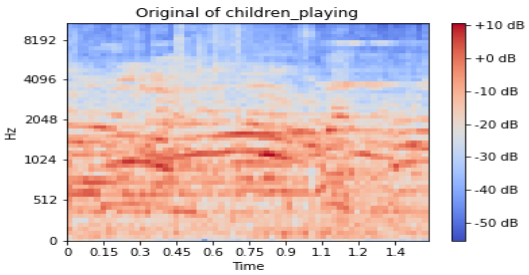
For input in size 64-by-64, the highest SNR is given by BIR-VAE with latent dimension of 128 and information rate of 256. SNR of 17.11 dB is achieved on the training set and 15.74 dB on the test set. Original 64-by-64-dimensional spectrograms in training set and test set are shown in Figure 5.6 and Figure 5.8 respectively. Their reconstructions are illustrated in Figure 5.7 and Figure 5.9. New samples with dimension of 64-by-64 are generated by the decoder of this model, which are shown as Figure B.2 in Appendix B.



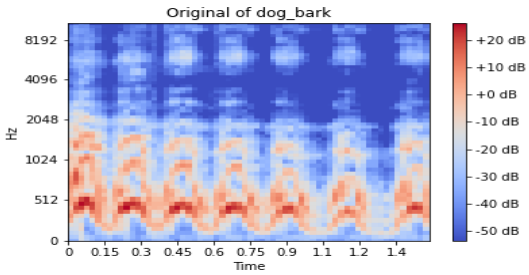
(a) Original of air conditioner



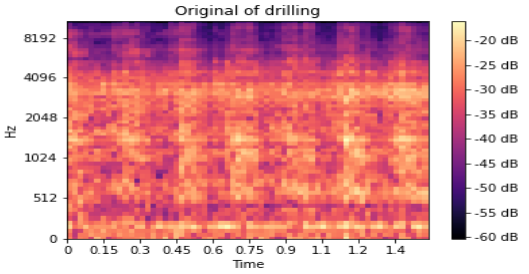
(b) Original of car horn



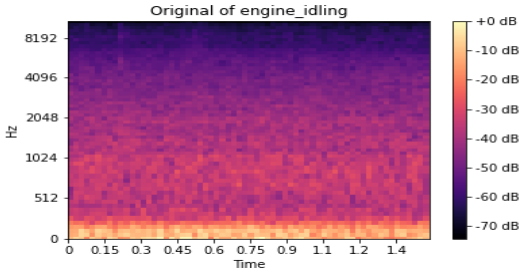
(c) Original of children playing



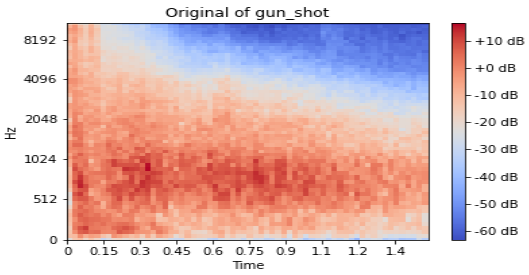
(d) Original of dog bark



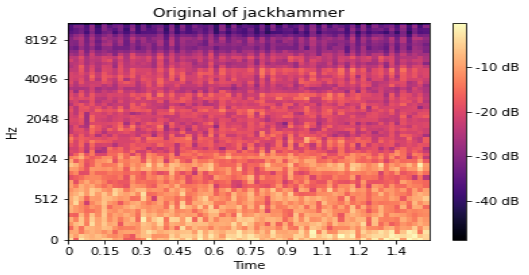
(e) Original of drilling



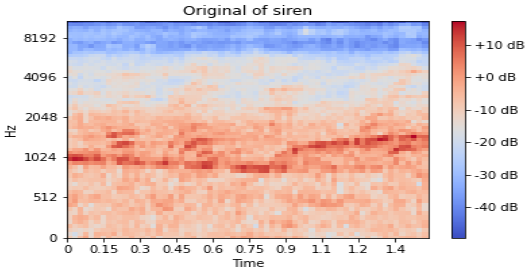
(f) Original of engine idling



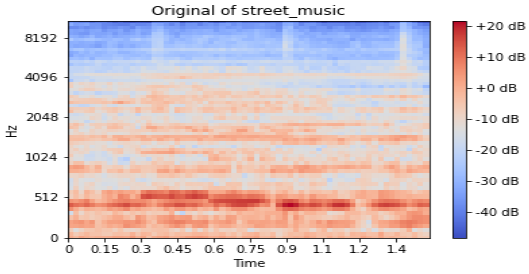
(g) Original of gun shot



(h) Original of jackhammer

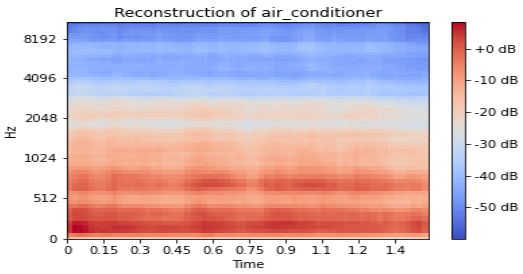


(i) Original of siren

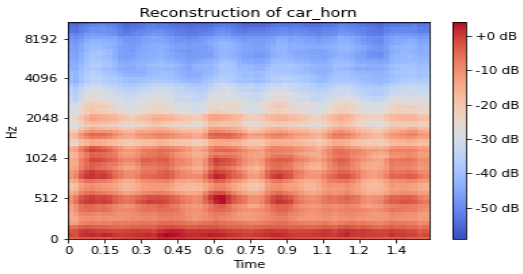


(j) Original of street music

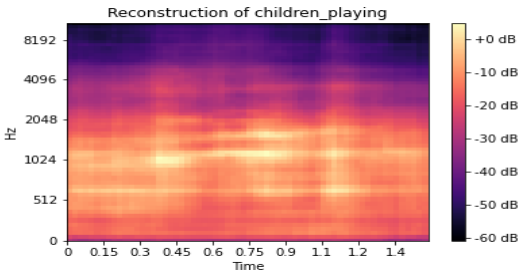
Figure 5.6: Original 64-by-64 log-Mel spectrograms in training set



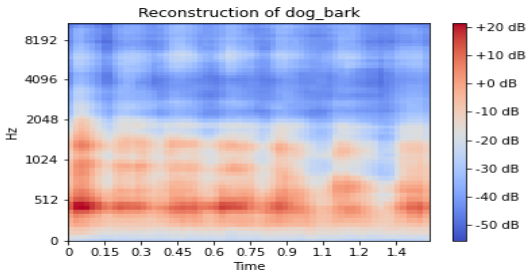
(a) Reconstruction of air conditioner



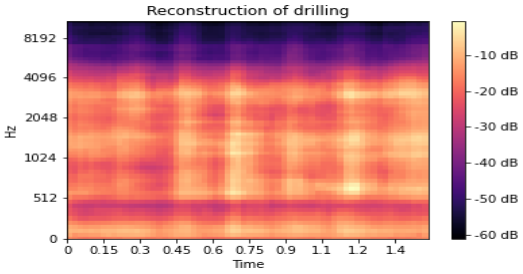
(b) Reconstruction of car horn



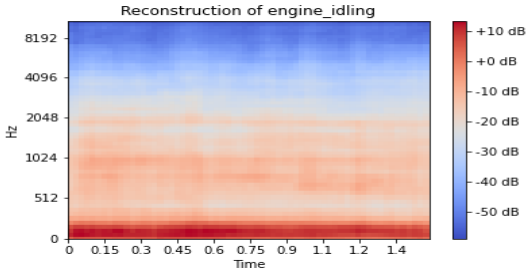
(c) Reconstruction of children playing



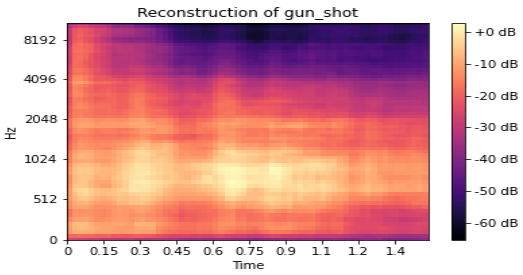
(d) Reconstruction of dog bark



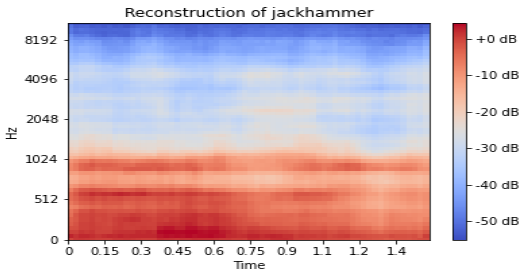
(e) Reconstruction of drilling



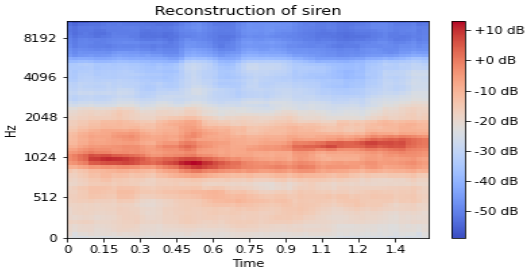
(f) Reconstruction of engine idling



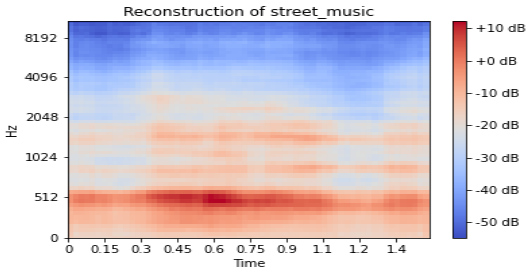
(g) Reconstruction of gun shot



(h) Reconstruction of jackhammer

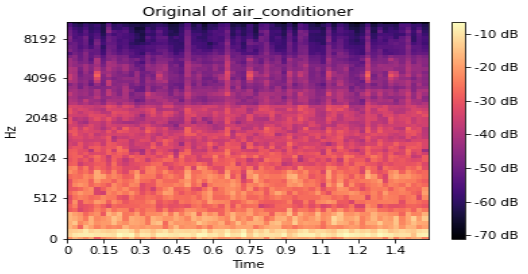


(i) Reconstruction of siren

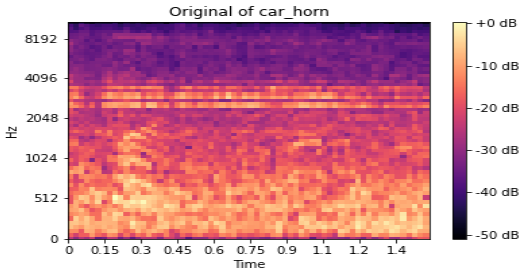


(j) Reconstruction of street music

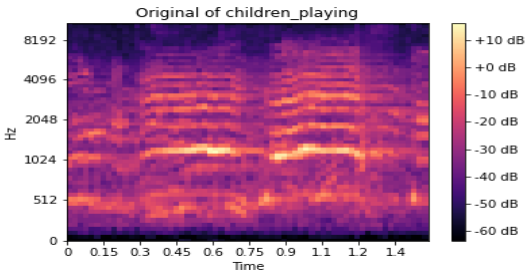
Figure 5.7: Reconstruction of 64-by-64 log-Mel spectrograms in training set with SNR = 17.11 dB



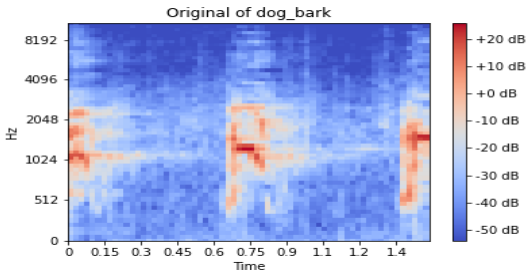
(a) Original of air conditioner



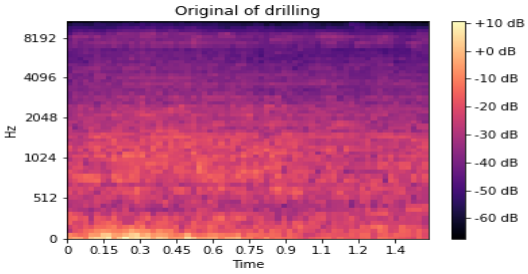
(b) Original of car horn



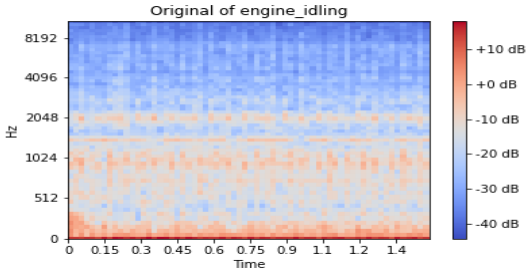
(c) Original of children playing



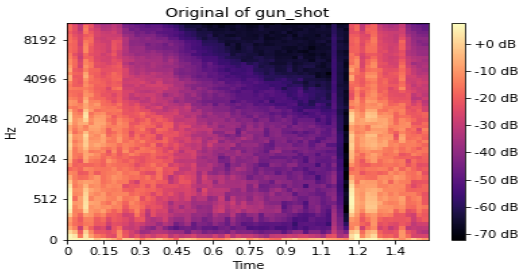
(d) Original of dog bark



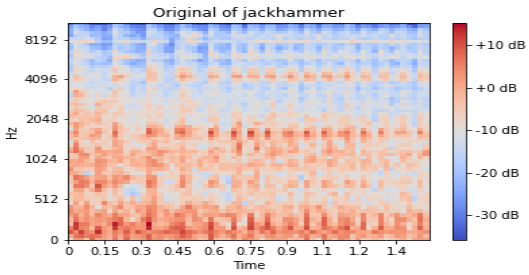
(e) Original of drilling



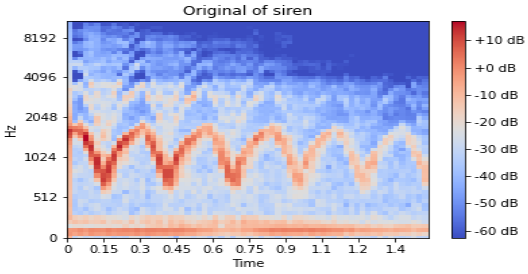
(f) Original of engine idling



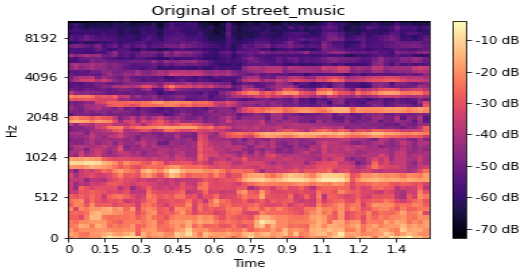
(g) Original of gun shot



(h) Original of jackhammer

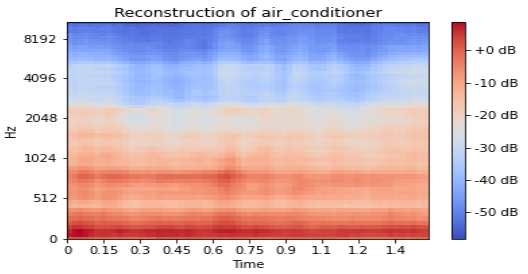


(i) Original of siren

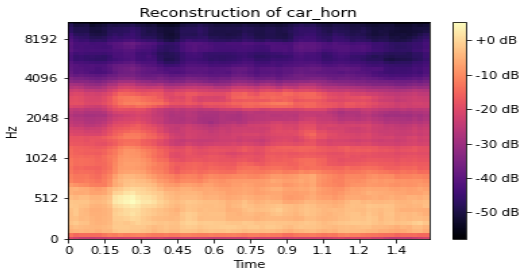


(j) Original of street music

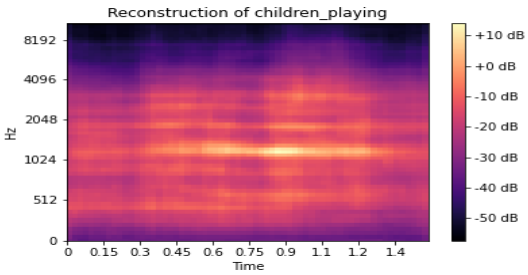
Figure 5.8: Original 64-by-64 log-Mel spectrograms in test set



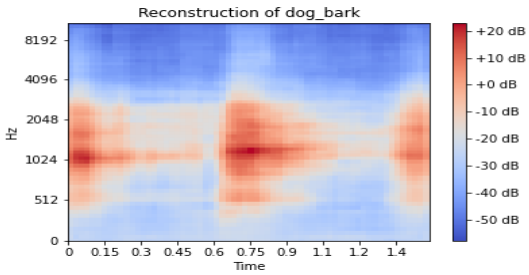
(a) Reconstruction of air conditioner



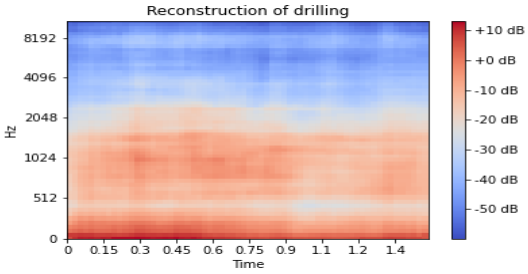
(b) Reconstruction of car horn



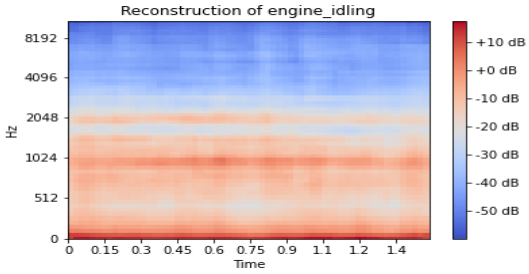
(c) Reconstruction of children playing



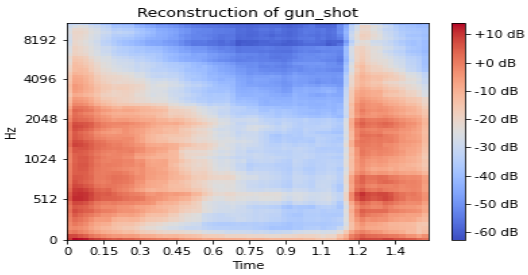
(d) Reconstruction of dog bark



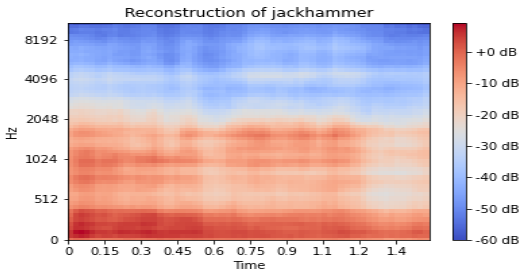
(e) Reconstruction of drilling



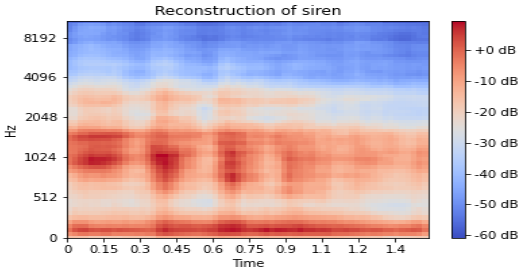
(f) Reconstruction of engine idling



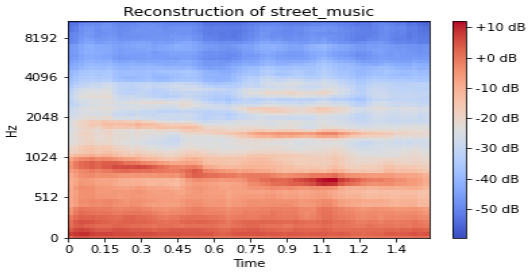
(g) Reconstruction of gun shot



(h) Reconstruction of jackhammer



(i) Reconstruction of siren



(j) Reconstruction of street music

Figure 5.9: Reconstruction of 64-by-64 log-Mel spectrograms in test set with SNR = 15.74 dB

The highest accuracy is 65.34%, achieved by the β-VAE with 256 latent dimension and a β value of 10 using 64-by-64 sized input. We give its confusion matrix in Figure 5.10 and classification report in Table 5.8. Using the decoder of this model, new spectrograms with dimension of 64-by-64 are generated, which are shown as Figure B.3 in Appendix B.

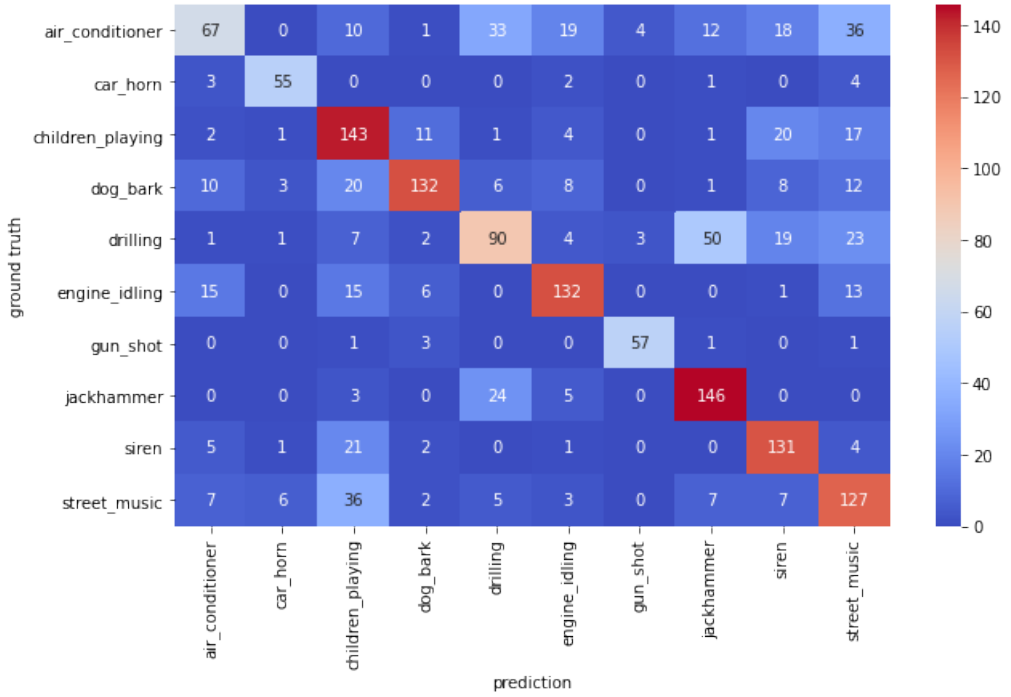


Figure 5.10: Confusion matrix of autoencoder-based model

Class name	precision	recall	F1-score	support
air conditioner	0.61	0.34	0.43	200
car horn	0.82	0.85	0.83	65
children playing	0.56	0.71	0.63	200
dog bark	0.83	0.66	0.74	200
drilling	0.57	0.45	0.50	200
engine idling	0.74	0.73	0.73	182
gun shot	0.89	0.90	0.90	63
jackhammer	0.67	0.82	0.74	178
siren	0.64	0.79	0.71	165
street music	0.54	0.64	0.58	200

Table 5.8: Classification report of autoencoder-based model.

5.2.4 Discussion

The reconstruction performances of autoencoder-based models using 28-by-28-dimensional input reached the highest SNR of 17.44 dB. This result came from the BIR-VAE with 128 latent dimension and an information rate of 256. β-VAEs obtained the highest SNR of 17.36 dB, which is competitive to the best result from BIR-VAE. When autoencoder-based models are implemented with 64-by-64-dimensional spectrograms under the same hyper-parameters, the reconstruction performances degraded. When reconstructing the spectrograms with size 64-by-64, the best achieved SNR was 15.85 dB. Observing the reconstructed images and compare them with the original ones, we can find that the reconstructions looked smoother and lower in resolution and some details may be lost, but the basic structure and pattern of the spectrograms are retained.

The highest accuracy achieved by autoencoder-based models was 65.34%. This result came from the β-VAE with latent dimension of 256 and a β value of 10 using spectrograms of size 64-by-64. Compared to the accuracy 75.20% obtained by the CNN model using the same input, this result is lower. A best result obtained by autoencoder-based model using spectrogram in 28-by-28 dimension was 64.85%, compared with an accuracy of 68.24% achieved by CNN model using the same data, the result of autoencoder-based model is 3.4% lower but comparable. Reviewing the confusion matrices and classification reports, we can discover the lowest precision value was scored by the class of street music, while the lowest recall value was obtained by air conditioner.

5.3 Summary

In this chapter, we have conducted the experiments of sound classification based on CNN and autoencoder-based models for different input representations. We first implemented CNN to perform classification on log-Mel spectrogram. The best accuracy obtained by CNN was 75.26% using 64-by-64-dimensional log-Mel spectrogram combined with its delta features, while the CNN using only the log-Mel spectrogram with the same dimension achieved 75.20%, indicating that the choice of input providing extra informative features may improve the classification performance. Compared with the results using the spectrogram of size 28-by-28, the accuracy achieved by CNN was 68.24%, which was 7% lower than that using 64-by-64 sized input, suggesting that the spectrogram with more Mel filter banks and longer audio segment may provide more information about the audio and contribute to higher accuracy. The best accuracy achieved by autoencoder-based models was 65.34% and the best reconstruction performance was 17.44 dB in average SNR. Comparative visualizations between original and reconstructed spectrograms were shown. This comparison showed that the fundamental pattern and the power distribution in the spectrogram can be reconstructed, but at a lower resolution and with some details lost. In general, the CNN yielded higher accuracy than autoencoder-based models, thus CNN is the most promising model to classify the spectrograms in our setup. The encoders were able to learn meaningful features about the spectrograms and classification could be realized.

6 Conclusion and Future Work

6.1 Conclusion

In this project, we studied the task of urban sound classification and identification using different deep learning-based methods. The feature of urban sound that we mainly used was the log-Mel spectrogram as it aligns with human's auditory system where the perception is much more sensitive for lower frequencies. The Mel-scaled spectrogram highlighted variations in lower frequencies. Other inputs such as delta features and augmented data were also considered. Several neural network models were implemented to achieve classification of urban sounds based on audio representations. The models involved in this task were CNN and autoencoder-based models. CNN is the most straightforward and efficient model to perform classification tasks. Autoencoder-based models included VAE, β-VAE and BIR-VAE, which were trained to reconstruct the original input spectrogram at their output and learn a low-dimensional feature set in the latent space. After finishing the training process of an autoencoder-based model, the encoder had learned meaningful features about the spectrogram and a fully connected classifier was applied to achieve classification based on the latent features learned by the encoder. To have knowledge on the performances of different models, we adopted evaluation metrics to evaluate the results with respect to classification and reconstruction. The classification was evaluated by accuracy in the beginning, in addition a confusion matrix gives a visualization of where misrecognition occurs. A classification report includes precision, recall and F1-score. The average SNR was used to evaluate the reconstruction performance of autoencoder-based models in a communication and signal processing perspective, and visualizations of comparison between the original spectrogram and their reconstructed ones were also provided.

In our experiments, we first conducted pre-processing on audio samples, the audio samples were processed into the log-Mel spectrograms in a consistent dimension. We considered two spectrum sizes: the first was 28-by-28 the second was 64-by-64. These dimensions indicate 28 or 64 Mel filters were applied to the spectrogram, and the lengths of sub-segment extracted from original audio were 0.65 seconds and 1.48 seconds respectively. Then, we defined the architecture of neural network models including CNN, encoders and decoders of the autoencoder-based models. Both the CNN and the encoder were built up by three convolutional layers and two fully connected layers, and decoder consisted of four transposed convolutional operations. For autoencoder-based models, we tested a set of hyper-parameters to investigate how the performance might be affected. Our results showed that the best classification performance was achieved by a CNN using 64-by-64-dimensional log-Mel spectrogram combined with its delta feature as input, where accuracy of 75.26% was reached. Accuracy of 75.20% was achieved by CNN using only the 64-by-64-dimensional log-Mel spectrogram, which is close to the best result. Compared

to the accuracy 68.24% obtained from CNN using 28-by-28-dimensional input, the result from 64-by-64 was 7% higher. The best accuracy achieved by autoencoder-based models was 65.34%. This came from β-VAE with β value of 10 and latent dimension of 512 using log-Mel spectrogram in size 64-by-64. The reconstruction performance was evaluated by average SNR, the highest achieved value was 17.44 dB by the BIR-VAE with latent dimension of 128 and information rate of 256 using 28-by-28 sized input. The SNR degraded when applied with larger input size. BIR-VAE with the same latent dimension and information rate achieved the highest SNR on 64-by-64 sized input, which was 15.74 dB.

From the results we can conclude that among our experimental implementations, the most promising model to perform classification was CNN, with the input of log-Mel spectrogram larger in number of Mel filters and longer in length. Moreover, the delta features describing dynamical changes in a spectrogram could aid the classification. The autoencoder-based models could learn the underlying features of the spectrogram by the encoder and provide meaningful latent features to perform classification, achieving competing accuracies compared to CNN. A reasonable quality reconstruction of the log-Mel spectrogram can be realized. Although this is perceived as a smoother and more blurry reconstruction, the fundamental time-frequency patterns and structures in the spectrogram can be recreated.

6.2 Future work

Research directions that are worthy of further study and have the potential to improve performance mainly focus on two aspects, one is to investigate the signal processing and audio features, the other is to explore the neural network models. The experimental results showed that CNN using combination of log-Mel spectrogram and its delta feature as input have achieved the best accuracy. This accuracy was slightly higher than the accuracy of using only log-Mel spectrogram. This is likely because the dynamical information of the spectrogram provided extra features about an audio sample in the second channel. Thus, the accuracy may be further improved by using a combination of different audio representations, allowing CNN to learn multiple features in different channels. Other audio representations that could be further investigated are Chroma features, gammatone spectrogram and raw waveform. With more input features, neural network models can be built deeper in order to sufficiently learn the features in the multi-channel input. As the input and neural network are both larger, the classification accuracy has a huge possibility to be improved.

The autoencoder-based models also have the potential to be further improved. A more complex model can be obtained by adding more layers to the encoder and decoder or changing the number of filters in each convolutional layer. A larger model can better learn the underlying features or achieve a reconstruction preserving more details in the spectrogram. Hyper-parameters such as latent dimension, β and information rate can be further tested by setting more different values and find the best combination that leads to the highest accuracy in terms of SNR. Other parameters such as filter sizes and stride sizes in each convolutional layer reduce the size of feature map, which can also be modified and train with different values when the input dimension of the spectrogram differs.

Bibliography

[1] Muqing Deng et al. "Heart sound classification based on improved MFCC features and convolutional recurrent neural networks". In: *Neural Networks* 130 (2020), pp. 22–32.

[2] Daniel Chamberlain et al. "Application of semi-supervised deep learning to lung sound analysis". In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2016, pp. 804–807.

[3] Jia-Ming Liu et al. "Cough signal recognition with gammatone cepstral coefficients". In: *2013 IEEE China Summit and International Conference on Signal and Information Processing*. IEEE. 2013, pp. 160–164.

[4] Regunathan Radhakrishnan, Ajay Divakaran, and A Smaragdis. "Audio analysis for surveillance applications". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE. 2005, pp. 158–161.

[5] Elizabeth Baum et al. "Sound identification for fire-fighting mobile robots". In: *2018 Second IEEE international conference on robotic computing (IRC)*. IEEE. 2018, pp. 79–86.

[6] Egor Lakomkin et al. "On the robustness of speech emotion recognition for human-robot interaction with deep neural networks". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 854–860.

[7] Letizia Marchegiani and Ingmar Posner. "Leveraging the urban soundscape: Auditory perception for smart vehicles". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 6547–6554.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Gerhard P Hancke, Gerhard P Hancke Jr, et al. "The role of advanced sensing in smart cities". In: *Sensors* 13.1 (2013), pp. 393–425.

[10] Wei Dai et al. "Very deep convolutional neural networks for raw waveforms". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 421–425.

[11] Muhammad Huzaifah. "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks". In: *arXiv preprint arXiv:1706.07156* (2017).

[12] Ossama Abdel-Hamid et al. "Convolutional neural networks for speech recognition". In: *IEEE/ACM Transactions on audio, speech, and language processing* 22.10 (2014), pp. 1533–1545.

[13] Antonio J Torija, Diego P Ruiz, and Ángel F Ramos-Ridao. "A tool for urban soundscape evaluation applying support vector machines for developing a soundscape classification model". In: *Science of the Total Environment* 482 (2014), pp. 440–451.

[14] Michael J Bianco et al. "Machine learning in acoustics: Theory and applications". In: *The Journal of the Acoustical Society of America* 146.5 (2019), pp. 3590–3628.

[15] Jia-Ching Wang et al. “Environmental sound classification using hybrid SVM/KNN classifier and MPEG-7 audio low-level descriptor”. In: *The 2006 IEEE international joint conference on neural network proceedings*. IEEE. 2006, pp. 1731–1735.

[16] Justin Salamon and Juan Pablo Bello. “Unsupervised feature learning for urban sound classification”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 171–175.

[17] Karol J Piczak. “Environmental sound classification with convolutional neural networks”. In: *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2015, pp. 1–6.

[18] Karol J. Piczak. *ESC: Dataset for Environmental Sound Classification*. Version V2. 2015. DOI: 10.7910/DVN/YDEPUT. URL: https://doi.org/10.7910/DVN/YDEPUT.

[19] *URBANSOUND8K DATASET*. URL: https://urbansounddataset.weebly.com/urbansound8k.html.

[20] Zhichao Zhang et al. “Deep convolutional neural network with mixup for environmental sound classification”. In: *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer. 2018, pp. 356–367.

[21] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[22] Venkatesh Boddapati et al. “Classifying environmental sounds using image recognition networks”. In: *Procedia computer science* 112 (2017), pp. 2048–2056.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[24] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[25] Sajjad Abdoli, Patrick Cardinal, and Alessandro Lameiras Koerich. “End-to-end environmental sound classification using a 1D convolutional neural network”. In: *Expert Systems with Applications* 136 (2019), pp. 252–263.

[26] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[27] J. Salamon, C. Jacoby, and J. P. Bello. “A Dataset and Taxonomy for Urban Sound Research”. In: *22nd ACM International Conference on Multimedia (ACM-MM’14)*. Orlando, FL, USA, 2014, pp. 1041–1044.

[28] *freesound*. URL: https://freesound.org/.

[29] Marvin HJ Gruber. *Statistical digital signal processing and modeling*. 1997.

[30] Eileen Daniel. “Noise and hearing loss: a review”. In: *Journal of School Health* 77.5 (2007), pp. 225–231.

[31] Stuart Rosen and Peter Howell. *Signals and systems for speech and hearing*. Vol. 29. Brill, 2011.

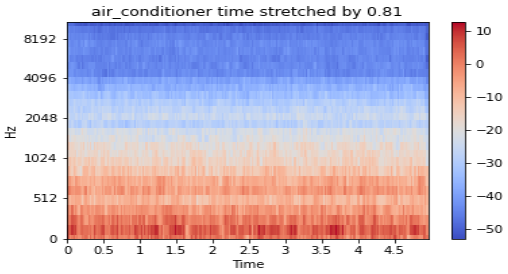
[32] Kshitiz Kumar, Chanwoo Kim, and Richard M Stern. “Delta-spectral cepstral coefficients for robust speech recognition”. In: *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2011, pp. 4784–4787.

[33] Yoonchang Han and Kyogu Lee. "Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation". In: *arXiv preprint arXiv:1607.02383* (2016).
[34] Justin Salamon and Juan Pablo Bello. "Deep convolutional neural networks and data augmentation for environmental sound classification". In: *IEEE Signal Processing Letters* 24.3 (2017), pp. 279–283.
[35] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.
[36] Carl Doersch. "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908* (2016).
[37] Diederik P Kingma and Max Welling. "An introduction to variational autoencoders". In: *arXiv preprint arXiv:1906.02691* (2019).
[38] Geoffrey E Hinton and Richard S Zemel. "Autoencoders, minimum description length and Helmholtz free energy". In: *Advances in neural information processing systems*. 1994, pp. 3–10.
[39] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.
[40] Irina Higgins et al. "Towards a definition of disentangled representations". In: *arXiv preprint arXiv:1812.02230* (2018).
[41] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: (2016).
[42] Christopher P Burgess et al. "Understanding disentangling in β-VAE". In: *arXiv preprint arXiv:1804.03599* (2018).
[43] Xi Chen et al. "Variational lossy autoencoder". In: *arXiv preprint arXiv:1611.02731* (2016).
[44] Shengjia Zhao, Jiaming Song, and Stefano Ermon. "Infovae: Information maximizing variational autoencoders". In: *arXiv preprint arXiv:1706.02262* (2017).
[45] Daniel T Braithwaite and W Bastiaan Kleijn. "Bounded information rate variational autoencoders". In: *arXiv preprint arXiv:1807.07306* (2018).
[46] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
[47] Arthur Gretton et al. "A kernel two-sample test". In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 723–773.
[48] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. Vol. 8. 2015, pp. 18–25.
[49] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

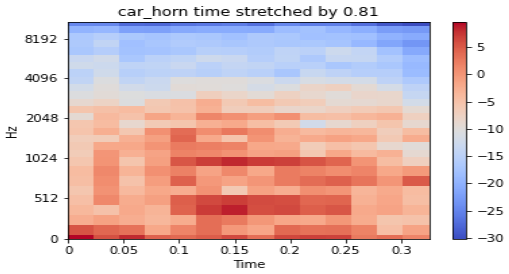
Visualization of Data Augmentation

A

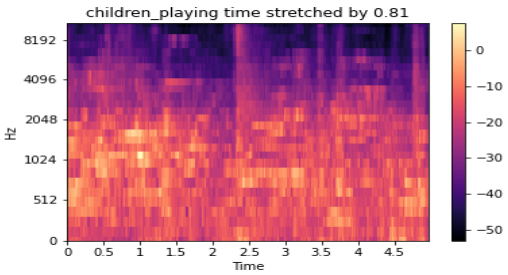
In this appendix, we visualize the audio samples that are applied with data augmentation, which includes time stretch by factors of 0.81 and 1.23 and pitch shift by factors of 2 and -2, which were discussed in Chapter 2. Comparing with the original log-Mel spectrograms in Figure 2.5, the visualizations of time stretched samples by factors of 0.81 and 1.23 are shown in Figure A.1 and A.2 respectively, the effect of pitch shift on audio sample by factors of 2 and -2 are visualized in Figure A.3 and Figure A.4.



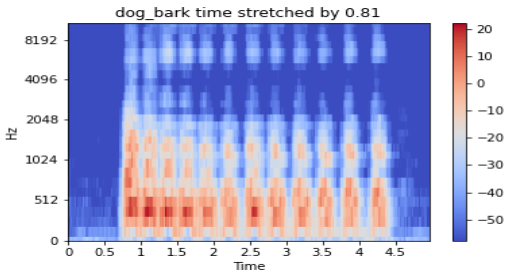
(a) Air conditioner time stretched by 0.81



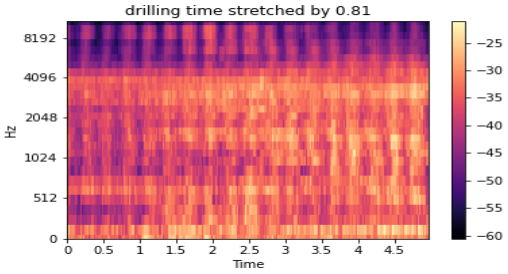
(b) Car horn time stretched by 0.81



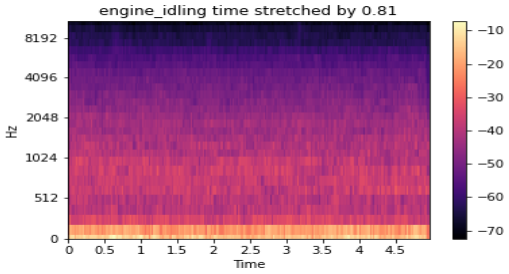
(c) Children playing time stretched by 0.81



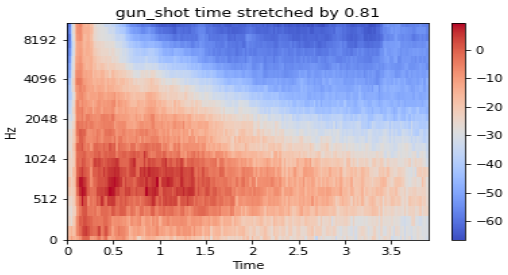
(d) Dog bark time stretched by 0.81



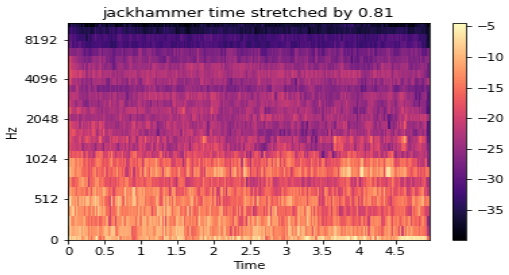
(e) Drilling time stretched by 0.81



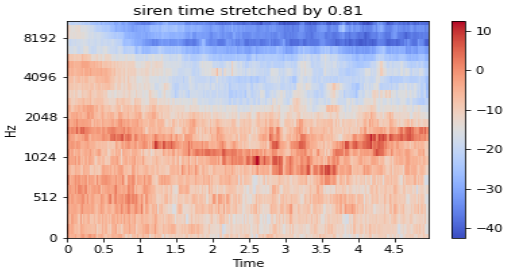
(f) Engine idling time stretched by 0.81



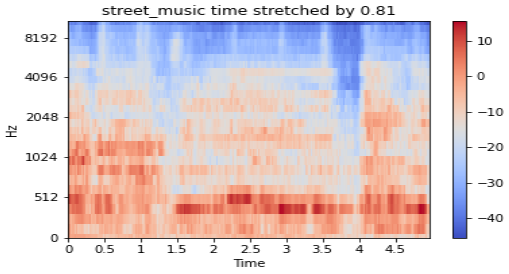
(g) Gun shot time stretched by 0.81



(h) Jackhammer time stretched by 0.81

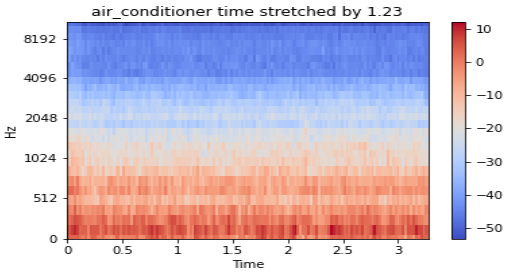


(i) Siren time stretched by 0.81

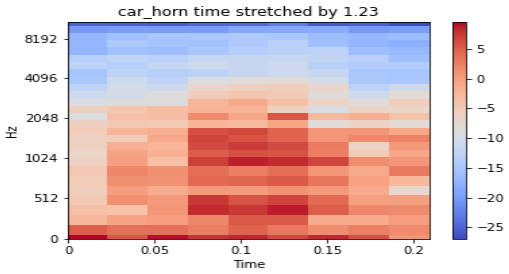


(j) Street music time stretched by 0.81

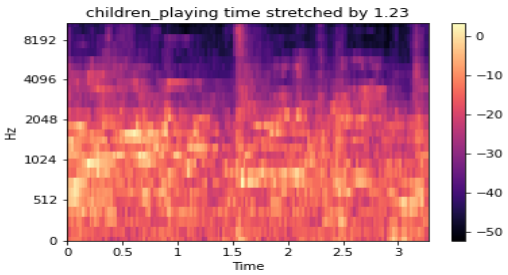
Figure A.1: Visualization of audio time stretched by 0.81



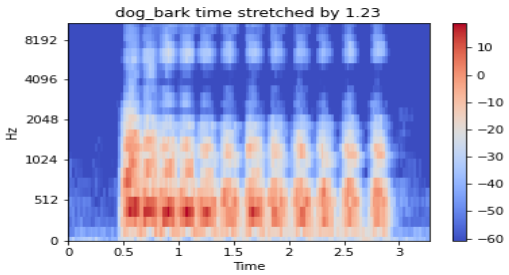
(a) Air conditioner time stretched by 1.23



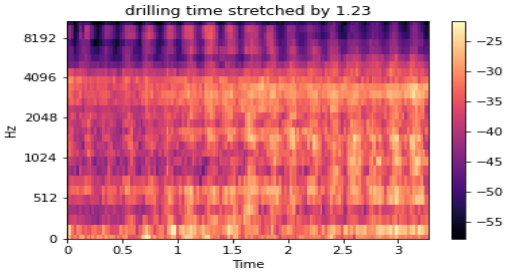
(b) Car horn time stretched by 1.23



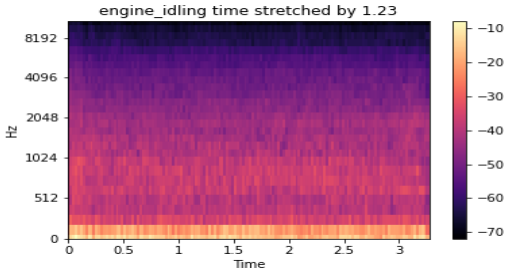
(c) Children playing time stretched by 1.23



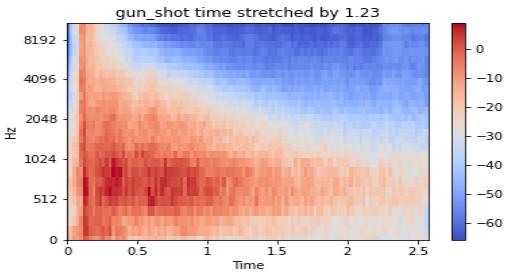
(d) Dog bark time stretched by 1.23



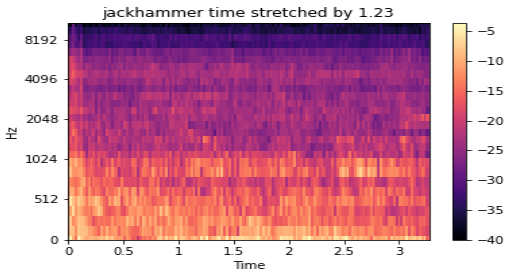
(e) Drilling time stretched by 1.23



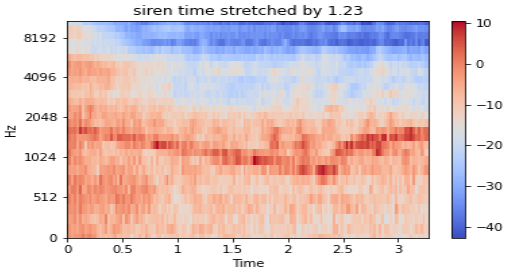
(f) Engine idling time stretched by 1.23



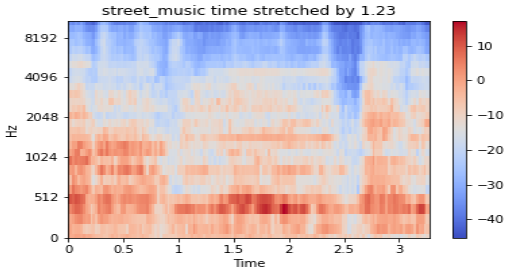
(g) Gun shot time stretched by 1.23



(h) Jackhammer time stretched by 1.23

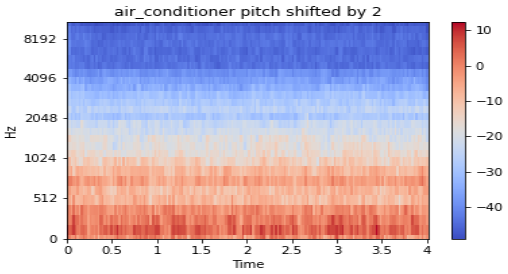


(i) Siren time stretched by 1.23

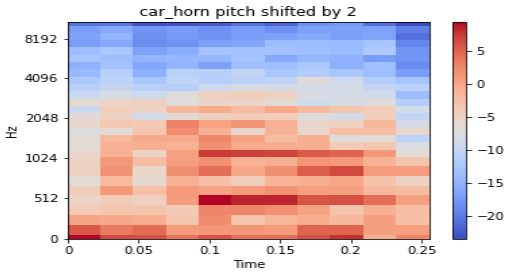


(j) Street music time stretched by 1.23

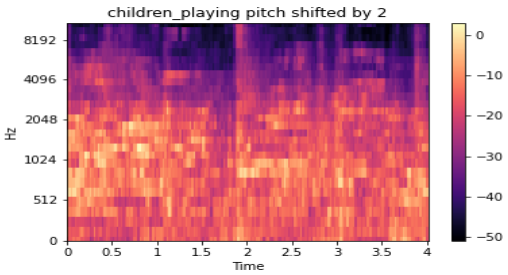
Figure A.2: visualization of audio time stretched by 1.23



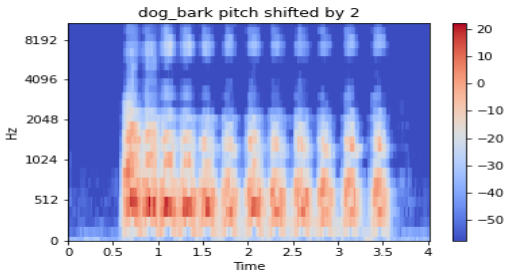
(a) Air conditioner pitch shifted by 2



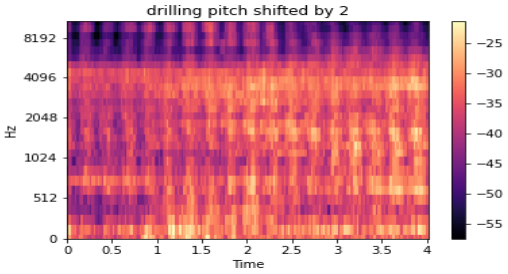
(b) Car horn pitch shifted by 2



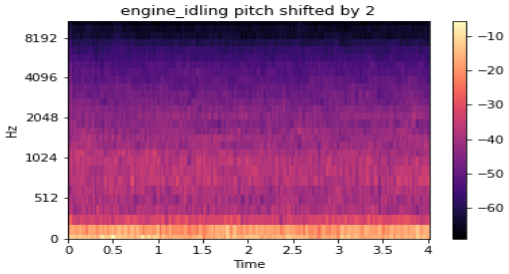
(c) Children playing pitch shifted by 2



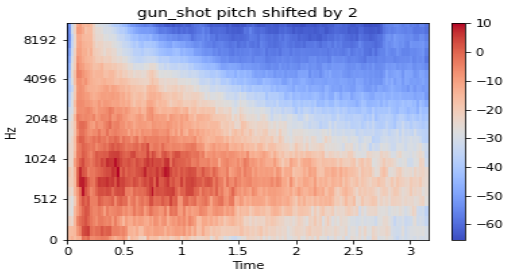
(d) Dog bark pitch shifted by 2



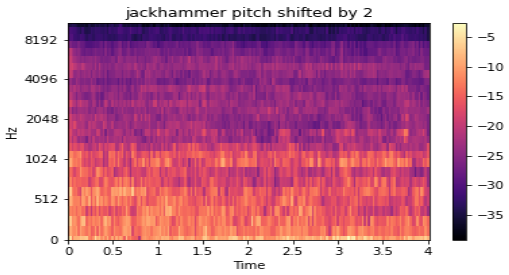
(e) Drilling pitch shifted by 2



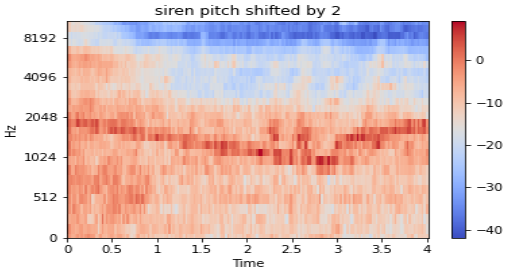
(f) Engine idling pitch shifted by 2



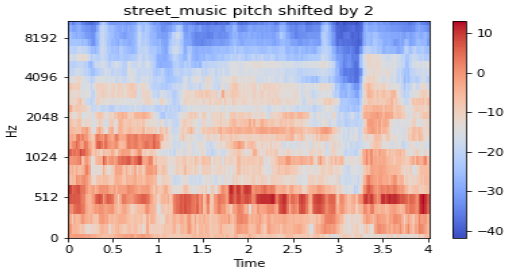
(g) Gun shot pitch shifted by 2



(h) Jackhammer pitch shifted by 2

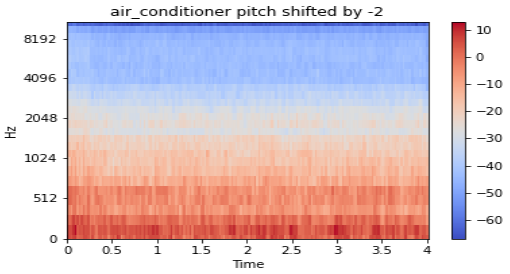


(i) siren

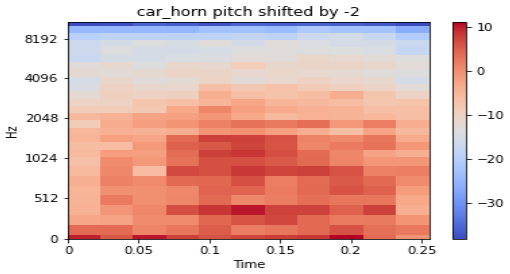


(j) Street music pitch shifted by 2

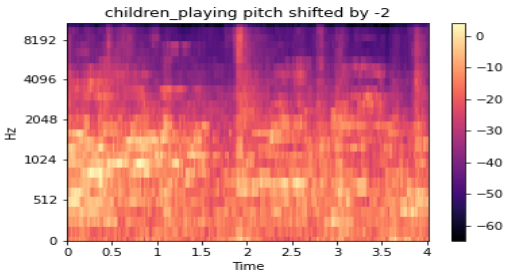
Figure A.3: Visualization of of audio pitch shifted by 2



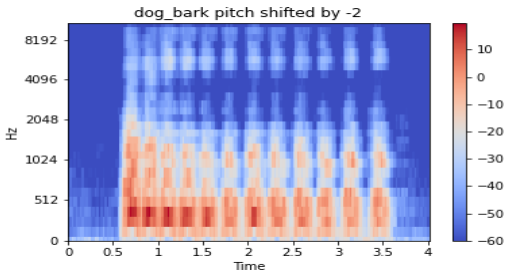
(a) Air conditioner pitch shifted by -2



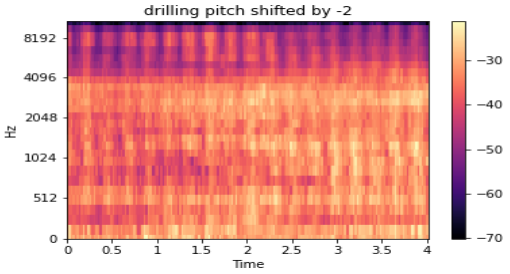
(b) Car horn pitch shifted by -2



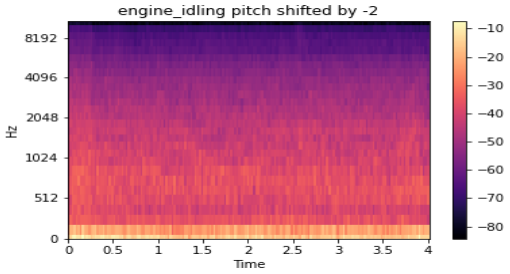
(c) Children playing pitch shifted by -2



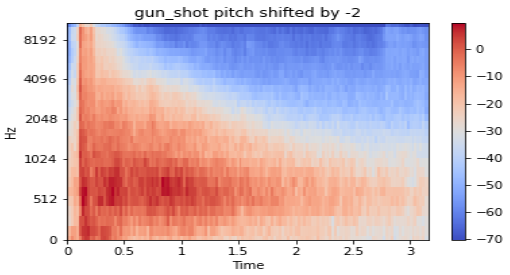
(d) Dog bark pitch shifted by -2



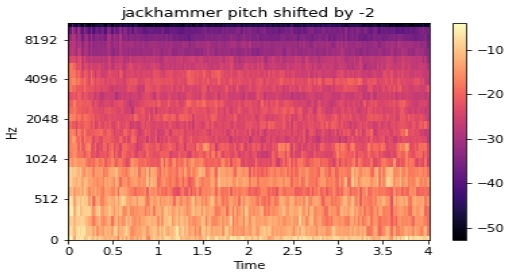
(e) Drilling pitch shifted by -2



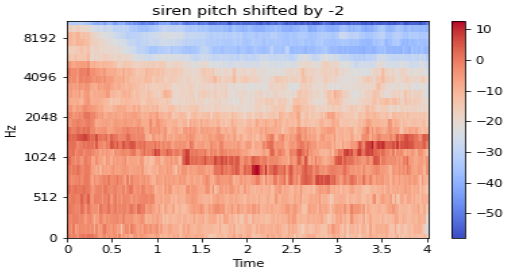
(f) Engine idling pitch shifted by -2



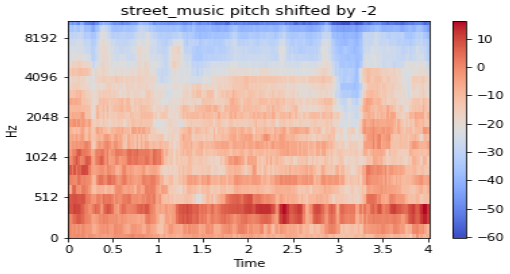
(g) Gun shot pitch shifted by -2



(h) Jackhammer pitch shifted by -2



(i) Siren pitch shifted by -2



(j) Street music pitch shifted by -2

Figure A.4: Visualization of audio pitch shifted by -2

Visualization of Generated Spectrograms

B

In this appendix, we visualize the generated log-Mel spectrograms by the autoencoder-based models that achieved the best reconstruction performance and highest classification accuracy in previous experiment. On both dimensions of 28-by-28 and 64-by-64, the BIR-VAE with latent dimension of 128 and information rate of 256 achieved the highest SNR. The visualization of generated 28-by-28-dimensional log-Mel spectrograms are shown in Figure B.1 and 64-by-64-dimensional ones are shown in Figure B.2. The highest classification accuracy was achieved by the β-VAE with 256 latent dimension and a β value of 10 using 64-by-64 sized input, the generated 64-by-64-dimensional spectrograms are shown in Figure B.3

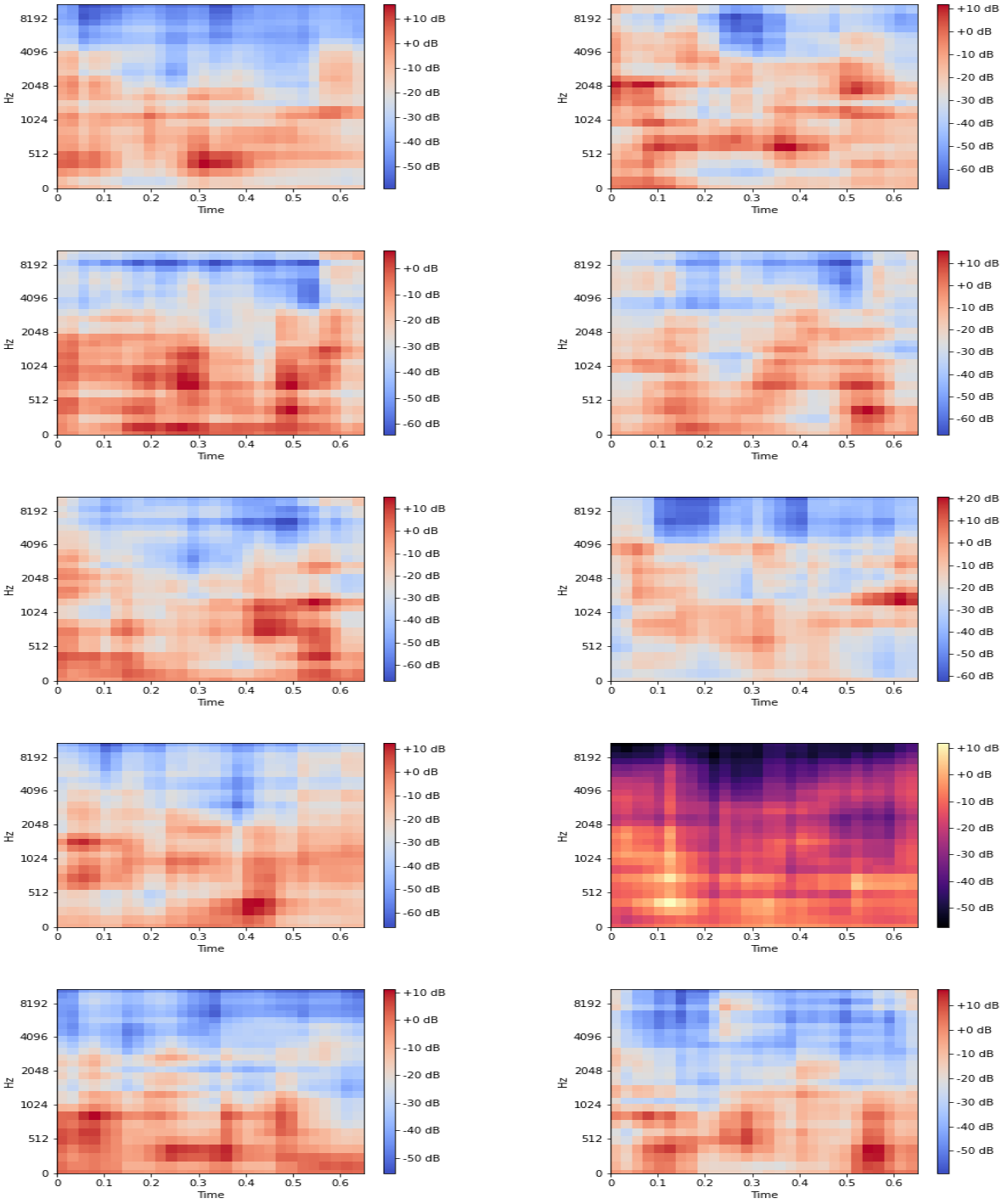


Figure B.1: Generated 28-by-28-dimensional log-Mel spectrogram

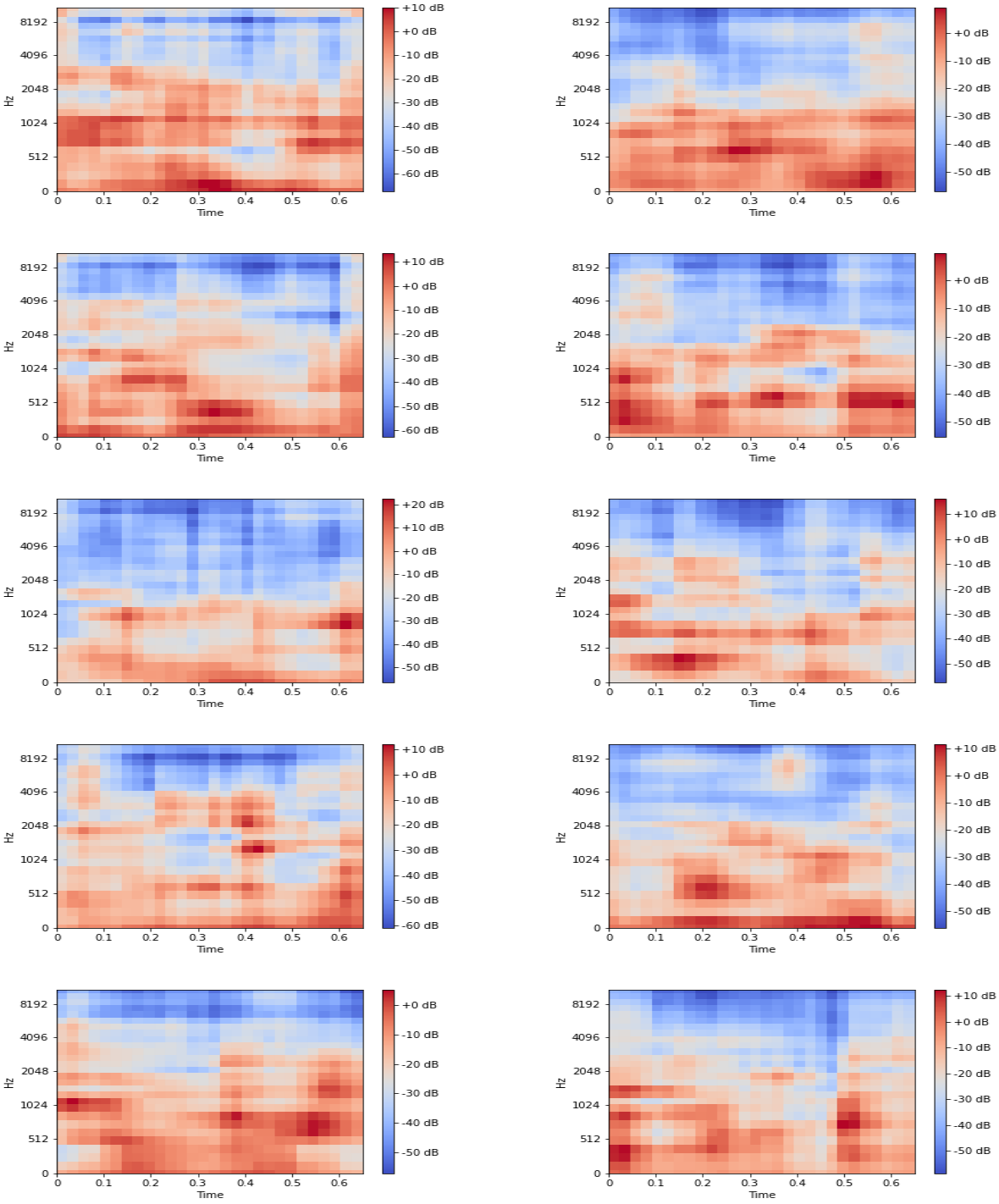


Figure B.2: Generated 64-by-64-dimensional log-Mel spectrogram

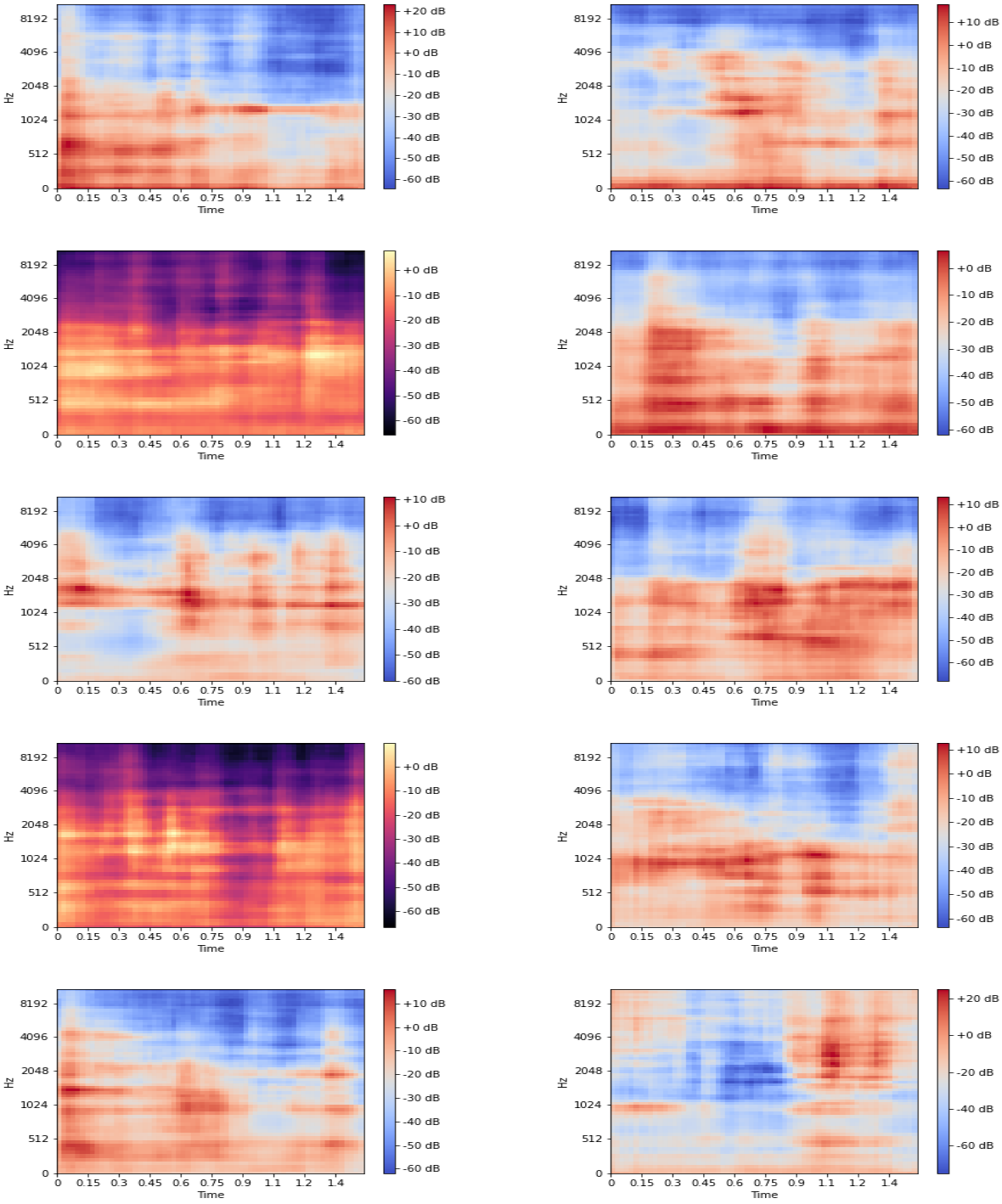


Figure B.3: Generated 64-by-64-dimensional log-Mel spectrogram