AUDIO CLASSIFICATION AND EVENT DETECTION BASED ON SMALL-SIZE WEAKLY LABELED DATA

A Thesis
Presented to
The Academic Faculty

By

Chieh-Feng Cheng

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2020

AUDIO CLASSIFICATION AND EVENT DETECTION BASED ON SMALL-SIZE WEAKLY LABELED DATA

Approved by:

Dr. David V. Anderson, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark A. Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Elliot Moore II
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Eva Dyer
Department of Biomedical Engineering
Georgia Institute of Technology

Dr. Ghassan AlRegib
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Abbas Rashidi
Civil and Environmental Engineering
University of Utah

Date Approved: November 21, 2019

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Audio event detection and classification are critical tasks in the analysis of multimedia data. Most current research on these topics focuses on processing strongly labeled data and using fully supervised machine learning techniques. However, many sources of multimedia data lack detailed annotation and rather have only high-level meta-data describing the main content of various long segments of the data. We propose a novel framework to perform audio classification when working with such weakly labeled data, especially for small datasets. A traditional approach to this problem is to use techniques for strongly labeled data and then to deal with the weak nature of the labels via post-processing. In contrast, our approach directly addresses the weakly labeled aspect of the data by classifying longer windows of data based on the clustering behavior of the acoustic features over time. We evaluate our framework using both synthetic datasets and real data and demonstrate that our method works well under both situations. Also, it outperforms other existing methods when using small size datasets.

CHAPTER 1
INTRODUCTION

1.1 Motivation

The purpose of this research is to develop a method for audio classification based on weakly labeled data, especially for small training datasets. The motivation for this research is in response to the deluge of self-recorded multimedia data now available. For example, many popular upload sites contain video and audio that lacks detailed annotation but rather only has high-level meta-data describing the significant content of the entire signal. Similar issues arise in recordings recorded in realistic soundscapes, such as from popular digital home services like Google home and Amazon Alexa. In these and many other contexts, given clip-level metadata, we only know that the described objects and events occur in the recording, but we have no information about how often and exactly when they occur. Such data is often said to be *weakly labeled*. There are many unexplored approaches that could potentially derive targeted information we need from these weakly labeled datasets.

This work develops signal processing and machine learning techniques for weakly labeled acoustic data. In these datasets, sections of data are labeled as containing a signal of interest, but this signal may be intermittent and occur at one or more locations which are not clearly delineated. An example of weakly labeled image data could be a picture labeled as "dog" in which the dog is only a part of a larger scene.

Weakly labeled data is common in many application areas but is particularly common in audio classification tasks. For example, one might have training data consisting of clips of audio labeled "horns" that contain many other noises along with some intermittent horn sounds. Given such weakly labeled data, our goal is to be able to classify segments of audio according to the content they contain, even if this content is only intermittent. Following

the example above, we would like to be able to recognize when an audio segment contains a horn sound even if it only occupies a small fraction of the segment.

More broadly, weakly labeled audio data arises in numerous other applications simply as a result of the difficulty and expense involved in manually annotating the precise content of audio data. Due to the difficulty of obtaining strong labels in real-life multimedia data, most of the multimedia or audio recordings will only contain clip-level metadata which can be treated as weak labels. Thus, by developing techniques to learn from weakly labeled data, we can avoid the time-consuming and expensive process of manual annotation. In this research, we will consider several particular example applications.

1.2 Contributions

In this research, we present a structured and flexible framework which combines multiple machine learning techniques as an approach to deal with these weakly labeled recordings. The primary concern of the proposed framework is its classification accuracy of unlabeled recordings in new environments or soundscapes. Our approach involves classifying longer segments of data by considering the clustering behavior of the acoustic features across a window of time to create a foreground / background model. The challenges we face in making such algorithms practical are open questions that arise in many domains:

- How to best leverage small amounts of data?
- How to make use of hand-recorded training data?
- How to extract important information from weakly labeled data?
- How to deal with significant background / environmental noise?
- How to distinguish different events in an audio clip?
- How to make algorithm flexible enough to fit different audio scenes?

We believe that the proposed framework will provide effective ways of addressing many of these questions.

This research also contributes to the field of civil engineering. One of our collaborative applications is acoustic monitoring of large construction sites. The goal in this context is to learn to identify the typical sounds of specific pieces of construction equipment and, where possible, their actions. Given weakly labeled training data for different pieces of equipment/actions, we would then like to be able to automatically monitor and characterize the activity at a construction site from simple audio recordings. Below, we will evaluate our proposed framework on real-world data we have collected from construction sites [1].

1.3 Organization

This document is organized as follows: Chapter 2 reviews the related research and several algorithms that are considered practical options in the proposed framework. Chapter 3 gives the overview of the proposed framework and details each stage. In Chapter 4, we will describe the various datasets we collected for our experiments. Since various algorithms can be applied with each step in our proposed work, chapter 5 evaluates the performance of different algorithms which are popular and applicable choices for each step in our proposed framework. Chpter 6 shows that our proposed framework achieves great performance to both a more traditional or a currently prevalent approach on both synthetic and real-world data. In addition to a general comparison of detection performance on a complete dataset, we also show the performance difference of our proposed framework and other popular algorithms when the size of a dataset decreases. Chapter 7 summarizes the contributions of our work and lists ways it might be expanded in future work.

CHAPTER 2

BACKGROUND

2.1 Audio event detection

Two main concepts are addressed in this research: "audio event detection" and "weakly labeled data." Audio event detection is also known as sound event detection and audio classification. In general, an audio event detection system can be classified as monophonic and polyphonic. Simple illustrations of monophonic and polyphonic audio event systems are shown in Figure 2.1 and 2.2. Monophonic audio event detection systems handle the polyphonic data by detecting the prominent event; while the polyphonic audio event detection system has the ability to detect or classify multiple events in complex auditory scenes. Audio classification and sound event detection is a key component of auditory scene analysis, which is the study of how we decompose an audio into its component events. Auditory scene analyses are typically performed on audio segments containing more than one auditory event. Some classical applications include speech and language recognition, automatic music transcription, and sentiment/emotion recognition, just to name a few. Conventional audio event detection often uses Gaussian Mixture Models (GMMs) on Mel-Cepstral Coefficient (MFCC) features [2, 3]. The problem of automatic speech recognition has largely dominated classic research in this field (e.g., see [4, 5, 6]), with dramatic improvements in the performance obtained by machine learning based approaches in recent years [7, 8, 9]. However, many other important applications in auditory scene analysis still remain. One example that we address in this thesis is the analysis of sounds made in construction sites. Heavy construction equipment often generates unique sound patterns while performing various tasks, by processing this data we can potentially extract a great deal of information regarding the underlying activities at a construction site [1].

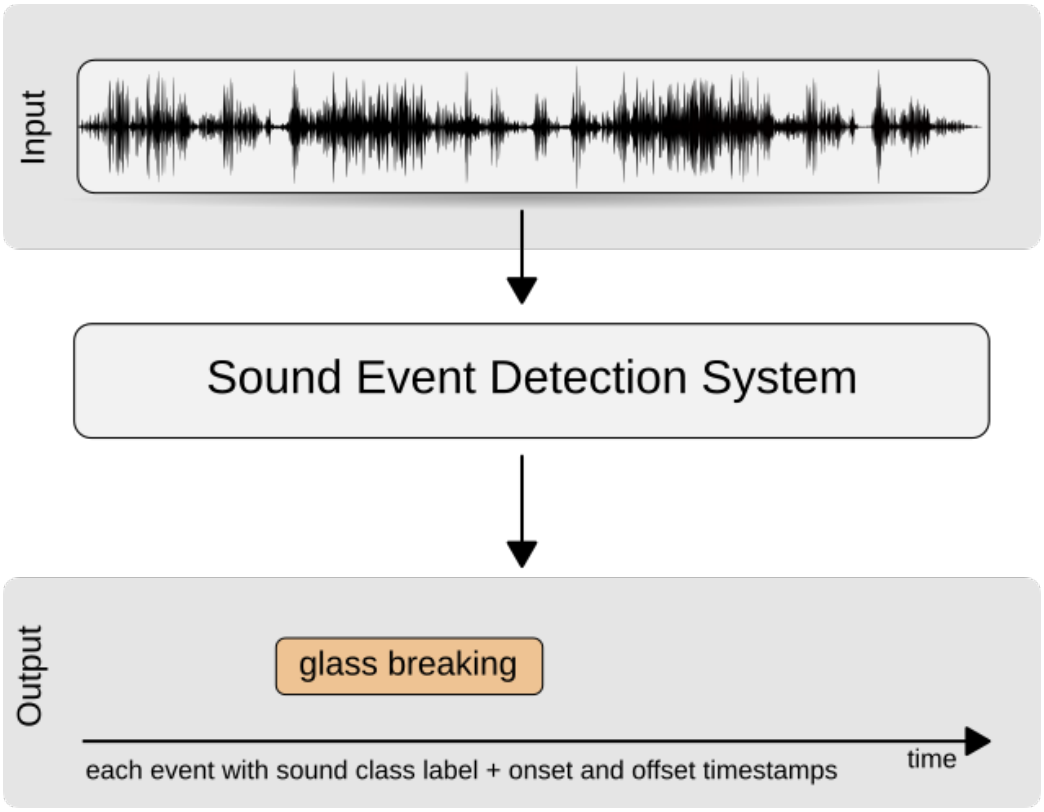


Figure 2.1: An overview of a monophonic sound event detection system, provided by *DCASE 2017 Challenge* (http://www.cs.tut.fi/sgn/arg/dcase2017/)

2.1.1 Audio event detection using weakly labeled datasets

Audio classification has traditionally been studied using datasets which contain detailed temporal information of each sound event present [10, 11]. These are known as strongly labeled datasets. In strongly labeled datasets, detailed time labels of occurrences of the audio event in the recordings are given so that event-specific parts can be detected from the whole recordings. However, as noted above, many audio datasets are only weakly labeled in that the labels only indicate that some specific sound events are presented in the audio, but do not contain the exact time the events occur in the recording [12, 13]. However, creating a large amount of strongly labeled data is an extremely time consuming, difficult and expensive process. In fact, most publicly available strongly-labeled datasets have less than an hour of audio data for each event [14, 15, 16, 17]. In most cases, only

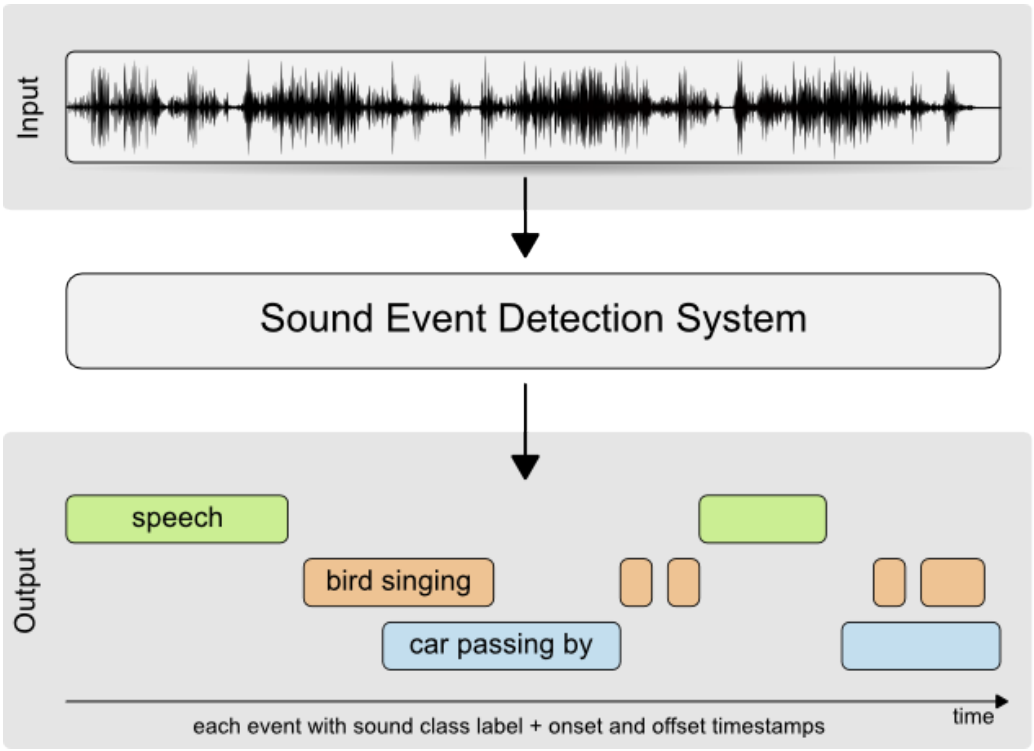


Figure 2.2: An overview of a polyphonic sound event detection system, provided by *DCASE 2017 Challenge* (http://www.cs.tut.fi/sgn/arg/dcase2017/)

a few minutes of audio data per acoustic event is available. Recently, the use of weakly-labeled audio data has received increased attention. Indeed, it was one of the subjects of the recent DCASE (Detection and Classification of Acoustic Scenes and Events) Workshop and Challenge [18].

Almost all current literature on audio classification relies on supervised methods using strongly labeled datasets. In this case, labeled examples of each different audio event class are available and then some supervised machine learning technique is applied for detecting events and performing audio classification. This reliance on strongly labeled data severely limits the scale and scope of audio classification works and is currently one of the most important challenges faced by the research community. The motivation for using weakly labeled datasets is that manually annotating audio recordings with weak labels is much easier than annotating with strong labels. Furthermore, weakly labeled datasets can

be directly obtained from popular websites such as Youtube.com. Most of the prior work on learning detection models from audio event search has focused on the task of creating strong labels from weak labels, for example, [11, 19, 20]. Recently, there have been attempts toward weakly supervised learning of audio events [21]. Chou *et al.* [22] and Lee *et al.* [19] used deep convolutional neural networks to solve weakly supervised audio event learning problems. Most of the multimedia or audio clips on the Internet contain some clip-level metadata which can be treated as weak labels. Thus, unlike with strongly labeled datasets, the time consuming and expensive process of manual annotation may no longer be required and a large amount of weakly labeled data can be directly obtained from the Internet.

In [23] the authors propose a multiple instance learning approach for sound event detection using weakly labeled data. The main idea is that audio event detection using weakly labeled data can be formulated as a multiple instance learning (MIL) problem. In MIL, instances are given in groups called bags, and labels are available for each bag. In a negative bag all instances are known to be negative, whereas, in a positive bag it is only known that at least one instance is positive. Thus, in a positive bag both positive and negative instances can be present. The goal is to learn a classifier technique using data in bag-label form. In [24] the authors use a fully connected neural network (FCN) to recognize instruments and tempo for each time frame of an audio clip with only the clip-level labels, extending this network to other sound event detection problems in [25]. Convolutional and recurrent neural networks (CNN and RNN) have also been used in the related context of audio tagging tasks [11, 26]. Different from the proposed framework in this research, these existing works can be understood as first using weakly labeled data to build strong labels, and then applying standard machine learning techniques.

2.2 Audio enhancement

Captured audio is assumed to contain the signals of interest along with environmental and other background noise sources. Usually the noise will have a negative effect on identifying certain activity patterns; thus, an effective noise estimation algorithm can have a significant impact on the proposed procedure, and may be used in our first processing step. Of course, the enhancement should be tuned carefully since low-level enhancement will still keep most of the background noise; while if the enhancement is too aggressive, the audio in the dataset might be distorted, degrading performance.

Noise estimation has been studied in signal processing for decades. Some traditional algorithms are based on optimal smoothing and calculating minimum statistics of the power spectrum [27]. Actually, most noise estimation algorithms are dealing with speech enhancement problems and many assume stationary noise models [28, 29, 30]. In our case, we need to address highly non-stationary noise environments such as what might be encountered in real-life recordings. Rangachari and Loizou proposed a noise estimation algorithm for highly non-stationary environments and it has been shown to be very effective [31]. Cohen also developed a popular noise estimation algorithm which can be used in adverse environments [32].

2.3 Audio features

The raw or enhanced audio recordings are rarely passed directly into learning algorithms because they contain far more information than is relevant for most tasks. Instead, a set of features are typically derived from the raw clips, which are then fed into subsequent algorithms. Feature extraction attempts to summarize the relevant information while discarding irrelevant information, and will be the first stage in our framework. Several practical features are listed below.

2.3.1 Short-time Fourier transform (STFT) spectrogram

The short-time Fourier transform (STFT) is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time [33]. In practice, the procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment, which results in the simple Fourier spectrum of each segment. The signal to be transformed is multiplied by a window function; for example, the Gaussian window and the Hamming window are two of the popular choices. The Fourier transform (a one-dimensional function) of the resulting signal is taken as the window is slid along the time axis, resulting in a two-dimensional representation of the signal. Important considerations for the STFT process are the window used, the size of Fourier transform, and the amount of overlap for each windowed segment.

2.3.2 Mel-frequency cepstral coefficients (MFCCs)

MFCCs are a set of features that can be computed from audio data and are designed to match how humans perceive sound. MFCCs are based on the mel scale, which relates the frequency of sound to a measure of the perceived pitch. The mel scale was originally developed by Stevens *et al.* in 1937 [34], and later refined in 1940 [35]. A few different formulas have been fit to the data that defines the mel scale. One of the most commonly used ones was given by O'Shaughnessy [36]. The corresponding frequency in mels is calculated using the formula

$$m = 2595 \times log\Big(1 + \frac{f}{700}\Big),$$

where f is the frequency in hertz. This formula will be used in our experiments when calculating MFCCs.

The word "cepstral" in MFCCs was defined by Bogert [37], which is obtained by re-

versing the first four letters of the word "spectral." It is originated from the "cepstrum," which is obtained by applying a frequency-based transform (e.g. the Fourier or cosine transform) to the power spectrum of a signal. For example, a simple cepstrum could be computed as

$$cepstrum = DCT\Big\{ \log\Big(|\mathscr{F}\{x(n)\}|^2\Big)\Big\},$$

where $x(n)$ denotes the targeted signal, $\mathscr{F}\{\cdot\}$ denotes the Fourier transform, and $DCT\{\cdot\}$ denotes the discrete cosine transform. A mel-scaled cepstrum inserts a triangular filter bank whose frequency bands are linearly spaced in the mel domain immediately prior to the log operation. In 1995, dynamic features (deltas and delta-deltas) for MFCCs were proved to provide significant performance gains under all the different signal conditions by Sandhu and Ghitza [38]. After MFCCs were applied to speech recognition [39], it was shown that MFCCs are superior to several other feature types used for speech recognition because of their ability to represent perceptually relevant aspects of the sound [40]. As of now, MFCCs are still one of the most commonly used features in speech recognition and other audio scene problems. The process we use to calculate MFCCs can be summarized as follows:

1. Calculate the short-time Fourier transform (STFT) of the audio clips. Discard phase information and only preserve the power.

2. Map the frequency axis onto a mel scale using triangular windows to combine frequency bins from the Fourier transform according to the mel scale spacing.

3. Compute the logarithm of the results.

4. Calculate the discrete cosine transform (DCT) on the results to decorrelate them.

5. The MFCCs are the amplitudes of the resulting spectrum.

The MFCCs are calculated using the RASTA-PLP, an acronym for Relative Spectral Transform - Perceptual Linear Prediction, program provided by Hynek Hermansky [41,

42] under MATLAB environment. The "melfcc.m" function supports various options to calculate MFCCs from sound signals. For Python environment, LibROSA is also applied in our work. LibROSA is a popular and great Python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. The details of LibROSA can be found in [43].

2.3.3 Delta features

When MFCCs are used as features of an audio, the estimation of the first and second derivatives are often included with the original MFCCs. The estimated first and second temporal derivatives of a set of features are typically called deltas and delta-deltas. The deltas and delta-deltas are also referred to as the dynamic features since they capture information about how the underlying features are changing over time. The combination of original features and dynamic features has been shown to improve various recognition tasks [44].

The delta feature is calculated as

$$\Delta[t] = \frac{\sum_{n=1}^{N} n(y[t+n] - y[t-n])}{2\sum_{n=1}^{N} n^2},$$

where $y[t]$ represents the value of a given feature for the tth sample and N is the number of samples on either side of the current sample to use in estimating the derivative. This formula represents the slope of a least-squares linear regression over the computation window. For the first and last N samples of a signal, the computation window will extend past the beginning or the end of the signal. In these cases, we repeat the corresponding first or last sample in place of all the missing samples. Delta-deltas are computed by applying the above equation a second time. Since delta-delta features are computed by second derivatives, they are also called acceleration coefficients.

Most of our work in Chapter 5 and 6 use features that include MFCCs, the deltas, and the delta-deltas. The derivative operation in the deltas and the delta-deltas acts like a high-pass filter. In this work, the original MFCCs, deltas, and delta-deltas of input audio data

tend to fall in different magnitude ranges, causing one to dominate the others when used together before they are fed into clustering algorithms. To prevent from this problem, we scaled the delta-deltas by a factor of 5 to bring them into similar magnitude ranges with the deltas. When the original MFCCs were also included, we scaled both the deltas and delta-deltas by an additional factor of 5.

2.3.4 Log-mel energy spectrum

Log-mel energies are computed in the same way as MFCCs, except that the DCT step is skipped. Different from coefficients obtained by MFCCs procedures, the calculated results here directly correspond to the amount of energy in different frequency bands. Comparing to MFCCs, Log-mel energy spectrum is a more intuitively understandable feature and it is also a commonly choice among audio feature types. Log-mel energy spectrum has several modified versions due to different types of acoustic scenes. A common example is using the magnitude spectrum instead of the power spectrum, and omitting the log scaling at the end of the log-mel energy spectrum computation. This is often be called as mel-magnitude spectrum.

2.4 Dimensionality reduction

In general, extracting features from original data can be viewed as a form of dimensionality reduction. However, we might want to tune the extracted features in a specific situation if the features are not designed manually. For this purpose, feeding features into a dimensionality reduction algorithm can help extract the information most relevant to the current task, which can speed up learning and reduce overfitting. Also, the dimensionality reduction process can help save the calculation and time cost in the following learning steps. Several popular dimesionality methods are introduced below.

2.4.1 Singular value decomposition

The singular value decomposition (SVD) is a factorization of a data matrix, and it has many useful applications in signal processing and statistics [45]. Formally, the SVD of an $m \times n$ matrix $\mathbf{X}$ is a factorization of the form $\mathbf{U\Sigma V^T}$. The function can be written as

$$\mathbf{X} = \mathbf{U\Sigma V}^T.$$

where $\mathbf{U}$ is an $m \times m$ unitary matrix, $\mathbf{\Sigma}$ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and $\mathbf{V}$ is an $n \times n$ unitary matrix. The diagonal entries σ_i of $\mathbf{\Sigma}$ are known as the singular values of $\mathbf{X}$ and they are typically sorted in decreasing magnitude. The columns of $\mathbf{U}$ and the columns of $\mathbf{V}$ are called the left-singular vectors and right-singular vectors of $\mathbf{X}$. To perform dimensionality reduction using SVD, we can preserve only the first R columns of $\mathbf{U}$, the first R rows and columns of $\mathbf{\Sigma}$, and the first R rows of $\mathbf{V}$ to be used in the following processes, where R is selected by examining the matrix $\mathbf{\Sigma}$.

After the value of R is selected, the first R columns of $\mathbf{U}$ will be preserved as $\mathbf{U}'$ and the first R rows and columns of $\mathbf{\Sigma}$ will be $\mathbf{\Sigma}'$. When the new data $\mathbf{X}_i$ comes in, we can use $\mathbf{U}'$ and $\mathbf{\Sigma}'$ to factorize $\mathbf{X}_i$ into $\mathbf{U'\Sigma'V_i^T}$, where V_i^T can be treated as the new feature matrix for the input data.

For example, performing SVD on the magnitude of the STFT matrix $\mathbf{X}$ in order to reduce the dimension of the STFT spectrogram ($\mathbf{X} = \mathbf{U\Sigma V}^T$.) By examining the matrix $\mathbf{\Sigma}$, which contains the singular values along the diagonal, we can determine how many components are sufficient to provide a good approximation to the original $\mathbf{X}$. We can then truncate the SVD by including only the first R components of $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ as $\mathbf{U}'$, $\mathbf{\Sigma}'$, and $\mathbf{V}'$. The columns of the truncated $\mathbf{V}^T$ will be treated as a low-dimensional set of features for each time bin of the STFT. Also, using $\mathbf{U}'$ and $\mathbf{\Sigma}'$ to decomposed the testing data can make sure the training data and testing data are both being projected onto a same space.

2.4.2 Principal component analysis and Independent component analysis

Principal component analysis (PCA) and Independent component analysis (ICA) are dimensionality reduction methods which are associated with SVD. Principal component analysis (PCA) reduces the dimensionality of data by projecting it onto a small set of orthogonal bases that maximizes the amount of variance preserved through the projection. The idea was originally presented by Pearson [46], and later expanded upon by Hotelling [47]. The PCA algorithm relies on the assumption that the directions of greatest variance contain the most useful information, which may not always be true. The full principal components decomposition of X can be given as

$$\mathbf{T} = \mathbf{XW},$$

where W is a m-by-m matrix of weights whose columns are the eigen-vectors of $X^T X$. The transpose of W is sometimes called the whitening or sphering transformation. This transformation maps a data vector $x(i)$ from an original space of m variables to a new space of m variables which are uncorrelated over the dataset. However, for dimensionality reduction purpose, we only need to keep the first L principal components. The truncated transformation produced by using only the first L eigen-vectors can be written as

$$\mathbf{T}_L = \mathbf{XW}_L$$

where the matrix T_L now has n rows but only L columns. That is to say, PCA learns a linear transformation $t = W^T x, x \in R^m, t \in R^L$, where the columns of $m \times L$ matrix W form an orthogonal basis for the L features (the components of representation t) that are de-correlated.

When a training data $\mathbf{T}_{train}$ has been factorized into $\mathbf{X}_{train}\mathbf{W}_L$ using PCA, the $\mathbf{W}_L$ will be preserved, where $\mathbf{W}_L$ is produced when we strip off all but the first L columns

of $\mathbf{W}$. To make the testing data project onto the same space as the training data, $\mathbf{W}_L$ is the matrix we need to perform the correct projection. When the testing data $\mathbf{T}_{test}$ comes in, it will be projected onto the principal components we derived from the training data $\mathbf{T}_{test} = \mathbf{X}_{test}\mathbf{W}_L$.

Unlike PCA, independent component analysis (ICA) seeks to decompose a signal into the sum of individual components, much like sparse representations. However, ICA uses the assumption that the components will be statistically independent for regularization instead of an assumption of sparsity. The idea of ICA was originally addressed by Jutten and Herault [48] in 1991. After that, ICA has been frequently applied to blind source separation (BSS) problems. For more details about the methods, applications, and extensions of BSS and ICA algorithms, Choi et al. provide a comprehensive review of this topic [49].

2.4.3 Non-negative matrix factorization

Non-negative matrix factorization (NMF or NNMF) is another matrix decomposition method which is developed by Lee and Seung [50, 51]. NMF is related to ICA, BSS, and sparse representations. The factorization of NMF can be usually be represented as

$$\mathbf{X} = \mathbf{W} \times \mathbf{H},$$

where $\mathbf{X}$ is the m-by-n data matrix, $\mathbf{W}$ is the m-by-k dictionary matrix, and $\mathbf{H}$ is the k-by-n activation or expansion matrix. The trick for applying NMF is determining the number of k we need when performing the factorization.

After the training data $\mathbf{X}$ being decomposed into $\mathbf{W} \times \mathbf{H}$, the $\mathbf{W}$ matrix is preserved to be used as the dictionary matrix. After the testing data $\mathbf{X}_t$ comes in, we could use $\mathbf{W}$ to decomposed $\mathbf{X}_t$ into $\mathbf{W} \times \mathbf{H}_t$. The learned dictionary $\mathbf{W}$ will keep fixed, and $\mathbf{H}_t$ could be used as the new feature matrix of testing data. This is similar to the idea of dictionary learning. The non-negative factors in the matrices induce the sparsity and lead to part-based

decompositions. In audio event detection applications, Smaragdis used NMF [52, 53] and probabilistic latent component analysis (PLCA) [54] to perform audio source separation and speech recognition. PLCA [55] is a straight forward extension of Probabilistic Latent Semantic Indexing (PLSI) [56], which deals with an arbitrary number of dimensions and can exhibit various features, such as sparsity or shift-invariance. The basic PLCA model is defined as:

$$P(X) = \sum_{z} P(z) \prod_{j=1}^{N} P(x_j|z),$$

where $P(x)$is an N-dimensional distribution of the random variable $x = x_1, x_2, ..., x_N$. The z is a latent variable, and the $P(x_j|z)$ are one dimensional distributions. This model can Effectively represent a mixture of marginal distribution products to approximate an N-dimensional distribution, where the challenge is to discover the most appropriate marginal distributions within PLCA models.

Smaragdis et al. found that NMF and PLCA provide excellent separation of vocals and a piano accompaniment [52]. They are also proved to be effective for sound recognition applications [55]. NMF relies on the assumption that the inner dimension between the dictionary and coefficient matrices is small (relative to the other dimensions) in order to prevent the problem from being under-determined. This effectively limits the number of dictionary atoms that may be used.

2.4.4 AutoEncoder

Considering the progress made in neural network based algorithms in recent decades, an *AutoEncoder* is also commonly used in recent years for the purpose of dimensionality reduction. An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner [57]. The aim of an AutoEncoder is to learn a representation (encoding) for a set of data by training the network to ignore signal "noise". In short, AutoEncoders are neural networks that copies their inputs to their outputs. They

work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. An AutoEncoder can be simply viewed as a combination of encoders and decoders. In an encoder, the network compresses the input into a latent-space representation, which can be represented by an encoding function $h = f(x)$. For the decoder part, it aims to reconstruct the input from the latent space representation, which can be represented by a decoding function $r = g(h)$. The AutoEncoder as a whole can thus be described by the function $g(f(x)) = r$, where you want r as close as the original input x. Several variants exist to the basic model of the AutoEncoder, with the aim of forcing the learned representations of the input to assume useful properties [58]. Within these years, data denoising and dimensionality reduction for data visualization, especially related to image processing aspect, are considered as the main practical applications of AutoEncoders [59, 60, 61]. With appropriate dimensionality and sparsity constraints, AutoEncoders can learn data projections that are more interesting than PCA or above mentioned techniques [62].

2.5 Clustering

Clustering or cluster analysis is a task of grouping a set of objects in such a way that objects in the same group, which is called a cluster, are more similar in some sense to each other than to those in other groups. It is a common technique for statistical data analysis that can be used in many fields, including machine learning, pattern recognition, image analysis, ..., etc. One prominent method is Gaussian mixture models (GMMs). For GMMs, the data set is modeled with a fixed number of Gaussian distributions that are initialized randomly and whose parameters are iteratively optimized to better fit the data set. Usually, the expectation-maximization (EM) algorithm is implemented for fitting mixture-of-Gaussian models; note, the EM algorithm can also give confidence ellipsoids for multivariate models. The Bayesian Information Criterion can also be compute to assess the number of clusters in the data. After learning a GMM from training data, it can be used for clustering by as-

signing any given testing data sample to the Gaussian to which it mostly probably belongs.

K-means clusters are each defined by a single data point (centroid). Data samples are then assigned to the cluster according to which of the cluster centroids they are closest to. K-means training involves starting with arbitrary centroids and then alternating between assigning observations to the nearest cluster centroid and updating the cluster centroids based on the new membership. The GMM may be thought of as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. K-d trees is a popular technique for the nearest neighbor search and intelligent initialization to speed up the process of k-means clustering.

Different from the basic idea of GMM and K-means, DBSCAN [63] starts with finding core samples of high density and expands clusters from them. After grouping together points in high-density regions, outliers that lie alone in low-density regions points are then marked. The assumption of FACED framework is different acoustic features in the signal will correspond to distinct clusters. We do not need to find centroids for each cluster but we need to find clusters that have dense regions in the data space. Regions of the lower density of points are then separated. The DBSCAN algorithm is based on the intuitive notion of "clusters" and "noise". Since the core key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points, this characteristic is much more fit as the intuition of FACED framework. Also, it is a applicable choice in our experiments since each audio event can be treated as data which contains clusters of similar density.

2.6 Classification

Many machine learning algorithms are available to build classifiers. One of the most popular classifiers is the Support Vector Machine (SVM), which seeks to maximize the margin for error between the decision boundary and training samples. Besides dealing with linearly separable data, the introduction of soft margins allow SVMs to handle non-separable

data [64] and the application of the kernel trick allows them to learn non-linear decision boundaries [65]. The sequential minimal optimization algorithm provides a computationally efficient method of learning SVM models [66]. C.C. Chang and C.J. Lin later proposed a popular library for SVMs which is called LIBSVM [67]. LIBSVM is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM).

Decision tree learning is also a simple, quick and popular way to train a classifier that has a straightforward interpretation. Most decision tree algorithms select rules that maximize the information gained by the split, as done by Quinlan's Iterative Dichotomiser 3 (ID3) algorithm [68]. Quinlan later developed the C4.5 algorithm, which extends ID3 to add support for missing values, continuous attributes, and attributes with different costs [69]. C4.5 algorithm also performs pruning at the end, which tries to remove unnecessary rules and reduce over-fitting.

Hidden Markov models (HMMs) are also very commonly used for classification. HMMs provide a relatively simple way to model sequential data. The system being modeled by an HMM is assumed to be a Markov process with unobserved (hidden) states. More specifically, we only know observational data and not information about the states. HMMs can provide a flexible way to model how a signal changes sequentially over time. Rabiner provides an excellent introduction [70] and tutorial [71] on HMMs.

In recent years, various types of neural-network based classifiers are frequently used in sound event detection problems. The simplest example is a naive neural network, which consists of only an input layer, a hidden layer, and an output layer. The neural networks calculated dot products and followed it up with a non-linear Rectified Linear Unit (ReLU) function to learn the weights and bias. Some of our preliminary work makes use of the naive neural network algorithm. Considering the development of deep learning these years, the most popular neural networks are convolutional neural networks (CNNs), recurrent neural networks (RNNs), or several ensembles of CNNs and RNNs [72]. In general, CNN

is a popular algorithm that pulls out the convoluted embeddings from the input data, and it is usually used in image processing aspect. Since we transformed the audio clips into time-frequency domain representations, CNNs can treat these representations as images and pull out the deep figures as it does in image processing problems. RNNs are mostly applied after the CNN pulling out the deep figures. Recurrent neural networks were based on David Rumelhart's work in 1986 [73]. RNNs come in many variants but one of the most popular structure of RNNs used in audio event detection problems is long short-term memory (LSTM) networks. LSTMs were discovered by Hochreiter and Schmidhuber in 1997 [74] and set accuracy records in multiple applications domains. LSTM can learn to recognize context-sensitive languages unlike previous models based on hidden Markov models (HMM) and similar concepts [75]. The outstanding applications for LSTM are improving machine translation [76], language modeling [77] and multi-lingual language processing [78]. In recent years, LSTM combined with convolutional neural networks (CNNs) has improved the accuracy of automatic image captioning and audio event detection. The non-linear Rectified Linear Unit (ReLU) function is frequently used within all the above mentioned neural networks. The softmax function is also a popular function used in neural networks, especially applied after the outpur layer. It is a non-linearity, but it is special in that it usually is the last operation done in a network. This is because it takes in a vector of real numbers and returns a probability distribution. The definition of softmax is as follows. Let x be a vector of real numbers, the i'th component of Softmax(x) is

$$\frac{e^{x_i}}{\sum_j e^{x_j}}.$$

The output of the softmax is a probability distribution; each element is non-negative and the sum over all components is 1.

CHAPTER 3
FLEXIBLE AUDIO CLASSIFICATION AND EVENT DETECTION (FACED) FRAMEWORK

3.1 Overview

Our Flexible Audio Classification and Event Detection (FACED) framework for training on weakly labeled data assumes that a common or similar background exists across multiple audio clips and that the signal of interest differs across the clips. For example, if there are two clips, one with a dog barking and one with people chatting, it is expected that they will have both been recorded under similar circumstances (*e.g.*, standing near the same road or in a same cafe). We recognize that many situations will not meet this assumption but there are many that will, such as recordings made in the same household environment or, as is the case with one of our experiment datasets, at the same construction site. After training is complete, it is also possible to perform the detection in new environments. The general approach pipeline is illustrated in Fig. 3.1.

In the FACED framework, we advocate a new approach to training sound event detectors and classifiers on weakly labeled data based on jointly analyzing entire segments of weakly labeled data to create foreground / background models that implicitly learn the weakly-labeled events. Our proposed method for training on weakly labeled data comprises a sequence of interrelated steps. Since different audio data may have significantly different presenting features, the proposed method is designed to be flexible so that we can adjust each step based on the audio dataset. The following is the general overview for the proposed framework. As an initial step, we note that it is typically helpful (but not required) to perform signal enhancement to reduce the background noise level. After the enhancement, we then extracted specific features from the output data. Usually, these

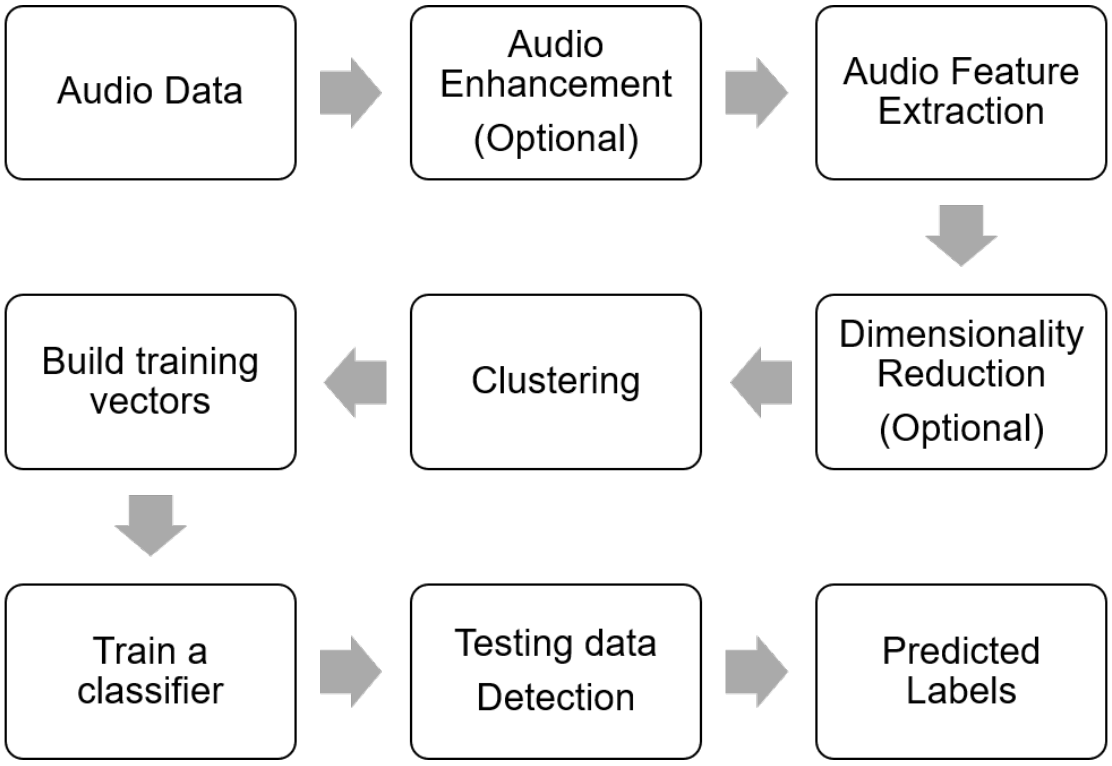


Figure 3.1: Block diagram

features are chosen by what kind of audio clips we are processing, and they are typically in a time-frequency representation, such as the *Mel-frequency cepstral coefficients (MFCCs)*, *short-time Fourier transform spectrogram (STFT)* and the *Log-mel energy spectrum*. If the dimensionality of the time-frequency representation is too high, we will apply a dimensionality reduction technique to produce a low-dimensional set of features for each column (time bin/window) of the transferred spectrogram. The most commonly used techniques here are truncated *singular-value decomposition (SVD)* and an autoencoder. We then apply a clustering algorithm to these feature vectors. From the output of the clustering, we can construct training vectors for segments corresponding to different categories by examining the distribution across the different clusters. A *fully connected neural network* is then trained using these training vectors to identify different sound patterns (*e.g.*, corresponding to the various sound events of interest such as activities of each machine in jobsite recordings or glass breaking in a shop). Each of these steps are described in detail below.

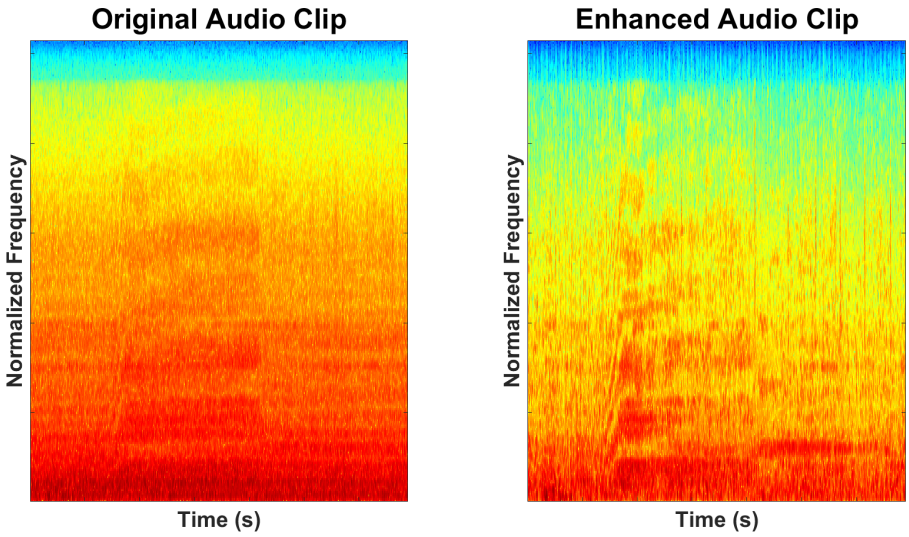


Figure 3.2: Comparison between the original recording and enhanced recording.

3.2 Signal enhancement

This signal enhancement step is not a necessary step in the proposed method but if there is a significant amount of noise, performance may be improved by applying basic noise suppression as a first step. Of course, the enhancement should be tuned carefully since low-level enhancement will still keep most of the background noise; while if the enhancement is too aggressive, the audio in the dataset might be distorted, degrading the performance. Here we introduce a classic signal enhancement algorithm developed by [31] because it is suitable to be used in highly non-stationary noise environments such as what might be encountered in real-life recordings. An estimate of the noise is continuously updated in every frame using time-frequency smoothing factors computed based on signal-presence probability in each frequency bin of the noisy spectrum. More details about this algorithm can be found in [31]. As shown in Figure 3.2, the frequency pattern is more distinct in the enhanced recording than the original recording.

3.3 Feature extraction

To extract features from the enhanced audio signal, several transforming techniques are considered. Since different types of recordings will present different characteristics, we need to select carefully depending on what sound event is recorded in the audio clips. In general, *Log-mel energy spectrum (MFS)*, *Mel-frequency cepstral coefficients (MFCCs)*, *short-time Fourier transform (STFT)* spectrograms, and simple spectrograms are commonly used for realistic recordings. Figures 3.3 and 3.4 are examples for STFT spectrogram and log-mel energy spectrum for a same audio clip. After performing one of the mentioned feature extraction techniques, we will have a feature matrix which can be fed into the clustering step in the proposed method. If the dimensionality of the feature matrix is too high, we will consider applying a dimensionality reduction technique to produce a low-dimensional set of features for each column (time bin/window).

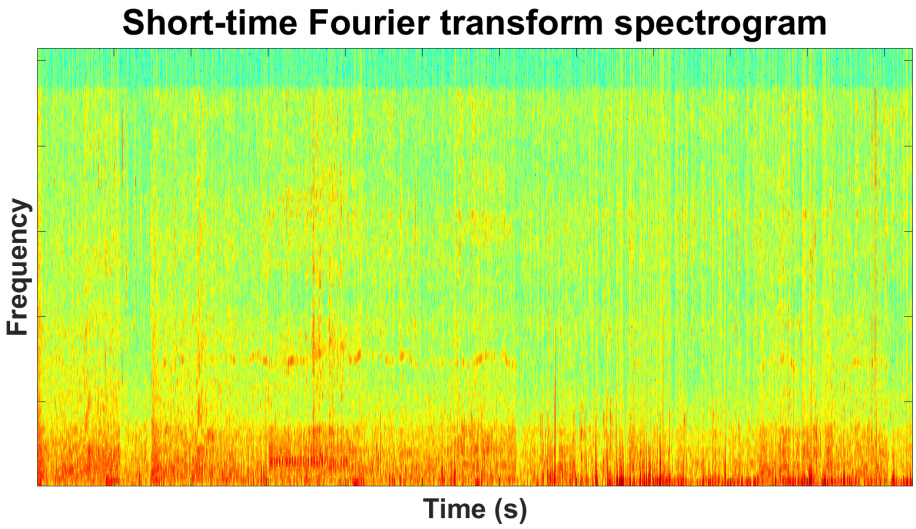


Figure 3.3: Short-time Fourier transform (STFT) spectrogram

At first, we convert the audio signal into a time-frequency representation using the STFT. The STFT reveals the frequency content of a signal of local windows in time and allows us to track this content as it changes over time. We use a Hann window with size 512, a 1024-point DFT (discrete Fourier transform), and a 50% overlap (256 overlapped

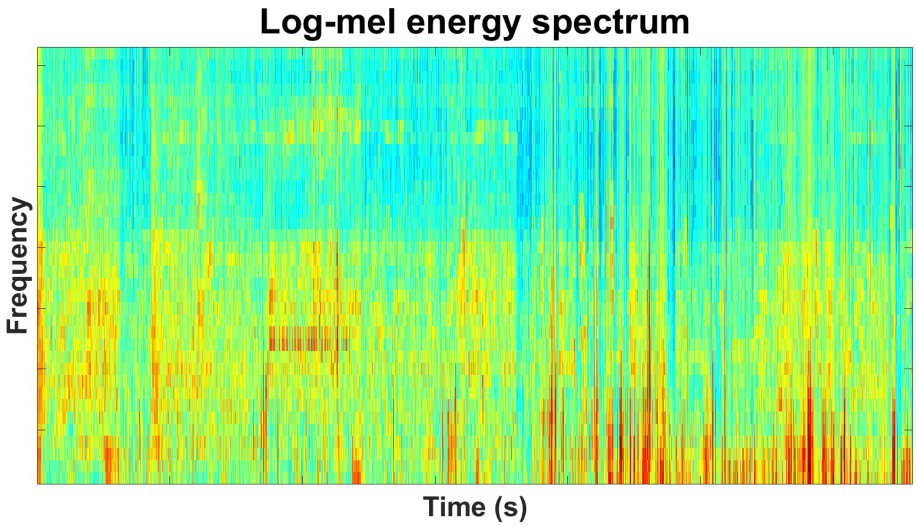


Figure 3.4: Log-mel energy spetrum

samples). The window size is not critical but must be long enough to provide sufficient frequency resolution; however, if it is too long, the temporal aspects of the signal are blurred. The 512 sample length met this criterion so we choose it. The 50% overlapping for Hann window has the advantage that the sum of the overlapping window functions is exactly one everywhere. The constant sum implies a proper reconstruction of the signal after inverse Fourier transform and summation of the overlapping windows. The output of the STFT consists of both magnitude and phase, here we discard the phase and consider only the magnitude. We also test log-mel energy spectrum on the same dataset; however, we finally chose MFCCs as the extracted audio feature to fed into the next stage. The MFCCs function uses a filter bank of 40 half-overlapped triangles. The number of samples in analysis window is set to $round(fs \times 0.05)$, while the number of overlapping samples between adjacent windows is set to $round(fs \times 0.03)$, where $round(\cdot)$ means round to the nearest integer and fs is the sampling frequency in Hertz. The number of coefficients returned for each window of data is set to be 20, specified as an integer in the range $[2, v]$, where v is the number of valid passbands. The number of valid passbands is defined as

$$sum(BandEdges <= floor(\frac{fs}{2}) - 2,$$

where $floor(x)$ rounds each element of x to the nearest integer less than or equal to that element. A passband is valid if its edges fall below $fs/2$. When MFCCs are calculated, the estimation of the first and second derivatives (delta and delta-delta) are included with the original MFCCs in our experiments. The combination of original MFCCs features and dynamic features will be simply referred to as MFCCs in this work.

To summary, our first experiment used the STFT spectrogram. The spectrogram transforms the input audio signal into time-frequency representation and then the audio clip will be represented as a sequence of spectral vectors. The reason for the popular usage of spectrogram is that it can help us visually study sounds and their properties much better. Different from the intuition of transforming audio signal into spectrograms, the mel-frequency based analysis of audio signal, e.g. mel-log energy spectrum and MFCCs, is based on human perception experiments. The human ear naturally acts as a set of filters that are non-uniformly spaced on the frequency axis with more filters located in the low frequency regions. The mel-frequency filters are designed to imitate this pattern. Typically, an audio signal is analyzed over short analysis window using FFT to obtain its spectrum, and then the spectrum is passed through mel-filters to obtain the mel-spectrum. For MFCCs, cepstral analysis is further performed on the obtained mel-spectrum. In these decades, MFCCs are mostly used features in state-of-art speech recognition system.

The reason why we finally choose MFCCs is that there is no significant performance difference when using STFT spectrogram, log-mel energy spectrum, or MFCCs in our proposed framework. However, STFT spectrogram and log-mel energy spectrum require dimensionality reduction to save computational time and cost in clustering and training stage but MFCCs does not. Thus, MFCCs are used as our audio feature. In actual applications of MFCCs, the deltas and the delta-deltas will both be appended to the MFCCs. The MFCCs combined with the deltas and the delta-deltas will be referred as only MFCCs in the following paragraphs. The details for the evaluation on different feature extraction methods can be found in section 5.2.

3.4 Dimensionality reduction

As mentioned in the above section, considering the time-consuming and high computational demand of the training process, we may wish to apply some dimensionality reduction techniques on the feature matrix if we choose STFT spectrogram or log-mel energy spectrum. Similar to the signal enhancement step, the dimensionality reduction step is not always required in the proposed method and it is interrelated with which feature extraction technique is selected in the previous step. For dimensionality reduction techniques, the simplest way is to apply *singular-value decomposition (SVD)* or *Principal Component Analysis (PCA)* on the targeting matrix. For these two dimension reduction techniques, we can eliminate those dimensions that are less important by inspecting the eigenvalue / singular value matrix in the reduced matrix. More complex dimensionality reduction techniques such as *Non-negative Matrix Factorization (NMF)* and *Probabilistic Latent Component Analysis (PLCA)* can also be practical choices. The above mentioned methods are all based on matrix factorization.

In our experiments, if we choose STFT spectrograms or log-mel energies as the acoustic features, SVD, PCA, or NMF are then applied to perform the dimensionality reduction. For SVD, the decomposed feature matrix $\mathbf{U\Sigma V}^T$ will be truncated by a value R, which is typically a number between 20 to 30. We found that 25 is a general and reasonable choice for our datasets. If PCA is applied, we will only preserve and use the first 10 components, Unlike SVD and PCA, we did not find a general solution for NMF since the number of k, which is used to decompose the feature matrix into $\mathbf{W} \times \mathbf{H}$, varies a lot when dealing with different sound events. Considering this nature, we will not use NMF in the following chapters. The AutoEncode is also applied on our FACED framework. The simplest one hidden-layer AutoEncode with 10 neurons in the hidden layer is sufficient for our time-frequency representation matrix. The transfer function for the encoder and decoder are

both logistic sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Considering the dimensionality of different features, the dimensionality reduction step is combined with feature extraction steps. One exception is MFCCs, which we do not apply a dimensionality reduction algorithm on them since the dimensionality is already low enough for our framework. To sum up, the simplest way to perform dimensionality reduction is matrix factorization based methods, such as the singular value decomposition (SVD), principal component analysis (PCA), and non-negative matrix factorization (NMF or NNMF) algorithms. PCA reduces the dimensionality of data by projecting it onto a small set of orthogonal bases that maximizes the amount of variance preserved through the projection; while NMF seeks to decompose a signal into the sum of individual components. Since NMF relies on the assumption that the inner dimension between the dictionary and coefficient matrices is small (relative to the other dimensions) in order to prevent the problem from being under-determined, it effectively limits the number of dictionary atoms that may be used. The extracted audio features can also be view as a set of images; thus, an AutoEncoder is considered as a dimensionality reduction algorithm in our work. Several of dimensionality reduction methods could potentially be applied in our work; however, we would like to use those can provide the most flexible and natural fit for modeling general sound environment. Furthermore, the method should be able to easily track the number of dictionary atoms we need to use in each situation. In the evaluation experiments presented in Chapter 5, we present the results for SVD, PCA, and autoencoder for dimensionality reduction methods since they are relatively easy to tune and track the number of dictionary that may be used. Also, they show the high flexibility to deal with the variety of sound environments in evaluation datasets.

After performing any of the mentioned dimensionality reduction methods on the feature matrix, the size of the feature matrix can be reduced by a certain amount. Since we chose MFCCs as our audio feature, this dimensionality reduction step now becomes an optional

step before we feed the matrix into the next stage. The another reason for dimensionality reduction step being optional in FACED framework is the development of neural network based algorithms. Several architectures of convolutional neural network involve pooling processes, which can be viewed as the a dimensionality reduction step built in the networks. Without using a separate dimensionality reduction method, the extracted feature will be directly fed into the learning step in these kind of methods [79, 80].

3.5 Clustering and forming the training data

After the feature extraction and dimensionality reduction steps, we then apply a clustering algorithm on the feature matrix to partition the columns into several clusters. Our intuition is that different acoustic features in the signal will correspond to distinct clusters. If this is true, then as shown in Fig. 3.5, different categories will have some overlapping clusters—resulting from the common background elements shared by the different categories—and will have some non-overlapping clusters, which can be treated as representatives for each different category. *Following the clustering, data that appears in clusters that are shared by several differently labeled audio clips is assumed to be background data (see cluster 3 and 4 in Fig. 3.5). Clusters associated with only a single class of audio clip are assumed to correspond to the corresponding weak label.*

Clustering algorithms can be categorized based on their cluster model. The most prominent clustering algorithms can be roughly categorized into four types: connectivity-based clustering (hierarchical clustering), centroid-based clustering, distribution-based clustering, and density-based clustering. The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another. In our experiments, we are applying *Gaussian mixture models (GMM)*, *K-means clustering*, and *density-based spatial clustering of applications with noise (DBSCAN)* on our dataset. After testing all the three methods on our experiment datasets, we found no significant performance difference using any of them.

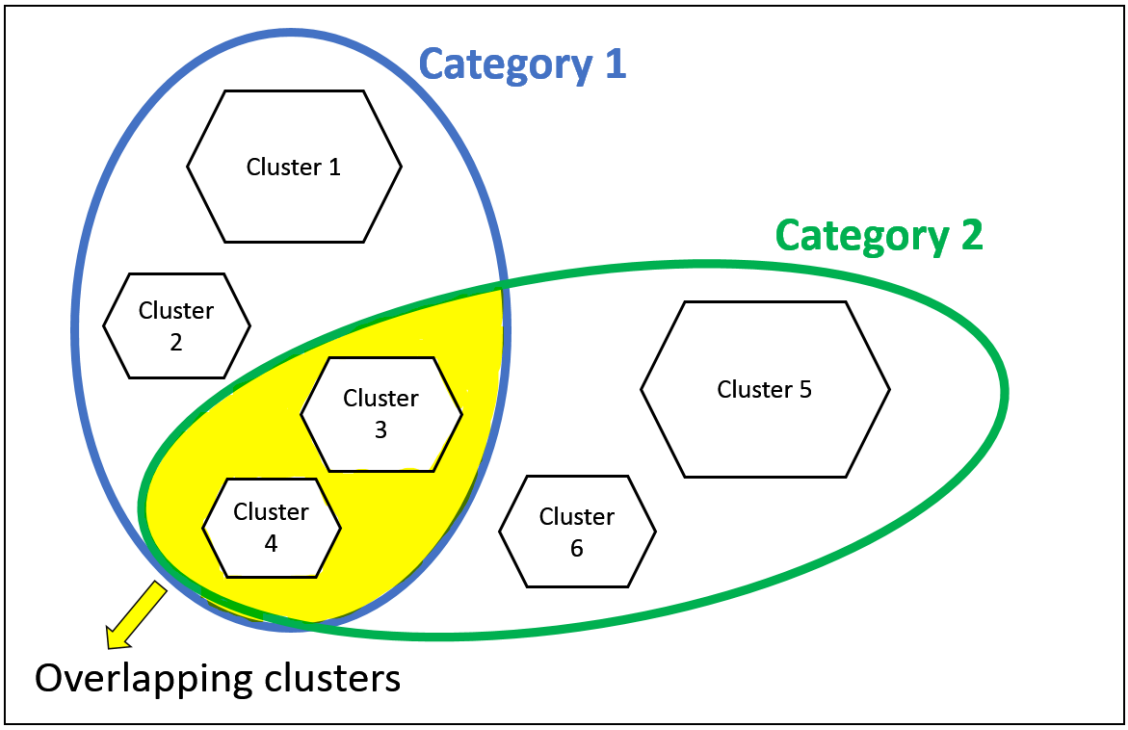


Figure 3.5: Illustration depicting clustering behavior. Note that the two categories share some clusters (corresponding to background features) but also contain clusters unique to each category.

The only concern is how we tune each clustering method on different datasets.

At first, we applied GMM and K-means clustering algorithms on our datasets. They are both simple, popular clustering algorithm that transforms multidimensional data into cluster indices. After performing several experiments, we found that the tuning process of K-means clustering is slightly easier than GMM. The K-means algorithm alternates between assigning observations to the nearest cluster centroid and updating the cluster centroids based on the new membership. In practical cases [81], we apply K-means clustering to partition the columns of $\mathbf{V}^T$ into K clusters. This can be viewed as a way of characterizing the distribution of the columns of $\mathbf{V}^T$, The number of clusters is the only one parameter that we need to carefully consider in K-means algorithm. The number is selected experimentally, and we tested it from a large number such as thirty to a more reasonable number like six, eight, or ten to see which number is suitable for the dataset. Finally, we found that six to eight clusters worked well for every case that we explored. However, we note that

more complex signals may require more clusters, that is to say, the number of clusters still needed to be tuned for each different datasets.

Since K-means assumes the clusters are spherical, it does not work efficiently with complex geometrical shaped data or non-linear data. Thus, we then consider applying density-based clustering algorithm: DBSCAN. Unlike the K-means clustering algorithm, DBSCAN doesn't require us to specify the number of clusters in the initialization process. DBSCAN identifies three kinds of points: core points, border points, and noise points from the input data. What we need to define for DBSCAN algorithm is the value of ε and the minimum number of neighbors required for a core point, where ε is called the epsilon neighborhood of a point. The epsilon neighborhood is specified as a numeric scalar that defines a neighborhood search radius around the point. If the epsilon neighborhood of a point contains at least minimum number of neighbors required for core point neighbors, then DBSCAN identifies the point as a core point. In our experiments, the practical value of ε is 0.3 to 0.5, and the minimum number of neighbors is ten to twenty. They are both selected experimentally depending on the dataset. However, after testing for all these mentioned clustering methods, we found out that the performance difference is really close from each other (within two percent). Thus, what actually influences the experiment results in this step is how well we tuned the parameters in each clustering methods.

The clustering algorithm results in each time bin being assigned a cluster label; but, this process is somewhat noisy and having data belong to a particular cluster is not necessarily a good class indicator. However, in practice what is often needed is a label associated with a slightly longer time period such as the duration of the sound or a short audio segment. For our data, the time period for a specific activity can last for seconds, but each second will have hundreds of time bins. Thus, to construct training vectors for all the time period, we calculate empirical histograms which capture the distribution across the clusters within each time window. The length of the time window is set to be half second – long enough to capture a brief impact sound or a sustained sound according to the activities that we were

trying to detect and classify. The half-second periods are then used in the final classification step.

3.6 Classification algorithm

Following clustering, we then form a set of training data to be used by standard supervised learning techniques. The simplest and most common algorithm is support vector machine (SVM) [82, 83]. The input to the SVM is the normalized cluster membership histogram over the time-period of interest (1 second in our case). Two example training vectors are shown in Fig. 3.6. The two training vectors (rows) correspond to different events or categories in the audio clip and the columns capture the percentage of time bins that belonged to each cluster over a one-second window. We can repeat this process for many such windows for both categories of interest to form training data for each class, which can then be used to build a simple decision rule for classifying future data using SVMs.

More concretely, we let $(\mathbf{x}_i, y_i)$, $i = 1, 2, \ldots, n$ denote the training data where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector and $y_i \in \{+1, -1\}$ indicates the class of $\mathbf{x}_i$. The SVM training procedure involves solving the following optimization problem [64]:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\begin{aligned} \text{s.t.} \qquad & y_i(k(\mathbf{w}, \mathbf{x}_i) + b) \geq 1 - \xi_i \qquad && \text{for } i = 1, 2, \ldots, n \\ & \xi_i \geq 0 && \text{for } i = 1, 2, \ldots, n \end{aligned}$$

Above, $C \geq 0$ is a tradeoff parameter that controls overfitting, and $k(\mathbf{w}, \boldsymbol{\xi})$ represents a *kernel* function which mathematically captures the similarity between the vectors $\mathbf{w}$ and

$\boldsymbol{\xi}$. The solution to the SVM optimization problem yields a vector $\mathbf{w}$ and an offset b from which we can make future predictions via the simple decision rule given by

$$f_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(k(\mathbf{w}, \mathbf{x}) + b).$$

We use the *radial basis function (RBF)* kernel which was found to yield better performance than the linear kernel in our tests. The *RBF* kernel is given by

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2),$$

where $\gamma \geq 0$ is a bandwidth parameter which we must select.

To train the SVM, we use the LIBSVM package in MATLAB [67]. The parameters C (trade-off parameter) and γ (bandwidth parameter) are selected by considering a log-scale range from 2^{-7} to 2^6. (Note that we select the parameters independently for each dataset.) We use 10-fold cross validation to select the appropriate values of C and γ.

Training Vector

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Category 1	0.4	0.2	0.2	0.18	0.01	0.01
Category 2	0.02	0.03	0.17	0.18	0.22	0.38

Figure 3.6: Illustration of training vector

Besides SVM, several neural network based classification methods are also considered in our framework. As noted before, most publicly available datasets only have less than an hour of audio data for each event. Moreover, for realistic recordings, only a few minutes of audio data per acoustic event is available in general. Thus, considering the small data size, deep neural network will not be a good choice. Also, since we do not need to reduce the input matrix size to lower dimension in the layers, several types of convolution neural network (CNN) will not be considered. In our experiments, we applied a fully connected

neural network (FNN) with one or two hidden layers to perform the classification. The number of neurons in the hidden layers is selected experimentally. We found that sixty-four to sixty is practical choices for our datasets in single layer case; while for two hidden layer networks, sixty four to sixty neurons in the first hidden layer and thirty two to thirty neurons in the second hidden layer are our experimental setups. The details for selecting parameters and structures will later be discussed in Chapte 5. All the above mentioned neural networks are implemented under the PyTorch environment.

CHAPTER 4

DATA

In order to evaluate the performance of the proposed research, we applied it to several different datasets.

4.1 Synthetic dataset

The first dataset we used is a synthetic dataset. The synthetic dataset consisted of audio spectrograms, generated so that each looked as though it contained multiple segments of sound each containing sound events from one of two classes interspersed with background sounds. To generate the synthetic dataset, a random sequence of states (corresponding to a sequence of sound types) was generated and then for each labeled segment we further generate a random sequence of background and sound events consistent with the label. For different event types, the spectral peaks were noticeably different. For each event type, the spectral peaks also varied but in a smaller range. State 0 corresponded to environmental noise in the real recordings and so consisted of randomly generated Gaussian distributions with large standard deviation σ. For state 1 and state 2, both consisted of randomly generated Gaussian distributions with similar standard deviation but different mean μ of the distribution so that each state can represent different categories in real-life datasets. Finally, the synthetic spectrograms are are blurred temporally (convolution kernel [0.5 1 0.5]) to make the transitions between states less distinct and more realistic. Each synthetically generated spectrogram column was labelled according to its type, for example, event 1, or event 2 (state 1 and state 2 in Gaussian distributions). Figure. 4.1 illustrates the spectrogram of a short segment of the synthetic dataset.

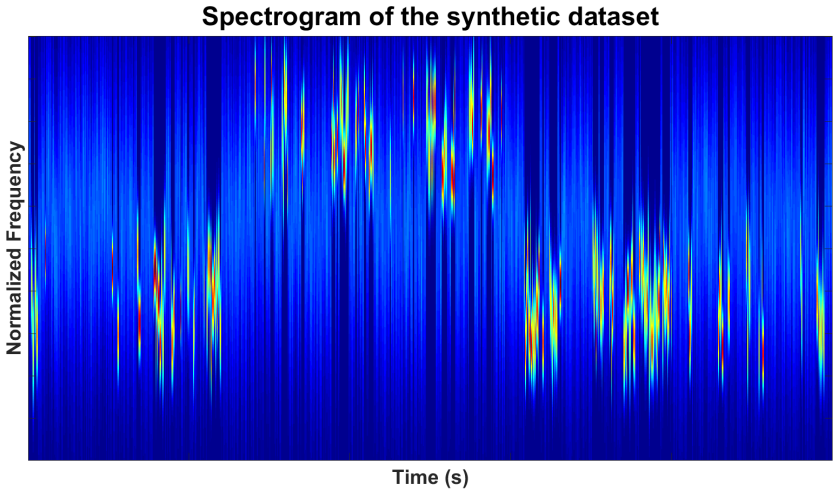


Figure 4.1: A segment of the synthetic dataset

4.2 Real-world dataset

The second dataset is provided by Dr. Rashidi and his students in Department of Civil and Environmental Engineering at the University of Utah. It consists of audio recordings of several different pieces of construction machines operating at various jobsites selected as case studies. Ten of the construction machines are listed below: 1) JD333E Compact Loader, 2) JD50D Compact Backhoe, 3) Ingersoll Rand Compactor, 4) CAT 320E Excavator, 5) Komatsu PC200 Excavator, 6) JD 700J Dozer, 7) Hitachi 50U Excavator, 8) Concrete Mixer, 9) JD 270C Backhoe, and 10) Bobcat 331 Mini Excavator. Each machine was carefully monitored and the generated sounds while performing routine tasks were captured using a commercially available recorder (Tascam DR-05). Figure 4.2 shows the microphone set up in a construction jobsite. In parallel to recording generated sound patterns, a smart phone was used to video tape the entire scene. The captured video files will be used later to manually label the audio file and classify different activities and thus, generate the validation benchmark (or ground truth data). The manual label was used as ground truth label in the testing experiments. Heavy construction equipment usually performs one major task (digging, loading, breaking, etc.) and one or more minor tasks (maneuvering, swinging,

moving, etc.) in each cycle. Therefore, each audio file had two labels based on the two activities: major and minor (or activity 1 and activity 2). Also, within each activity time period, there will be some inactive times which only contain environmental noise in the recording. Thus, we will have audio clips each labeled with up to two activities, but these labels do not contain the information as what time the specific events occur in the clip. A large portion of the recordings might be environmental noise, which corresponds to state 0 in our synthetic dataset. For example, one recording for JD 700J Dozer could be manually labeled as "digging" from 0s to 30s but it might only dig for 10 seconds in this 30 seconds period. Each labeled audio file was sent through our proposed framework and divided into activities 1 and 2. Finally, the performance of the algorithm for each case study has been compared to manually labeled files.

Figure 4.2: The microphone array used for collecting audio files for this project (left) and placing audio recorders at a jobsite (right)

4.3 DCASE dataset

A publicity available dataset provided by *Detection and Classification of Acoustic Scenes and Events Challenge (DCASE)* is also used to test our proposed framework. The one we used is *TUT Sound Events 2017 dataset*. Audio in the dataset consists of recordings of street acoustic scenes with various levels of traffic and other activity. Each scene was se-

lected as representing an environment of interest for detection of sound events related to human activities and hazard situations. The dataset was collected in Finland by Tampere University of Technology between June 2015 to January 2016. The recordings were captured each in a different streets. For each recording location, a three– to five–minute–long audio recording was captured. The equipment used for recording consists of a binaural Soundman OKM II Klassik/studio A3 electret in-ear microphone and a Roland Edirol R-09 wave recorder using 44.1 kHz sampling rate and 24 bit resolution. Individual sound events in each recording were annotated by the same person using freely chosen labels for sounds. Nouns were used to characterize the sound source, and verbs to characterize the sound production mechanism, using a noun-verb pair whenever this was possible. The annotator was instructed to annotate all audible sound events, decide the start time and end time of the sounds as he sees fit, and choose event labels freely. This resulted in a large set of raw labels. Target sound event classes were selected to represent common sounds related to human presence and traffic. Mapping of the raw labels was performed, merging sounds into classes described by their source before selecting target classes. Target sound event classes for the dataset were selected based on the frequency of the obtained labels, resulting in selection of most common sounds for the street acoustic scene, in sufficient numbers for learning acoustic models. The mapping of the raw labels merged sounds into classes described by their source, for example "car passing by," "car engine running," "car idling," etc. into "car," sounds produced by buses and trucks into "large vehicle," "children yelling," and "children talking" into "children," etc. Thus, for a time period which is labelled as a specific class like "car", it still contains the sparse manifestation during the on-set time. In the "car" case, there will be about ten seconds long segment labelled as "car" class in the recording but only two to three seconds in the segment are actually contain active sound. We cannot hear "car" sound in the rest of the labelled segment, which might be corresponding to "car idling" or "car passing by" events in the manually labeling procedure. With this labelling methodology, we treated this dataset as the weakly-labeled.

The sound classes in this dataset are:

1. brakes squeaking
2. car
3. children
4. large vehicle
5. people speaking
6. people walking

Besides *TUT Sound Events 2017 dataset*, we also used *TUT Rare Sound Events 2017 dataset* in our experiments. The labelling methodology are the same but *TUT Rare Sound Events 2017 dataset* contains different sound classes, which are "baby crying", "glass breaking", and "gunshot." *TUT Rare Sound Events 2017 dataset* consists of source files for creating mixtures of above mentioned events with background audio, as well a set of readily generated mixtures and recipes for generating them. The background recordings are from 15 different acoustic scenes, and the recordings with the target rare sound events from three classes, accompanied by annotations of their temporal occurrences. The dataset consists of two subsets (training and testing), each containing 1500 mixtures (500 per target class in each subset, with half of the mixtures not containing any target class events).

CHAPTER 5
EVALUATION ON FACED FRAMEWORK

5.1 Evaluation experiments setup

To evaluate our FACED framework, we performed several experiments on *TUT Rare Sound Events 2017 dataset* provided by the DCASE challenge. The experiments can help us know which algorithm is suitable in each step within FACED framework. Since TUT Rare Sound Events 2017 dataset contains various different sound events, the choices of algorithms used for this dataset can be viewed as a general choice for audio classification and event detection. We used eighty percent of TUT Rare Sound Events 2017 development dataset as our training data, and the remaining twenty percent is used as a testing dataset. This dataset was developed for audio tagging but FACED framework is focused on audio classification and event detection. Thus, we made a modification to this dataset to fit our evaluation purpose. The ground truth is provided in the dataset but we do not use the strong labels. We only preserve the clip-level meta-data for each recording; for example, the class *label* "children" is preserved but *not* the precise on-set and off-set times. To generate the training dataset used for audio classification, we randomly selected ten recordings from the TUT Rare Sound Events 2017 development dataset and then concatenated them together to build a longer recording. These longer recordings contain different sound events and each section of the concatenated recording is weakly-labeled. Each original audio recording has only one sound event; thus, we built these longer recordings with several different events. Also, we ensured the concatenated recordings have at least two different sound events. The concatenating process was performed on the whole TUT Rare Sound Events 2017 development dataset with each clip being used only once. These concatenated recordings are used in both training and testing experiments. This process is then repeated for ten

times in the following evaluation experiments. We averaged the ten experimental results and show them in the following result tables.

5.2 Feature extraction and dimensionality reduction

The first step in the FACED framework is extracting audio features from the provided recordings, thus, we considered different audio features to see which are most suitable under different background environment and sound events. Considering the dimensionality of different features, the dimensionality reduction step is combined with feature extraction steps. As mentioned in chapter 2, although feature extraction methods can be viewed as a form of dimensionality reduction, they are typically not tuned to a specific situation unless the features are designed by hand. Passing extracted features through a dimensionality reduction algorithm can help us gain further information which is more relevant to the current task. This process can also speed up the following learning process and reduce the overfitting situation.

In our case, the input recordings contain not only human speech but with various types of sounds; thus, the evaluation experiments help us find a general choice within these feature extraction methods. Since experimental recordings will have at least two different sound events, the evaluation experiments show the ability for each feature extraction method when dealing with multi-event recordings. Each concatenated recording in the testing dataset was fed into the FACED framework to detect and label the sound events in it. The labelling window size is a half second in our experiments. We implemented the FACED framework with different feature extraction methods on the training dataset, and the experimental results are shown in the following Table 5.1. We use a Hann window with size 512, a 1024-point DFT (discrete Fourier transform), and a 50% overlap (256 overlapped samples) for the STFT spectrogram. For log mel-band energies, the audio clips are analyzed by 40 ms frames with 50% hop size, and 40 bands are implemented. The MFCCs in Table 5.1 consist of 20 MFCC coefficients, 20 delta MFCC coefficients, and 20

delta-delta coefficients. Thus, total of 60 coefficients are used as the acoustic features. In the other hand, the dimesionality reduction methods, we preserved the first 25 columns of $\mathbf{U}$ and the first 25 rows and columns of $\mathbf{\Sigma}$ when applying SVD on the extracted acoustic feature matrices. For PCA, the first thirty pricipal components are taken in the experiments. A one hidden-layer AutoEncode with 10 neurons in the hidden layer is used in this evaluation experiments. The transfer function for the encoder and decoder are both logistic sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}.$$

For the evaluation purpose, we fixed other steps in the FACED framework. The clustering algorithm is set to be DBSCAN; while the classification method is fully-connected neural network (FNN) in this evaluation experiments. The values in the following tables are the average labelling accuracy using FACED framework for audio event detection.

	STFT spectrogram	log mel-band energies	MFCCs
SVD	84.6%	85.2%	x
PCA	84.1%	84.9%	x
Autoencoder	82.1%	83.6%	x
Without dimensionality reduction	85.3%	88.7%	86.8%

Table 5.1: Detecting performance using FACED framework under different feature extraction methods on TUT Rare Sound Events 2017 dataset

As shown in the Table 5.1, the best feature extraction method for the evaluation dataset is using mel-log energy spectrum without any dimensionality reduction methods. Especially in the no dimensionality reduction cases, the detecting and labelling accuracy indicates that log mel-band energies outperform other methods among the feature extraction methods. After we apply any dimensionality reduction methods on the original features, the accuracy will slightly decreased. The reason might be the elimination and distortion of several information in the features during the dimensionality reduction process. Considering the feature matrix size and time cost to run this step in FACED framework, MFCCs

can give us a good result with a relatively low dimensionality and computational cost; thus, we chose MFCCs as our audio features to be fed into further experiments. Noted that the MFCCs here refer to the combination of MFCCs, delta features, and delta-delta features.

5.3 Clustering algorithm

For clustering algorithms, we compared three popular and efficient methods: GMM, K-means, and DBSCAN. For Gaussian mixture models (GMMs), it is assumed that all the data points are generated from a probability distribution that can be modeled as a mixture of a finite number of Gaussian distributions with unknown parameters. We implement the expectation-maximization (EM) algorithm for fitting mixture-of-Gaussian models, and compute the Bayesian Information Criterion to assess the number of clusters in the data. For K-means clustering, we use k-d trees for the nearest neighbor search and intelligent initialization to speed up our process. All the DBSCAN, GMM, and k-means clustering are using the "scikit-learn" machine learning tools in Python environment [84].

Similar to the evaluation process in the feature extraction step, the features that fed into these methods are MFCCs of the concatenated recordings; while the classification method is fully-connected neural network (FNN). The experimental results are show in Table 5.2. We set up several different experimental parameter sets for the evaluation process. The ideal case is that the clusters can clearly present different sound events. However, in real cases, part of the clusters are corresponding to the shared environmental and background sounds. Thus, the number of cluster should be sufficient enough to present both non-overlapping and overlapping components of the audio clips. The cluster numbers for both GMM and K-means are first set to be a large number and then reduced to a certain amount. In our case, we evaluated cluster numbers from thirty to five to see how the clustering methods perform on the evaluation datasets. As mentioned in section 3.5, we found that about eight clusters give a suitable mapping of clusters for general usage. In general, two to three clusters can present a specific sound events. Increasing the number of clusters

will relatively decrease the distance between each cluster, which will make it more difficult for us to classify each cluster into different sound events. By setting a suitable number of clusters in GMM and K-means (eight to ten clusters in our cases), we can better indicate each cluster to be the member of each sound events. For evaluation purpose, we fixed the number of clusters to eight in the experiments for both GMM and K-means. Different from GMM and K-means, DBSCAN needs to initialize the the value of ε and the minimum number of neighbors. Considering the size of data points in the extracted features from the previous step, ε is set to be 0.4 and the minimum number of neighbors is fifteen. All the above mentioned parameters are selected by experiments.

	Detection accuracy
GMM	84.5%
K-means	85.9%
DBSCAN	86.8%

Table 5.2: Detecting performance using FACED framework with different clustering methods on TUT Rare Sound Events 2017 dataset

As mentioned in section 3.5, the performance for the three clustering methods are close but K-means and DBSCAN are slightly better than GMM in our experiments. Since the results in Table 5.2 are average result values, DBSCAN is not always the best choice but it is a great choice in general. For GMM and K-means, the centroids of clusters will highly influence the results of clustering so we need to carefully consider the initialization process of GMM and K-means. For DBSCAN, we can think of the high density clusters in DBSCAN as representing the different sound events with their distinct features, and we do not need to tune the centroids and initialization process. One concern for DBSCAN in our experiments is that if only a few segments of an audio clip are the target sound event and most of the clip is background environmental sound, DBSCAN cannot construct the clusters for features of the targeted sound event well. Since the targeted sound events in the evaluation datasets are long enough and clearly distinguishable, this is not a problem with our evaluation datasets but it is a concern to be considered for other datasets. After

performing the DBSCAN on the training datasets, we will have the mappings for the data points of the extracted audio features. When feeding a testing data into the clustering step in FACED framework, the k-nearest neighbors (KNN) algorithm is performed to label each data point to a specific cluster in the constructed mapping. Similar processing is conducted for both GMM and K-means. In this clustering step, one clustering algorithm is first performed on the training datasets, and then the KNN is performed for each testing data point to locate its cluster. Based on the experimental results, we used DBSCAN in all future evaluation experiments. However, both K-means clustering and DBSCAN performed well and are also effective choices for use in the FACED framework. The only one concern for the clustering step is how well we tuned the parameters in the chosen clustering algorithm to deal with different soundscapes.

5.4 Classification methods

Similar evaluation experiments are also performed to evaluate classification methods for the FACED framework. We used different classification methods on the constructed training vectors generated by MFCCs and DBSCAN clustering. Current state-of-art classification algorithms are all based on neural-network structures. As we can see on the popular sound event detection challenges in these years, the convolutional neural network (CNN), recurrent neural network (RNN), or even joint neural networks, i.e. convolutional recurrent neural network (CRNN) [85], are frequently used in others' experiments. These algorithms performed really well on large-scale datasets, but they are not quite fit our research intuition. In our experiments, we targeted relatively small size datasets of actual recordings. Thus, we chose support vector machines (SVMs), a traditional classification algorithm, and a shallow fully connected neural networks (FNNs) in our experiments. The network structure in the fully connected neural network is not that complex as CNN and RNN, but it is still able to construct a robust network model to perform the classification task. For SVMs, we use the LIBSVM package in MATLAB [67]. The trade-off parameter C and

bandwidth parameter γ are selected experimentally between 2^{-5} to 2^{5} depending on each epoch. The kernel is set to be the RBF kernel. Besides SVMs, we tested several different structures of neural network based classification method. We found that a fully connected neural network (FNN) with two hidden layers performs well in the evaluation experiments. The FNN consists of input layer, a hidden layer, a bottleneck layer, and a output layer. The number of neurons in the hidden layer and the bottleneck layer is selected based on the number of neurons in the input layer. Typically, there will be 100 neurons in the hidden layer and 50 neurons in the bottleneck layer. A non-linear Rectified Linear Unit (ReLU) function is applied in the FNN to learn the weights and bias. The optimizer is Adam [86] with 10^{-3} learning rate. The above mentioned neural networks are implemented under PyTorch environment. The evaluation results are shown in the following table 5.3.

Method	Detection accuracy
SVM	82.8%
FNN (one hidden layer)	86.8%
FNN (two hidden layers)	87.3%

Table 5.3: Detecting performance using FACED framework with different classification methods on TUT Rare Sound Events 2017 dataset

As shown in the Table 5.3, FNN performs better than SVM on the evaluation datasets. The SVM in the above table is using RBF kernel with trade-off parameter $C = 2^{-2}$ and bandwidth parameter $\gamma = 2^{9}$. We tested several kernels and combination of SVM parameters and we found this combination is a general choice for this evaluation dataset. To avoid the overfitting problem, the ten-fold cross validation is applied during the classification step. Considering the comparison of SVM and FNN, the performance of FNN does not drastically outperform SVM but it still shows a five percent difference between SVM and FNN. Since our datasets are multi-class datasets, SVM is not a great choice when facing multi-class cases, which performs better on binary classification problems. Different from SVM, FNN has no limitation on the number of classes. We tested several different structures of FNN and we found that increasing the number of hidden layers will not help

to improve the performance much after two hidden layers. For FNN with three more hidden layers, the computation time will significantly increase to five more times than using FNN with one or two hidden layers, however, the classification performance will only be improved by less than one percentage. The computation cost for SVM and FNN with one or two hidden layers are pretty similar. We selected FNN with two hidden layers as the classification method in FACED framework because it is more flexible and adjustable than SVM when facing different soundscapes. Typically, FNN can provide multi-class outputs but SVM is fundamentally a binary classifier.

After performing the evaluation experiments on each step in FACED framework, we selected a general combination to test the FACED framework on other datasets. The selected framework starts with transforming the input audio clip into MFCCs representation, and then feeding the MFCCs into DBSCAN clustering to construct the training vectors. The training vectors are then used as the input for FNN with two hidden layers to construct the classification model. The illustration for training steps is shown in Figure. 5.1.

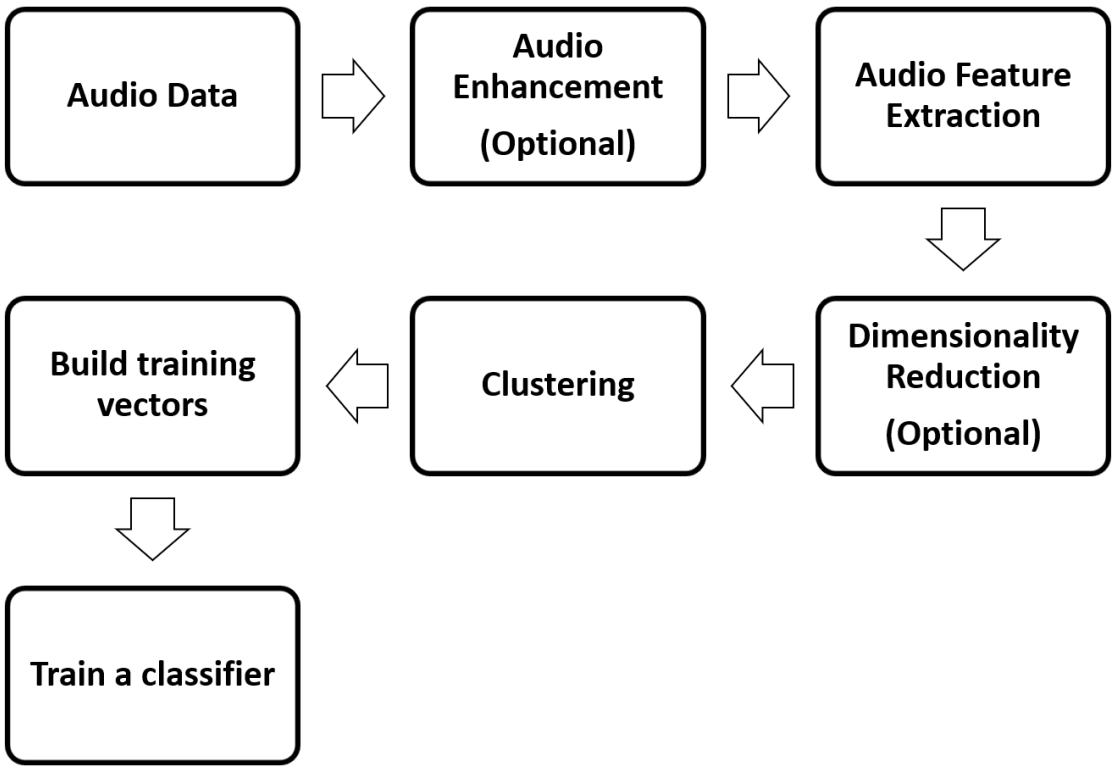


Figure 5.1: Block diagram for training steps

For the testing process, the first step is also transforming the audio clip into MFCCs representation. After extracting the MFCCs, we applied the KNN algorithm to label each data point into the trained clustering mapping to construct the testing vector. We then used the trained FNN model to detect the sound event present in this input window. The window segment is set be be 0.5 second in our experiments. The comparison of this combination of FACED framework and other research is discussed in the next chapter. Figure. 5.2 briefly illustrates the testing process.

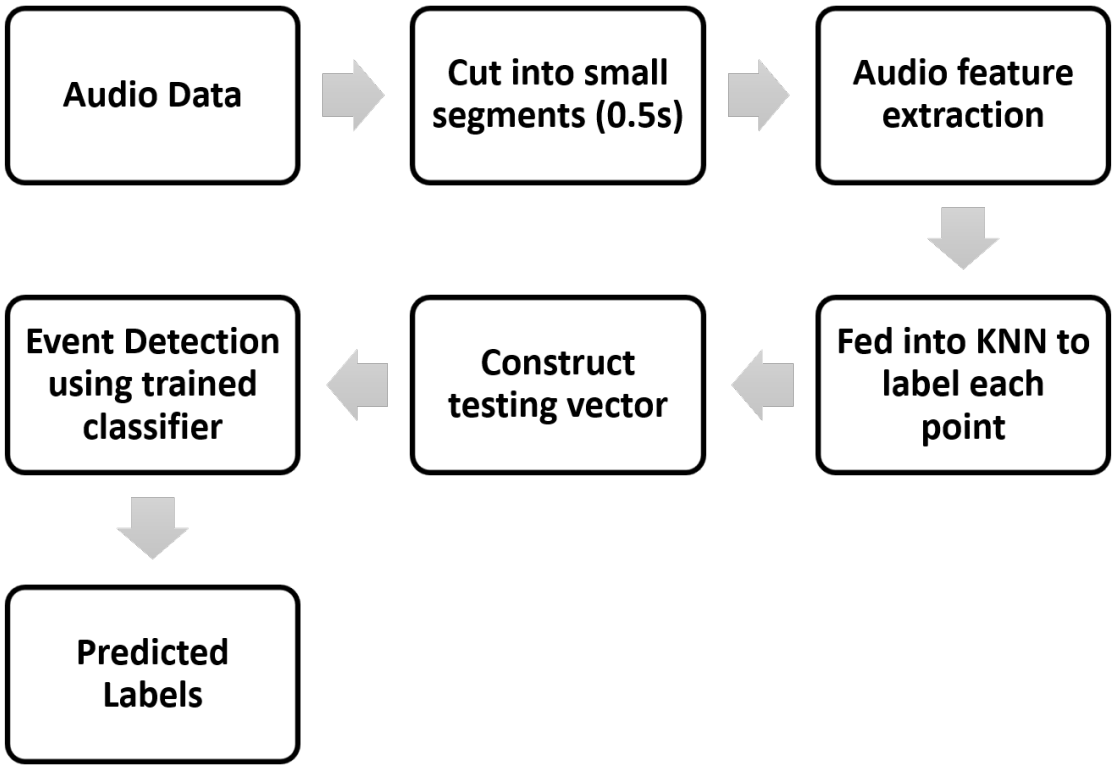


Figure 5.2: Block diagram for testing steps

As mentioned in previous sections, the advantage of the FACED framework is that it is flexible to face different soundscapes. The selected methods above are used for experimental purpose. It is found to be a general choice but not a best one. For real cases, we need to test and tune each step carefully in FACED framework when dealing with different datasets.

CHAPTER 6
COMPARISON

6.1 Introduction

The FACED framework has two major applications. The first one is to detect what kind of audio events occur in a given audio clip. The second one is labeling the time period of audio events in audio clips. The main difference is whether we care about the time label for each audio event or not. Since the datasets we are facing are generally weakly-labelled, the major challenge of the FACED framework appears when we try to do the second application (audio tagging) on any given audio clips. However, due to the nature of this work, it is difficult to get definitive quantitative results on the performance of our FACED framework for tasks such as weakly labeled sound event detection. This is primarily because the vast majority of our experimental data is weakly labeled, but also in part because the question of exactly what should and should not constitute a specific sound event is highly subjective and task-dependent. For weakly labelled datasets, we lack the ground truth of event labels to quantitative precisely the performance of FACED framework. To reduce the subjectivity, we used several strongly labeled datasets but we removed the ground truth and treated it as weakly labeled datasets when training the classification model. The only one information preserved for training is the clip-level label for each audio recording. For example, the label of a randomly selected audio clip will be only “street”, “dog bark”, or “glass breaking” without any temporal information. In order to compare the performance of the FACED framework with other existing methods for detecting and labelling, we chose two popular frameworks from the DCASE challenge to compare against. In general, the popular existing audio event detecting methods consist of data augmentation, feature extraction, a classifier, and a decision maker. Unlike these existing methods, the FACED

framework does not perform data augmentation on either training nor testing data. Furthermore, several deep learning techniques such as convolutional recurrent neural networks, deep neural networks, and long short term memory networks (LSTMs) are frequently used in audio event detection challenges these years. We will apply both FACED framework and two popular audio event detection methods on datasets provides by DCASE challenge [87] to see the differences and their pros and cons. The comparison will be in two parts: performing audio event detection (training and testing) using full datasets and using reduced datasets. The reason for using reduced datasets is to imitate the small size datasets and hand-recorded audio clips. The details of audio event methods, datasets, and experimental results are listed in the following sections.

6.2 Methods

Audio event detection methods vary significantly due to the variety of sound events. However, several popular choices still exist for people when facing with audio event detection problems. We chose two popular state-of-art methods to compare with the FACED framework. The first method consists of four parts: extracting log mel-band energies, converting spectral features with a convolutional neural network, incorporating temporal dependency with fully connected layer, and determining the presence and the onset time of audio event with post-processing [11, 88, 79, 80]. There are multiple choices for the temporal dependency processing: a simple dense layer, a fully recurrent neural network, long short-term memory (LSTM) network, or various types of artificial recurrent neural network. Since the architecture of RNN-based networks varies a lot, we used a fully-connected dense layer in our comparison experiments. The second method is based on MFCCs and GMM [15, 72, 89, 90]. The reason for choosing these two methods is that the first method appears frequently in the top results of DCASE challenges from 2017 to 2019; while the second method is usually selected as the baseline system in sound event detection challenges. The following sections describe these two methods in greater detail. We also apply our prelim-

inary work [81] on these datasets to compare against the FACED framework work in this chapter.

6.2.1 Method 1

Method 1 first cuts each audio into 10 seconds segments, and then extracts log mel-band energies are extracted for each segment. A network consisting of two CNN layers and one fully connected layer is trained to assign scene labels to the audio signals. The audio clips are analyzed by 40 ms frames with 50 percent hop size, and 40 bands are implemented for log mel-band energies. The architecture for the neural network is listed below.

1. First CNN layer
    - 2D convolutional layer with kernel size 7 (single channel with 32 filters)
    - Zero padding
    - Batch normalization
    - Rectified Linear Unit (ReLU) activation function
    - 2D max pooling with pool size $(5, 5)$
    - 25% Dropout
2. Second CNN layer 2
    - 2D convolutional layer with kernel size 7 (single channel with 64 filters)
    - Zero padding
    - Batch normalization
    - Rectified Linear Unit (ReLU) activation function
    - 2D max pooling with pool size $(4, 100)$
    - 25% Dropout

3. Flattening

4. A dense layer

    - 100 units
    - Rectified Linear Unit (ReLU) activation function

5. Output layer

6. Softmax

The learning process contains 300 epochs with 16 batch size. We also shuffled the data between each epoch. The optimizer used in experiments is Adam [86] with 10^{-3} learning rate. In order to validate the learning process, about 30% of the original training data is assigned to be the validation set. After each learning epoch, the performance is evaluated on the validation set, and best performing model is finally selected. The post processing is the decision stage and the decision is made based on a threshold value 0.5. If there are multiple class values over the threshold, the most probable target class is chosen. If all the values are under the threshold, the segment will be labeled as unknown. The method is implemented using Python version 3.6.

6.2.2 Method 2

The second method is relatively straight forward compared to the first method. The audio clips are also analyzed by 40 ms frames with 50 percent hop size. A total of sixteen Gaussian distributions are used to create each acoustic event class model. The feature vector contains the following three components.

- 20 MFCC static coefficients (including 0th)
- 20 delta MFCC coefficients
- 20 acceleration MFCC coefficients (delta-delta)

There are total of sixty values in the feature vector. The test and training process is processed on the full audio clip with cross-validation. The classification accuracy is averaged over folds. The Python environment is also 3.6 for the second method.

6.2.3 FACED framework

The details of methods used in FACED framework have been evaluated in chapter 3 and 5. The specific methods we chose to be used in the comparison are listed below.

1. Signal enhancement: No (using the original data)
2. Acoustic feature: MFCCs with delta and delta-deltas
3. Dimensionality reduction: No
4. Clustering: DBSCAN
5. Classification model: A fully connected neural network
    - 100 neurons in the hidden layer and 50 neurons in the bottleneck layer, Rectified Linear Unit (ReLU) activation function used in the layers
    - Output layer
    - Softmax
    - Adam optimizer with 10^{-3} learning rate

6.3 Datasets

As mentioned in chapter 4, we have three different datasets in our work: synthetic dataset, real-world recordings, and DCASE challenge datasets. In the comparison experiments, we will first use all of the datasets to see how FACED framework performs compared to other methods. After using the full dataset, the synthetic dataset and DCASE challenge datasets will be shrunk gradually. The reason for not shrinking the real-world recordings is that

the data size is already small compared to DCASE challenge datasets. The length for each sound event in DCASE challenge datasets can be more than 4 hours; while the length for each real-world recording may only last for a half hour. The shrunk datasets will be trained and tested using the same method listed in previous section to see their performance.

6.4 Using full dataset

6.4.1 Synthetic dataset

For the synthetic dataset, we generated a equivalent 12-hour training data and an 1.5-hour testing data. This data size is selected to be close to the data size of DCASE challenge dataset. We found that our method worked very well even if we made the classes relatively similar, noisy, and blurred. Under those circumstances, it still performed in a similar manner with other methods. As shown in Table 6.1, it is obvious that the popular algorithms have the ability to identify different categories with this synthetic weakly-labeled training data, while the FACED framework can complete the identification task slightly better than our preliminary work. The detection accuracy in Table 6.1 is calculated by comparing the predicted labels with the ground truth.

	Detection accuracy
Method 1	99.8%
Method 2	98.5%
Preliminary work	99.0%
FACED framework	99.6%

Table 6.1: Detection performance using different methods with synthetic dataset

6.4.2 Real-world dataset

The real-world dataset consists of 10 different pieces of construction machines operating at various jobsites selected as case studies: 1) JD333E Compact Loader, 2) JD50D Compact Backhoe, 3) Ingersoll Rand Compactor, 4) CAT 320E Excavator, 5) Komatsu PC200

Excavator, 6) JD 700J Dozer, 7) Hitachi 50U Excavator, 8) Concrete Mixer, 9) JD 270C Backhoe, and 10) Bobcat 331 Mini Excavator. For these recordings, as a more realistic dataset, we would generally expect the accuracy of all approaches to be lower than in the synthetic datasets. The event Detection performance of the different methods for each case study has been compared to manually labeled files. The comparison results are depicted in Table 6.2.

Machine	Method 1	Method 2	Preliminary Work	FACED framework
JD333E	85.50%	79.10%	81.79%	83.25%
JD50D	84.81%	68.79%	78.27%	80.05%
IR compactor	85.66%	81.22%	79.37%	82.69%
CAT320E	82.18%	74.39%	79.33%	81.75%
Komatsu PC200	83.97%	78.44%	79.36%	81.94%
JD700J	80.57%	76.87%	79.99%	79.81%
Hitachi 50U	86.11%	80.17%	80.25%	82.57%
Concrete Mixer	82.37%	80.26%	80.12%	80.63%
JD 270C	81.08%	76.89%	77.00%	80.98%
Bobcat 331	80.22%	70.09%	75.69%	78.88%
Average	82.95%	76.62%	79.81%	81.26%

Table 6.2: Detection performance using different methods with real-world dataset

As shown in Table 6.2, the FACED framework performs relatively well and generally outperformed both the method 2 and our preliminary work. The average accuracy of the FACED framework was 81.59%. This is better overall than the method 2 (GMM-based classification), which can be viewed as the baseline system within there method. In general, method 1 (CNN-based classification) performs the best when using this dataset. This shows the similar results as presented in DCASE challenges this years. The top-notched results are all using convolutional neural networks jointed with other neural networks as their classification algorithm. Although the FACED framework can not outperform the CNN-based method in this dataset, it still presents a great performance in this comparison. As presented in Table 6.2, method 2 cannot provide a stable classification accuracy while the rest of methods all present relatively stable detecting results. The possible reason is

that only method 2 directly used the acoustic features as its training data when building the classifier. The actual training data for method 1 is the flattened vectors which consist of the output results from any architecture of CNNs. For our preliminary work and the FACED framework, the training vector is the cluster distributions not the original acoustic features. As mentioned in Chapter 2, the extracted acoustic features can not always fit into a specific situation if the features are not designed manually. For this nature, feeding extracted acoustic features into another data-processing algorithm might help extract the information most relevant to the current task, which can help improve the performance of the whole framework.

In the other hand, it is as expected that the accuracy of all approaches is lower when using the real-world recordings than they did with the synthetic dataset. We believe that this is an artifact of how the data was collected and the interference made by background environments. Although the synthetic dataset has been generated with artificial noises and blurred to imitate real-world conditions, it still cannot simulate some unexpected situations in real-world. For example, people talking, clicking the buttons on the microphone, pen writing on papers, and car whistling sound are all presented in this real-world dataset. It is difficult to take all the conditions into consideration when generating the synthetic dataset, thus, the real-world dataset contains more bountiful background sounds compared with the synthetic dataset, which will largely influence the Detection accuracy of the classification methods. Also, the background sounds between recordings might be correlated to the activity, for example, similar activities were recorded near the same time and place. This will cause a classifier using the background not as a confuser but to actually help in the classification. Since the testing data might be recorded in a different place and different date, how well a classification method deals with the background sounds will largely influence the detection accuracy. Table 6.2 only presents the overall Detection accuracy of all the methods. However, in this real-world datasets, each audio file has two labels based on the two groups of activities: major and minor (or activity 1 and activity 2.) A large portion of

activity 2 in the recordings is environmental noise because activity 2 contains several activities which are not contributed to the productivity of a specific construction equipment, for example, machine idling, arms swinging, or back-and-forth moving. For activity 1, the main portion in the recordings are productive activities that contain relatively out-standing features compared with activity 1, i.e., rock breaking, maneuvering, and digging. Due to this nature, we also inspect how these methods work with these two types of activities. The results are shown in Table 6.3 and 6.4.

Table 6.3: Detection performance for Activity 1 and 2 using method 1 and method 2 with real-world dataset

Machine	Method 1		Method 2	
	Activity 1	Activity 2	Activity 1	Activity 2
JD333E	85.97%	79.03%	82.55%	75.65%
JD50D	88.56%	81.06%	79.68%	57.90%
IR compactor	90.02%	81.30%	83.22%	79.22%
CAT320E	87.11%	77.25%	76.37%	72.41%
Komatsu PC200	88.92%	79.02%	79.99%	76.89%
JD700J	87.79%	73.35%	79.88%	73.86%
Hitachi 50U	91.13%	81.09%	83.16%	77.18%
Concrete Mixer	86.68%	78.06%	84.18%	76.34%
JD 270C	83.28%	78.88%	80.99%	72.79%
Bobcat 331	81.09%	79.35%	81.12%	59.06%
Average	87.06%	78.84%	81.11%	72.13%

As shown in the Table 6.3 and 6.4, the main performance difference among all the methods involves identifying activity 2 (minor activities) in real-world recordings. A simpler illustration is depicted in Figure 6.1. For all the methods, the Detection accuracy for activity 1 is better overall than for activity 2, which contains much more environmental noises. In some cases, the method 2 (GMM-based classification) has difficulty when identifying minor activities in construction equipment recordings, i.e., JD 700J Crawler Tractor and Bobcat 331 Mini Excavator. The activity 2 (minor activities), which often contain significant environmental noise, inactive periods for a specific machine, and non-productive actions such as moving and swinging arms. These activities often present similar fea-

Table 6.4: Detection performance for Activity 1 and 2 using our preliminary work and the FACED framework with real-world dataset

Machine	Preliminary Work		FACED framework	
	Activity 1	Activity 2	Activity 1	Activity 2
JD333E	80.03%	83.55%	85.64%	80.86%
JD50D	79.79%	76.74%	82.36%	77.74%
IR compactor	82.47%	76.26%	84.19%	81.19%
CAT320E	80.36%	78.29%	83.99%	79.51%
Komatsu PC200	81.24%	77.48%	85.72%	78.16%
JD700J	80.06%	79.91%	82.61%	77.01%
Hitachi 50U	81.62%	78.88%	88.07%	77.07%
Concrete Mixer	80.16%	80.08%	83.39%	77.87%
JD 270C	79.31%	74.69%	82.57%	79.39%
Bobcat 331	76.24%	75.14%	80.87%	76.89%
Average	80.56%	78.10%	83.94%	78.57%

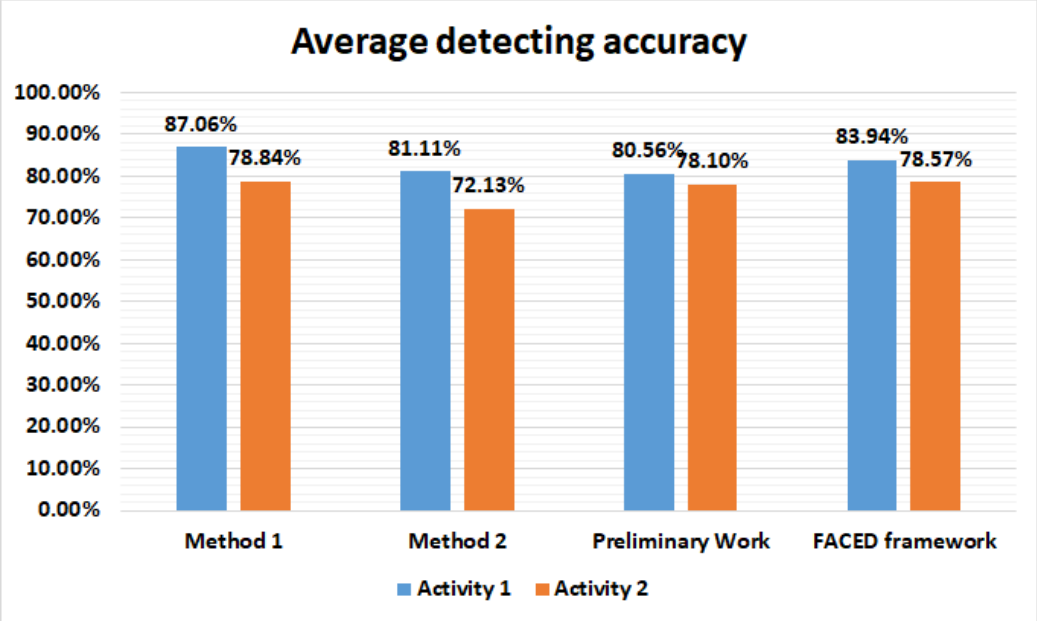


Figure 6.1: Average Detection performance using real-world dataset

tures and these features are corresponding to the overlapping clusters in clustering stage within the FACED framework. In general, the lower performance when identifying activity 2 results from that activity 2 has a high probability of being confused with other target activities. In practice, this might result from the overfitting to the background signal in

these recordings. As illustrated in Figure 6.1, both method 1 and 2 have a relatively large difference between detecting activity 1 and 2 (around 10%). In contrast, in the FACED framework and our preliminary work, difference between the detection accuracy of activity 1 and 2 is about 5%. This result can support our assumption that each different category will separate out the background into the "overlapping" clusters and the classification performance is determined more by the "non-overlapping" clusters. Also, since the FACED framework uses several up-to-dated algorithms and different acoustic features compared to our preliminary work, it shows an improved performance in this real-world dataset.

6.4.3 DCASE challenge dataset

TUT Sound Events 2017 dataset and *TUT Rare Sound Events 2017 dataset* are both real-world recordings from the DCASE challenge. There is a difference between the construction equipment recordings with TUT sound events dataset: the duration of targeted audio events. In the construction equipment dataset, an activity for a sound event can last for ten or even twenty seconds. However, for TUT datasets, each sound event presented in the audio clips might be only less than five seconds. Also, the targeted sound events in TUT datasets are specific events, for example, brakes squeaking, baby crying, and glass breaking, rather than a group of similar events recorded in construction equipment datasets. Since we randomly selected and concatenated the audio clips in *TUT Rare Sound Events 2017 dataset*, the detecting and classifying methodology is different from the one used in DCASE challenges. Also, DCASE challenge uses evaluation datasets for contest but it did not provide the ground truth of it. Thus, we split the development datasets provided by DCASE into training and testing data in this experiments. For the purpose of consistency, we present our results using the same detection accuracy as presented in DCASE challenges. DCASE challenges tend to use segment-based error rate calculated in one-second segments and segment-based F-score as the competition rubric. Segment based evaluation is done in a fixed time grid, using segments of one second length to compare the ground

truth and the system output. In each segment **k** we need to count the following parameters.

- True positives **TP**: events indicated as active by both the ground truth and system output
- False positives **FP**: events indicated as active by the system output but inactive by the ground truth
- False negatives **FN**: events indicated as inactive by the system output but active by the ground truth
- Substitutions **S**: system output indicating as active a wrong label events; one substitution is equivalent to one false positives and one false negative, meaning the system did not detect the correct event (false negative for the correct class) but detected something (false positive for another class)
- Insertions **I**: false positives after subtracting the substitutions
- Deletions **D**: false negatives after subtracting the substitutions
- Reference events **N**: number of events in the ground truth (segment)

After calculating the above parameters, we can use these parameters to calculate the **Error rate (ER)** and **F-score (F1)**. Error rate is calculated as described in [91] over all test data based on the total number of insertions **I**, deletions **D** and substitutions **S**. The formula is

$$\mathbf{ER} = \frac{\sum S(\mathbf{k}) + \sum D(\mathbf{k}) + \sum I(\mathbf{k})}{\sum N(\mathbf{k})}.$$

In the other hand, F-score (**F1**) is calculated over all test data based on the total number of false positive **FP**, false negatives **FN** and true positives **TP**:

$$\mathbf{F1} = \frac{2P \cdot R}{P + R},$$

where

$$P = \frac{\sum TP(\mathbf{k})}{\sum TP(\mathbf{k}) + \sum FP(\mathbf{k})},$$

$$R = \frac{\sum TP(\mathbf{k})}{\sum TP(\mathbf{k}) + \sum FN(\mathbf{k})}.$$

An alternative but similar way to represent the results is event-based evaluation metrics, which consider true positives (TP), false positives (FP) and false negatives (FN) with respect to event instances. The parameters are calculated as:

- True positives **TP**: correctly detected events
- False positives **FP**: events in the system output that are not correct according to the definition
- False negatives **FN**: events in the ground truth that have not been correctly detected according to the definition
- Substitutions **S**: events in system output that have correct temporal position but incorrect class label
- Insertions **I**: events in system output that are not correct nor substitutions
- Deletions **D**: events in ground truth that are not correct nor substituted
- Reference events **N**: number of events in the ground truth

The formula to calculate the event-based error rate (ER) and F-score (F1) are

$$\mathbf{ER} = \frac{S + D + I}{N}.$$

$$\mathbf{F1} = \frac{2P \cdot R}{P + R},$$

where

$$P = \frac{TP}{TP + FP},$$

$$R = \frac{TP}{TP + FN}.$$

In short, for **ER**, the lower is the better; while for **F1**, the higher is the better. The details of the above mentioned evaluation metric can be found in [92]. The calculating procedure is automatically done by the provided *sed_eval toolbox* published by Forman and Scholz [93].

	Segment-based (development dataset)	
	ER	F1
Method 1	0.61	66.1%
Method 2	0.73	56.8%
FACED	0.64	65.8%
DCASE top 1	0.20	80.3%
DCASE top 5	0.59	67.0%
DCASE baseline	0.69	56.7%

Table 6.5: Segment-based error rate (ER) and F-score (F1) using *TUT Sound Events 2017 dataset*

	Segment-based (development dataset)	
	ER	F1
Method 1	0.18	90.2%
Method 2	0.35	85.0%
FACED	0.21	89.9%
DCASE top 1	0.07	96.3%
DCASE top 5	0.16	92.8%
DCASE baseline	0.53	72.7%

Table 6.6: Segment-based error rate (ER) and F-score (F1) using *TUT Rare Sound Events 2017 dataset*

Table 6.5 presents the segment-based evaluation results for *TUT Sound Events 2017 dataset*, while Table 6.6 is for *TUT Rare Sound Events 2017 dataset*. Due to the different

characteristics of the two datasets, the results seem drastically different. The recordings in *TUT Sound Events 2017 dataset* involves multi-source conditions similar to the everyday life. The sound sources are rarely heard in isolation in this dataset, which causes the difficulty to detect the targeted sound event correctly. In contrast, *TUT Rare Sound Events 2017 dataset* is consisted with artificially created mixtures. Targeted sound events are "baby crying", "glass breaking", and "gunshot" mixed with different background scenes. It is easier to hear the targeted sound events in this dataset. As you can see in Table 6.6, the Detection performance for all the methods are pretty great when using *TUT Rare Sound Events 2017 dataset*. Since DCASE did not provide the class-wise performance, the results in Table 6.5 and 6.6 are the overall results using the development dataset.

In general, the FACED framework, method 1, and 2 cannot outperform the best result in DCASE challenge. However, the FACED framework and method 1 can reach about top 20 even top 10 performance. The core idea of method 1 is from the top DCASE participants. It is designed to be a general architecture without sophisticated tuning to fit any specific situations. Similarly, the FACED framework is designed to be flexible so that it can work under any acoustic scenes. It is reasonable that the general approaches do not have a better result compared to the carefully designed system. However, the FACED framework and method still present great results in both datasets. For method 2, the GMM-based classification, it is a relatively conventional baseline; thus, it performs overall worse than the other compared methods. As mentioned in previous section, it is a tough task for a classifier to deal with background/environmental noises. In *TUT Sound Events 2017 dataset*, more complex situations frequently present in the audio clips, for example, overlapping targeted sound events, or large noise which dominate the targeted sound event. These conditions largely influence the low detection performance shown in Table 6.5. Without the high level environmental noises, all the methods present favorable results in when using *TUT Rare Sound Events 2017 dataset*.

Different from *TUT Sound Events 2017 dataset*, *TUT Rare Sound Events 2017 dataset*

can make use of the class-wise event detection. *TUT Sound Events 2017 dataset* contains too many overlapping mixtures in the recordings, it is difficult to separate each audio event out. In the other hand, the sound events in *TUT Rare Sound Events 2017 dataset* are much more clear and distinguishable. DCASE provides the class-wise detecting results using the TUT evaluation dataset. However, both the ground truth of the evaluation dataset and the class-wise detecting results of the development dataset are not provided. Furthermore, the system which can provide the overall best detecting results is different from the system that has a better performance on detecting a specific class. As a result, the results of class-wise detection using method 1, method 2, and the FACED framework are compared against the DCASE baseline. Again, the DCASE baseline results are using the TUT evaluation dataset, which is different from the TUT development dataset we are using, thus it is only an informative benchmark.

	Class-wise error-rate (ER)		
	Baby cry	Glass break	Gunshot
Method 1	0.6	0.24	0.623
Method 2	0.784	0.39	0.698
FACED framework	0.65	0.271	0.652
DCASE baseline	0.804	0.38	0.728

Table 6.7: Class-wise error rate (ER) using development dataset from *TUT Rare Sound Events 2017 dataset*

	Class-wise F-score (F1)		
	Baby cry	Glass break	Gunshot
Method 1	76.9%	84.7%	67.8%
Method 2	65.4%	72.20%	55.4%
FACED framework	74.17%	82.17%	65.39%
DCASE baseline	66.80%	79.10%	46.50%

Table 6.8: Class-wise F-score (F1) using development dataset from *TUT Rare Sound Events 2017 dataset*

Table 6.7 and 6.8 depicted the results of class-wise error-rate (ER) and F-score (F1). In general, all the method 1, 2, and the FACED framework performs better than the DCASE

baseline system. Although the dataset used are different, the audio events in the dataset are in same class. Method 1 and the FACED framework outperform the baseline system a lot thus we can have a rough idea that these two systems have a great ability performing audio event detection and classification. These results are also consistent with Table 6.6. The only difference is that, in some cases, method 2 performs even worse than the baseline system. Figure 6.2 and 6.3 illustrate the overall and class-wise ER and F1. As shown in the figures, the detection performance varies a lot when facing different audio events. A great overall detection result does not guarantee great results in every case. Within these three classes, the "gunshot" and "baby cry" have a high possibility to be confused as other sound events or background sounds. The "baby cry" might have similar features with people's sound, such as talking or murmuring. For the "gunshot", it seems like the detecting classifier has the chance to confuse it with some sounds with large volume, for example, dropping things onto the ground. In these comparison experiments, the FACED framework has similar detecting and classification results with method 1, the popular top-notched detecting system. To better inspect the performance difference between all the methods, several different experiments are conducted and presented in the following section.

Noted that the results in Table 6.5, 6.6, 6.7, 6.8 are using the full DCASE datasets with full meta-data. Although the FACED framework does not produce the best result in experiment, it still shows its competitiveness against other methods. In order to test the performance of the FACED framework when using weakly-labeled data, we eliminate the detail temporal information from the *TUT Rare Sound Events 2017 dataset* and then perform the experiments again. Since the *TUT Sound Events 2017 dataset* contains overlapping sound events and labels, we will not use it in the following experiments. The audio recordings in *TUT Rare Sound Events 2017 dataset* will be labeled only with a short description such as "glass breaking" and "baby crying" without any temporal information. Due to the modified meta-data, we cannot directly compare our results again the DCASE results. However, we will still include the DCASE baseline results here as a general benchmark. The over-

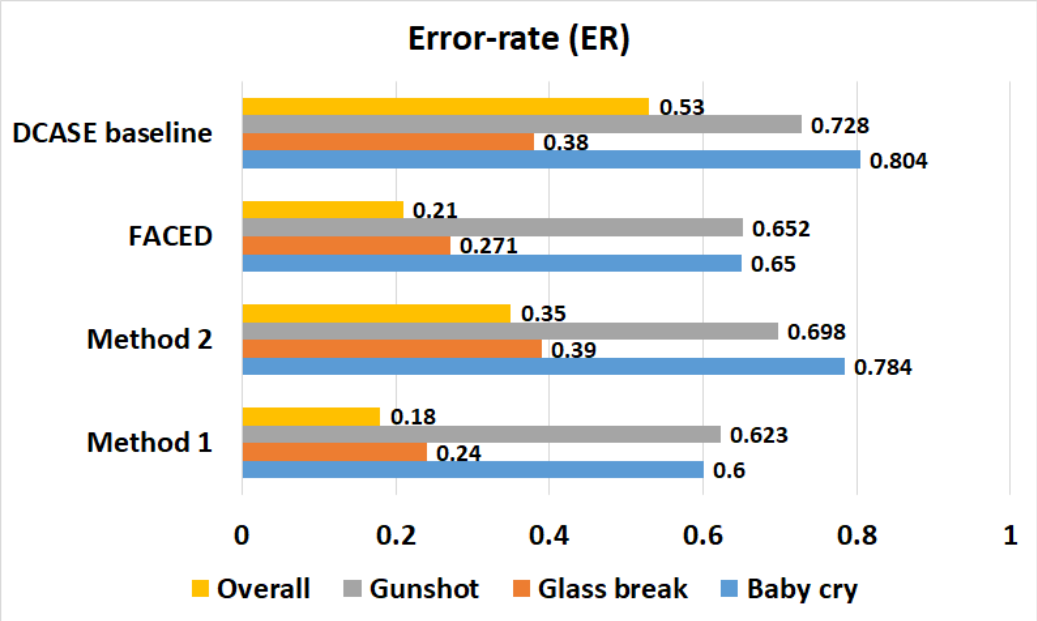


Figure 6.2: The comparison of error-rate (ER) between using *TUT Rare Sound Events 2017 dataset*

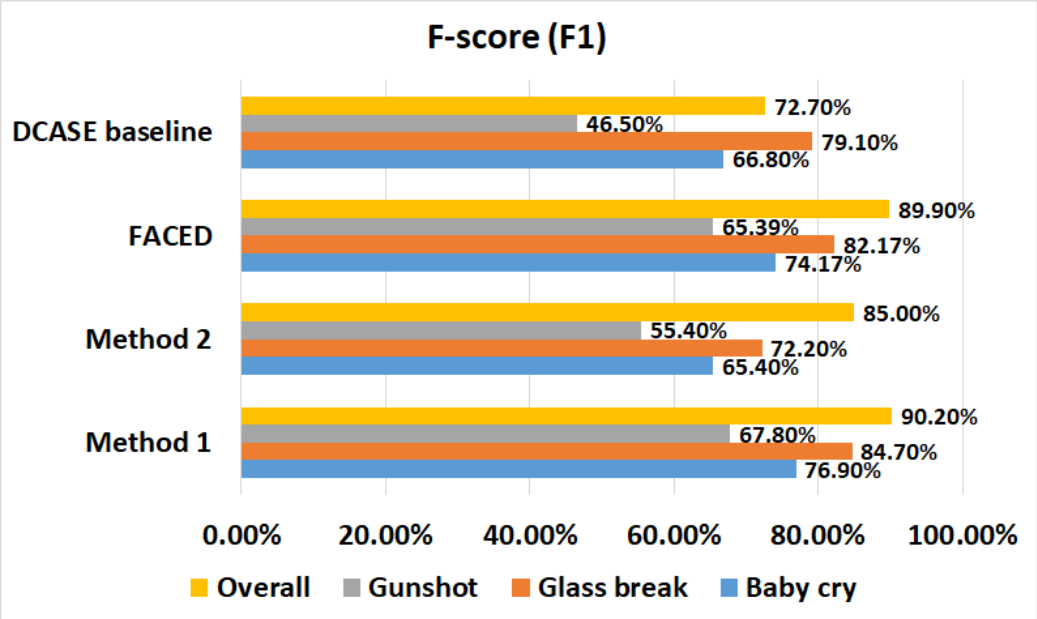


Figure 6.3: The comparison of F-score (F1) using *TUT Rare Sound Events 2017 dataset*

all DCASE baseline results are using the TUT development dataset, which is the same as what we used in this experiments. Although it is not a fair comparison, we also include the DCASE baseline class-wise event-based evaluation results in the following tables. These

class-wise results are using TUT evaluation dataset, not the TUT development dataset, thus they can only be treated as a general benchmark not an equivalent comparing target. Table 6.9, 6.10, and 6.11 present the detecting results using the weakly labeled *TUT Rare Sound Events 2017 dataset*.

	Overall dataset	
	ER	F1
Method 1	0.36	83.2%
Method 2	0.61	70.1%
FACED framework	0.4	80.1%
DCASE baseline	0.53	72.7%

Table 6.9: Error rate (ER) and F-score (F1) using weakly-labeled *TUT Rare Sound Events 2017 dataset*

As presented in Table 6.9, the method 1 and the FACED framework can still outperform the DCASE baseline system even using the weakly labeled data. Noted that the meta-data used for DCASE baseline here is strong labels. In contrast, method 2 has a difficulty detecting the targeted events correctly after removing the detail temporal information in the labels. This might result from the GMM failing to build reasonable distributions for each targeted event. Also, it is as expected that the detection performance is overall lower than using the original strongly labels. Figure 6.4 and 6.5 illustrated the comparison of error-rate (ER) and F-score (F1) between using the original meta-data and weak labels. The ER of the weakly labeled data is about double than it of the original data, and the F-score is also decreased a certain amount. It is obvious that the weak labels highly interfere with the detecting and classification process among all the methods.

6.5 Decreasing the size of dataset

Besides the weakly labeled nature of the dataset, the other topic of our work is the small size dataset. In this section, we will only used *TUT Rare Sound Events 2017 dataset* and the synthetic dataset. The reason for choosing these two dataset is the original time length.

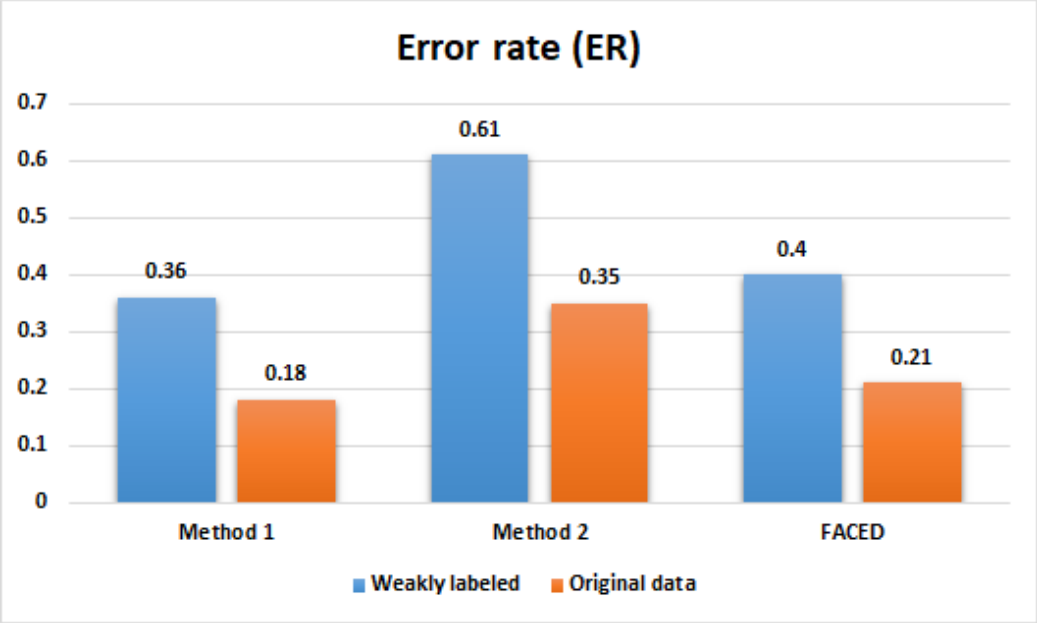


Figure 6.4: The comparison of error-rate (ER) between the original and weakly-labeled *TUT Rare Sound Events 2017 dataset*

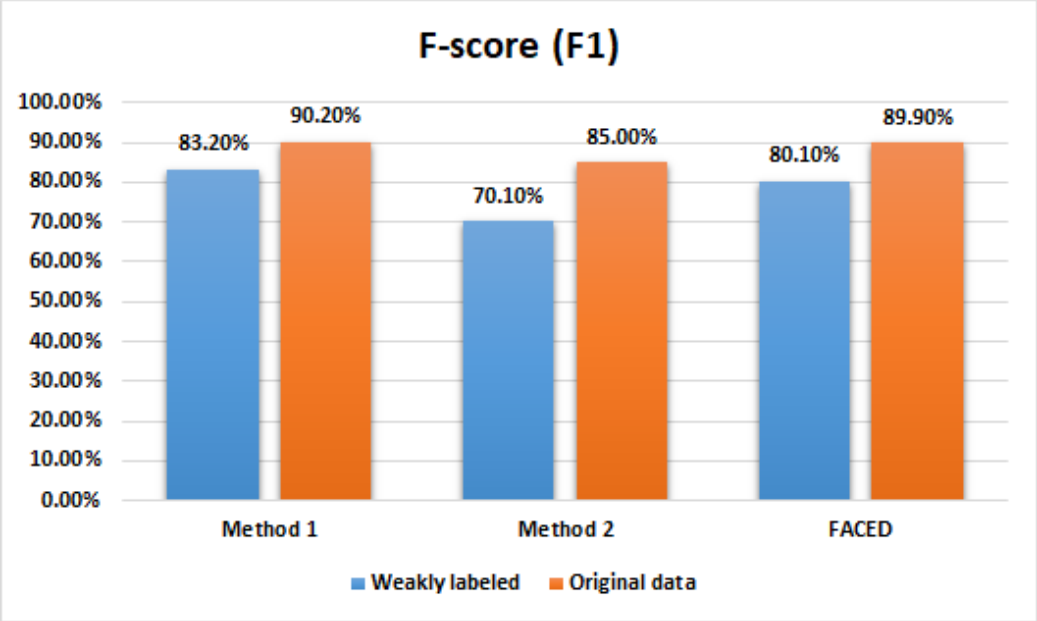


Figure 6.5: The comparison of F-score (F1) between the original and weakly-labeled *TUT Rare Sound Events 2017 dataset*

The total time length of the *TUT Rare Sound Events 2017 dataset* is about 13 hours, and the synthetic dataset can be generated based on this length. For *TUT Sound Events 2017*

	Class-wise error rate (ER)		
	Baby cry	Glass break	Gunshot
Method 1	0.781	0.498	0.703
Method 2	0.914	0.667	0.85
FACED framework	0.799	0.528	0.73
DCASE baseline	0.804	0.38	0.728

Table 6.10: Class-wise error rate (ER) using weakly-labeled *TUT Rare Sound Events 2017 dataset*

	Class-wise F-score (F1)		
	Baby cry	Glass break	Gunshot
Method 1	70.49%	80.02%	50.84%
Method 2	60.02%	71.97%	39.28%
FACED framework	69.67%	78.57%	49.17%
DCASE baseline	66.80%	79.10%	46.50%

Table 6.11: Class-wise F-score (F1) using weakly-labeled *TUT Rare Sound Events 2017 dataset*

dataset and construction equipment dataset, the original time length is already below half the size of the *TUT Rare Sound Events 2017 dataset*. Thus, only the *TUT Rare Sound Events 2017 dataset* and the synthetic dataset will be shrunk gradually in this section. The intuition of the development of FACED framework is dealing with the hand-recorded small size data. Thus, the shrunk datasets can help us inspect the influence of the data size and to compare the performance of the FACE framework with other methods under this situation.

6.5.1 Synthetic dataset

First, the full size of the synthetic dataset is equivalent to 12-hour recordings. The data size is then shrunk to 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, and 10%. The 10% data size is equivalent to a 72 minutes recording. The testing data is generated to be a fixed 30 minutes-long data. We will use different size of training data and a same testing data to perform the comparing experiments. The data size is selected based on the total length of *TUT Rare Sound Events 2017 dataset* which will be used later in our work. Table 6.12

presents the Detection accuracy when the data size is shrinking. To better visualizing the results, a line chart is plotted as Figure 6.6. As shown in Table 6.12 and Figure 6.6, the Detection accuracy is also decreased when the data size is decreased. When the data size is over 50% (6 hours), the detection accuracy only has a little fluctuation. Actually, the detection accuracy almost remains the same when using full size, 90%, 80%, and 70% size of data. However, when the data size is decreased under 40% (4 hours and 48 minutes), the detection accuracy falls down drastically. In the last case, 10% data size, the FACED framework has the best performance. It can be observed in Figure 6.6 that the FACED framework will outperform the other two methods when the data size is below 30%. The structures of all the three methods are not changed compared to the previous section. A possible reason for the results under 30% data size is that the neural network architecture and epochs in method 1 remains fixed during the experiments, thus the shrunk training data cannot construct networks as well as using full size of data. Similar reason might cause method 2 build preferable distributions to perform the classification. Compared to method 1 and 2, the Detection performance of the FACED framework decreased relatively smooth. The synthetic data results support our assumption that the FACED framework can still perform well when the size of dataset is small. For a small size data, a more complex or sophisticated framework does not guarantee better Detection performance. As mentioned previously, the FACED framework has intended to work on hand-recorded audio clips, which might not have a large amount of data compared to publicity available datasets. The experimental results here potential indicate that the FACED framework has the ability to outperform other methods when facing small size dataset. After the experiments with synthetic dataset, the following experiment will be conducted with *TUT Rare Sound Events 2017 dataset*.

	Method 1	Method 2	FACED framework
Full data	99.8%	98.5%	99.6%
90% data	99.7%	98.3%	99.6%
80% data	99.8%	98.3%	99.4%
70% data	98.0%	96.9%	97.6%
60% data	96.5%	96.2%	96.3%
50% data	94.1%	94.3%	94.2%
40% data	91.7%	90.2%	91.5%
30% data	90.3%	85.5%	91.3%
20% data	87.6%	83.2%	90.3%
10% data	84.2%	79.7%	88.6%

Table 6.12: Detection performance using synthetic dataset, the data size is gradually shrunk from 100% toward 10%

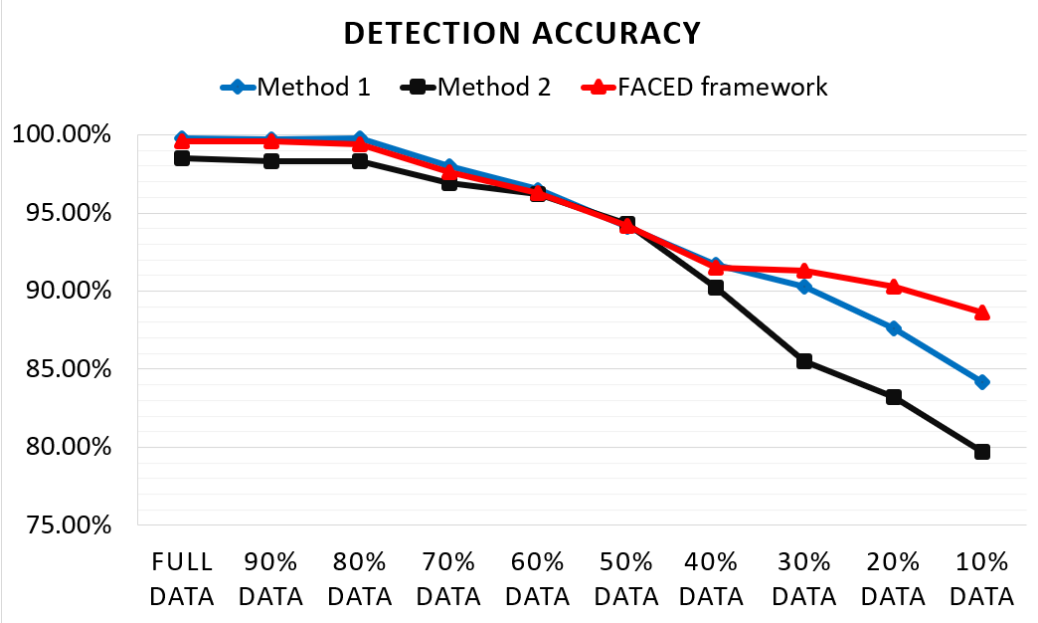


Figure 6.6: The Detection accuracy of gradually shrinking synthetic dataset

6.5.2 DCASE dataset

Same as the shrinking process when dealing with synthetic dataset, we also decrease the size of *TUT Rare Sound Events 2017 dataset* 10%-by10% from the full data size toward a 10% data size. The overall results of error-rate (ER) and F-score (F1) are listed in Table 6.14 and 6.13. Notably, the tendency of the results are pretty similar to it in

the synthetic dataset. For the purpose of an easier inspection, Figure 6.8 and 6.7 are plotted. As shown in the figures, the FACED framework will outperform the method 1 when the data size is decreased under 50% data size. Since *TUT Rare Sound Events 2017 dataset* is consisted with real-world recordings, the Detection performance is expected to be worse than using synthetic dataset. Compared to the overall DCASE baseline results ($ER = 0.53, F1 = 72.70\%$), the method 1 and the FACED framework still present a good performance even using only 10% of original training data. However, when decreasing the training data size to a certain amount, the FACED framework starts to be superior than the method 1. The possible reason has been discussed in the previous section. These results indicate that the FACED framework has an outstanding performance when facing with small size dataset, which is the intuition of designing the FACED framework. The results of the class-wise error-rate (ER) and F-score (F1) are also presented in below.

Table 6.15, 6.17, and 6.19 present the results of class-wise error-rate (ER) of three sound events in *TUT Rare Sound Events 2017 dataset*; while Table 6.16, 6.18, and 6.20 are for F-scores (F1). Figure 6.9 to Figure 6.14 can help inspect the results much easier. In general, the overall tendency of the error-rate (ER) and F-score (F1) remain the same as the result of overall dataset. Both the method 1 and the FACED framework perform better than the method 2, which can be treated as the baseline here. The class-wise results between the method 1 and the FACED framework are pretty close to each other. The relatively special case is the "gunshot" class. The ER and F1 results have a distinct fluctuation when the data size is decreasing.

Based on all the results presented in this chapter, we can conclude that the FACED framework has the competitiveness against other audio event detection and classification methods. In particular, the relatively-well performance when dealing with the small size data is the advantage of the FACED framework. Several improvements can be applied on the FACED framework and we will discuss in the next chapter.

Table 6.13: The F-score (F1) using *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	90.2%	85.0%	89.9%
90% data	90.1%	84.8%	89.8%
80% data	90.1%	84.9%	89.5%
70% data	89.6%	82.2%	88.3%
60% data	88.6%	80.1%	88.2%
50% data	86.1%	78.4%	85.9%
40% data	82.0%	76.5%	84.3%
30% data	80.7%	72.2%	83.3%
20% data	80.10%	71.7%	80.7%
10% data	76.0%	69.4%	79.6%

Figure 6.7: The F-score (F1) of gradually shrinking *TUT Rare Sound Events 2017 dataset* dataset

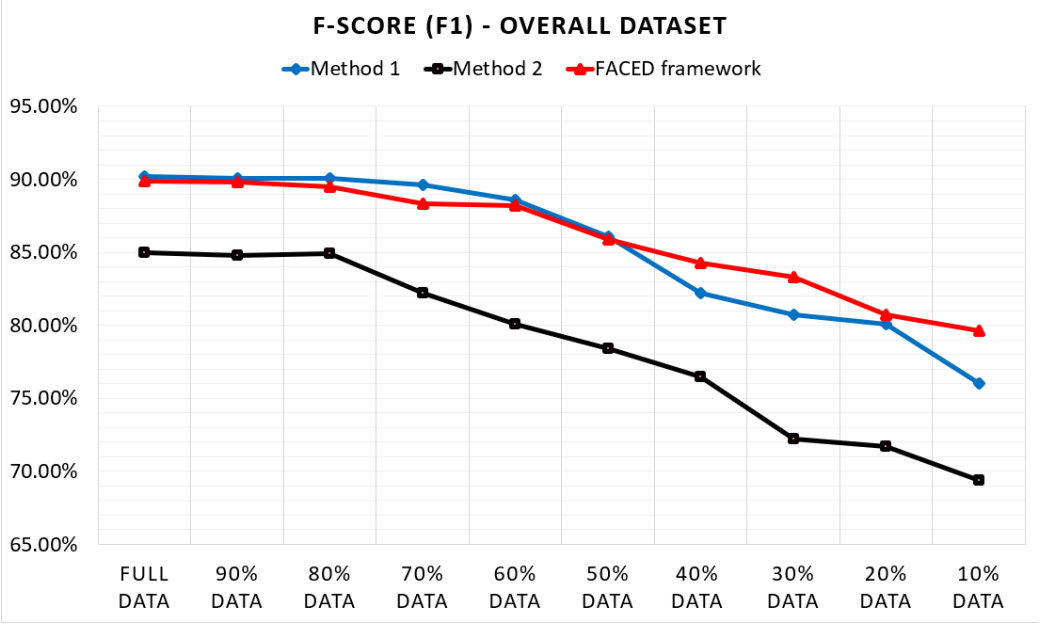


Table 6.14: The error-rate (ER) using *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	0.18	0.35	0.21
90% data	0.19	0.36	0.22
80% data	0.22	0.39	0.24
70% data	0.26	0.44	0.3
60% data	0.32	0.47	0.36
50% data	0.44	0.5	0.4
40% data	0.5	0.58	0.48
30% data	0.56	0.66	0.5
20% data	0.59	0.75	0.54
10% data	0.61	0.8	0.55

Figure 6.8: The error-rate (ER) of gradually shrinking *TUT Rare Sound Events 2017 dataset* dataset

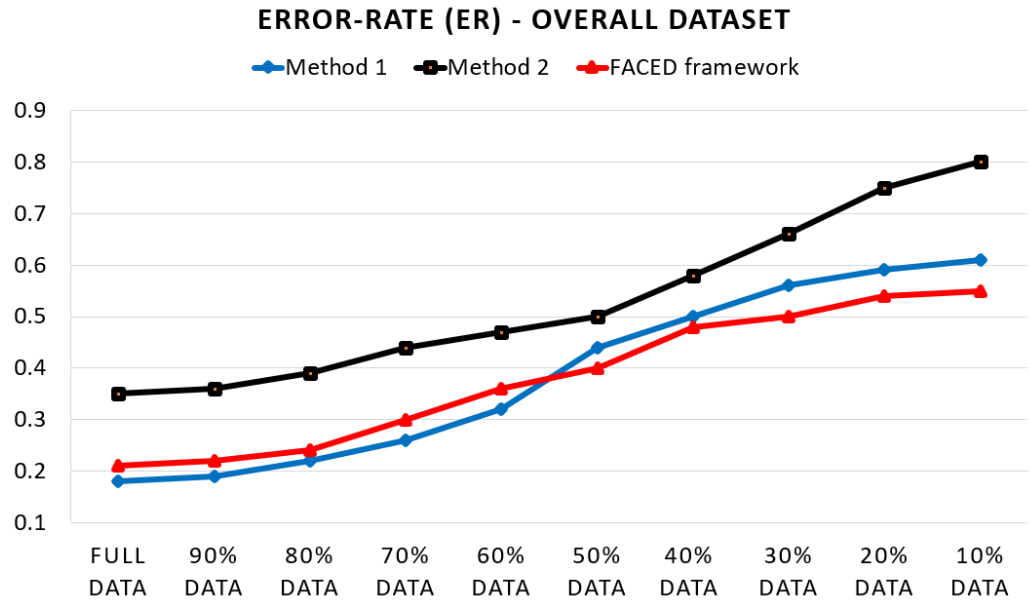


Table 6.15: The class-wise error-rate (ER) for "**Baby cry**" event within *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	0.6	0.784	0.65
90% data	0.59	0.78	0.63
80% data	0.61	0.78	0.66
70% data	0.63	0.8	0.69
60% data	0.66	0.83	0.65
50% data	0.69	0.87	0.71
40% data	0.77	0.89	0.73
30% data	0.81	0.92	0.78
20% data	0.81	0.95	0.8
10% data	0.84	0.99	0.79

Table 6.16: The class-wise F-score (F1) for "**Baby cry**" event within *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	76.9%	65.4%	74.17%
90% data	76.8%	64.9%	74.0%
80% data	76.6%	62.0%	73.5%
70% data	72.8%	61.8%	72.9%
60% data	70.2%	58.7%	70.9%
50% data	67.8%	57.6%	69.7%
40% data	67.9%	55.4%	67.1%
30% data	65.4%	53.1%	65.9%
20% data	63.0%	50.6%	65.3%
10% data	62.7%	49.9%	64.5%

Figure 6.9: The class-wise error-rate (ER) for "**Baby cry**" event within the gradually shrinking *TUT Rare Sound Events 2017 dataset*

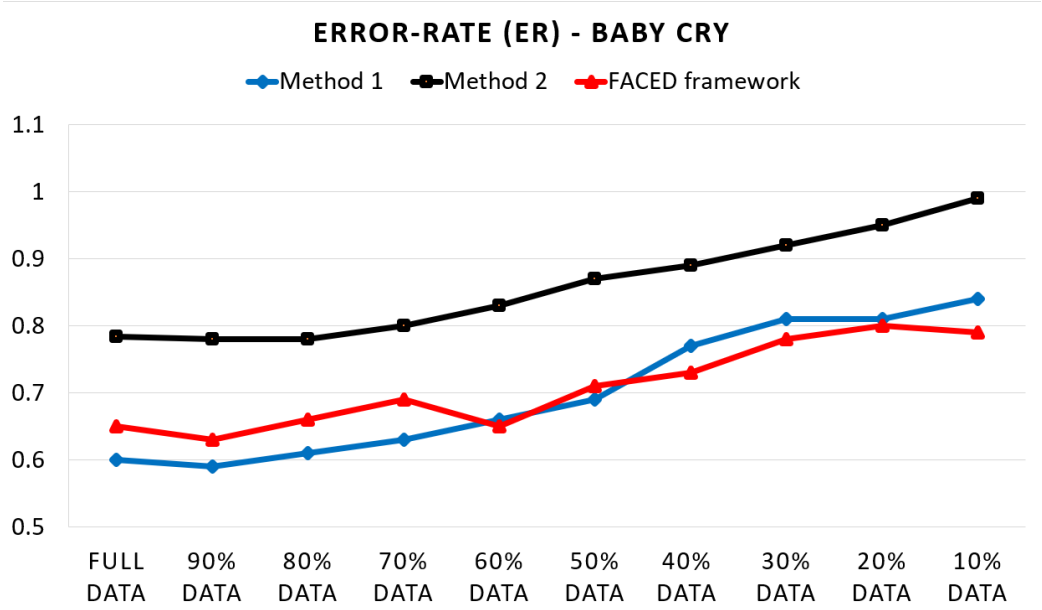


Figure 6.10: The class-wise F-score (F1) for "**Baby cry**" event within the gradually shrinking *TUT Rare Sound Events 2017 dataset*

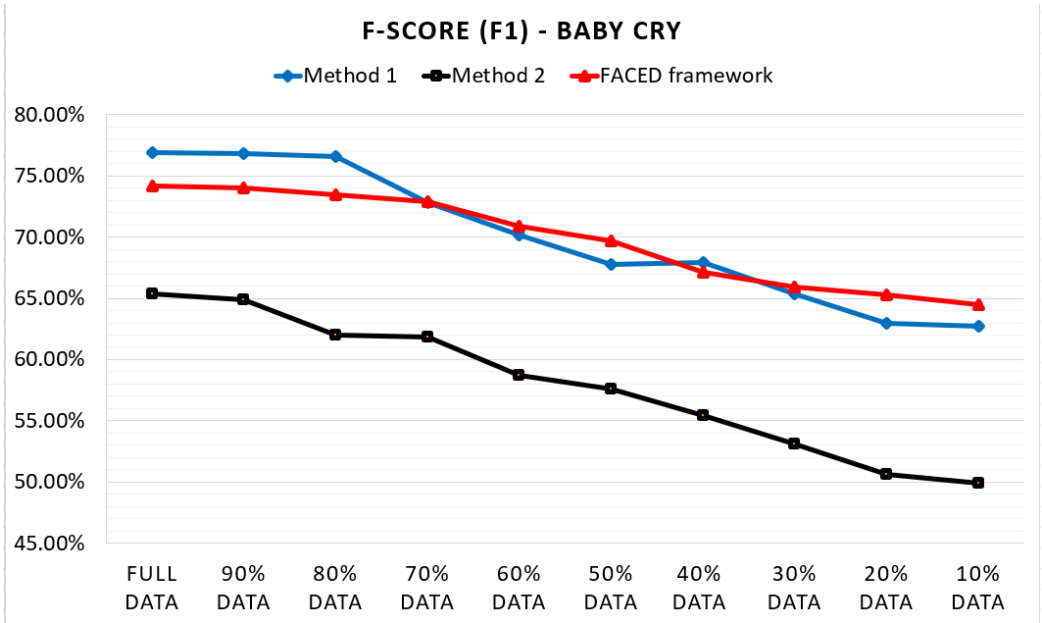


Table 6.17: The class-wise error-rate (ER) for "**Glass breaking**" event within *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	0.24	0.39	0.271
90% data	0.24	0.40	0.27
80% data	0.26	0.44	0.28
70% data	0.30	0.48	0.33
60% data	0.33	0.49	0.34
50% data	0.35	0.51	0.37
40% data	0.36	0.51	0.38
30% data	0.39	0.55	0.39
20% data	0.41	0.57	0.39
10% data	0.42	0.58	0.40

Table 6.18: The class-wise F-score (F1) for "**Glass breaking**" event within *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	84.7%	72.2%	82.17%
90% data	84.5%	72.3%	82.1%
80% data	84.6%	71.9%	81.8%
70% data	84.0%	72.0%	81.9%
60% data	83.4%	71.6%	81.6%
50% data	82.7%	71.2%	81.3%
40% data	81.4%	71.8%	81.0%
30% data	79.1%	70.4%	80.5%
20% data	78.3%	69.9%	79.4%
10% data	78.1%	68.7%	79.0%

Figure 6.11: The class-wise error-rate (ER) for "**Glass breaking**" event within the gradually shrinking *TUT Rare Sound Events 2017 dataset*

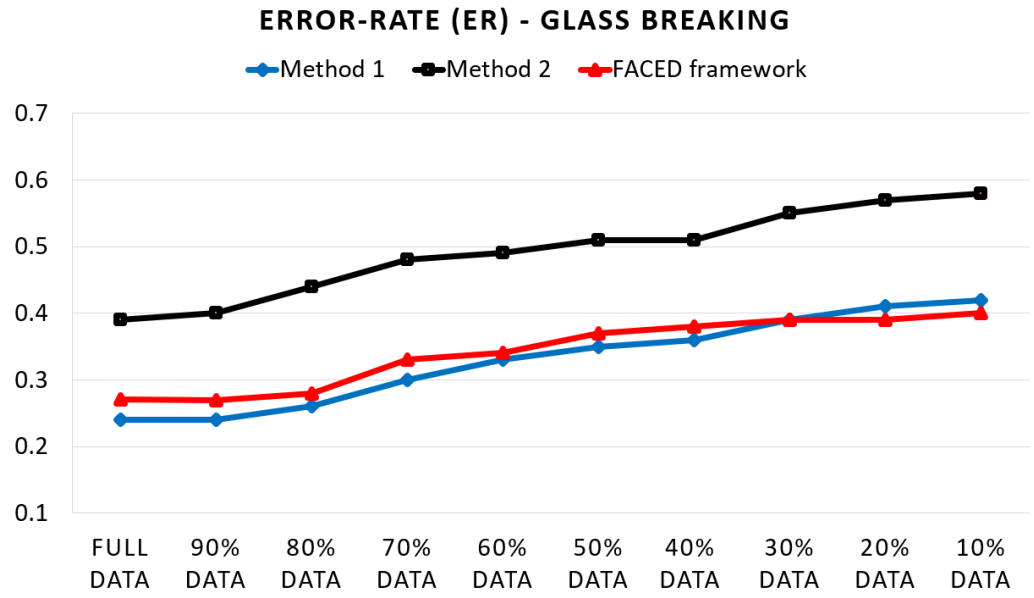


Figure 6.12: The class-wise F-score (F1) for "**Glass breaking**" event within the gradually shrinking *TUT Rare Sound Events 2017 dataset*

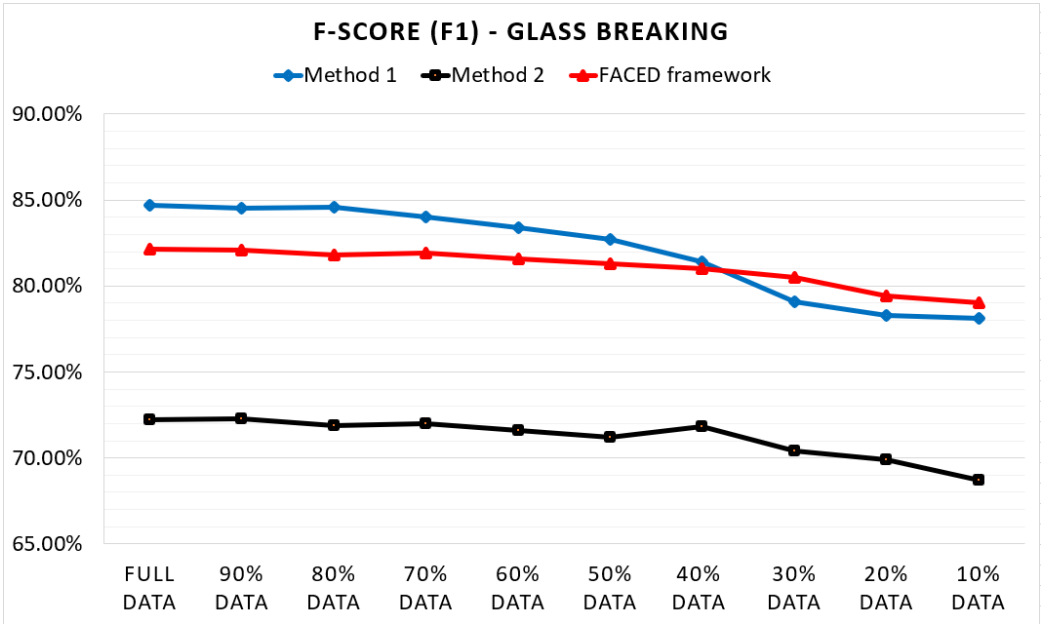


Table 6.19: The class-wise error-rate (ER) for "**Gunshot**" event within *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	0.623	0.698	0.652
90% data	0.63	0.70	0.68
80% data	0.67	0.72	0.69
70% data	0.70	0.75	0.71
60% data	0.75	0.81	0.73
50% data	0.80	0.85	0.75
40% data	0.82	0.93	0.79
30% data	0.85	0.95	0.80
20% data	0.87	0.99	0.82
10% data	0.88	1.01	0.83

Table 6.20: The class-wise F-score (F1) for "**Gunshot**" event within *TUT Rare Sound Events 2017 dataset*, the data size is gradually shrunk from 100% toward 10%

	Method 1	Method 2	FACED framework
Full data	67.8%	55.4%	65.39%
90% data	67.6%	55.0%	64.9%
80% data	65.6%	54.1%	64.7%
70% data	59.9%	53.0%	62.7%
60% data	57.3%	49.0%	60.0%
50% data	50.1%	48.0%	57.7%
40% data	52.8%	45.7%	57.5%
30% data	50.0%	41.9%	54.9%
20% data	46.8%	39.4%	53.0%
10% data	46.9%	37.9%	53.1%

Figure 6.13: The class-wise error-rate (ER) for "**Gunshot**" event within the gradually shrinking *TUT Rare Sound Events 2017 dataset*

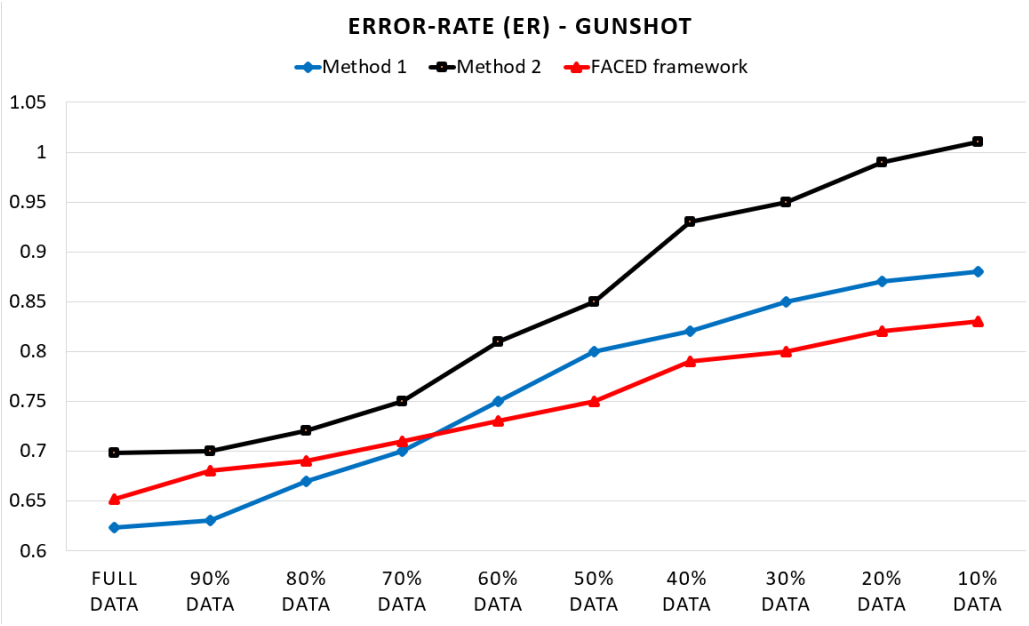
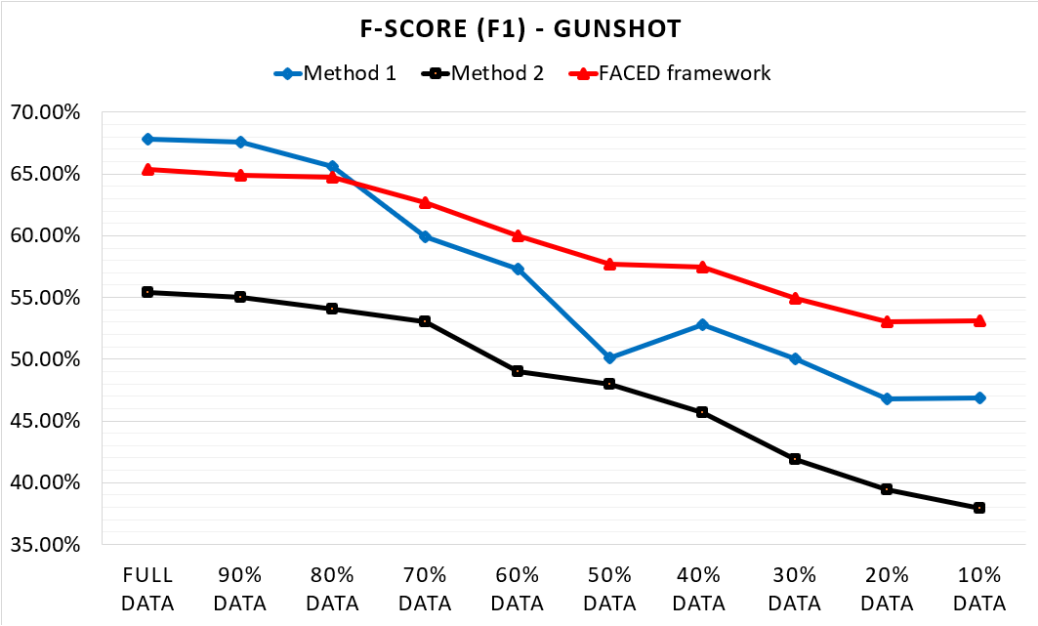


Figure 6.14: The class-wise F-score (F1) for "**Gunshot**" event within the gradually shrinking *TUT Rare Sound Events 2017 dataset*



CHAPTER 7
CONCLUSION

The research presented in this thesis makes contributions to the fields of signal processing and audio event detection.

7.1 Contributions

The FACED framework represents novel and practical ways to perform audio event detection on weakly labeled dataset. Conventionally, the audio event detection system tends to first create strong labels before training when dealing with weakly labeled data. In contrast, the FACED framework reduce the need for costly efforts of manually labeling the individual sound events within a dataset, allowing the user to leverage general knowledge of clip-level meta-data with certain sound events may have occur. The FACED framework has been tested and found to be useful both in synthetic and real-world environments, which could be non-stationary, uncontrolled, and with plentiful noises. It is flexible in how the framework designed and can make use of different types of algorithms or knowledge about the acoustic data. It can also be leveraged with relatively small size data to generate a robust classifier of the sound events present in the data and to help label it. The FACED framework has a great potential to be adapted to new situations, and adjust itself well to being cooperated as a processing steps in other systems.

7.2 Future work

There are many ways in which this work could be extended. The FACED framework could be tried in various other environments where it might be applicable, such as domestic scenes or a continuously moving settings. There are countless different acoustic feature types that

could be used in the feature extraction step (including potential applications to multi-media data). Also, various algorithms and many parameter configurations could both be explored when there is any up-to-dated approaches published by other researchers. Ways to make the FACED framework more robust involves changing in classification methodology, especially for overlapping sound events presented in the audio data, in some way that could be explored. There may also be potential to use some post-processing algorithms to help adapt the FACED framework to polyphonic scenes. The prominent event and the underlying events in different environments could both be targeted while retaining information about all the sound events and adding some ways of post processing. One practical choice of post processing used in current experiments is the decision stage and the decision is made based on a threshold value 0.5. For now, if there are multiple class values over the threshold, the most probable target class is chosen; while if all the values are under the threshold, an "unknown" label will be attached on to the audio segment. By adjusting this decision rules, the classification step in the FACED framework might have an improved ability to label multiple overlapping sound events.

Future work could also include efforts to cooperate the FACED framework into other systems or as a user-friendly machine learning tools. Considering the intuition for developing the FACED framework, the substitution from manual labelling with automatically detecting is the users' hope. Compared to other deep learning techniques, the FACED framework has relatively low time cost and computational demand. It has the potential for the FACED framework performs smoothly as a toolkit on a portable and offline computation environment. Since there are bountiful options with the FACED framework, effort would still be needed to inspect the demand for hardware environment and to develop interfaces or GUIs that would allow users to interact much easier with the FACED framework.

REFERENCES

[1] C. Cheng, A. Rashidi, M. Davenport, and D. Anderson, "Activity analysis of construction equipment using audio signals and support vector machines," *Automation in Construction*, vol. 81, pp. 240–253, 2017.

[2] X. Zhuang, X. Zhou, M. A. Hasegawa-Johnson, and T. S. Huang, "Real-world acoustic event detection," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1543–1551, 2010.

[3] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, p. 1, 2013.

[4] D. Gerhard, *Audio signal classification: History and current techniques*. Citeseer, 2003.

[5] G. Guo and S. Li, "Content-based audio classification and retrieval by support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 209–215, 2003.

[6] L. Lu, H. Zhang, and H. Jiang, "Content analysis for audio classification and segmentation," *IEEE Transactions on speech and audio processing*, vol. 10, no. 7, pp. 504–516, 2002.

[7] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[8] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[9] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[10] D. Ubskii and A. Pugachev, "Sound event detection in real-life audio," *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events*, 2016.

[11] S. Adavanne and T. Virtanen, "Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network," *arXiv preprint arXiv:1710.02998*, 2017.

[12] Q. Kong, Y. Xu, W. Wang, and M. Plumbley, "A joint detection-classification model for audio tagging of weakly labeled data," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, 2017.

[13] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. Plumbley, "Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging," *arXiv preprint arXiv:1703.06052*, 2017.

[14] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: An ieee aasp challenge," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*, IEEE, 2013, pp. 1–4.

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

[16] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, ACM, 2015, pp. 1015–1018.

[17] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia*, ACM, 2014, pp. 1041–1044.

[18] *Workshop on Detection and Classification of Acoustic Scenes and Events*, New York, USA, Oct. 2019.

[19] J. Lee, J. Park, S. Kum, Y. Jeong, and J. Nam, "Combining multi-scale features using sample-level deep convolutional neural networks for weakly supervised sound event detection," *Proc. DCASE*, pp. 69–73, 2017.

[20] D. Lee, S. Lee, Y. Han, and K. Lee, "Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[21] A. Kumar and B. Raj, "Weakly supervised scalable audio content analysis," in *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1–6.

[22] S.-Y. Chou, S. Jang, and Y.-H. Yang, "Framecnn: A weakly-supervised learning framework for frame-wise acoustic event detection and classification," *Recall*, vol. 14, pp. 55–4, 2017.

[23] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*, 2016.

[24] J. Liu and Y. Yang, "Event localization in music auto-tagging," in *Proceedings of the 2016 ACM on Multimedia Conference*, 2016.

[25] T. Su, J. Liu, and Y. Yang, "Weakly-supervised audio event detection using event-specific Gaussian filters and fully convolutional networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, 2017.

[26] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *Neural Networks (IJCNN), 2017 International Joint Conference on*, 2017.

[27] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on speech and audio processing*, vol. 9, no. 5, pp. 504–512, 2001.

[28] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE transactions on acoustics, speech, and signal processing*, vol. 33, no. 2, pp. 443–445, 1985.

[29] I. Cohen and B. Berdugo, "Noise estimation by minima controlled recursive averaging for robust speech enhancement," *IEEE signal processing letters*, vol. 9, no. 1, pp. 12–15, 2002.

[30] P. C. Loizou, *Speech enhancement: theory and practice*. CRC press, 2007.

[31] S. Rangachari and P. Loizou, "A noise-estimation algorithm for highly non-stationary environments," *Speech communication*, vol. 48, no. 2, pp. 220–231, 2006.

[32] I. Cohen, "Noise spectrum estimation in adverse environments: Improved minima controlled recursive averaging," *IEEE Transactions on speech and audio processing*, vol. 11, no. 5, pp. 466–475, 2003.

[33] E. Sejdić, I. Djurović, and J. Jiang, "Time–frequency feature representation using energy concentration: An overview of recent advances," *Digital signal processing*, vol. 19, no. 1, pp. 153–183, 2009.

[34] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[35] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[36] W. J. Poser, "Douglas o'shaughnessy, speech communication: Human and machine. reading, massachusetts: Addison-wesley publishing company, 1987. pp. xviii+ 568. isbn 0-201-16520-1.," *Journal of the International Phonetic Association*, vol. 20, no. 2, pp. 52–54, 1990.

[37] B. P. Bogert, "The quefrency alanysis of time series for echoes; cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," *Time series analysis*, pp. 209–243, 1963.

[38] S. Sandhu and O. Ghitza, "A comparative study of mel cepstra and eih for phone classification under adverse conditions," in *1995 International Conference on Acoustics, Speech, and Signal Processing*, IEEE, vol. 1, 1995, pp. 409–412.

[39] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[40] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," in *Readings in speech recognition*, Elsevier, 1990, pp. 65–74.

[41] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[42] H. Hermansky and N. Morgan, "Rasta processing of speech," *IEEE transactions on speech and audio processing*, vol. 2, no. 4, pp. 578–589, 1994.

[43] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[44] A. Eronen, "Comparison of features for musical instrument recognition," in *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*, IEEE, 2001, pp. 19–22.

[45] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra*, Springer, 1971, pp. 134–151.

[46] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[47] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[48] C. Jutten and J. Herault, "Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture," *Signal processing*, vol. 24, no. 1, pp. 1–10, 1991.

[49] S. Choi, A. Cichocki, H.-M. Park, and S.-Y. Lee, "Blind source separation and independent component analysis: A review," *Neural Information Processing-Letters and Reviews*, vol. 6, no. 1, pp. 1–57, 2005.

[50] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[51] ——, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[52] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*, IEEE, 2003, pp. 177–180.

[53] P. Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *International Conference on Independent Component Analysis and Signal Separation*, Springer, 2004, pp. 494–499.

[54] ——, "Probabilistic decompositions of spectra for sound separation," in *Blind Speech Separation*, Springer, 2007, pp. 365–386.

[55] P. Smaragdis, B. Raj, and M. Shashanka, "A probabilistic latent variable model for acoustic modeling," *Advances in models for acoustic processing, NIPS*, vol. 148, pp. 8–1, 2006.

[56] T. Hofmann, "Probabilistic latent semantic indexing," in *ACM SIGIR Forum*, ACM, vol. 51, 2017, pp. 211–218.

[57] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[58] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[59] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 1096–1103.

[60] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2016, pp. 241–246.

[61] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, ACM, 2016, pp. 153–162.

[62] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[63] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, 1996, pp. 226–231.

[64] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[65] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, ACM, 1992, pp. 144–152.

[66] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[67] C.C.Chang and C.J.Lin, *LIBSVM: A library for support vector machines*, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[68] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[69] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[70] L. R. Rabiner and B.-H. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[71] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

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

[73] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[74] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[75] F. A. Gers and E Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[76] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[77] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[78] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya, "Multilingual language processing from bytes," *arXiv preprint arXiv:1512.00103*, 2015.

[79] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[80] E. Cakır and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[81] C.-F. Cheng, D. V. Anderson, M. A. Davenport, and A. Rashidi, "Audio classification based on weakly labeled data," in *2018 IEEE Statistical Signal Processing Workshop (SSP)*, IEEE, 2018, pp. 568–572.

[82] A. Rashidi, M. Sigari, M. Maghiar, and D. Citrin, "An analogy between various machine-learning techniques for detecting construction materials in digital images," *KSCE Journal of Civil Engineering*, vol. 20, no. 4, pp. 1178–1188, 2016.

[83] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," *Data mining techniques for the life sciences*, pp. 223–239, 2010.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

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

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

[87] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Nov. 2018, pp. 9–13.

[88] H. Lim, J. Park, and Y Han, "Rare sound event detection using 1d convolutional recurrent neural networks," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017, pp. 80–84.

[89] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, "Experiments on the dcase challenge 2016: Acoustic scene classification and sound event detection in real life recording," *arXiv preprint arXiv:1607.06706*, 2016.

[90] S. Adavanne, G. Parascandolo, P. Pertilä, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," *arXiv preprint arXiv:1706.02293*, 2017.

[91] G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, p. 048 317, 2006.

[92] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[93] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 49–57, Nov. 2010.