University of Sheffield

Sound Event Detection in Real Life Audio

Jack Deadman

Supervisor: Dr Jon Barker

A report submitted in fulfilment of the requirements for the degree of MComp Computer Science

in the

Department of Computer Science

May 3, 2017

Declaration

Abstract

The real world is full of sound events, cars passing, glass breaking, birds tweeting. This report presents a polyphonic sound event detection system to detect these events in real life audio. The system is compared against the state of the art techniques used in the 2016 DCASE challenge, this is achieved through using 1 second segment-based performance metrics using Error Rate and F-Score.

The project aimed to experiment with novel feature extraction and data augmentation techniques to find where the performance benefits can be achieved.

By introducing novel feature extraction and data augmentation techniques the sound event detection system successfully showed a considerable improvement over the baseline system on a completely new dataset, achieving an Error Rate of 0.76, compared to the higher Error Rate of 0.96 achieved by the baseline system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Introduction

The aim of sound event detection (SED) is to recognise distinct events in continuous acoustic signals to which a label can be attached. For example, a given acoustic signal may be of an elderly person at home and an event could be of a person falling. It is evident that a system that can detect a person falling can help improve people's quality of life [1]. The elderly person would not be stranded on the floor for hours, and help could be notified immediately. Unfortunately even with the current state-of-the-art sound event detection techniques [2], a reliable system that achieves this is not yet feasible.

Further benefits of SED is that it aids other research areas. The field of acoustic scene classification can benefit from the ability to use an event in the environment as a heuristic in determining the scene. Embedded systems can be improved with a better perception of the environment they are in by using their auditory features [3]. Further applications of sound event detection include smart home devices [4], military hardware [5], indexing files with audio [6] and even a smart phone application that monitors sleeping patterns [7] to name a few.

However, sound event detection is a difficult problem to solve because a sound event will rarely occur in isolation. For example, in a given segment of audio, a car could be passing by, a bird could be singing and a person could be talking. Audio like this is known as *polyphonic* or *multi-sourced* and is very challenging as it is difficult to find strong features that are not corrupted by the mixture of events. The difficulty of the problem is increased by the fact that the same event can sound very across different instances. For example, the amount of energy at the start of plosive sounds like in doors slamming can vary a lot, changing its acoustic characteristics. Another challenge in SED is handling context. Sound events do not sound exactly the same through-out their life time. Imagine placing a glass in the drying rack after washing it. The initial sound may be of the glass impacting with another glass, the glass may then further slide into place generating another sound. These sounds may all be classed as one sound event of “Washing Dishes”.

In 2016, the DCASE [8] challenge was introduced to encourage research in the field. The organisers provided the TUT dataset [9] and invited researchers to take part in a challenge to create a SED system to detect events in two scenes: a home and a residential area. This project will use and extend the dataset provided and benchmark the performance of the proposed system against the baseline system created for the challenge. The system will be scored using the performance metrics [10] provided by the organisers in order for it to be a fair comparison.

Sound event detection is a fairly well established field of research. Early techniques focused on classifying events in isolation [11], i.e., the systems assumed only one event could occur at a given time. These systems are known as monophonic. The real world, however, has a high level of polyphony; most events do not occur in isolation, they occur with numerous other events at the same time.

1.1.1 Handling Polyphony

A monophonic system will fail to detect two events of interest that occur at the same time; this could be detrimental for some applications. There are two main approaches to solving the polyphony problem: using multiple *single label classifiers* and using *multiple label classifiers* [12].

Single Label A single label classifier can be trained for each of the events, creating a different model for each type of event. Thus, single label classifiers can essentially only classify one type of event. When a test signal is being classified, the frames can then be tested against each of the classifiers. If several classifiers return a high probability of the frame belonging to their particular class then the data is assigned to those classes. This approach was used in [13].

Multiple Label The next approach involves training a classifier with an output vector $(x_0, x_1, x_2, \cdots, x_n)^T$ where n is the number of classes. The value of each element can be either 0 or 1 depending on if the event is present, or a confidence value between 0 and 1.

For example, an ideal classifier for the data in figure 1.1 would be a three-class classifier and would return the vector $(1, 1, 0)^T$ at time t_0 and $(0, 1, 1)^T$ at time t_1 assuming the dimensions of the vector as “People walking”, “Car passing” and “Bird singing” respectively. This approach was used in [14] and [15] and was successful.

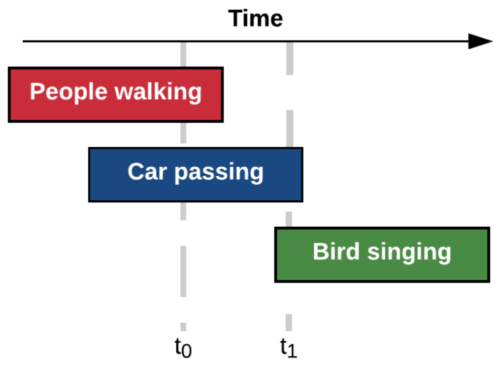


Figure 1.1: Diagram showing how different events can occur at the same time. People walking and car passing occurs at the same time at t_1 and car passing and bird signing occurs at the same time at t_2.

1.1.2 Handling Context

When determining an event, knowing what has occurred prior to the event helps classify the next event more accurately. A common approach to add this contextual information is to use a Hidden Markov Model (HMM) [16][17] which uses a Finite State Machine with transitions which have associated probabilities. Some events are more likely to keep occurring once they are in an active state. "Washing dishes" is one such event; a person does not normally wash dishes for a fraction of a second.

Another approach to add this contextual information is to concatenate the feature vectors of the previous n frames when classifying an event, this was used in [18] and [13].

More advanced modern techniques include using a Recurrent Neural Network [19][20] which use the current frame as well as weighted inputs from the previous time step as inputs in the network.

1.2 Aims and Objectives

The project aims to fulfil the following aims and objectives, the project's main aims are 1 and 2, however in order to achieve these objectives 3 and 4 must also be achieved.

1. Experiment with extracting useful features from audio signals that are not commonly used in this problem domain.
2. Construct data augmentation techniques to artificially increase the limited data available in this field while being precautious of mismatch in the process.

3. Establish an extensible sound event detection (SED) framework to allow for experiments to be easily carried out.
4. Develop an online system to run and compare sound event detection systems with audio uploaded from a user.

Features used in SED systems are very uninspired with Mel-Frequency Cepstral Coefficients (MFCCs) being the most popular choice. Given that sound events are complicated signals that can contain multiple different events at the same time strong features are required for an effective performance, this is why this is one of the main research topics of the project. As well as this, the amount of data available in the field is very limited compared to other fields such as automatic speech recognition. A lack of data makes it difficult to create robust SED systems as some events could have very few instances in training, which can make it difficult to classify unseen data.

1.3 Overview of the Report

Chapter 2 provides the background material for creating a sound event detection system, which includes the fundamentals of signal processing and the challenges in the field and how these are being solved. The chapter then goes onto explain the literature of feature extraction and data augmentation techniques, creating a good foundation for the following research. Following this, in chapter 3 the visualisation tool that has been created to better understand the performance of a SED system is discussed. The chapter details the requirements for the tool, the design and how the final result was achieved. The report then continues on to describe the baseline system in chapter 4. This chapter details how the MFCC features are implemented as well as the underlying classification system and then goes into detail about how the system was re-engineered into a better framework for experiments. The following two chapters, chapter 5 and 6 describe the main experiments of the project. In chapter 5 the acoustic feature experiments are explained. Different features are experimented by firstly extending the MFCC features used in the baseline system by simply concatenating different features onto the existing feature vectors. The better performing features are then combined together in an attempt to create an overall better performing system. In chapter 6 the work in chapter 5 is continued with experiments with data augmentation. Various augmentation techniques are explored by augmenting the audio in the frequency domain and the time domain. The effect of the augmentation on different features is assessed to see how augmentation improves or worsens the results with different feature types. The various augmentation techniques are then combined to see how the different augmentation techniques complement, or alternatively, conflict with each other. Finally, using what was learnt in these experiments, a combined best system is created, which is then compared against the baseline system on unseen data using the optimal configurations for the two systems.

Chapter 2

Literature survey

This chapter aims to provide the background information for creating a sound event detection system. It begins by describing the fundamentals of processing audio for sound event detection, explaining how useful information can be retrieved from an audio signal. The chapter goes on to describe the literature for feature extraction and data augmentation which will be the main focus for the project. The chapter then concludes with a discussion on evaluating SED systems. This is shown in two ways, firstly using formal methods by using F-Score and Error Rate to give a quantifiable measure, and then by exploring graphical systems which visually represent the output labels given by a sound event detection system. This can be used to better understand the types of mistakes the systems are making and to give a more subjective evaluation of performance.

2.1 Sound Event Detection Background

2.1.1 Signal Processing

Audio is a continuous signal. In order to capture this, the signal needs to be digitalised; this is achieved through sampling. A standard sample rate is 44.1 kHz which means that for every second of audio there have been 44100 samples recorded. To detect audio events, the samples are grouped into frames; these frames normally contain 40 ms worth of samples. This frame is then put through a classifier, where it would be assigned a label. The frame is moved along, usually by 20 ms [21] or 50% the frame size; this is known as the hop length. The new frame is then classified. This process is repeated until all the data has been completely classified. The last frame is padded with zeroes if it does not line up exactly with the audio. Between some frames of an event there may be no distinguishing features, as happens with footsteps, where there is a no distinguishing sound between steps. To solve this issue, the classifier needs to either use the context of the previous frames or the frames need to be smoothed in a post-process stage [18], where smoothing fills in the gaps between labels if they occur close enough.

Analysing audio in the frequency domain through framing can cause unwanted side effects. The Discrete Fourier Transform (DFT) [22] is used to transform a signal from the time domain to the frequency domain by assuming the signal in that frame is periodic. This is because all periodic signals can be represented as an infinite sum of sinusoidal components. In practise, infinite components are not possible; therefore a finite set of components need to be chosen, usually 2048. The number of components are usually a power of two as this allows for optimisations with the Fast Fourier Transform (FFT) implementation. When using 2048 components, the actual frame size is around 46.6 ms (2048/44100) worth of samples; this also shows the trade-off between time and frequency resolution. The more components, the better the frequency resolution, except with the downside of a poorer time resolution. Each component consists of a real sine wave and an imaginary cosine wave, which can be compressed using Euler's identity i.e., $e^{i\theta} = sin(\theta) + icos(\theta)$. Formally the DFT is defined as:

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{i2nk\frac{n}{N}} \text{ for } K = 1 \cdots N$$

Where N is the total number of samples in the frame (e.g 2048), x_n is sample n in the time domain signal x and k is number of cycles that component has in the frame i.e, the frequency. Therefore, using a 2048 component DFT on a time domain signal will result in a 2048 element sized complex vector. The first component X_0 indicates the average energy in the signal i.e., $X_0 = \frac{1}{N}\sum_{n=0}^{N-1} x_n$. For each complex component the absolute value $|X_k|$ is the amplitude of the frequency k and the argument $arg(X_k)$ is the phase of frequency, i.e., the shift. The frequency components are indexed from 0 to 2047. To calculate their corresponding frequencies in terms of audio they need to be converted with respect to the sample rate the audio was captured at e.g., component 10 for a sample rate of 44.1 kHz corresponds to 215.3 Hz ($10 \times (44100/2048)$), Nyquist theorem [23] states that the sampling rate must be at least twice the maximum frequency, therefore only components up to 1024 (22050 Hz) are kept. More detail on this is specified in the description of the implementation of the baseline system in chapter 4.

For the DFT to work, the signal needs to be statistically stationary. This means that the frequencies that make up the signal do not change over time. This is clearly not the case for a complicated sound, such as a sound event which varies over time. However, it is assumed to be quasi-stationary, that is, over a short period of time the signal is stationary. In practise this can cause issues; the maths behind the DFT assumes the signal in the frame repeats infinitely in the future and in the past in the exact same way it appears in that frame [24]. In figure 2.1 a simple sinusoid wave is used to demonstrate the discontinuity caused by this assumption. Discontinuities cause issues because frequencies can bleed into other frequencies,

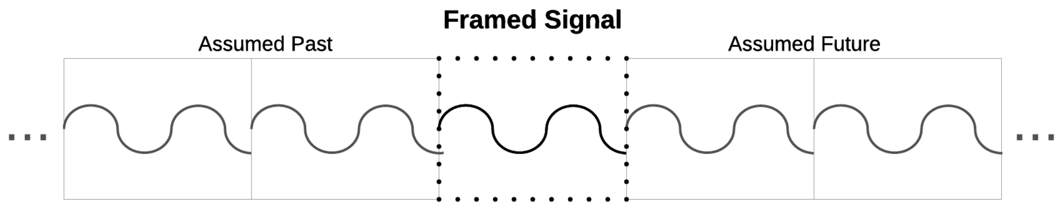


Figure 2.1: Diagram showing the discontinuities caused by framing a signal. The framed signal is assumed to be repeating infinitely to the left and to the right.

creating frequencies which are not actually present. To fix this issue a windowing function can be used. Windowing functions dampen the start and end of the frame to make the repetition free of discontinuities The Hamming window [25] is the most common windowing function. The Hamming window is defined with the following formula, where N is the width of the window and n is the current sample index in the window:

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi\, n}{N-1}\right)$$

An example of this windowing has been generated and displayed in figure 2.2. The example shows a sine wave that does not complete a full cycle. The sample is assumed to repeat like this and the discontinuity can been seen. The Hamming window that will be applied to the first frame is shown in the centre of the image. The final image on the right shows the new dampened signal and the repeated frame. The discontinuity reduction between the two frames can clearly be seen, along with the distortion created.

In the baseline implementation, each frame is 40 ms of the audio which is then padded with 0s to be 46.4 ms long (2048 DFT components); these resulting frames are then processed.

2.2 Structure of Sound Event Detection Systems

Every sound event detection system works roughly in the same distinct stages. The flow of the stages are illustrated in figure 2.3. The blue procedures in the diagram represent the main focus points of this project.

The organisers of the DCASE challenge provided baseline Gaussian Mixture Model systems written in Python [13] and MATLAB [26]. The baseline systems were made for the competition and provided useful functionality such as automatically downloading the dataset and caching the progress of the system when it is running. However this system is quite rigid and it is not flexible for experiments outside of the challenge's constraints.

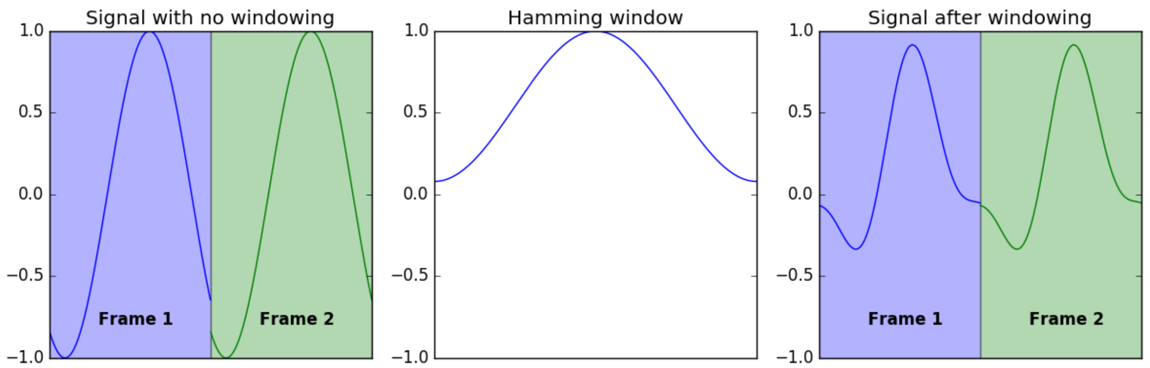


Figure 2.2: Simple plot showing a signal before and after windowing. The image on the left shows a shifted sine wave that does not complete a full cycle. The image shows the discontinuities when frame 1 is assumed to be repeated in frame 2. The image in the middle shows the Hamming window that will be applied to frame 1. Finally the image on the right shows the sine wave in frame 1 after an element wise multiplication; the new frame is then repeated in frame 2. The image clearly shows the new distortion created from the windowing, however the discontinuity between the frames has severely reduced.

Dataset The starting point of a SED system is the dataset. If the training data is not very good it does not matter how good the other components are. Also, if there is not much data to train the system on, the system will struggle to generalise well. Datasets can be created by recording real life audio, which will of course produce the most reliable data. The dataset can also be extended in artificial ways, such as augmenting the existing data or simulating new data through combining isolated instances of sounds and placing them into a scene.

Pre-processing Pre-processing involves removing data from the audio that is of no use and will affect the classification performance negatively; this stage is often ignored as robust classifiers should be invariant to such differences in the audio. However is has been shown to be effective in [27] where removing noise from the audio was used.

Feature Extraction Feature extraction is needed by all the systems. This stage involves finding useful properties in the data that can distinguish sounds with different labels. This will be one of the main focus points of the project as most modern SED systems do not experiment with many different feature types.

Classifier The classification stage can be implemented using a variety of different techniques. A classification system can learn from labelled data; this is known as supervised learning. Alternatively, it can learn from unlabelled data which is known as unsupervised learning. As the labelled TUT dataset will be used in the project, a supervised method will be used for classification. In particular a Gaussian Mixture Model (GMM). The GMM is a distribution

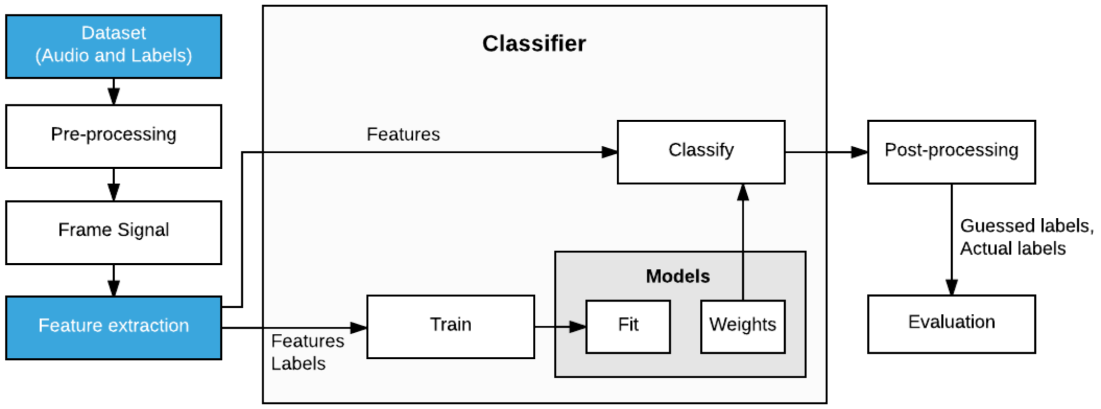


Figure 2.3: Typical stages of a SED system, starting with pre-processing and ending with post-processing. The results are then evaluated to judge the performance of the system.

that is learnt in an unsupervised manner, however the overall classification procedure is supervised as a different model will be trained for each label.

Post-processing Classification systems are not perfect and some of their mistakes can be eradicated by using some simple techniques, such as removing events that the system detected to be too short to be real [13] [18]. The systems entered into the DCASE challenge often used hard-coded knowledge for their post-processing techniques, rather than learning from the training data.

2.3 Feature Extraction

The features chosen to be used in classification can greatly affect their performance as they are used to distinguish different sound events. There are three main types of features that can be used in sound event detection. These are: Spectral (Frequency Domain), Temporal and Spatial.

2.3.1 Spectral Features

For sound event detection the most common features used are Mel-Frequency Cepstral Coefficients (MFCC) [28]. They are used to smooth the spectrum; this is normally used in an attempt to model the vocal tract of a speaker. This focuses on what is being said rather than how it is being said. This property makes them an odd choice to be the most popular feature in sound event detection, this is because the audio source is a scene made up of many sources and not a speech signal. MFCCs throw away a lot of information to achieve smoothing, such as high frequency sounds like noisy fricatives. Noise can be useful when determining the class of an event. For example, object rustling and water tap running are very noisy sounds. Additional features can be generated by using the previous frames. This is achieved through using the differential (Δ) and the acceleration coefficients ($\Delta\Delta$) of the MFCCs. These indicate how the features are changing with respect to time. The lack of creativity in the features used in the DCASE competition and sound event detection in general is an area where all the systems could potentially improve. However MFCCs do seem to perform well and they are also an attractive feature choice because they are very decorrelated making them ideal for the GMM based classifier. The project aims to see if they are valid to be the state of the art, or whether performance benefits can be gathered through better feature choice.

Apart from MFCCs, other spectral features are possible to extract. Task 1 of the DCASE challenge (Scene Classification) had participants which used a larger range of different features. For example in [29] spectral flux and centroid were used in the scene classification part of the DCASE challenge. Centroid indicates the 'Center of Mass' of the spectrum and flux measures how quickly the power spectrum changes between frames. Additionally Rolloff was used in [30] along side other features such as flatness and energy. Other spectral features include, Slope, Spread and Sharpness of the spectrum, these all describe different properties of the spectrum.

2.3.2 Temporal Features

Another area of feature extraction and where the entrants of the DCASE failed to take advantage of was using temporal features. Most of the systems entered used spectral features, neglecting the time domain. Simple features such as the zero crossing rate [31] can aid in differentiating events. Zero crossing rate is the rate at which a signal's value changes sign. It is normally used in determining whether a frame of speech is voiced or unvoiced. However, it could be used in sound event detection. For example, a noisy signal such as a car engine will have a high zero crossing rate. The scenes in the dataset are very noisy in general, this means that the zero crossing rate may be high most of the time during the scene, which is a downside to this feature.

2.3.3 Spatial Features

One entry in the DCASE competition took advantage of the fact that the audio recordings were captured on a binaural device [20]. They used spatial features to aid classification in a polyphonic environment. If two sounds come from different parts of the room, they are likely to be separate events. The spatial feature they used was time delay of arrival (TDOA) which works in the same way humans determine the location of the source of a sound. TDOA uses the fact that a sound will reach one channel before the other channel. If the human body detects a sound in their right ear before the left then it is likely to be coming from the direction to their right.

2.4 Data Augmentation

As has been stated before, the lack of data in the challenge was a major issue in the challenge. The entrants of the challenge did not take great advantage of the option of augmenting the given data to artificially create more data; thus data augmentation is another potential area where improvement is possible. The performance gain from augmentation will vary between the different classifiers. Ideally a good classifier should be invariant to small changes and would therefore not benefit much from data augmentation. Machine learning systems are treated like black boxes, it is not possible to completely understand what the system has learnt. However if data augmentation is used then certain characteristics of the audio can be learnt by the classifier on purpose. For example the fact that pitch of certain sound event types can vary slightly can be learnt by the classifier by simply training the classifier on the same samples with a augmented pitch, this it called pitch perturbation.

2.4.1 Audio Mixing

Mixing the sound recordings together was used in [32], however it was not very effective with the Convolutional Neural Network (CNN) classifier implemented. Their systems learnt the effect of simultaneously occurring events; this is the opposite approach to [33] which attempted learning sounds in isolation. Event mixing allows for the classifier to learn the features that are more important for that sound event. This is because the classifier will learn if there is no correlation between certain features when the sound event is mixed with other events and the classifier will learn the more important features through their being correlation between the event in isolation and the event when mixed with other events. For example, the sound of rustling is noisy. If a frame contains just rustling there will be a high zero-crossing-rate. If the event is mixed with something such as an object impact, the frame will still contain the high zero-crossing-rate, indicating it is a useful feature.

2.4.2 Vocal Tract Length Perturbation (VTLP)

Another spectral augmentation technique used is Vocal Tract Length Perturbation (VTLP). This is a technique used in automatic speech recognition to create speaker independent systems. It is derived from Vocal Tract Length Normalisation which aims to normalise the signal to be speaker independent. In [34] the experiments showed that VTLP improved the performance for speech recognition. It has also been shown to improve performance in sound event detection in [35]. VTLP can be used to create sound events that are similar to the original sound event by augmenting the size objects involved. For example the sound of an object hitting a table could be augmented to sound like an object hitting a slightly larger table.

2.4.3 Speed Perturbation

Speed perturbation was also used to augment data in [32]; a small improvement was achieved through this. Speed perturbation involves speeding up the data and slowing it down through re-sampling the audio. When performing this procedure, the pitch is also increased as a side-effect. However there are techniques to mitigate this effect [36], to create more natural sounding audio.

Along with using the existing data, more data can be added to the dataset. This data will need to be simulated, such that it sounds like it was produced in the same room as the other events by using the reverberation characteristics of the room that the other events were present in.

2.5 SED Evaluation

What makes a good SED system is quite a subjective task. One might want a SED system to only detect events when they occur and any mistakes are critical. On the other hand a SED system detecting mistakes may not be critical as long as the important events are being detected. An analogous to this is with cancer screenings. Mistakenly detecting someone has cancer will only cause unnecessary worry. It is far more critical if someone with cancer is not detected. The DCASE challenge provided resources and constraints for creating a SED system and then presented the evaluation metrics to use.

2.5.1 DCASE Challenge

In 2013 [37] a group of researchers created the DCASE challenge to encourage research in sound event detection. The challenge was a success and led to another challenge in 2016 [8]. The organisers of the competition considered the opinion of the research community to shape the rules of the competition in order to better suit the current research interests in the field.

The challenge involved participants creating systems to detect acoustic events in two different locations: a home and a residential area. For the competition, the organisers provided the TUT [9] dataset for the participants to train their systems on as well as a program to evaluate their systems objectively [38].

The systems were then judged by similar data, recorded using the same hardware. The competition had strict rules stating that the systems entered into the competition could only be permitted to use provided data to train their systems. However, augmenting the provided data was allowed.

The competition had unrealistic constraints. The participants were forced to augment individual data samples to create variability in the dataset. In a real-life scenario, more data would be gathered before augmentation. The organisers had to put this constraint on participants to ensure focus on the performance of the systems rather than the amount of data they could gather.

2.5.2 SED Metrics

To judge a sound event detection system a quantitative measure is needed. For the DCASE challenge, two types of metrics were used: event based and segment based [10]. The segment based metrics will be used to evaluate the proposed system.

The systems are scored based on their output labels. The labels are produced by creating a text file in which each line has the onset, offset and event name respectively, separated by a tab character. Both metrics used this output file.

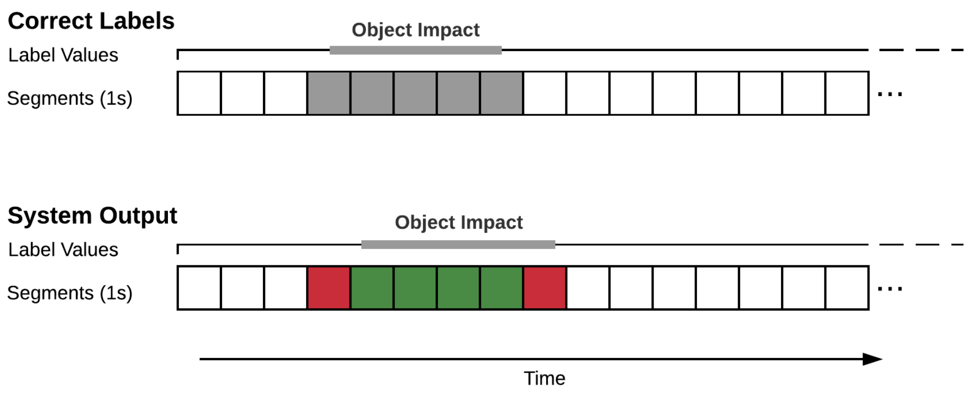


Figure 2.4: Diagram showing segment-based evaluation metrics. The solid red box indicates a segment as being incorrect and the green boxes indicate a segment as being correct.

Segment Based Metric For the segment-based metric, the labels are broken down into 1 second segments. If an event occurs at any point during this 1 second segment then that segment contains that event.

In figure 2.4 an “Object Impact” event is used as an example. The event occurs between around 3.5 s to 7.5 s. This means the 4th-8th segments contain the event, indicated by the solid green squares in the diagram.

Event Based Metric The event based metric evaluates the performance of a system by considering the labels event by event rather than second by second. An event is considered to be correct if the onset is within 200 ms of the correct onset and the offset is within 200 ms of the correct offset or within half the duration of the event for longer events.

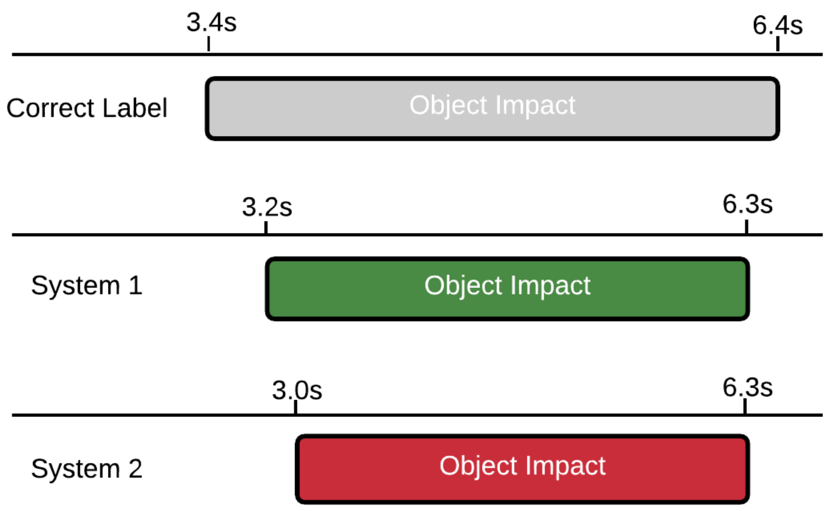


Figure 2.5: Diagram showing how event labels are evaluated event by event

Figure 2.5 shows how events are evaluated in the event based metric. The event "Object impact" is labelled as having occurred between the timestamps 3.4 and 6.4 seconds. The first system correctly classified the event as both the onset and offset were within the 200 ms. System 2, however, failed to detect the event because the onset was too late. This metric is perhaps too harsh on the second system. The system still managed to detect the majority of the event; the segment based evaluation metric would have rewarded the system for that.

The best metric to use depends on the application of the desired system. For a system where detecting the exact time an event occurs is important, using the event based metric would be best. A system where detecting the rough duration and the rough start time is good enough, would benefit from the segment based metric. For the DCASE challenge, they used both types of rating metrics, which was quite a bizarre choice. It would have been better to choose one metric. Using both metrics resulted in entrants performing well for one metric and not so well for the other metric, as they optimised their particular system for that metric. This made it difficult to select an overall winner.

Segment based and event based metrics are different perspectives on looking at the data. Both these perspectives can be used to calculate scores. The scores that were decided by the organisers were *F-Score* and *Error rate*.

Error Rate This score evaluates a system based on how many errors it makes. The lower the score, the better the system has performed. A system can easily get the good error score of 1 by not labelling anything, making it difficult to interpret, therefore it should be used with a complementary score.

$$ER = \frac{\text{substitutions} + \text{deletions} + \text{insertions}}{\text{No. of events to detect}}$$

Where *substitutions* are the number of correct detections with incorrect labels. *Deletions* are failing to detect an event that occurred and *insertions* are detecting an event that did not occur.

F-Score This score uses the harmonic mean of precision and recall, where precision is the percentage of events detected that are correct and recall is the percentage of the correct events detected.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F-Score} = \frac{2 \cdot \text{Precision} \cdot Recall}{\text{Precision} + \text{Recall}}$$

Where TP is the total number of True Positives, FP is the total number of False Positives and FN is the total number of False Negatives.

The organisers of the challenge provided the SED Eval program [38] to automatically evaluate the systems using these metrics. For this project the 1 second segment metrics will be used to evaluate and compare the system. This is because it better represents the overall performance of a system; it was also the preferred metric by the organisers.

Please note the difference in the similar sounding terms sample, frame and segment. A sample of audio is a snapshot of the continuous real world audio signal. These snapshots typical happen 44100 times a second. A frame is a group of these samples in a usually 40ms block, often processed with a windowing function. The underlining classification system being used works with features extracted from these frames. This means that the classifier learns from a dataset of frames, therefore when the term data sample is used, this refers to a frame and not an individual sample in the original audio file. Finally the term segment is only used in evaluating a system. The decisions of the SED system are broken down into 1 second segments and then judged using the metrics that have been just discussed.

2.6 Sound Event Detection Visualisation

Numeric measures are useful for determining how well a system is performing objectively. However, being able to see how well a system is performing using a visual system will allow specific weaknesses of the system to be seen, such as graphically being able to see an event always detected late. Heittola created a Flash visualisation tool [39] to graphically render the results of their experiments. The system only works with three predetermined scenes but provides a lot of tools to analyse results. The waveform and the current labels being detected can be seen to aid seeing how the system can be improved. This improvement can then be measured objectively using an evaluation metric. A screenshot of the tool is displayed in figure 2.6.

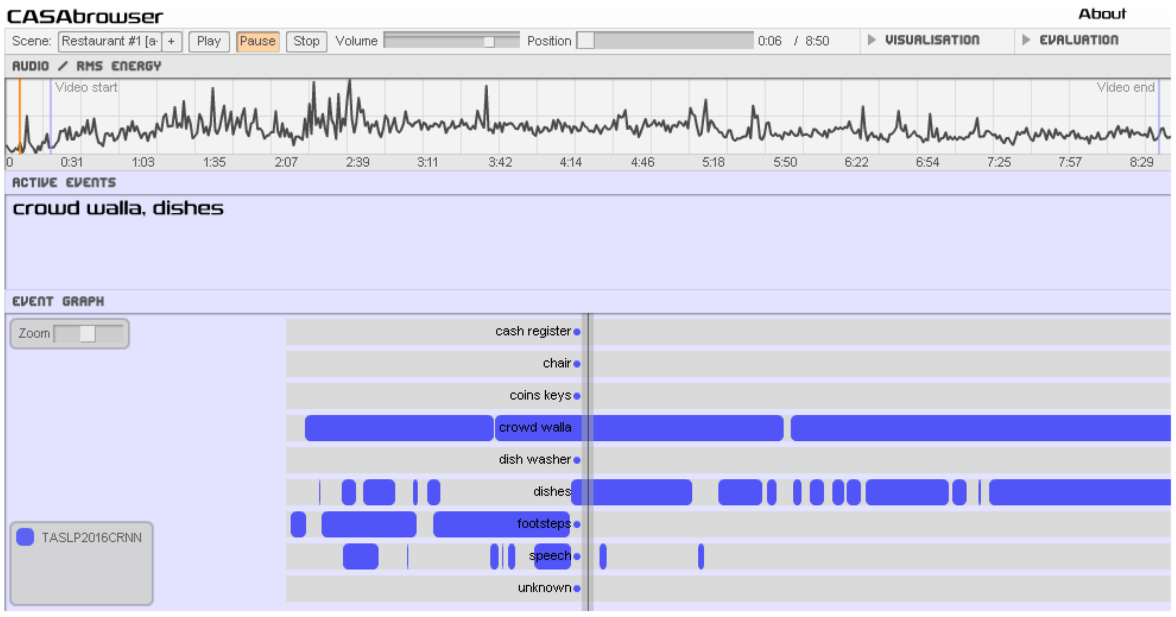


Figure 2.6: Screenshot taken from the flash visualisation tool created by Toni Heittola, demonstrating the results of classifying events in a restaurant.

Heittola has also created another sound event detection visualisation tool called *sed_vis* [40]. This tool is a Python module that was recommended by the organisers of the DCASE challenge. Compared to the flash visualisation tool, this one is more flexible; it allows for any audio file and two annotation files to be selected: the correct labels and the system's labels. However this tool lacks some fundamental features such as being able to pause and seek to specific locations in the audio. A screenshot of the tool is displayed in figure 2.7.

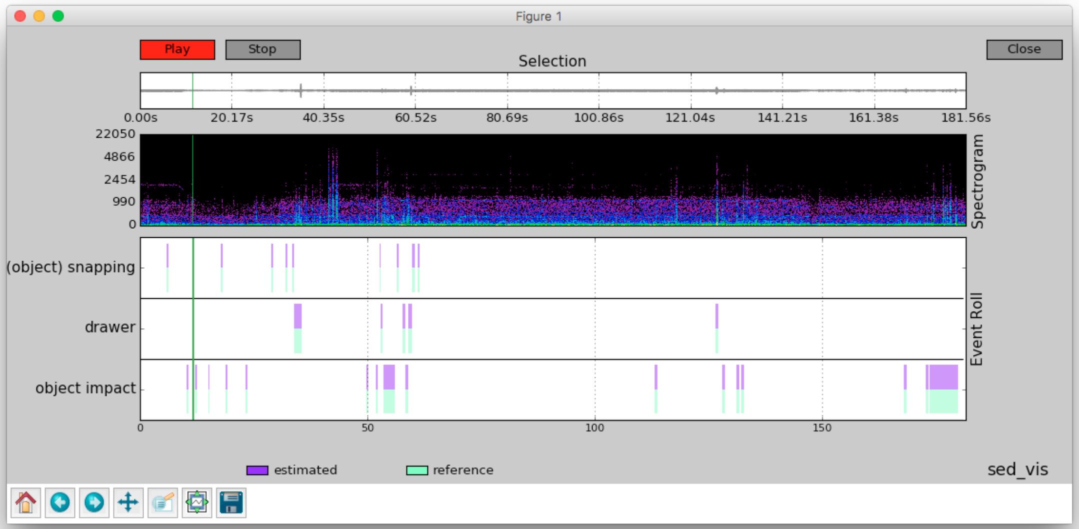


Figure 2.7: Screenshot taken from sed_vis Python library

2.7 Summary

In this chapter the techniques to go from a noisy continuous signal into useful features was explained. The chapter began by explaining how the audio is grouped into frames and how the Discrete Fourier Transform is used to turn the time domain signal into the frequency domain. The chapter then went onto explain how a complete sound event detection system can be created, highlighting all the key stages such as: the dataset, pre-processing, feature extraction, classification, post-processing and evaluating. The chapter then went onto explain further the literature of the main focus points of the report, feature extraction and data augmentation. Then the chapter then concluded with further details of the DCASE challenge and the performance metrics that will be used for this project.

Chapter 3

SED Visualisation

This chapter is a self-contained description of the SED Visualisation tool that has been developed to aid displaying the performance of SED systems. This tool will be used for a deeper analysis of the systems, further than what can be achieved through looking at performance metrics. The aim of the SED visualisation tool is to graphically display how well Sound Event Detections systems are performing by allowing the user to listen to the audio and watch the labels appear in real time. The system aims to have a low barrier to entry, in particular to allow users to compare these systems with no installation; the web is a great platform for this. A frontend application will be created to visualise the events, while the labels will be generated by running an audio file through the SED systems on the server. The chapter begins by setting out the requirements of the tool, followed by the design and then concludes with the implementation created.

3.1 Requirements and Analysis

The system will require two endpoints to function, a frontend and a backend. The frontend is an interactive web application which allows the user to upload an audio file and select which Sound Event Detection system should be evaluated. The backend system needs to store the uploaded audio file and then run the audio file using the selected Sound Event Detection system. After classification the resulting labels will be returned to the frontend which will display the results to the user. The user will then be able to play the audio file and then watch the labels of the classifiers scroll right to left as time passes giving the user an intuition of how well the system is performing. The user will also have the option to upload a labels file in the format specified by the competition organisers. The visualisation tool should fulfill the following list of requirements:

1. The user should be able to upload an audio file.
2. The user should be able to select which SED systems should be run on the file.

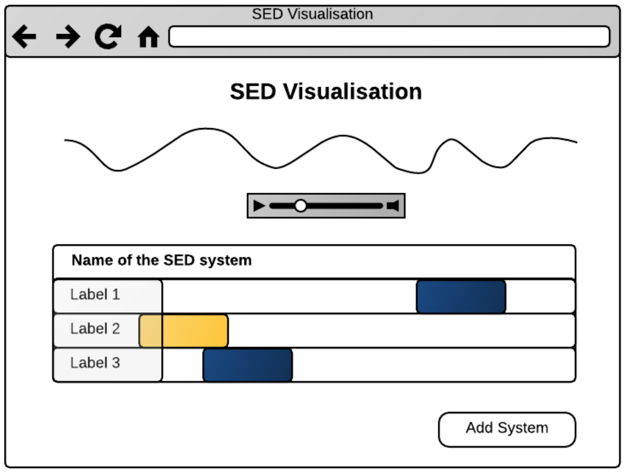


Figure 3.1: Mockup of the design of the main page of the visualisaion tool. The user has uploaded an audio file and selected a SED system. The mockup shows that at this current timestep the event 'Label 2' is currently active. 'Label 3' will soon be active at the end of this event.

3. The server should run the selected SED systems on the uploaded files.
4. The user should be able to see the labels relative to the time position of the audio player.
5. The user should be able to seek to different positions in the audio.
6. The user should should be able to play and pause the audio.
7. The user should be able to remove a SED system from being displayed.
8. The server should store the uploaded audio files for reuse.
9. The user should also be able to interact with the audio through clicking on a waveform.
10. The user should be able to optionally upload a labels file.

3.2 Design

Allowing a low barrier to entry is a key requirement of the tool, because of this it is important that the user is able to view the application without uploading any files. On the homepage the user will be presented with the option to upload their own file or to use an example file already on the server. Once uploaded the user will be redirected to the main part of the application. In figure 3.1 a mockup of the main page is presented. The user will be able to see a soundwave at the top of the page - this is useful to allow the user to quickly see points

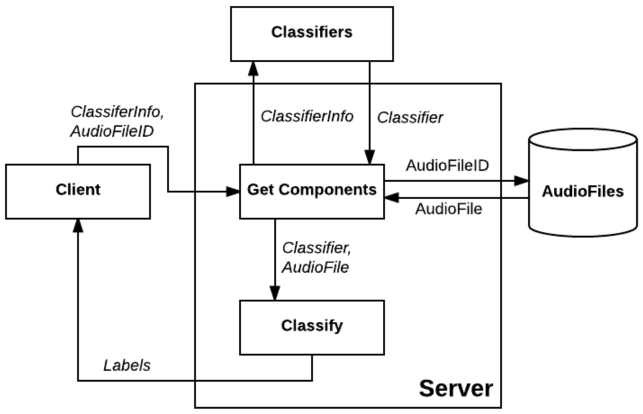


Figure 3.2: Diagram indicating the flow of data between the client and the server. The client starts by requesting the labels for the audio file with the specified ID using the specified classifier parameters. The server then runs the classifier and returns the labels.

of interest in the audio, such as seeing spikes to locate loud plosive sounds. Underneath this are the controls for the audio player. Finally below this is the main component of the page, the labels slider. The currently active events are coloured yellow while non-active events are coloured blue. As the audio is played the events segments slide from the right to the left.

The interaction between the frontend of the application and backend of the application is the most key part of the architecture of the visualisation tool. The flow of data is depicted in the diagram in figure 3.2. Once a user has uploaded an audio file it will be stored on the server and is given a unique code called the audioID. When the frontend wants to run a classifier on the audio file the frontend will pass the audioID to the server along with the parameters of the classifier. The parameters of the classifier will include data such as the number of components used in the GMM, the features to extract from the data and what data the classifier was trained on, these parameters will be explored further in the following chapters. On the server, the server will retrieve the corresponding classifier to the specification given by the user and retrieve the audio file corresponding to the audioID. From this the server will then run the classifier using the audio file. The classifier will output the labels; this is then relayed onto the frontend.

3.3 Implementation

The frontend of the application is a web application written mainly JavaScript. The JavaScript framework React.js was used to speed up the development, allowing robust and reusable components to be constructed. The server side of the application was written in Python using the framework Flask. Writing the server side code in Python allowed the classifiers to be easily be integrated into the application as they are simply imported as a module. A live version of the application is available at `https://sedvisualise.com`. Instructions on how to install the application locally are located in the appendix.

The user is first presented with the option to upload an audio file, this page is shown in figure 3.3. Currently the only supported file type is *wav*. The user also has the option to use an example file.

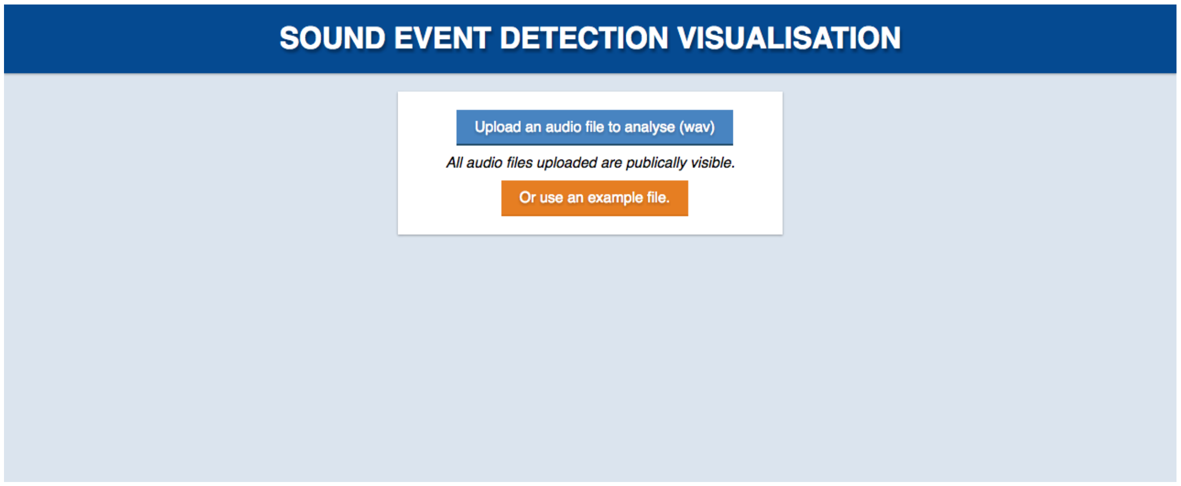

SOUND EVENT DETECTION VISUALISATION
Upload an audio file to analyse (wav)
All audio files uploaded are publically visible.
Or use an example file.

Figure 3.3: Homepage of the application, the user has the option to upload their own audio file or to use the example file on the server.

When an audio file has been uploaded to the server a random code is generated to identify the file; the file is then stored in a folder with the name being that random code. The user is then redirected to */play/*[*audioID*] to view the main part of the application, this link is shareable allowing users to share their results easily. This page is shown in figure 3.4. In this image a labels file has been uploaded and the events are shown in the top slider. The baseline system for the home scene is being displayed below this. Immediately from inspection it can be seen that the baseline system is able to detect the event of a tap running really well. The sound of dishes were also detected but not for the full duration. The sound event of cutlery was completely missed by the baseline system. This is quite understandable as the event sounds very similar to dishes. Just from this quick look the value of this tool can be seen as already potential areas of improvement can be seen. Viewing the F-Score and Error Rate does not give this kind of insight. A design decision was made to only show events in the slider that the system detected in the whole of the audio file. This is the reason why the cutlery event is not visible at all in the baseline system slider. By doing this space has been saved to allow for more classifiers to be compared at once. The user is also able to seek through the audio both by interacting with the waveform and through using the audio controls below the waveform. The user can also remove systems that have been added to the page, which concludes that all the requirements set out in the first section have been met.

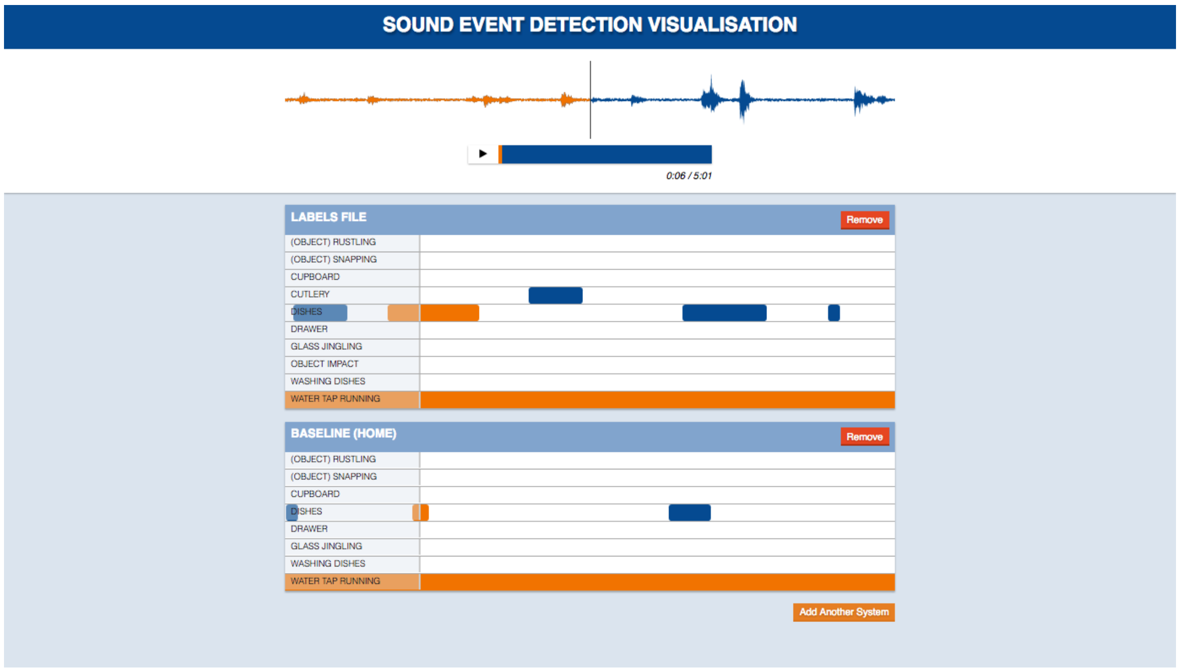


Figure 3.4: Main part of the application. The user has uploaded an audio file and the corresponding labels file. The baseline system has been selected and run on the audio file and the results are displayed below the labels file. The tool shows the system performing really well for the water tap running event and not so well for the other events.

3.4 Summary

In this chapter the visualisation tool that has been developed to aid the understanding of the performance of SED systems has been explored. The requirements were set out to begin with to establish what was needed in order to make the tool successful such as being able to easily upload audio files without any setup as well other basic functionality which other tools lacked such as the ability to seek through the audio. After this the design of the system was discussed and how the frontend application interacts with the backend system. Finally the chapter concluded with explaining how the final solution was implemented using React and Flask.

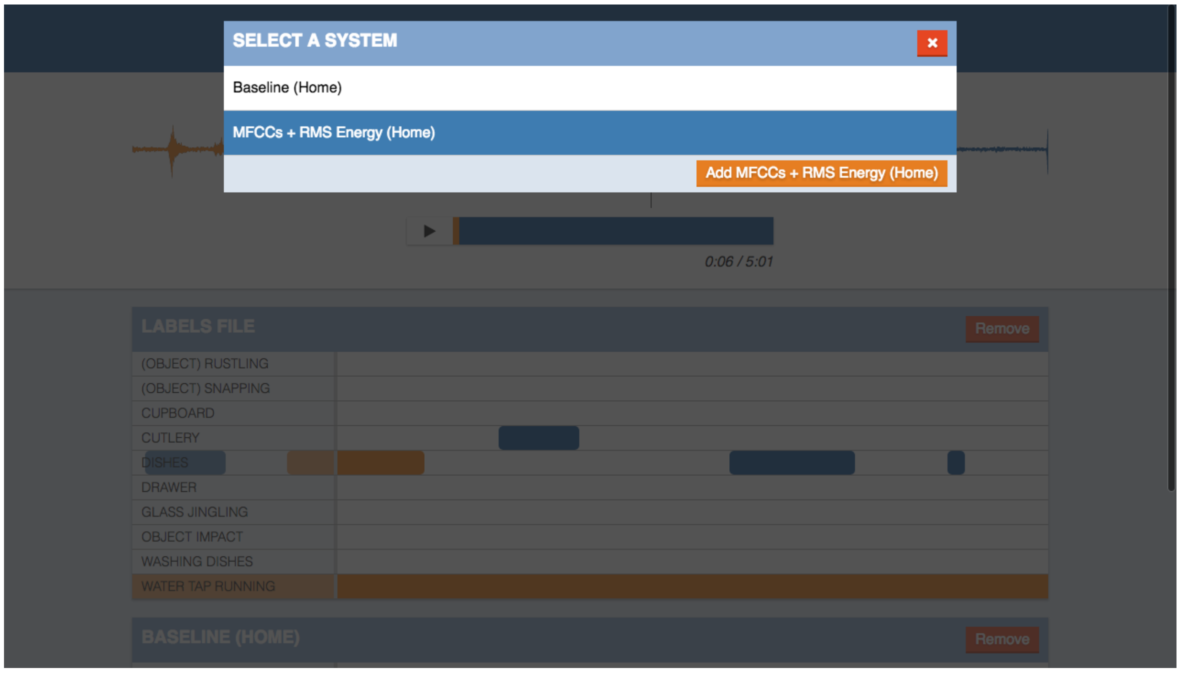


Figure 3.5: The menu for choosing SED systems to run. In this version only two systems are available. Once selected and confirmed the slider will be appended to the main page and the corresponding labels will be shown.

Chapter 4

The Baseline System

The chapter begins with describing the TUT dataset that was provided for the DCASE competition. The following chapters will build upon this dataset through augmenting the data to artificially grow the number of samples, but to start with, what is currently available is explored and the need for more data is emphasised. This chapter then describes the baseline system that was provided for the competition. The chapter goes on to detail how it was re-engineered to better fit future experiments. The following chapters will also build upon this baseline system using novel feature extraction techniques, because of this it is important to have a good understanding on the current level of performance being achieved by the baseline system. This is why the chapter concludes with further experiments to detail more specific performance achieved by the baseline system, than that was presented by the organisers. These experiments also set the foundations to how the future experiments will be carried out.

4.1 TUT Dataset

For the DCASE challenge the TUT dataset was provided by the organisers and will be used as the baseline dataset for this project. The recordings were captured at a sample rate of 44.1 kHz with a 24 bit resolution using a Roland Edirol R-09 wave recorder [9]. The database consists of 10 recordings inside of a home and 12 recordings in a residential area, each between 3 and 5 minutes long. Each of these recordings has a corresponding metafile which contains the events and when they occurred. In the training data, the events do not necessary occur in isolation.

The event labels were freely chosen by two research assistants. The most frequent labels chosen by these assistants were chosen to be the target sound events and similar sounds were grouped together. Due to using two lab assistants the resulting labels may be inconsistent as this is a very subjective task. It is also very difficult for humans to detect more than a few events at the same time. Unlike synthesised data, it is difficult to have a reliable ground truth.

The dataset is quite a small; therefore techniques discussed in chapter 2 will be used to extend the dataset such as using speed, pitch and vocal tract length perturbation.

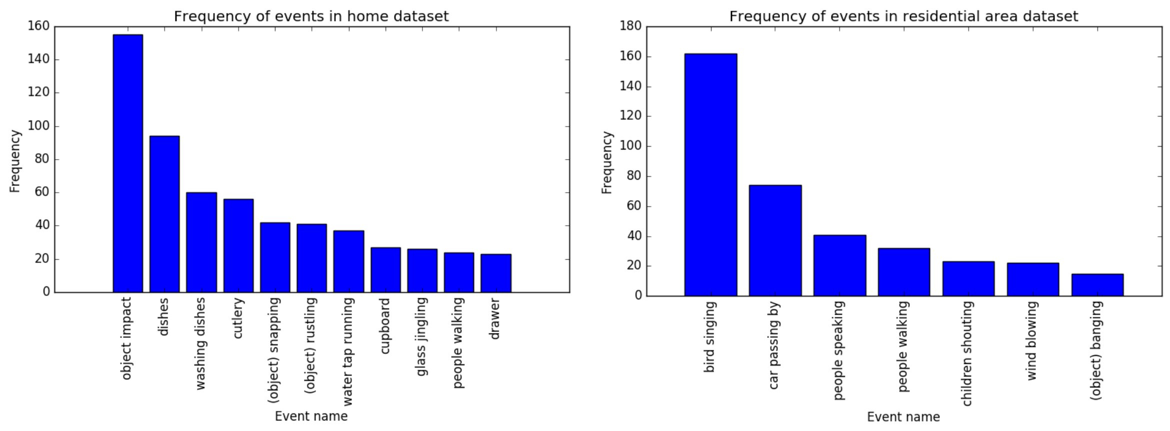


Figure 4.1: Frequency of the events in the TUT training dataset

The number of samples for each of the events in the dataset is not uniformly distributed. In figure 4.1 it is clear that some events are more prominent than others. For example in the home dataset "object impact" is by far the most common event, which is unsurprising as it is a very generic label compared to something specific like the sound of a drawer. It is a similar story with the residential area dataset with one event being more popular than the others by some margin.

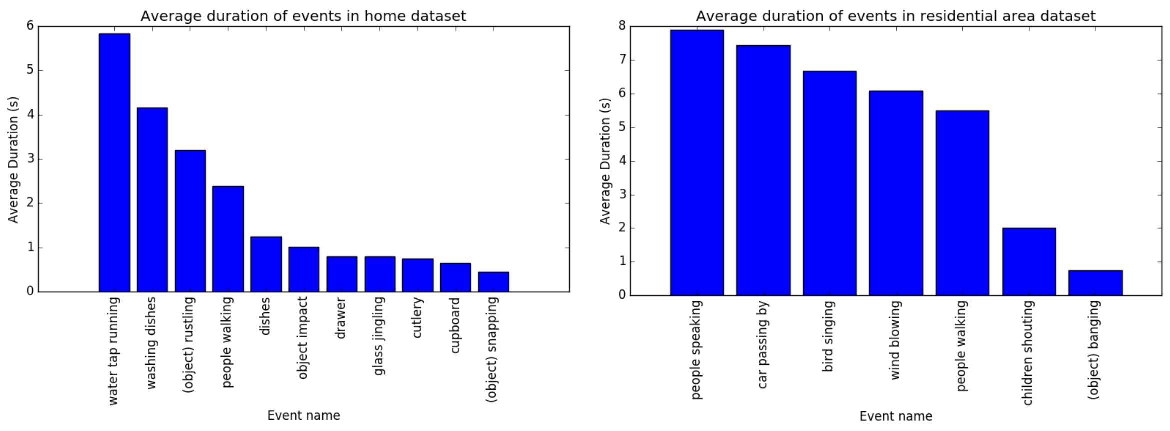


Figure 4.2: Duration of the events in the TUT training dataset

In figure 4.2 the average duration of all the events is shown. The chart clearly shows the disparity in the two categories. The chart on the left shows that the events in the home are far shorter on average compared to the residential area. The results of the challenge show that systems performed better on average for the residential area category. This is probably due to long standing events being easier to detect. If an event occurs for a long period of time the context of the previous frames can be used to detect the event. This is not the case for short plosive sounds found in the home, which can be easily missed. The sound of a cupboard is difficult to detect based on the previous frames. Another reason why systems performed better in the residential area category could be because there are fewer possible labels and less ambiguity. Most humans would struggle to tell the difference between the sound of 'dishes' and the sound of 'washing dishes'.

4.2 Feature Extraction

The only features used in the original baseline system were MFCCs. Using a Hamming window and a frame size of 40 ms and a hop size of 20 ms. The frames were calculated using a FFT with 2048 components. This results in a complex matrix S with the shape (1025, number of frames). 1025 is used as just under half of the components are thrown away after computing due to the symmetry caused by aliasing when sampling. This was implemented using the Python library Librosa, using the Short-time Fourier transform function provided by the library.

From the complex matrix S the power spectrum P of the signal is then calculated by $P = |S|^2$. A set of 40 mel filter banks are then created to give a matrix M with shape (40, 1025) where 40 is the number of filter banks and 1025 is the number of DFT components. The Librosa implementation that was used, normalises the traditional mel filter bank such that the triangular filter banks have an equal area of 1 instead of an equal height of 1. The difference is depicted in the image in figure 4.3.

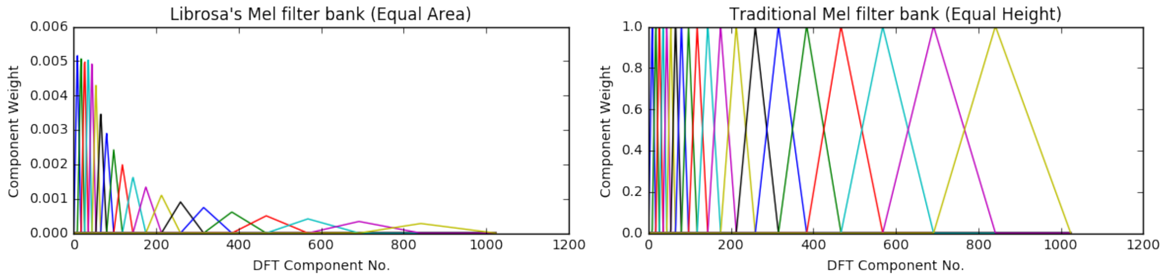


Figure 4.3: Comparison of the filter bank created by Librosa (Left) and the filter bank traditionally used (Right). The Librosa implementation is used in the baseline system implementation.

The amount of energy in each filterbank for the power spectrum is then calculated by simply multiplying the matrices MP resulting in the mel spectrum.

Suppose that an audio signal has some excitation to start the sound, which is then filtered by the cavity in which the sound was excited in, then the audio signal is simply the result of the convolution of the source and the filter. A convolution is simply a multiplication of the two signals (filter and source) in the frequency domain.

Because of this relationship and that the power spectrum is derived from the audio signal the next step is taking the log of the power spectrum which turns the multiplications in frequency space into additions in the log frequency space. In this space the high frequency components represent the noise in the signal including the source of the sound i.e lots of small additions. The low frequency components represent the shape of the filter.

The final step performed is taking the Discrete Cosine Transform of the signal to find the frequencies that make up the signal, this is similar to taking the DFT of the signal. In the baseline system's implementation 20 Cosine components are used here. The resulting signal is in the Cepstrum domain and the 20 coefficients for the Cosine components making up the signal are the desired Mel Frequency Ceptstrum Coefficients, often in speech processing only the first 13 of these components are used, however in the baseline system all 20 are used, as a sound event is signal is being analysed and not a speech signal. The lower components represent slower moving changes in the signal which describe more generic features of the sound and higher frequency components represent more specific features for that particular instance of the sound. The very first component represents the overall energy in the signal and is removed in the implementation as it is considered not very useful, this leaves 19 features.

As well as the 19 MFCCs the speed and acceleration (Δ, $\Delta\Delta$) of the features are also calculated. They are both calculated using a width of 9 frames. The Δ values are calculated using the following formula:

$$\Delta C_{i,t} = \frac{\sum_{n=-N}^{N} n(c_{t+n} - t_t - n)}{2\sum_{n=1}^{N} n^2}$$

Where $C_{i,t}$ is the value for coefficient i at time step t and N is the width. To calculate the double delta value, instead of feeding the coefficient into the formula, the delta value is fed in i.e, it is the derivative of the derivative. This was implemented in practise using the delta utility function provided by Librosa.

Even though the first MFCC is ignored the Δ and $\Delta\Delta$ of the first MFCC is used as this does carry useful information about the sound. This then results in a final feature vector with 59 features (19 + 20 + 20).

4.3 Normalisation

Features are not always on the same scale. This can cause issues when classifying samples, using humans as an example this will be explained. Let their features be height and head size. If the two measurements are recorded using centimetres, then the amount the values vary are clearly very different. Human head size will result in a few cms difference between people, opposed to human height which can vary by more than 30 cm. let human samples be plotted in a 2D space where the height and head size are the dimensions. If the distance between the points are measured using Euclidean distance (ℓ_2 norm). The resultant distance value will be more bias towards the change in height than the change in head size because humans height vary more in cm than what their head size varies in cm. This can cause issues because a change of 20 cms in head size is way more significant than a change in height by 20 cms, but would be considered identical in this space. To fix this issue the features are normalised by using the mean and variance in the dataset. This issue occurs for all features, including the ones used in SED, because of this the following formula is used to normalise the value such that they are on an even scale:

$$f' = \frac{(f - mean)}{std}$$

Where f is some unnormalised feature value, mean is the average value of the feature in the dataset and std is the standard deviation of the feature in the entire dataset.

4.4 Classifier

The chosen classifier of the baseline system was made using multiple Gaussian Mixture Models, in particular the Scikit learn implementation. The aim of this project is to explore how novel features and data augmentation can be used to improve performance, therefore the classifier will remain constant throughout the later experiments, however the classifier used will be explained here for completeness.

A Gaussian Mixture Model works by the assumption that the data comes from a set of N Gaussians. A weighted sum of the likelihood unseen data belongs to these Gaussians is used to determine if the sample belongs to the distribution. A separate Gaussian Mixure Model was created for each sound event label in the dataset. For the baseline system 16 Gaussian components were chosen for every sound event label in the dataset. This could probably be further fine tuned to use a different amount of components for each sound event type, as some sound events have more variety than others, however this is out of the scope of the project.

Fitting the data to the distributions is an iterative process. The set of Gaussians are randomly initialised in the feature space. The points are then adjusted using k-means clustering. K-means is solved in an iterative process using Lloyds algorithm [41]. This is achieved by iteratively updating the component centres to the centre of the closest points to them. Closest points are defined in terms of the Euclidean distance ($\ell_2$norm). After the points have been initialised to a stable set of means, the main iterative process starts. This starts by first finding the likelihoods that the data samples belong to each of the Gaussian components.

The weights, means and covariance values of the Gaussian components are then tweaked to improve these likelihoods, this process is then repeated further improving the likelihoods. This algorithm is known as Expectation Maximization. In the baseline system this iterative process is repeated 40 times.

Further parameters used in the baseline system includes using a diagonal covariance matrix and enforcing a minimum covariance value of 0.001 to prevent overfitting [42].

To train the classifier a GMM was fitted for each of the sound events types, these are the positive models. As well as the positive models, negative models were also trained. A negative model is trained for each of the sound event types, these models fit all the samples that are not that event.

Finally to determine the class of an unseen sample the ratio of the likelihood of the sample belonging to the positive model and the likelihood of the sample belonging to the negative model is calculated.

$$\text{likelihood ratio} = \frac{\text{positive likelihood}}{\text{negative likelihood}}$$

Samples are not independent, they depend on the previous samples seen. Because of this a mean smoothing window using the context of 1 second worth of samples is used to smooth the likelihoods. The final likelihoods are then compared against a threshold value of 160, if the likelihood is greater than this then that sample is classified as belonging to the class. A sample can have a high likelihood of belonging to multiple classes, this accounts for the polyphonic nature of the audio.

4.5 Post-Processing

Given that the samples are now classified, further post processing can clear up obvious mistakes. In the baseline system errors are corrected by throwing away events that occur for less than 100 ms and same event classes that occur within 100 ms of each other are merged together.

4.6 Re-engineering the DCASE system

The Baseline system provided by the organisers was very rigid and difficult to extend in a maintainable way. To allow for further experiments to be carried out, the system was re-engineered with the aim of more maintainable code by breaking the code into components. The original system was built by the organisers was very specific to the solution provided. By recreating the system in a maintainable way, the experiments can more easily be reproduced and further extended by other researchers. For example the classification part of the system is not a focus point of this project, however if someone wanted to continue this work with a different classification system that should be easily achievable. It was also important to maintain the integrity of the original system's performance level, this was achieved and results of those tests are shown in the appendix. The main component of the new architecture is the SoundEventDetector class. This class joins together the different aspects of a SED system discussed in the literature review, such as feature extraction, training and post processing. The full relationship of the system is depicted in the class diagram in figure 4.4. As shown in the diagram, the features being extracted by the system is are attribute of the SoundEventDetector class. These features are looped through, the features' extract method is called and the resulting feature vectors are concatenated together. This architecture allows for multiple combinations of features to be experimented with, in a maintainable way, unlike in the previous architecture where a series of conditionals in an extract function would have been required. In the class diagram only one feature has been shown, the Centroid. However the other feature classes are just as simple, and they all inherit from the same Feature base class. To run experiments the configurations of the experiments were defined using a specified *yaml* file. The file contains information such as the configuration of the classifier e.g the number of components, the features to extract and the cross fold validation information. When the results are saved this file is saved a long with the results, to allow for experiments to be easily reproduced later on.

The following features: Centroid, RMSE, Rolloff and Zero Crossing Rate were implemented using the Python library Librosa [43]. The spectral features Centroid and Rolloff were framed using a frame size of 40 ms and hop size of 20 ms using 2048 DFT components. The components are then reduced down to 1025. The windowing function used was a Hamming window. For the time domain features, the same frame and hop size was used as the spectral features.

The remaining features of Flux, Sharpness, Slope and Spread the features were implemented using the Python library Yaafe; parameters to frame the signal were the same as for other features.

4.7 Experimental Setup

The dataset provided by the baseline system is quite small, because of this it is easy to overfit. To ensure that the experimental scores are a fair reflection on how well the system is actually performing, the same cross fold validation setup will be used that was provided by the DCASE organisers. The crossfold validation works in 4 folds. For a fold the data is split into 2 sets, one containing around 75% of the data and another containing the remaining data, these sets of data are disjoint. The data is then trained on the section with around 75% of the data and tested on the section with around 25%. For the next fold a different split of

Components	**Substitutions**	**Insertions**	**Deletions**	**Events**	**ER**
4	246	330	1109	1572	1.07
8	206	311	1166	1572	1.07
16	141	123	1269	1572	0.98
32	119	100	1334	1572	0.99

Table 4.1: The results for the baseline system in the home scene using a threshold of 150 showing the error rate and the parts that make up the error rate. The error rate reduced with more components added. From the looking at the plots in figure 4.6, it can be seen that this was the general trend.

data is used. After all the folds are completed the total True Positives (TP), False Positive (FP), True Negatives (TN), False Negatives (FN) are calculated and then the corresponding F-Scores and Error Rates are calculated. This is known as the micro average and is a better reflection of the overall scored compared to averaging the F-Score and Error Rates of each of the folds when there is an in-balance in the amount of labels between folds. The same fold splits are used that were used in the baseline system as the organisers of the DCASE challenge chose splits based on where and when the audio files were recorded and it was recommended by the organisers to keep the same split.

4.8 Results and Discussion

The first set of experiments explore the performance of the baseline system in more detail. Recall that the classification system relies on a threshold value to determine whether or not the likelihood is high enough for the sample to be considered part of that class. If this threshold is varied the F-Score and the Error Rate can be plotted against the threshold creating a curve, this will allow the more general performance of the system to be understood. Threshold values between 0 and 300 are used with a resolution of 1 experiment per 5 threshold units. The experiments have also been repeated for different amounts of components to find the significance of the number of components.

The graphs in figure 4.5 and 4.6 clearly show the disparity in the two scenes. The system consistently performs a lot better in the Residential Area compared to the Home scene. The main metric that the following experiments with feature extraction and augmentation will try to optimise is the F-Score system however a low Error Rate ideally bellow 1 will also be desired. The Error Rate on its own cannot be considered a good metric because this is easy to optimise by being very cautious with decisions, which does not necessarily make for a good SED system. Not detecting any events gives an error rate of 1. A system that is able to detect lots of events, which however gets lots of FP is probably a better system than one that does absolutely nothing. This makes the error rate quite difficult to interpret.

Using different numbers of components gave interesting results. In figure 4.6 the higher

Components	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F-Score**
4	217	576	1355	0.27	0.14	0.18
8	200	517	1372	0.28	0.13	0.17
16	162	264	1410	0.38	0.10	0.16
32	119	219	1453	0.35	0.08	0.12

Table 4.2: The results for the baseline system in the home scene using a threshold of 150 showing the F-Score and the components that make the calculation. As more components were added the F-Score dropped, this is because the decline in the recall was more significant in the calculation than that of the precision because of how low the recall was already.

amount of components gave the better error rate and in figure 4.5 the lower amount of components gave the better F-Score. To better understand the reason for this in tables 4.1 and 4.2 a threshold of 150 for the home scene has been used. The parts that make up the F-Score and Error Rate have been shown to try get a better understanding for the cause for the differences in score. From looking at the tables, the error rate shows how poorly a system is doing and does not really give much indication of how well a system is doing, F-Score however does do a better job at showing how well a system is doing. For example the 4 component mixture found the most events but also made the most mistakes. The F-Score did not punish the 4 component classifier as much as the error score did. This is because the F-Score focuses on trying to get a good balance between precision and recall. Looking at the 32 components the precision increased a lot, 29.6% relatively to the 4 component GMM. However, looking at the recall this difference is dwarfed, the recall almost halves. This relative difference is not really accounted for with the error rate. The sound events in scenes are very sparse which means it is far easier to make a mistake than it is to get an event correct. In order for a system to improve at detecting events, it will make more mistakes. This does not mean the system is getting worse. The GMMs with the lower number of components managed to retrieve more correct events because the Gaussian's formed are wider and more general than that of a GMM with more components. Having wider and more general Gaussians means more events will have a likelihood to belong to the Gaussians of the mixture, compared to a GMM with lots of highly specific Gaussian. With highly specific Gaussians less events will have a high likelihood to belonging to these Gaussians, however when they do the probability of them actually being correct will be greater than that of the less specific ones. At this point it is difficult to tell the optimal number of Gaussians for this dataset however 8 or 16 components seem likely.

Finally to give a visual understanding of how well the system performed, screenshots of the visualisation tool are displayed in figure 4.7 and 4.8. This instance of the SED system is using the 16 components chosen by the organisers originally. The screenshot on the left shows the system performing quite well. There are a few false positives, such as detecting object impacts and the sound of cutlery. However other events such as one object impact, people walking and the sound of water tap running were quite well detected. On the other hand on the right shows the system performing not so well. An (Object) Snapping event has been missed as well and 3 false positives.

4.9 Summary

In this chapter the baseline system that was provided by the competition organisers has been dissected. The implementation of the MFCC feature extraction used in the baseline system has been explained as well as the classification system used. A discussion of how the system was re-engineered to allow for an extensible framework to be created. Following this the performance of the baseline system has been analysed including the two metrics that were chosen for the competition. It was decided that the F-Score showed a better representation of the overall performance of the system. However the error rate should not be neglected completely, as a high error rate can indicate a poor performing system, even if the F-score is good. It will be desirable to have an error rate value below 1.

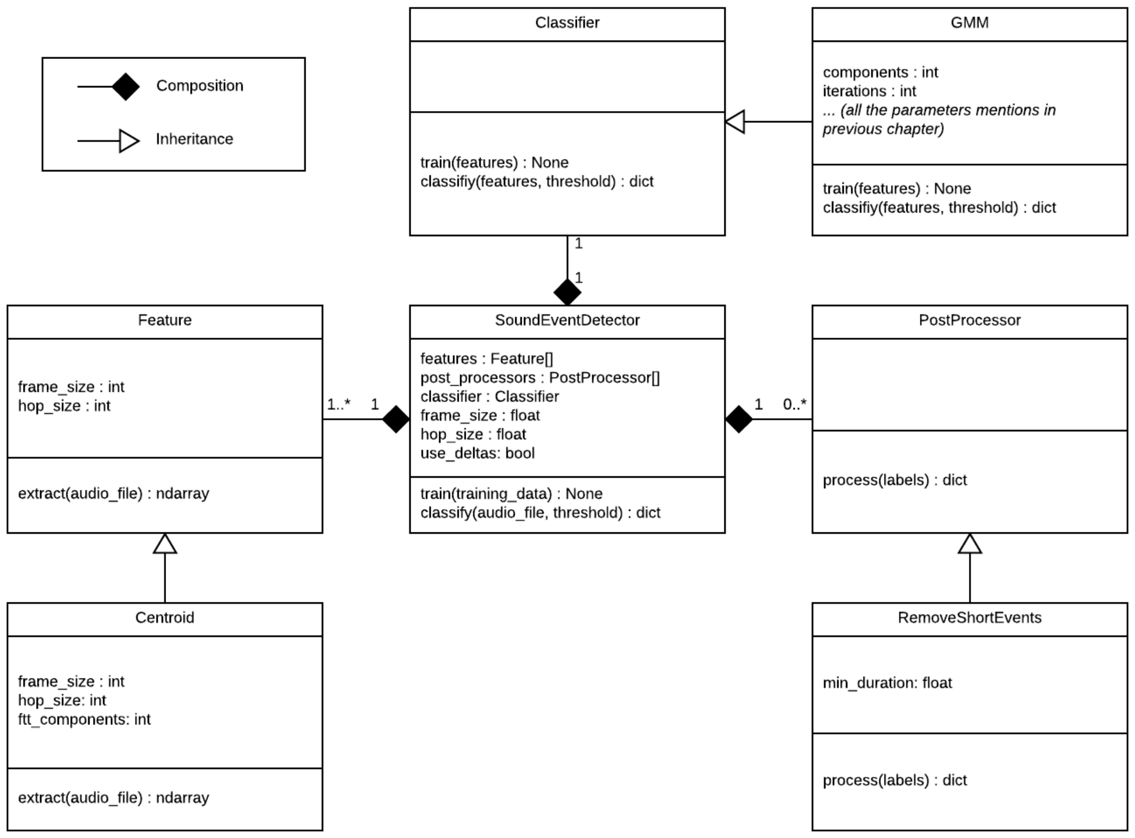


Figure 4.4: The class diagram demonstrates how the components in the system are connected. The core of the framework is the SoundEventDetector class, the rest of the application are modular components. A user of the framework can supply 1 or more Feature classes which are fed audio files. The user supplies a Classifier, this is given a dictionary of matrices, where the key is the corresponding audio file. The rows of the matrices are features and the columns are the frames. The Classifier will then return a dictionary with the key being the audio file and the value being a matrix of labels at each frame. The user can also optionally provide PostProcessors to tidy up mistakes at the end. This framework allows future work to be carried out such as replacing the classifier or experimenting with different post-processing techniques.

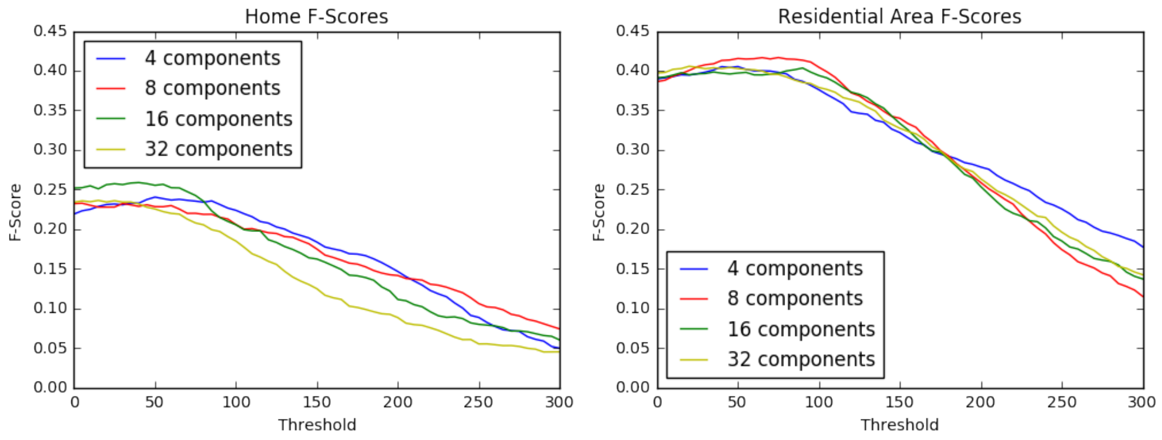


Figure 4.5: Results from running the baseline using multiple components. A threshold value between 0 and 300 has been used with a resolution of 1 experiment per 5 threshold units. Using more components lead to a slightly decreased F-Score.

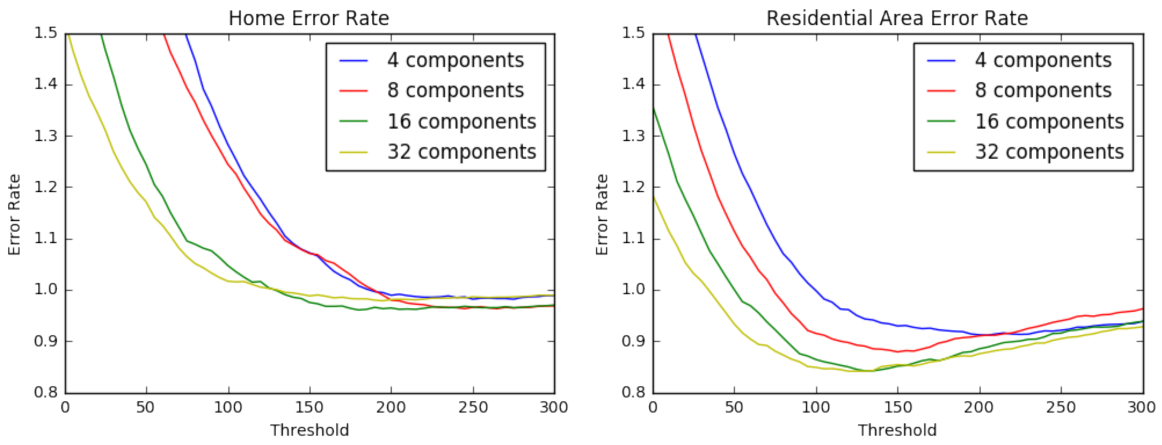


Figure 4.6: Graphs showing the general performance of the baseline system in the two scenes through looking at the error rate. The experiments have been rerun several times using different number of components. Generally the higher number of components leads to a better error rate.

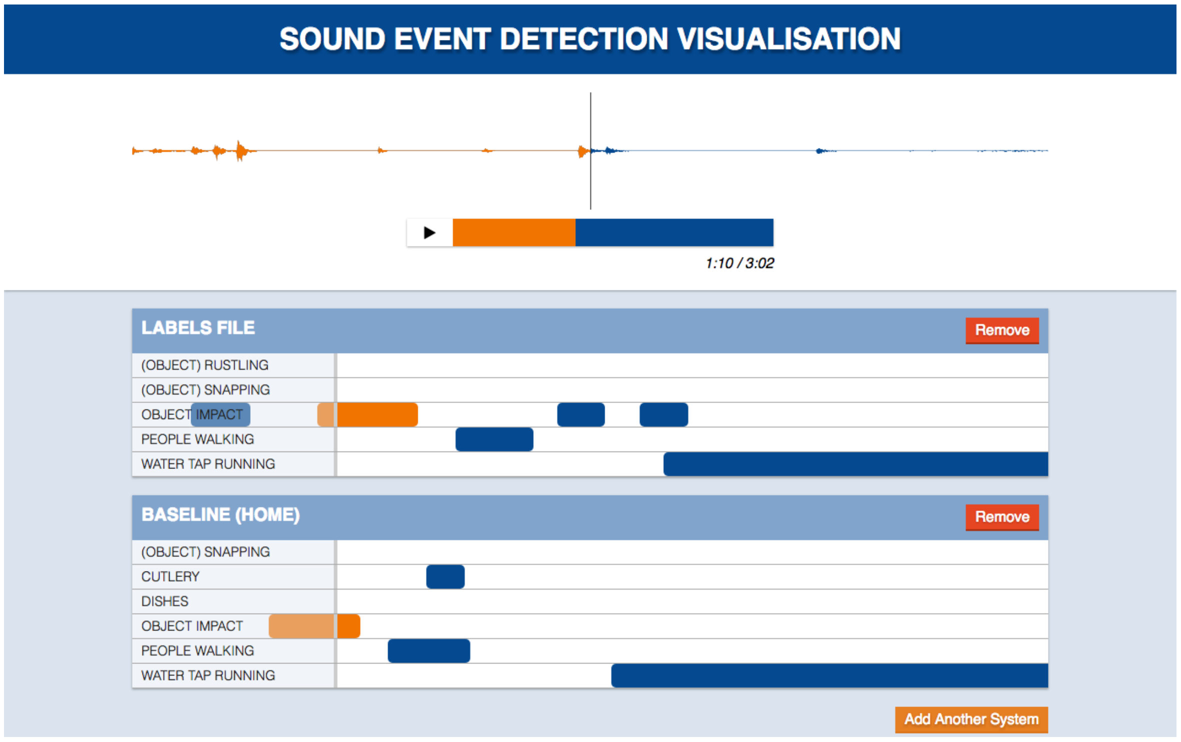


Figure 4.7: Screenshot the baseline system detecting events well.

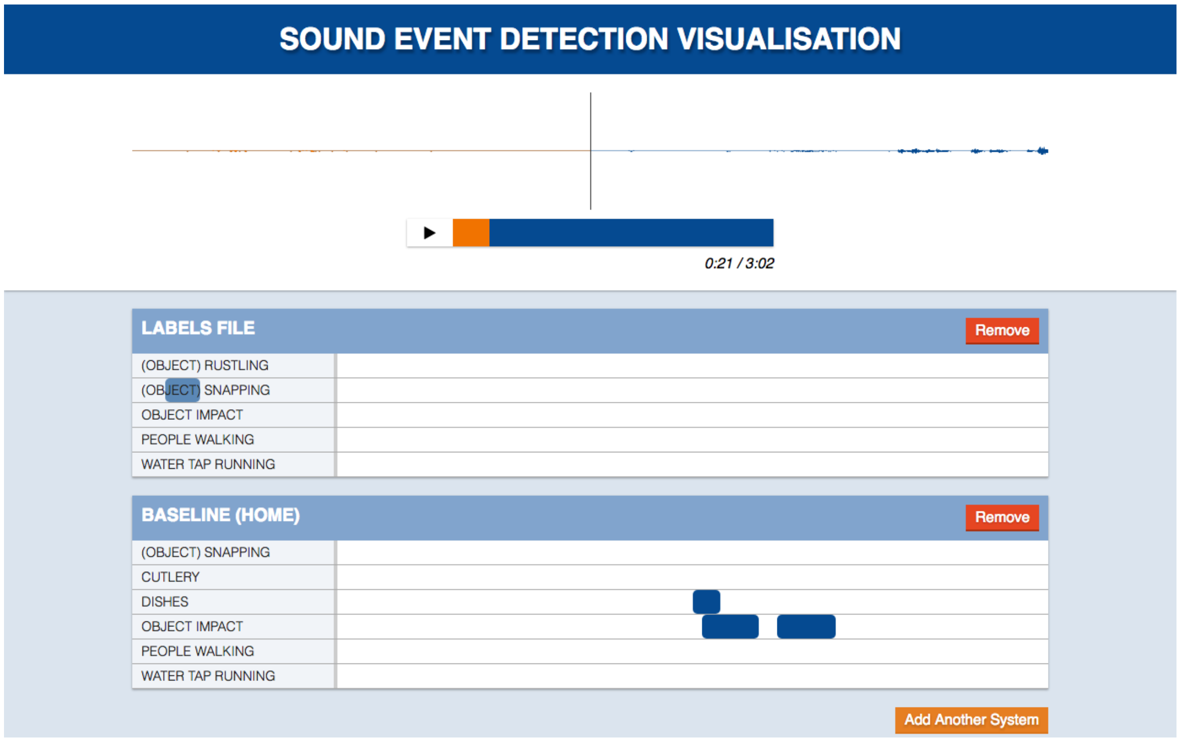


Figure 4.8: Screenshot the baseline system detecting events poorly.

Chapter 5

Acoustic Features for Sound Event Detection

5.1 Introduction

Using the baseline system as a starting point, novel feature extraction techniques not usually used in Sound Event Detection will be explored. Traditionally in SED the choice of features are usually just MFCCs and not much more than this. Considering the audio source is a scene and not a speech signal this is an interesting choice to be the state of the art. The features to be experimented with are the following:

1. Centroid
2. Root-Mean-Square Energy (RMSE)
3. Flux
4. Rolloff
5. Sharpness
6. Slope
7. Spread
8. Zero Crossing Rate

The majority of these features are spectral features apart from RMSE and Zero Crossing Rate, which are in the time domain.

5.2 Experiment Setup

The 8 features stated in the the introduction will be the main focus of the experiments. The spectral features are extracted from the exactly same spectrum. That is they all use the resulting spectrum from framing the time domain by signal using 40 ms frames with a hop length of 20 ms, a Hamming window and using 2048 DFT components. The time domain features were extracted from the 40 ms second frames with the hop length of 20 ms, this means that all the resulting frames for the features line up. Alternatively, features that require a longer time frame could have been used, however they would have not lined up with the other features, which would have meant a different approach would have been required. A separate system would have needed to be trained for each of the different feature types and then the likelihoods would be have been considered together. This is a potential direction for further work.

The initial experiments will extend the baseline system by concatenating each of the new features onto the baseline system's MFCC features. Along with the features the Δ and $\Delta\Delta$ of the new features will also be concatenated onto the feature vector. Recall in chapter 4 that the features in the system are normalised. The additional features will again be normalised using the same procedure, this is to put all features onto the same scale . In the previous chapter, it was discussed that the SED system relies on a threshold value to determine whether the likelihood of an event is high enough to mark a frame as containing that event; because of this the threshold, like in the previous chapter, will be tweaked and metrics will be recalculated creating a more general measure of performance. Again a threshold from 0 to 300 is used with a resolution of 1 experiment per 5 threshold units. To ensure that the results are a fair reflection the same cross fold validation setup provided for the baseline system will be used.

5.3 Extending the Baseline Features

The performance benefit of using additional features along with the MFCCs will be evaluated in the following experiments. Firstly a separate experiment will be carried out for each new feature, which involves simply concatenating the feature onto the MFCC vector. The thresholds used for these experiments will again be between 0 and 300 with a resolution of 1 experiment per 5 threshold units. This is because the thresholds below 0 give very high Error Rates and therefore do not give a very fitting solution even though the F-Score may be good. These experiments have also been repeated using multiple components to see how the number of components effect the performance of different features.

The results of the experiments are shown in figure 5.1, these plots show the average values for each of the components using the different feature types. The plots at the top of the figure show the F-Scores and the plots and the bottom of the figure show the Error Rate

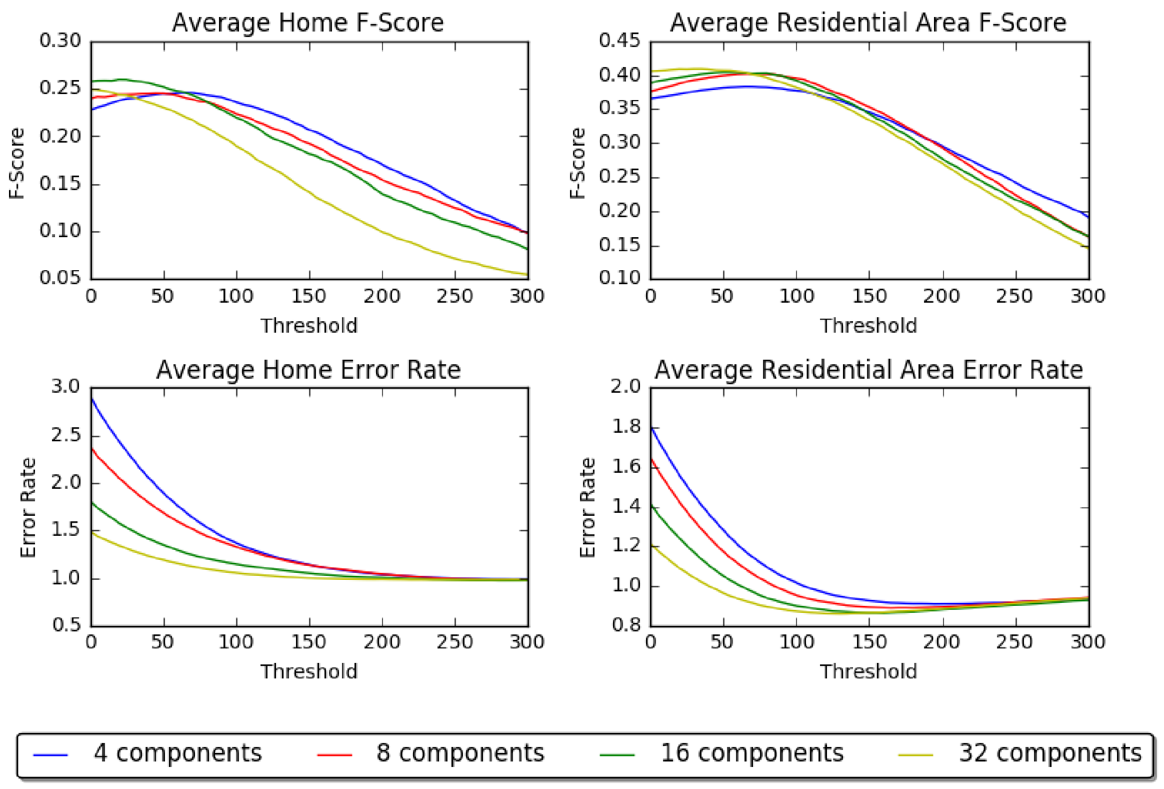


Figure 5.1: Results from using multiple different features averaged together, showing the performance different between different components.

scores. Plots of the individual feature results are shown in the appendix. The results back up the previous observations that in general, a higher amount of components produce worse F-Scores but produce better Error Rates. A good SED system should have a high F-Score but also ideally have an Error Rate below 1. An Error Rate above 1 means that the system is getting more events wrong than it is right. The graphs also highlight the lack of correlation between the two metrics used in the competition, this makes it difficult to judge a definitive better solution. Systems tended to perform best in terms of F-Score when the threshold was around 50, however at this point Error Rates were often above 1, and sometimes by a large margin. Likewise when the Error Rate is at its lowest the F-Score starts to decline.

For the following experiments, 16 Gaussian components will be used. The reason for 16 components is based on inspecting the plots in figures 5.1. The Error Rates of 16 and 32 components reach a respectable value at a far lower threshold. When taken into consideration with F-Scores, having a low Error Rate at a low threshold is a good property as F-Score tends to decrease as the threshold is increased. The F-Score of 32 components tended to be far lower than the rest of the component choices. However, 16 components performed with an F-Score close to that of the lower components and sometimes better. For this reason using 16 components gave the best balance between a respectable Error Rate and a good F-Score.

Now that the best number of Gaussian components to use for this SED system for the dataset provided has been decided to be 16. The F-Scores of 16 components from the previous experiments have been shown together in figure 5.2. This shows the relative difference between the different feature choices. The plots clearly shows that for the Home scene all the different feature types improved upon the F-Score. For the Residential Area scene improvement was a bit more difficult due to the system already performing well for this scene. However even with this being the case a slight improvement was still made by the majority of the features. Only features such as Centroid and Spread had a negative impact on the performance. This is when looking at the general performance of the F-Score in isolation and not the Error Rate.

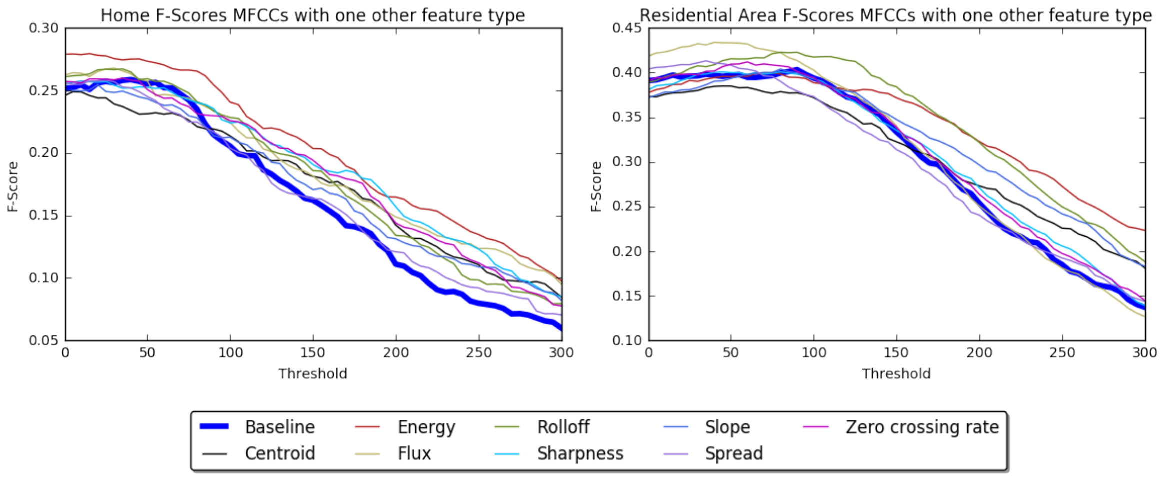


Figure 5.2: F-Score for different features over the two scenes. All the features improve upon the baseline system in the Home scene (left). For the Residential Area scene most of the features saw an improvement.

Now looking at the Error Rate of the systems in figure 5.3, the additional features struggled to get a low Error Rate in the Home scene. Most of the features needed a threshold of 250 to get a Error Rate below 1. By this point a lot of the F-Scores are lower than that achieved by the baseline system when it got to an Error Rate of below 1 at around a threshold of 125 in the Home scene. In contrast, for the Residential Area scene all the features achieved good Error Rates along with their improved F-Scores. Very interestingly, Rolloff managed to improve upon the impressive baseline system performance for Error Rate.

A lot of the features improved the F-Score of the SED system. However in doing so this led to a higher Error Rate. In table 5.1 the average scores over the two scenes are shown. Looking at the table, if the maximum F-Score is being used to judge the performance, then the best features to extend the system are Flux, Rolloff, Energy and Spread. However if the Error Rate is also being considered, this is not the case. The F-scores shown in the table correspond to the F-scores when the error rate is 1. This is a better judgement of the actual performance difference between the different features. The table shows quite a bit of difference between features that produce high F-Scores and features that produce a high F-Score when the Error Rate is 1. For example Root-Mean-Square Energy produces a maximum F-Score of 33.5%, which is the third highest F-Score. However when the Error rate is down to 1 this score has reduced to 30.0%, making it the fifth highest F-Score compared to the other F-Scores of features when their Error Rate is 1. A lot of the features produce a better maximum F-Score than the baseline system; however with Error Rate taken into consideration, only Flux and Rolloff managed an improvement. Flux managed a 3% relative F-Score improvement over the baseline system when the Error Rates are 1. Flux gives a 6.7% relative improvement over the baseline when using the maximum F-Scores. Comparing the maximum F-Score for Flux to the reported F-Score for the baseline system by the competition organisers, a massive 47.6% relative F-Score improvement has been made over the 23.7% F-Score reported.

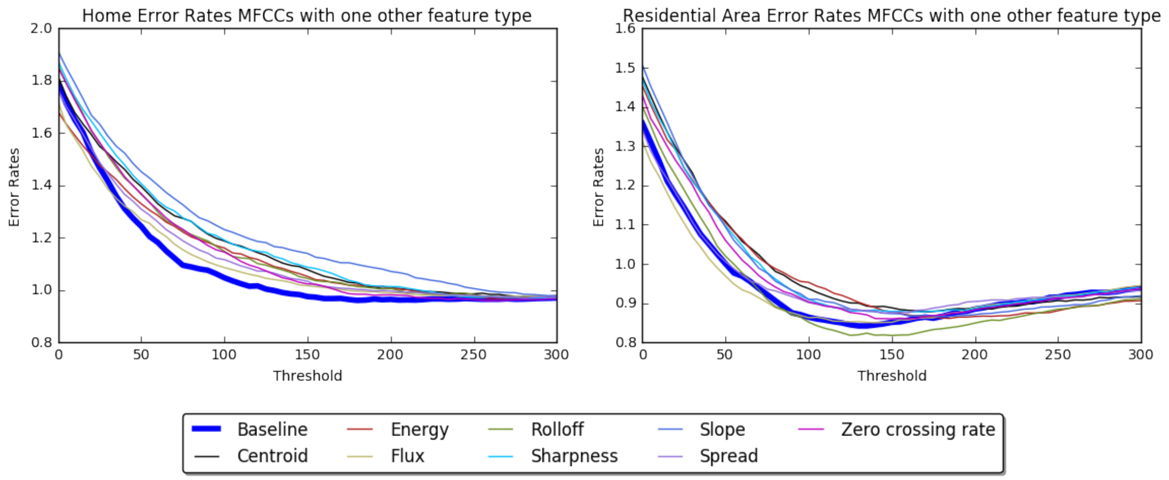


Figure 5.3: Error Rate of the two scenes, when features are concatenated onto the baseline systems' MFCCs. Most of the features failed to improve upon the Error Rate of the baseline system in both of the scenes. Rolloff managed a slight improvement over the baseline system in the Residential Area scene.

Figure 5.4 shows the visualisation tool demonstrating the difference in the performance of different features. The baseline system is being compared against the system with the additional Flux feature and the system with the additional RMS Energy feature. The Flux feature performed well, as shown in table 5.1. The RMS Energy also performed well with a high F-Score; however, it had a high Error Rate in general, letting the feature down. To understand the reason for these scores the visualisation tool will be used. The image on the left shows the Flux feature improving upon the Baseline system by removing some False Positives that were previously detected. The Energy feature also removed some of these False Positives; however, the False Positives of the "dishes" event were extended more, showing where the Energy feature is performing not so well. The images in 5.4 show the visualisation tool with the labels file loaded at the top of the page, followed by the results of the labels of the SED systems. In the right image, the image emphasises this difference

Features	Threshold	F-Score (%)	Error Rate	Max F-Score (%)
Flux	85	32.7	1.0	35.0
Rolloff	100	32.3	1.0	33.7
Baseline	**80**	**31.7**	**1.0**	**32.8**
Zero Crossing Rate	110	30.4	1.0	32.2
Energy	125	30.0	1.0	33.5
Spread	100	28.8	1.0	33.4
Sharpness	130	28.3	1.0	32.7
Centroid	130	26.8	1.0	31.2
Slope	155	25.7	1.0	31.9

Table 5.1: Results of the different features concatenated onto the MFCCs averaging the results from the two scenes. The results are normalised such that the F-Score are shown for the threshold which results in an Error Rate of 1.0

even more. Many False Positives were introduced; the energy levels of the "Drawer" events must be very similar to the Object-impact events. The Energy feature did however pick up the Object-impact events much better compared to the other features. This means that the energy feature may be a poor choice on its own with MFCCs, whereas with other additional features, the feature may become more beneficial. Flux, in the right image, slightly improved upon the baseline. This was the general result from playing all the audio files. This is a good property for an additional feature; it is nice to slowly improve the performance opposed to a more drastic change, like that what was introduced by the Energy feature.

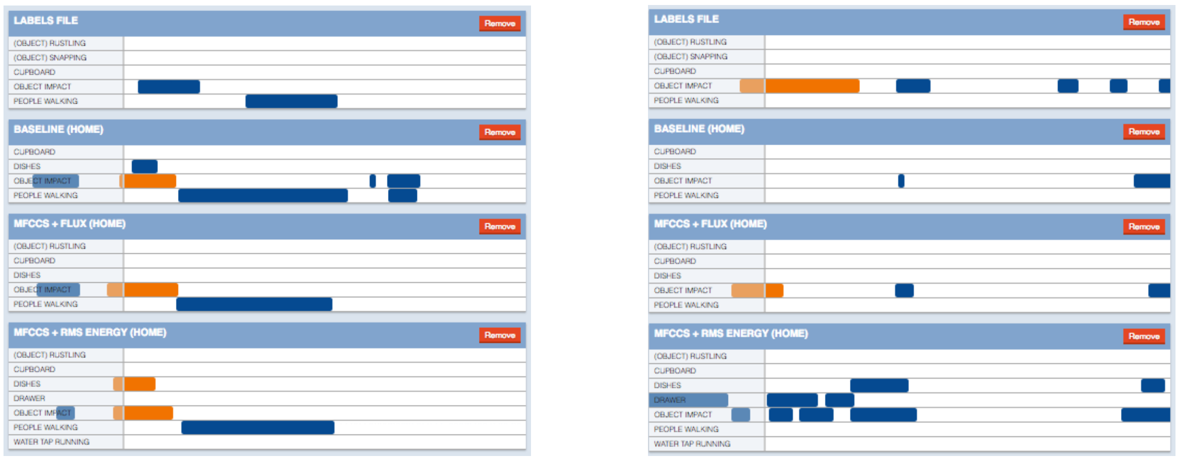


Figure 5.4: SED Visualisation tool demonstrating the the difference in performance between different feature types. The additional features corrected some of the mistakes made by the baseline. However they also introduced some of their own mistakes. The Energy feature added more errors compared to the Flux feature.

5.4 Combining Novel Features

Finally to conclude this chapter, experiments will further attempt to improve the baseline system through combining the features used in the previous section. Using the results in table 5.1 the features which performed the best will be combined. The best features in this context are defined in terms of their maximum F-Score. Four different mixtures of features will be initially used in the following experiments; these are:

1. **Mixture 1:** MFCCs + Flux
2. **Mixture 2:** MFCCs + Flux + Rolloff
3. **Mixture 3:** MFCCs + Flux + Rolloff + Energy
4. **Mixture 4:** MFCCs + Flux + Rolloff + Energy + Spread

Again, 16 components will be used, using the same cross fold validation setup. In addition to this the dimensionality of the data will be reduced using Principal Component Analysis (PCA) down to 59 features after concatenation. This is to make the experiment fairer, as with more features there are more parameters that need to be learnt for the GMM. The PCA implementation was created using Numpy's *linalg.eigh* function to find the eigenvectors from the centered covariance matrix; the data is projected onto the first 59 eigenvectors with the highest eigenvalue.

The results of these experiments are shown in figure 5.5. The plots show the mean values of the F-Scores and Error Rate over the two scenes. The results show that with PCA the additional features only make the system perform worse, both in respect to Error Rate and F-Score. The negative effect of reduction in dimensionality outweighs the benefit of the reduced amount of parameters. This is because by reducing the dimensionality, important information has been lost. Further experiments have been shown in figure 5.6 which do not have the PCA feature reduction. The graph shows that with more features, the SED system produces a more consistently better F-Score. Combining the 4 best features (MFCCs, Flux, Rolloff and Energy) produced a curve which consistently had the highest F-Score. However, looking at the Error Rate graph, the Error Rate again gets worse with the better F-Scores. The Error Rate is a difficult metric to interpret when the Error Rate is above 1. A system that detects events but gets a lot of False Positives is better than a system that detects nothing at all. Using the best F-Score as a measure, the optimal feature choice would be MFCCs with Flux as this produced the highest F-Score of the configurations. However, using the features in mixture 4 would, in theory, result in a more consistent system when using a different dataset. This is because the system is less reliant on the ideal threshold; an ideal trait of a SED system. A system that consistently performs well is probably a better system than one that performs really well some of the time.

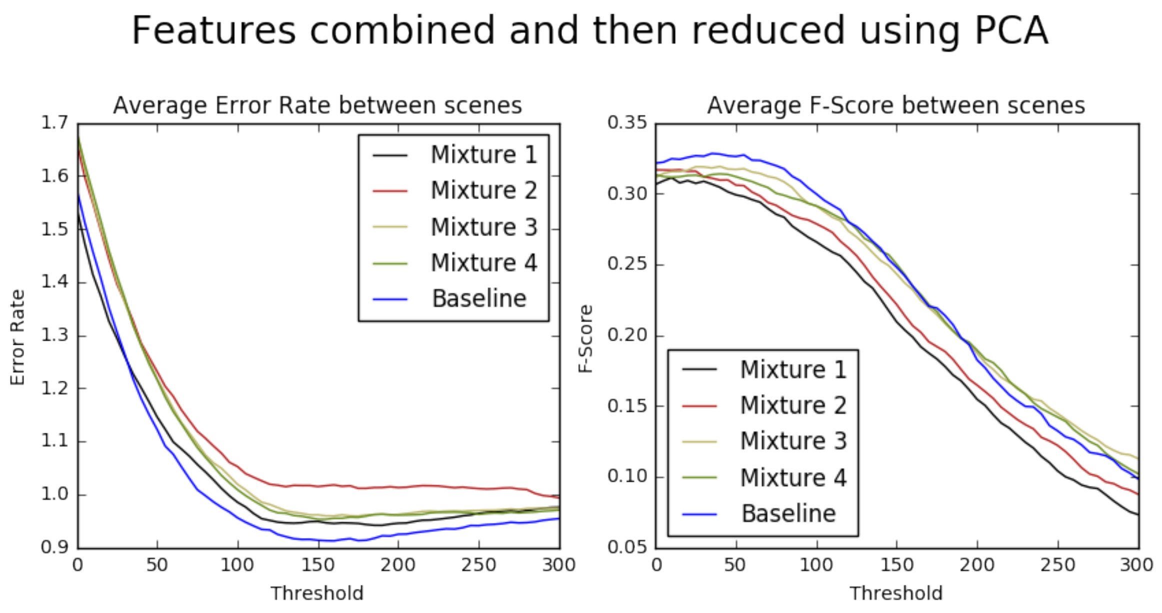


Figure 5.5: Results from combining features through concatenating them together into one larger feature vector. The feature vector is then reduced using PCA down to 59 features, which means all the experiments involve learning the same number of parameters.

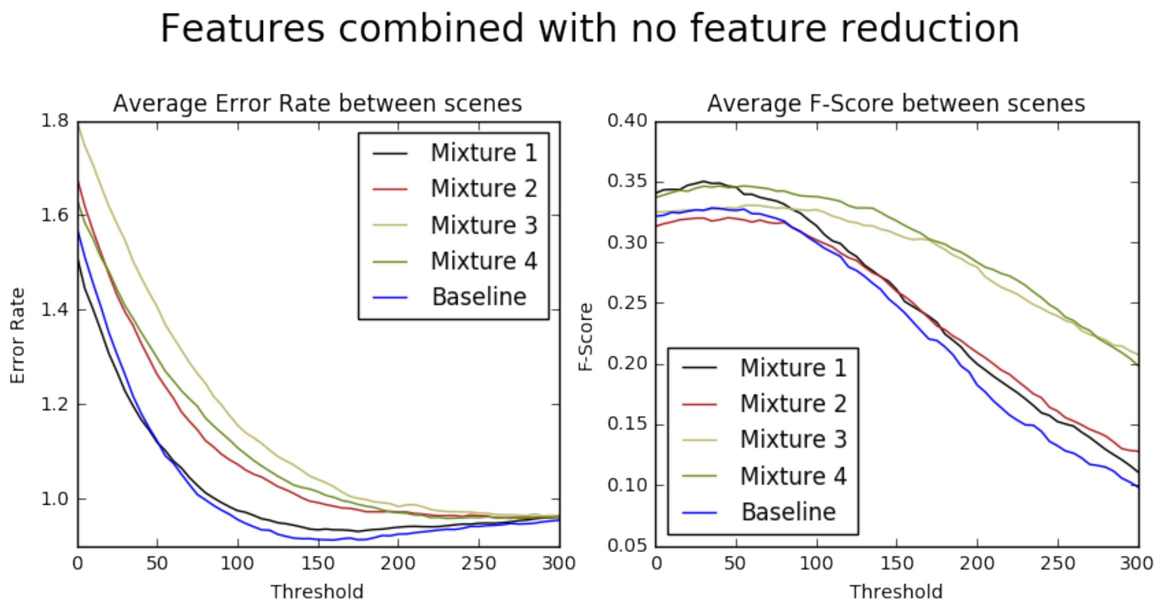


Figure 5.6: Results from combining features through concatenating them together into one larger feature vector with no feature reduction.

5.5 Summary

In this chapter various novel features have been explored. Mainly features in the frequency domain were used, with some from the time domain also being explored. The performance benefit of the one additional feature was experimented with, the results between the two scenes were quite different. An improvement was made when each of the additional features were used in the Home scene. The Residential Area proved to be more difficult to improve. Through repeating the experiments multiple times with different amounts of components, it was evident that using 16 Gaussian components gave the optimal results. This backs up the decision of 16 Gaussian which the organisers chose without justification.

Throughout the experiments a common theme between the Error Rate and the F-Score was the lack of correlation. The optimal F-Score was never the optimal Error rate. This lack of correlation has made the analysis of best system difficult.

Through concatenating one additional feature to the baseline feature vector, Flux, Rolloff and Energy showed the most improved F-Score at the best threshold. The features were then combined together to create an even larger feature vector. This combination resulted in a consistently well performing system when no PCA was used. However the maximum F-Score did decrease slightly. Using the knowledge gained through these experiments, the best performing features will be combined with the best performing data augmentation techniques in the next chapter.

Chapter 6

Data Augmentation for Sound Event Detection

6.1 Introduction

Given that the TUT dataset is a small dataset it is difficult to build SED systems which are robust enough to deal with the variety in the sound events. To counter this, the following chapter will explore techniques to artificially increase the dataset in a natural way. This will be achieved through augmenting the data available in the TUT dataset. This is an analogous to rotating images slightly to generate more data in image recognition. Like with images it is important that the data is not over augmented which would lead to unrealistic data. To augment the data the techniques speed and pitch perturbation which were used by some entrants in the DCASE challenge will be explored and a further technique called Vocal Tract Length Perturbation (VTLP) will be explored, VTLP is a technique used in speech recognition to generate utterances that sound like they come from different speakers by manipulating the length of their vocal tract. Sound events clearly do not have a vocal tract, however generating new sound events through this technique will, for example make the size of table change when augmenting an object impact on a table.

6.2 Implementation and Limitations

New data was generated by augmenting the whole audio file and therefore the whole scene, this is a rather naive approach. A better approach but far more difficult approach would be to take the sound events out of the scene, augment them and then place them back into the scene. This approach involves many difficult steps such as first separating the sources in the audio file to take out just the audio event with no noise or leakage from other events. Augmenting and then placing the event back into the scene, which is difficult without knowing the reverberation characteristics of the scene. However, this would be an interesting direction for further work to see the difference in the naive approach compared to this more advanced approach. Another disadvantage of this technique is that it does not factor in the distribution of a sound event. An analogous to this is for example if humans were being modelled and to generate a new human, a human is copied and their height is augmented, it would lead to some very short and very tall people. This is because, for example if a very short person is augmented to become even shorter, this could lead to an unrealistic human height. Instead a better approach would be to use a Gaussian ditribution which would be modelled based on the mean and variance of human height, then heights would be drawn from this distribution. Unfortunately figuring out the mean sound for a sound event and their distribution is very difficult. Also if the distribution is known then no training would even be required, because that is the aim of machine learning, to learn models. Therefore it is not a practical direction for augmentation. However, with caution, augmentation factors will be chosen to generate naturally sounding audio throughout the scenes. Given these known limitations, experiments will be carried out to see if an improvement can still be made.

Speed perturbation The speed perturbation was implemented using the command line utility *sox's* command called *speed* this command increases the speed of the audio file by some factor with the side effect of an increased pitch, this is simply implemented through re-sampling the audio. *Sox* also provides a command called *stretch* which performs a similar operation but keeps the pitch the same. When listening to the output of the audio files created from using *stretch* they sounded distorted, however, when ran through the SED system, the system performed with similar results to *speed.* Speed was decided to be the one used, however *stretch* is equally as good of a choice. When altering the speed files the corresponding label files were also altered by dividing the times of all the events by the speed factor being used.

Pitch perturbation For pitch perturbation *sox* is used again, this time using the *pitch* command. The pitch command shifts the frequencies of the audio file by some shift in cents. Where 100 cents is a semitone.

VTLP Perturbation There was no tool available to easily apply VTLP therefore it was implemented using Python. Using the following formula, as given in [34]. VTLP was implemented using Librosa.

$$vtlp(f,\alpha) = \begin{cases} f\alpha & \text{if } f \leq F_{\text{hi}} \frac{min(\alpha,1)}{\alpha} \\ F_{\text{max}} - \frac{F_{\text{max}} - F_{\text{hi}} min(\alpha,1)}{F_{\text{max}} - F_{\text{hi}} \frac{min(\alpha,1)}{\alpha}} (F_{\text{max}-f}) & \text{otherwise} \end{cases}$$

Where f is the frequency and α is the factor to augment the frequency. F_{max} is the maximum frequency of the signal which is set to 22050 Hz and F_{hi} is the peak of the augmentation, this will be explained further, later on. For now F_{hi} is set to 17640 Hz ($F_{\text{max}} \times 0.8$).

The first step of the implementation is to turn the time domain audio signal into the frequency domain. This was achieved by using a DFT, in particular using Librosa's *stft* function using a 40 ms frames with 20 ms hop length, Hamming window and 2048 components. After applying a DFT the signal is represented with 1025 bins rather than frequencies in Hz, the formula requires the frequencies to be in Hz. However, each bin has a corresponding centre frequency f_c which can be found with the following formula, where $freqs_i$ is the center frequency f_c for bin i:

$$freqs = linspace(0, F_{\text{max}}, 1025)$$

Where linspace gives a matrix of 1025 evenly spaced numbers between 0 and F_{max}.

These centre frequencies are then transformed using the formula stated before:

$$f_c' = vtlp(f_c, \alpha)) \text{ for each } f_c \in freqs, \text{ for some augmenting factor } \alpha$$

This gives a new frequency for each component. From this the frequencies can be transformed back from frequencies to a bin index. To do this the closest bin for that frequency is found using the following formula:

$$\text{freq2bin}(f_c') = \left\lfloor \frac{(F_{\text{max}} + 1) f_c'}{2F_{\text{max}}} \right\rfloor$$

Therefore the VTLP operation can be described as mapping from one bin index i to another bin index i' as described below with the function ψ:

$$i' = \psi(i, \alpha) = \text{freq2bin}(vtlp(freqs_i, \alpha)) \text{ for each } i \in bins$$

Finally the inverse FFT is used to put the frequency domain signal back into the time domain, where the time domain signal can then be saved to an audio file. The mapping of the frequency bins are shown in figure 6.1 to show the effect of changing α and F_{hi}. F_{hi} is the peak amount of augmentation, after this point the amount of change decreases until F_{max}, α is how extreme the augmentation is. Using 0.4 shows an extreme decrease in frequency, while 1.2 shows a more subtle increase. VTLP changes the frequencies more intensely before F_{hi} as this where the important resonance are, in speech these will be the formants and a F_{hi} would be chosen based on the location of the important formants for speech. After the point F_{hi} the amount of change in frequency is reduced to produce a more natural final result as most of the important variation occurs at the lower frequencies and not the higher frequencies.

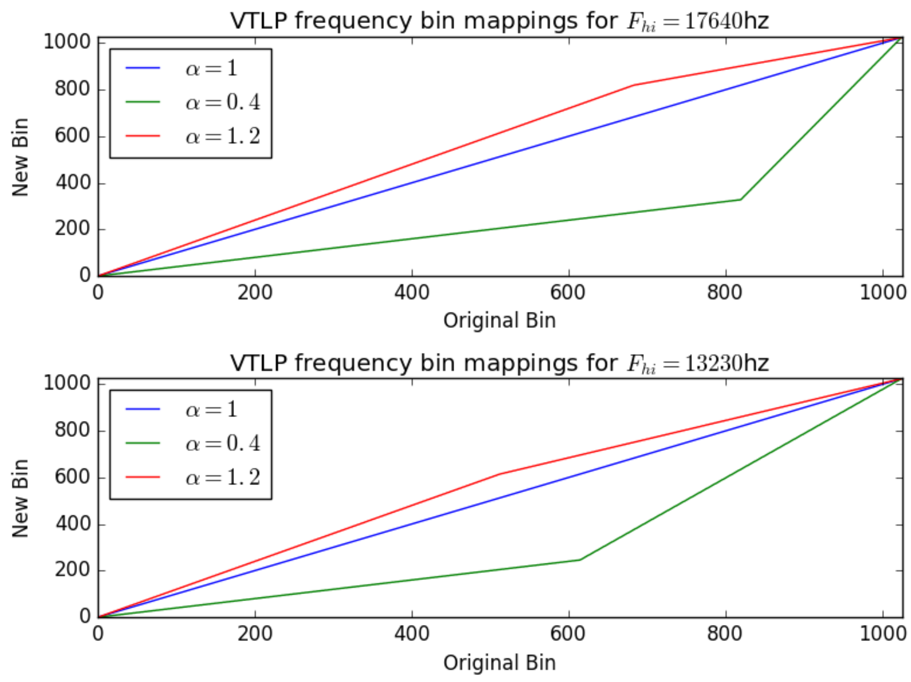


Figure 6.1: The diagrams above show the mapping between bins when VTLP is used. When a higher α is used the augmentation is more intense. The F_{hi} values gives the point of peak augmentation.

6.3 Experiment Setup

For the initial experiments the dataset will be augmented using the three techniques stated prior to this: speed, pitch and VTLP perturbation. Experiments will be carried out for each of the different techniques in isolation. For each augmentation technique different augmentation factors will be used to show the effect this has on performance. The different configurations are categorised into three groups, wide, narrow and dense. Wide augmentations perturb the audio using factors at the extreme ends of what sounds natural. Narrow augmentations perturb the audio using factors that keep the audio close to the original. Finally the dense category augments the audio using the same narrow range however, using more factors in-between the values. To be specific the following augmentation factors will be used creating 9 new datasets:

Speed (Speed factors)

1. Narrow: 0.9, 1.1
2. Wide: 0.8, 0.9, 1.1, 1.2, 1.3
3. Dense: 0.9, 0.95, 1.05, 1.1

Pitch (Cents)

1. Narrow: -100, 100
2. Wide: -300, -100, 100, 300
3. Dense: 0.9, 0.95, 1.05, 1.1

VTLP (α)

1. Narrow: 0.9, 1.1
2. Wide: 0.8, 0.9, 1.1, 1.2, 1.3
3. Dense: 0.9, 0.95, 1.05, 1.1

For the VTLP experiments the F_{hi} value has been set to 17640hz. The same cross validation setup will be used that has been used in all the previous experiments however with some slight alterations as more data is involved. For a fold, only the files in the training data will be augmented and the files in the test data will be untouched. No augmented versions of the test data will be present in the training data as this will be an unfair comparison with the unaugmented data. All the features that were experimented with in the previous chapters will also be experimented with in data augmentation. This is because some features may benefit from the added variety and some may just ignore the changes made, again all the experiments have been repeated using the threshold values between 0 and 300 with a resolution of 1 experiment per 5 threshold units.

6.4 Increasing the Dataset Through Augmentation

For each of the augmentation configurations specified in the previous section a set of experiments will be carried out, for each experiment the SED system will be trained on the original data along with the chosen artificially augmented data. All the different augmentation configurations will be repeated using the different features discussed in the previous chapter. That is for each configuration, experiments will be carried out using the MFCC features and the one additional feature with their corresponding Δ and $\Delta\Delta$s. They will again be combined by concatenating the new features onto the MFCCs. The 16 component GMM will again be used in the classification system.

6.4.1 Performance Difference Between Augmentation Techniques

After performing all the experiments with the different augmentation configurations and features, the F-Scores for using all the different features are averaged together at each threshold to give an overall result for the augmentation techniques.

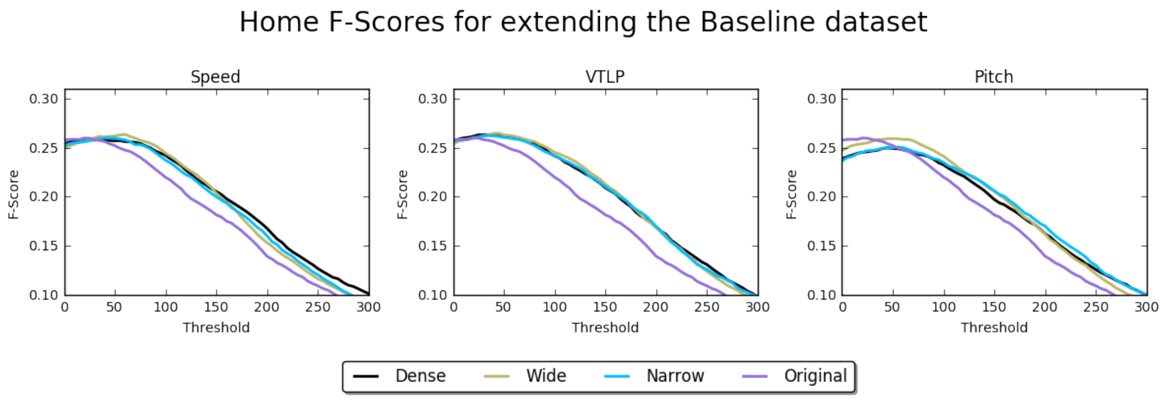


Figure 6.2: Plots showing the F-Score improvement made through data augmentation. All the different techniques showed an improvement over using the normal datasets.

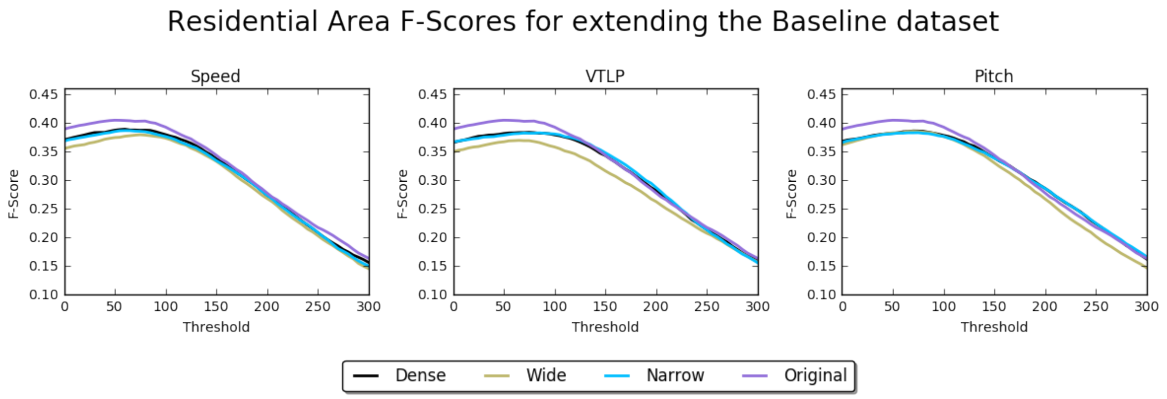


Figure 6.3: Plots showing the F-Score performance degrade in the Residential Area.

Figures 6.2 and 6.3 show the average F-Scores of all the features over the threshold values between 0 and 300, with again a resolution of 1 experiment per 5 threshold units. In the Home scene in figure 6.2 the graphs clearly show that all the augmentation techniques improve upon the F-Score of the original dataset. The different categories of augmentation increased the performance equally well in most of the augmentation techniques. Pitch using the wide category improved more than the others. These results show that the original SED system is not robust to small changes in the Home scene, training on the augmented data allowed the system to become slightly more robust to these small changes. This could be an indicator to why systems are performing poorer in the Home scene in general.

Now looking at the Residential Area scene, augmentation actually made the system perform worse. This means that the augmentation techniques used must have created unrealistic sounds for this scene. For the score to decrease the system would need to have been learning unrealistic sounds in the training stage, which are now overpowering the realistic sounds in the original dataset. From inspecting the graphs the Home scene improved relatively more than

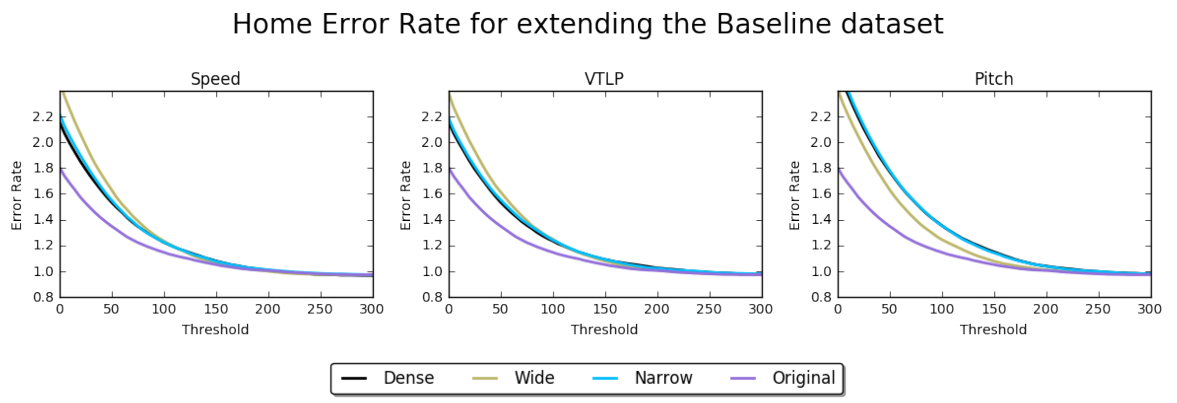


Figure 6.4: Plots showing the performance degrade in terms of Error Rate in the Home scene.

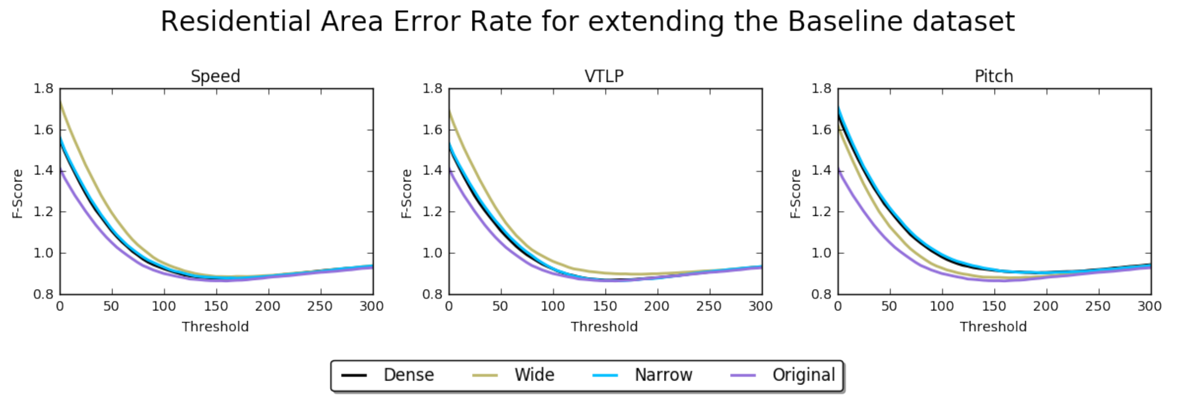


Figure 6.5: Plots showing the performance degrade in terms of Error Rate in the Residential Area scene.

then amount the Residential Area lost in performance. This indicates that augmentation is a worth while operation to perform, however care needs to be taken. For example VTLP Wide performed equally well in the Home scene as the other augmentation techniques, however when looking at the Residential Area scene, this configuration had the most detrimental effect on performance.

Once again looking at the error rates in figures 6.4 and 6.5, the Error Rate has worsened even though the F-Score has got better. Interestingly even though the F-Score decreased the most with the Residential Area scene, it did not lead to as big of an Error Rate increase compared to the Home scene.

6.4.2 Effectiveness of Augmentation with Different Features

The previous plots showed the improvements that the augmentation techniques made in general, however they did not show how the techniques effected each of the individual features.

Feature	**Speed**			**Pitch**			**VTLP**		
	Narrow	Wide	Dense	Narrow	Wide	Dense	Narrow	Wide	Dense
MFCCs	32.26	31.96	32.87	31.35	32.23	31.27	32.37	31.25	32.63
Centroid	31.55	31.51	31.72	31.48	31.90	31.45	31.70	31.44	31.64
Energy	34.18	34.05	33.06	32.95	32.78	33.37	33.54	32.52	32.43
Flux	33.01	32.71	32.59	31.25	32.50	32.13	32.95	32.67	32.74
Rolloff	31.49	33.03	31.44	31.93	32.17	31.49	31.75	31.12	32.76
Sharpness	32.80	31.77	32.12	31.99	32.29	31.99	31.75	30.84	32.24
Slope	31.30	30.12	31.97	30.84	31.96	30.77	31.41	31.38	31.66
Spread	32.92	32.38	33.24	32.40	33.01	31.99	31.95	32.20	32.45
ZCR	31.46	31.31	32.37	31.10	31.64	30.75	32.15	31.31	32.22

Table 6.1: Table of results of all the features extending the MFCCs with the various different augmentations. The table shows the maximum F-Score that was achieved after averaging the two scenes' scores first.

The results of the different features are shown in table 6.1. Features tended to perform slightly better when trained on data through speed augmentation.

Now just using the Dense category for each of the augmentation techniques the results are plotted in figure 6.6. The plots show the performance difference between the augmentation result and the result from the original dataset i.e (augmentation results − original result). Figure 6.6 shows that in the Home scene all the features improved using most of the thresholds which shows that these techniques do result is a better performing system. Interestingly when comparing the results of the Home scene with the results of the Residential Area scene a clear difference in the range of results can be seen. With the Home scene most of the features showed a fairly consistent improvement. The Residential Area scene did not show this same consistency. Some of the features showed quite a large improvement, such as Spread which shows a peak performance increase a 4% increase in F-Score. However, some features showed an equally large degrade in performance such as Rolloff which showed a 4% decrease in F-Score. Throughout all the experiments in the Residential area scene Rolloff and Flux tended to benefit the least from the augmentation, this is a shame considering they were the best performing features when trained on the original data. The features that performed less well on the normal data tended to perform better after augmentation for example Spread produced one of the worst results in the original data, however produced the most improvement consistently in the Residential Area scene.

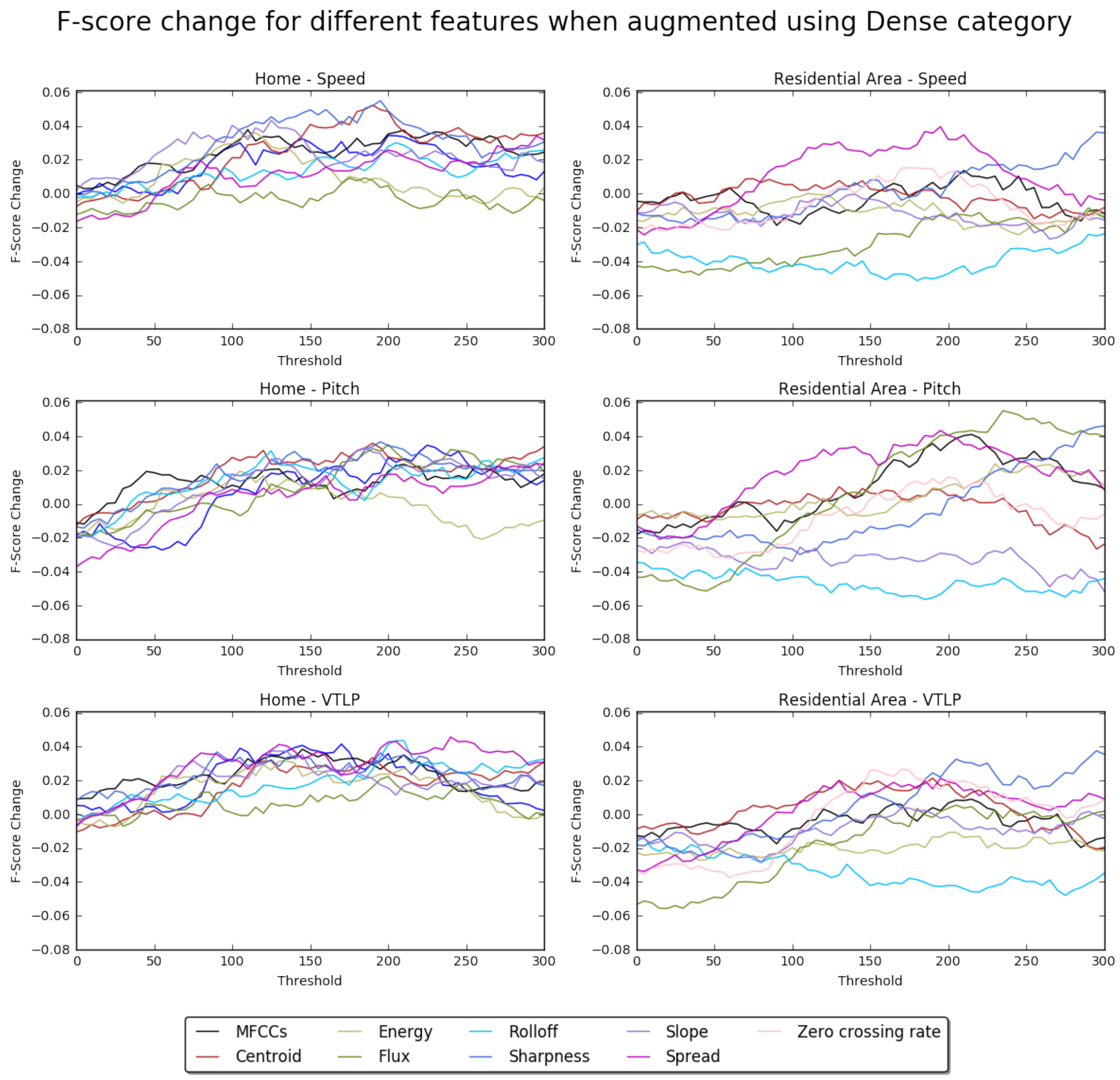


Figure 6.6: Plots showing the performance change of features' scores after augmentation. The Home scene showed a constant improvement with all the different features through-out all the augmentation techniques. The Residential Area has a range of some features performing better and some performing worse.

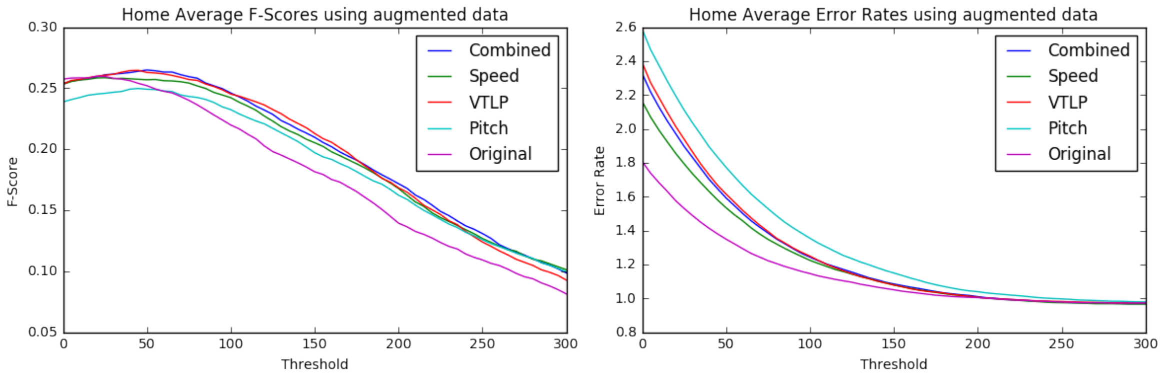


Figure 6.7: Plots comparing the performance of combining data augmentation techniques. The results show that combining techniques does not give much of an improvement over the individual augmentation techniques.

6.5 Combining Augmentation Techniques

The previous section showed that an improvement can be made through augmenting the data using the three techniques. This work will now be continued to combine the better performing augmentation configuration from the previous section and create one large dataset. It is expected that the Residential Area will continue to not improve through more augmentation, as all the different augmentations caused a decrease in performance. However, the Home scene did see an improvement through all the different augmentation techniques, it will be interesting to see if combining these augmented audio files creates an even better system, or whether they conflict and confuse the SED system. To find if combining augmentation techniques improve performance the following best augmentation configurations will be combined: Speed Dense, Pitch Wide and VTLP Dense along with the original dataset. The results are shown in figure 6.7. The scores are again showing the average F-Score/Error Rate from each feature concatenated onto the MFCCs. The plots show that the additional augmentation techniques together do not give any additional performance benefits. This suggests that there is a cap in the performance benefit that can be achieved through augmentation. This also suggests that there is an overlap in the information being added into the dataset through the different augmentation techniques. Training the system using the combined augmented data took around 20 hours for each system using MFCCs with one additional feature. The systems trained on one set of augmentations took around 6 hours to train. These current times are manageable, however if a larger base dataset is used the 20 hours may turn into an unmanageable training time for a small improvement. The results show that only a little amount of augmentation is needed to get most of the benefit that is achieved through combining lots of augmentation techniques.

Features	Dataset	Threshold	F-Score	Error Rate
MFCCs	Normal	35	50.5%	0.96
MFCCs + Flux + Rolloff + Energy + Spread	Augmented	110	50.4%	0.76

Table 6.2: Results of the training the baseline and new system on new data. The system with the additional features and augmentation performed a considerable amount better than the baseline system.

6.6 Running on a New Dataset

For the DCASE challenge the organisers provided a test dataset for the systems to be run on and this was used to score the systems. However, unfortunately these labels are yet to be released. This means the best system created in this report cannot be compared against the ones entered into the competition directly. However, as of writing this report, the new DCASE 2017 challenge has started and for this challenge a new dataset has been provided. The organisers have again provided a cross validation setup to fairly evaluate systems. Using what has been discovered in the previous chapter and in this chapter. The feature extraction and data augmentation techniques will be combined to see how well what has been discovered generalises. The scene for the new dataset provided is a street. This scene is similar to a Residential Area, unfortunately in the previous experiments the Residential Area is where the systems struggle to get the most improvement. A new system will be created by combining the best performing features with the best performing data augmentation techniques. The F-Score and Error Rate will then be stated with no further tweaking to parameters. The scores will again be produced after all four of the folds in the cross validation have been run, which should give a fair reflection of the overall performance of each of the systems.

The following systems will be run on the new dataset:

1. MFCCs trained on the base dataset for the new challenge (Baseline).
2. MFCCs + Flux + Rolloff + Energy trained on the new dataset with Speed Dense, Pitch Wide and VTLP Dense augmentations.

Note that the systems are being trained on the new dataset and not the original. This is because the techniques are being tested and not the instances of the previous systems. The optimal threshold has been decided to be the threshold that produced the highest F-Score in the previous experiments and not the optimal threshold for the new dataset. If just the optimal score for the new dataset is reported, it will not test the generalisability of the systems. The results in table 6.2 show the scores for the baseline system at its optimal threshold and the new system at its optimal threshold. The F-Score's of the two systems are very similar to each other, these score's are very good compared to what was achieved in the dataset used in the experiments before this. The baseline system shows a very slight F-Score improvement of 0.1%. The largest difference in score comes from looking at the

Error Rates. The baseline gets a quite respectable Error Rate of 0.96. The new system has considerably lower Error Rate of 0.76, which is better than any Error Rates found in the previous experiments. If the MFCC baseline threshold was increased it is likely that the Error Rate would get better with the side effect of the F-Score decreasing. Given that the new system has such a good Error Rate it is safe to conclude that this new system has improved upon the baseline system.

6.7 Summary

In this chapter various data augmentation techniques have been explored. The chosen techniques were: Speed, pitch and Vocal Tract Length perturbation. All of these techniques showed a good improvement in the Home Scene however, this was not the case in the Residential Area. These augmentation techniques actually made some of the systems perform worse. The effect of augmentation on each of the features was then explored further. It was found that in the Home scene all the features improved with augmentation. In the Residential Area scene there was a large disparity between the performance of different features, some features performed considerably better and some were performing considerably worse. It was concluded from this that data augmentation through these techniques can certainly improve upon the performance however, care needs to be taken in with certain features.

The various augmentation techniques were then combined together using their optimal configurations in order to find if this would bring further performance benefit. From these experiments it was shown that the same performance benefit can be achieved through very little augmentation as to what was achieved through combining the augmentations techniques. This implies that the performance gain of augmentation saturates very quickly.

Finally, this chapter concluded with running an overall best system on a new data. The best system was decided to be the system will all the best features (MFCCs, Flux, Rolloff, RMS Energy, Spread) and all the best augmentations (Speed Dense, Pitch Wide, VTLP Dense). The results of these experiments showed a considerable lower Error Rate while keeping a similar F-Score.

Chapter 7

Conclusions

7.1 Goals Achieved

The majority of the goals that were set out at the beginning of the project have been achieved. A new sound event detection visualisation tool has been developed which allows researchers to easily inspect their results and share their results with others in an intuitive manner. A new sound event detection framework has been created to allow for the work created in this project to be extended without the need to recreate an entire sound event detection system from fresh. The main research topics of the project have been thoroughly explored. The project aimed to find the effects of introducing new features had on the performance of SED systems, this was achieved through combining features through concatenation. Mainly spectral features were explored here, spatial features were also considered however due to time constraints could not be implemented. As well as features, various data augmentation techniques were explored. Techniques augmenting the audio signal in the time domain and the frequency domain were used to achieve this. At the start of the project, this part of the project also aimed to experiment with data simulation techniques to artificially create more data. However, it was decided that this part would not be carried out to allow for the augmentation sections to be more thorough.

7.2 Further Work

Combining the new features with the augmented data gave successful results on the new 2017 DCASE dataset. Unfortunately the label files for the test data for the 2016 DCASE challenge are yet to be released. The organisers of the challenge have stated that they plan to release the labels soon. Unfortunately it will be too late to score the systems created in this report on the test data. However, when they are released it will be interesting to see how well the systems would rank against the submitted systems. Furthermore, given that the techniques explored in this project were very successful on the new challenge dataset, the system created will be entered into the new 2017 DCASE challenge to see how well this new system ranks against the other systems submitted to this year's challenge. Submitting the system will involve condensing the work in this project into a 4 page paper.

The classification system used in this project used a GMM. Further work could be carried out by repeating the experiments performed in this project with other classification systems. For example a Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) could be the replacement. Given the experimental framework created for this report, such a change should be simple, as it is simply a component of the architecture created. These further experiments will seek to find if the additional features and the additional data through augmentation generalises well. Some classification systems such as ones using a CNN may be invariant to some of the augmentation techniques. However, this is yet to be explored.

Further work with feature extraction could be carried out by exploring the differences between combining features through concatenation and through combining outputs of multiple classifiers that are using different features. Only concatenation was used in this project, however, a system may perform better if say a system was trained using just MFCCs and a system was trained just using Flux and then having these two classifiers combine their likelihoods to make a final decision.

In addition to this, further work could be carried out with extending the dataset. In this project only augmentation was used. However, further work could be achieved through simulating data. Simulating data can be achieved through finding sound events in isolation and then placing them together in a scene. Some work was started with this in the early stages of the project however, unfortunately it was difficult to find enough audio clips that fit the labels of the challenge and the data augmentation sections became a bigger focus of the research.

Finally, throughout the project the analysis of the SED systems was difficult due to the performance metrics that were used. A high F-Score does not necessary give a good SED system. This is understandable as the F-Score metric was initially designed for evaluating document retrieval systems. The Error Rate is also lacking, the main difficulty is that wildly different systems can get similar Error Rates e.g., a system that produces no output will get the same Error Rate as a system which detects all the events but has an equal amount of False Positives as events correctly detected. This is quite likely to happen if there are very few events to be detected. This means that the metric has a bias towards systems that are very cautious. More cautious systems are not necessarily the better systems. For the field of sound event detection to grow further, better metrics of performance are needed.

7.3 Conclusion

The main aims of this project were to explore the effects of novel feature extraction and data augmentation. The results for feature extraction showed that the MFCC features perform really well and deserve to be used as the state of the art. Small performance benefits can be achieved through concatenating on an additional feature such as Flux. A 6.7% relative F-Score improvement was seen when using Flux. When multiple features are concatenated together, the optimal performance does not increase however a more robust system is created. The robustness of the system was determined by the system performing well using a wide range of thresholds. The performance benefits of using different features were a lot clearer in the Home scene compared to the Residential Area. Principal Component Analysis was used to see if reducing the dimensionality of the data effected the performance. It was shown that this reduction caused a negative effect on performance with the chosen features.

Following the research with feature extraction, data augmentation techniques were then explored. Speed, pitch and Vocal Tract Length Perturbation all produced better performing systems. Again the Home scene saw the majority of the performance increase. When inspecting the effect data augmentation had on the various features, in the Home scene a clear consistent improvement was shown throughout all the features. In the Residential Area an improvement was shown for some of the features however, an equal performance degrade was seen with other features. The better performing configurations of the augmentation techniques were then combined together, to give a far larger dataset. This dataset was then used to train systems using the various different features again. The performance gain from using this far larger dataset was just about the same as using any of the other techniques in isolation. From this it was concluded that the performance benefit from augmentation saturates quickly and that when these different techniques are combined it does not give much performance benefit and is not probably worth the considerable longer training time.

After augmentation techniques were explored. The discoveries found in the feature extraction chapter and the data augmentation chapter were combined. The best four performing features were then used in combination with the best augmentation configuration for each of the data augmentation techniques. This combined system was then run on a completely new dataset provided for the new 2017 DCASE challenge. The results from this system running on the dataset were then compared to the results of the 2016 DCASE baseline system on the new dataset. For the results to be a fair comparison the systems were both run on their optimal thresholds found in the experimental dataset. Using this threshold was decided as it also tested the generalisability of the systems, it is likely that the systems will not be at their optimal threshold with the new dataset. The results from these experiments showed a considerable improvement over the baseline. This concludes that performance benefits can be achieved through using more novel features and extending datasets through augmentation.

Bibliography

[1] Y.-T. Peng, C.-Y. Lin, M.-T. Sun, and K.-C. Tsai, “Healthcare audio event classification using hidden markov models and hierarchical hidden markov models,” in *2009 IEEE International Conference on Multimedia and Expo*, 2009.

[2] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Multi-label vs. combined single-label sound event detection with deep neural networks,” in *2015 23rd European Signal Processing Conference (EUSIPCO).* Institute of Electrical and Electronics Engineers (IEEE), 2015.

[3] S. Chu, S. Narayanan, C. c. Kuo, and M. Mataric, “Where am i? scene recognition for mobile robots using audio features,” in *2006 IEEE International Conference on Multimedia and Expo.* Institute of Electrical and Electronics Engineers (IEEE), 2006.

[4] M. E. A. Sehili, B. Lecouteux, M. Vacher, F. Portet, D. Istrate, B. Dorizzi, and J. Boudy, *Sound environment analysis in smart home.* Springer Berlin Heidelberg, 2012.

[5] S. Ntalampiras, I. Potamitis, and N. Fakotakis, “On acoustic surveillance of hazardous situations,” 2016.

[6] V. Barbosa, T. Pellegrini, M. Bugalho, and I. Trancoso, “Browsing videos by automatically detected audio events,” *2011 IEEE EUROCON - International Conference on Computer as a Tool*, 2011.

[7] T. Hao, G. Xing, and G. Zhou, “Isleep: Unobtrusive sleep quality monitoring using smartphones,” 2013.

[8] T. Heittola. Detection and classification of acoustic scenes and events 2016. DCASE2016. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/

[9] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” 2016.

[10] ——, “Metrics for polyphonic sound event detection,” *Applied Sciences*, 2016.

[11] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, “Audio context recognition using audio event histograms,” 2010.

[12] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Multi-label vs. combined single-label sound event detection with deep neural networks," 2015.

[13] T. Heittola, A. Mesaros, and T. Virtanen. (2016) DCASE 2016 baseline system. [Online]. Available: https://github.com/TUT-ARG/DCASE2016-baseline-system-python

[14] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.

[15] S. Adavanne, G. Parascandolo, P. Pertila, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," 2016.

[16] T. Heittola, A. Mesaros, T. Virtanen, and M. Gabbouj, "Supervised model training for overlapping sound events based on unsupervised source separation," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[17] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen, "Sound event detection in multisource environments using source separation," 2011.

[18] D. Wei, J. Li, P. Pham, S. Das, S. Qu, and F. Metze, "Sound event detection for real life audio DCASE challenge," 2016.

[19] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," 2016.

[20] S. Adavanne, G. Parascandolo, P. Pertila, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," 2016.

[21] A. Diment, T. Heittola, and T. Virtanen, "IEEE AASP challenge on detection and classification of acoustic scenes and events sound event detection for office live and office synthetic aasp challenge," 2013.

[22] K. Azad. An interactive guide to the Fourier Transform – BetterExplained. [Online]. Available: https://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/

[23] D. Marshall, "Nyquist's sampling theorem," 2001. [Online]. Available: https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node149.html

[24] R. Moore. The fourier transform. [Online]. Available: https://www.dcs.shef.ac.uk//campus_only/COM3502-4502-6502/Lecture%20Notes/COM3502-4502-6502_L15_Fourier-Transform.pdf

[25] (2012, 08) Hamming window - diracdelta science & engineering encyclopedia. [Online]. Available: http://www.diracdelta.co.uk/science/source/h/a/hamming%20window/source.html

[26] T. Heittola. (2016, 11) DCASE 2016 baseline system matlab. [Online]. Available: https://github.com/TUT-ARG/DCASE2016-baseline-system-matlab

[27] Y.-H. Lai, C.-H. Wang, S.-Y. Hou, B.-Y. Chen, Y. Tsao, and Y.-W. Liu, "DCASE report for task 3: Sound event detection in real life audio," 2016.

[28] (2009) Practical cryptography. [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[29] D. Vij, N. Aggarwal, B. Raman, K.K.Ramakrishnan, and D. Bansal, "Acoustic scene classification based on spectral analysis and feature-level channel combination," September 2016.

[30] J. H. Foleiss and T. F. Tavares, "Mel-band features for DCASE 2016 acoustic scene classification task," September 2016.

[31] R. Cachu, S. Kopparthi, B. Adapa, and B. Barkana, "Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal," 2016.

[32] A. Gorin, N. Makhazhanov, and N. Shmyrev, "DCASE 2016 sound event detection system based on convolutional neural network," 2016.

[33] M. Zöhrer and F. Pernkopf, "Gated recurrent networks applied to acoustic scene classification and acoustic event detection," 2016.

[34] N. Jaitly and G. E. Hinton, "Vocal tract length perturbation (vtlp) improves speech recognition," 2013.

[35] N. Takahashi, M. Gygli, and L. V. Gool, "Learning deep audio features for video analysis," 2017.

[36] Time stretching and pitch shifting of audio signals – an overview. [Online]. Available: http://blogs.zynaptiq.com/bernsee/time-pitch-overview/

[37] (2013) IEEE AASP challenge: Detection and classification of acoustic scenes and events. [Online]. Available: http://c4dm.eecs.qmul.ac.uk/sceneseventschallenge/

[38] Sed eval. [Online]. Available: https://github.com/TUT-ARG/sed_eval

[39] T. Heittola. (2016) Sound event detection. Toni Heittola. [Online]. Available: http://www.cs.tut.fi/~heittolt/research-sound-event-detection

[40] (2016, 11) sed_vis. GitHub. [Online]. Available: https://github.com/TUT-ARG/sed_vis

[41] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[42] Scikit learn mixture. [Online]. Available: http://scikit-learn.org/stable/modules/mixture.html

[43] (2016) Librosa. [Online]. Available: http://librosa.github.io/

Appendices

Appendix A

DCASE Challenge 2016 Results

	Submission Info	**Segment-based**		**System Characteristics**		
Rank	**Code**	**Error Rate**	**F-Score**	**Input**	**Features**	**Classifier**
1	Adavanne_1	0.8051	47.8%	binaural	Mel energy	RNN
2	DCASE	0.8773	34.3%	mono	MFCC	GMM
3	Adavanne_2	0.8887	37.9%	binaural	Mel energy + TDOA	RNN
4	Zoehrer	0.9056	39.6%	mono	Spectrogram	GRNN
5	Vu	0.9124	41.9%	mono	Mel energy	RNN
6	Liu	0.9287	34.5%	mono	MFCC	Fusion
7	Kong	0.9557	36.3%	mono	MFCC	DNN
8	Pham	0.9583	11.6%	mono	MFCC	DNN
9	Elizalde_4	0.9613	33.6%	mono	MFCC	Random forests
10	Elizalde_3	0.9635	33.3%	mono	MFCC	Random forests
11	Phan	0.9644	23.9%	mono	GCC	Random forests
12	Gorin	0.9799	41.1%	mono	Mel energy	CNN
13	Ubskii	0.9971	39.6%	mono	MFCC	Fusion
14	Elizalde_1	1.0730	22.5%	mono	MFCC	Random forests
15	Elizalde_2	1.1056	20.8%	mono	MFCC	Random forests
16	Kroos	1.1488	16.8%	N/A	N/A	Random
17	Schroeder	1.3092	33.6%	mono	GFB	GMM-HMM

Table A.1: Results of the systems for the DCASE challenge ranked by their Segment-based error score.

Appendix B

Installing the SED Visualisation tool

B.1 Front End

The front end code is built using modern ES6 functionality. This means it is not possible to run the raw Javascript files. The Javascript files need to be compiled using a build process which combines the files and converts them to a more compatible version of Javascript. A build script has be created which requires the node package gulp. If *node* is not already installed, install it from `https://nodejs.org/en/`. Once *node* has been installed the command line programs *node* and *npm* should be available. Now to install *gulp* run the following command:

```
$ npm install -g gulp
```

Now *gulp* has been installed, the other dependencies of the project need to be installed. To do this change to the directory where the project files are located. All the other dependencies can be installed using the following command:

```
$ npm install
```

This command installs all the dependencies located in the *package.json* file. The source can then be compiled using the following command:

```
$ gulp compileJS
```

This will then create a *main.min.js* file in the src directory. This should be used for deploying the application. When developing the application the following command can be used:

```
$ npm run dev
```

This boots up a development node server to compile the Javascript files when the files are saved.

B.2 Backend

In order to run the application the Python server needs to be running. The following packages are needed to be installed through pip or anaconda: pyyaml, librosa, numpy, hashids, scipy and flask to run the application. The *client.min.js* file that was compiled in the previous section needs to be placed into the static/js directory of the server folder in order for the latest version of the code to be used. The server is now ready to be run. From the root of the project run python3 server/server.py

Appendix C

Baseline System Results

This section shows that the re-engineered baseline system has not introduced any bugs. The first section shows the original systems results. This output was generated from the program and not using the evaluation tool that was provided by the competition. The second section shows the results from the new re-engineered system which does use the evaluation system provided.

C.1 Original Baseline System Results

```
| Main     |   | Secondary metrics
|          |   | Seg/Overall                                   | Seg/Class       | Event/Overall   |  Event/Class
Scene               | ER     |   | F1       : ER     : ER/S   : ER/D   : ER/I   | F1       : ER     | F1       : ER     | F1       : ER     |
--------------------+--------+   +----------+--------+--------+--------+--------+----------+--------+----------+--------+----------+--------+
home                |   0.97 |   | 15.4 % :   0.97 :   0.08 :   0.83 :   0.06 |   8.9 % :   1.06 |   4.8 % :   1.27 |   4.4 % :   1.27 |
residential_area    |   0.86 |   | 31.5 % :   0.86 :   0.05 :   0.74 :   0.07 | 17.6 % :   1.03 |   2.9 % :   1.92 |   1.5 % :   1.97 |
--------------------+--------+   +----------+--------+--------+--------+--------+----------+--------+----------+--------+----------+--------+
Average             |   0.91 |   | 23.4 % :   0.91 :                            | 13.2 % :   1.04 |   3.8 % :   1.60 |   2.9 % :   1.62 |
```

```
    Results per events

  HOME
  | Segment-based                      |   | Event-based
  Event                    | Nref  : Nsys  : F1       : ER     |   | Nref  : Nsys  : F1       : ER     |
  -------------------------+-------+-------+----------+--------+   +-------+-------+----------+--------+
  (object) rustling        |   170 :    29 :   7.0 % :   1.09 |   |    41 :    22 :   6.3 % :   1.44 |
  (object) snapping        |    61 :     0 :   0.0 % :   1.00 |   |    42 :     0 :   0.0 % :   1.00 |
  cupboard                 |    43 :     0 :   0.0 % :   1.00 |   |    27 :     0 :   0.0 % :   1.00 |
  cutlery                  |    84 :     2 :   0.0 % :   1.02 |   |    56 :     2 :   0.0 % :   1.04 |
  dishes                   |   204 :    38 :   2.5 % :   1.16 |   |    94 :    26 :   0.0 % :   1.28 |
  drawer                   |    42 :     8 :   0.0 % :   1.19 |   |    23 :     4 :   0.0 % :   1.17 |
  glass jingling           |    48 :     5 :   0.0 % :   1.10 |   |    26 :     3 :   0.0 % :   1.12 |
  object impact            |   291 :    92 :  19.3 % :   1.06 |   |   155 :    67 :   3.6 % :   1.38 |
  people walking           |    84 :    24 :  14.8 % :   1.10 |   |    24 :    16 :  10.0 % :   1.50 |
  washing dishes           |   297 :   107 :  20.3 % :   1.08 |   |    60 :    48 :   9.3 % :   1.63 |
  water tap running        |   248 :    63 :  34.1 % :   0.83 |   |    37 :    26 :  19.0 % :   1.38 |
  -------------------------+-------+-------+----------+--------+   +-------+-------+----------+--------+
  Sum                      |  1572 :   368 :                   |   |   585 :   214 :                   |
  Average                  |               :   8.9 % :   1.06 |   |               :   4.4 % :   1.27 |


  RESIDENTIAL_AREA
  | Segment-based                      |   | Event-based
  Event                    | Nref  : Nsys  : F1       : ER     |   | Nref  : Nsys  : F1       : ER     |
  ----------------------+--------+--------+---------+--------+  +--------+--------+---------+--------+
  (object) banging      |   26 :      0 :  0.0 % :   1.00 |  |    15 :      0 :  0.0 % :   1.00 |
  bird singing          | 1095 :    275 : 30.1 % :   0.87 |  |   162 :    131 :  4.8 % :   1.72 |
  car passing by        |  576 :    320 : 54.5 % :   0.71 |  |    74 :    119 :  3.1 % :   2.53 |
  children shouting     |   68 :      5 :  0.0 % :   1.07 |  |    23 :      4 :  0.0 % :   1.17 |
  people speaking       |  365 :     67 : 25.0 % :   0.89 |  |    41 :     47 :  0.0 % :   2.15 |
  people walking        |  207 :     35 :  1.7 % :   1.15 |  |    32 :     23 :  0.0 % :   1.72 |
  wind blowing          |  157 :    115 : 11.8 % :   1.53 |  |    22 :     57 :  2.5 % :   3.50 |
  ----------------------+--------+--------+---------+--------+  +--------+--------+---------+--------+
  Sum                   | 2494 :    817 :                 |  |   369 :    381 :                 |
  Average              |               : 17.6 % :   1.03 |  |                 :  1.5 % :   1.97 |
```

C.2 Re-engineered Baseline System Results

C.2.1 Home

```
Segment based metrics
========================================
Evaluated length  : 2033.7 sec
Evaluated files   : 10    files
Segment length    : 1.00  sec

Overall metrics (micro-average)
===============
F-measure
F-measure (F)     :  15.4 %
Precision         :  40.5 %
Recall            :   9.5 %
Error rate
Error rate (ER)   :  0.97
Substitution rate :  0.08
Deletion rate     :  0.83
Insertion rate    :  0.06
Accuracy
Sensitivity       :   9.5 %
Specificity       :  99.0 %
Balanced accuracy :  54.2 %
Accuracy          :  92.8 %

Class-wise average metrics (macro-average)
===============
F-measure
F-measure (F)     :  10.9 %
Precision         :  25.3 %
Recall            :   5.7 %
Error rate
Error rate (ER)   :  1.06
Deletion rate     :  0.94
Insertion rate    :  0.11
Accuracy
Sensitivity       :   5.7 %
Specificity       :  98.9 %
Balanced accuracy :  52.3 %
Accuracy          :  92.8 %
```

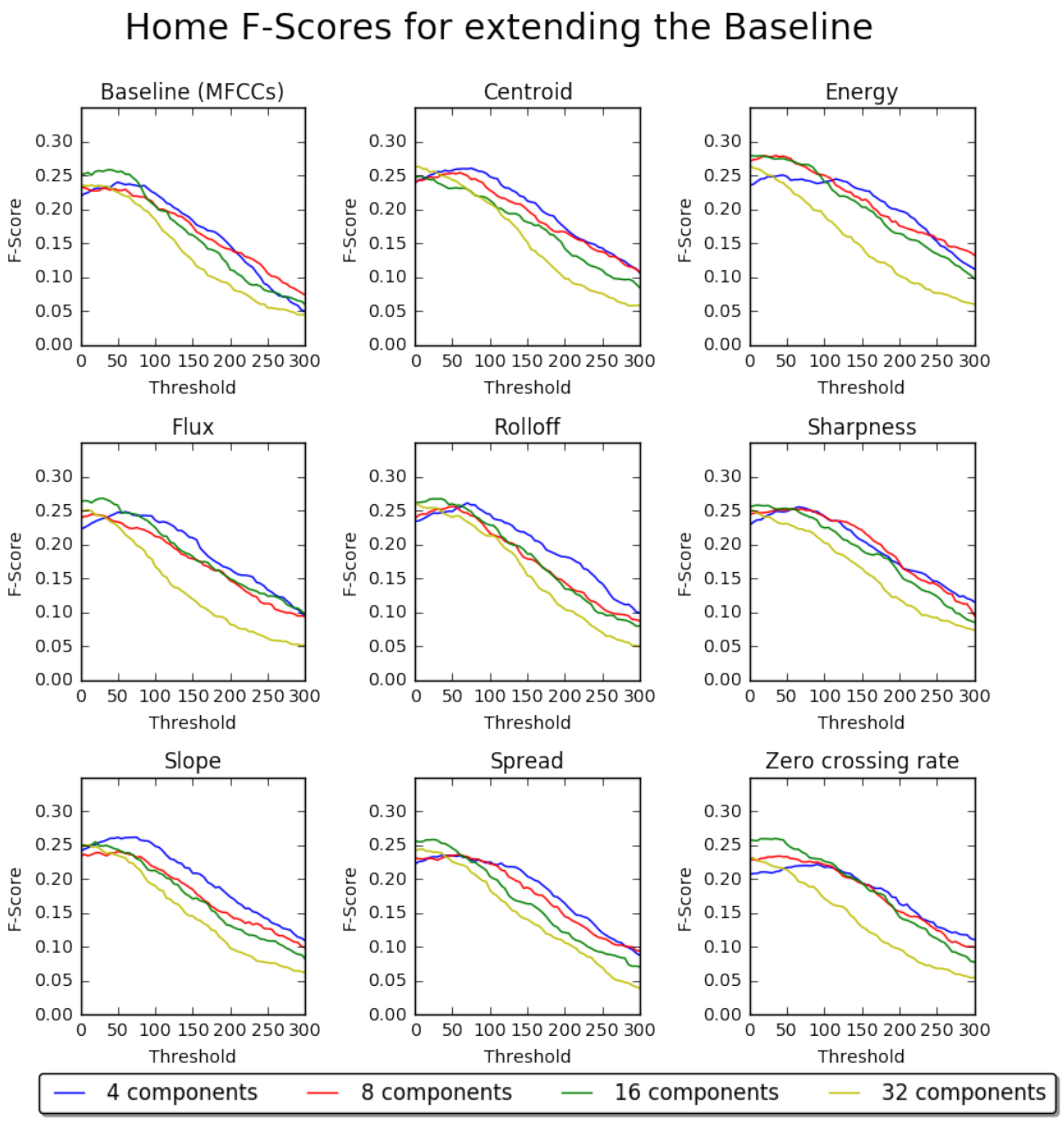
```
Class-wise metrics
===============
Event label        | Nref | Nsys |    F    :   Pre    :   Rec    |  ER   :  Del  :  Ins  |  Sens   :  Spec   :  Bacc   |   Acc   |
-------------------+------+------+---------+----------+----------+-------+-------+-------+---------+---------+---------+---------+
(object) rustling  |  170 |   29 |   7.0 %    24.1 %      4.1 % | 1.09    0.96    0.13  |   4.1 %    98.8 %    51.5 % |  91.0 % |
(object) snapping  |   61 |    0 |   nan %     nan %      0.0 % | 1.00    1.00    0.00  |   0.0 %   100.0 %    50.0 % |  97.0 % |
cupboard           |   43 |    0 |   nan %     nan %      0.0 % | 1.00    1.00    0.00  |   0.0 %   100.0 %    50.0 % |  97.9 % |
cutlery            |   84 |    2 |   0.0 %     0.0 %      0.0 % | 1.02    1.00    0.02  |   0.0 %    99.9 %    49.9 % |  95.8 % |
dishes             |  204 |   38 |   2.5 %     7.9 %      1.5 % | 1.16    0.99    0.17  |   1.5 %    98.1 %    49.8 % |  88.6 % |
drawer             |   42 |    8 |   0.0 %     0.0 %      0.0 % | 1.19    1.00    0.19  |   0.0 %    99.6 %    49.8 % |  97.6 % |
glass jingling     |   48 |    5 |   0.0 %     0.0 %      0.0 % | 1.10    1.00    0.10  |   0.0 %    99.8 %    49.9 % |  97.4 % |
object impact      |  291 |   92 |  19.3 %    40.2 %     12.7 % | 1.06    0.87    0.19  |  12.7 %    96.9 %    54.8 % |  85.0 % |
people walking     |   84 |   24 |  14.8 %    33.3 %      9.5 % | 1.10    0.90    0.19  |   9.5 %    99.2 %    54.4 % |  95.5 % |
washing dishes     |  297 |  107 |  20.3 %    38.3 %     13.8 % | 1.08    0.86    0.22  |  13.8 %    96.3 %    55.0 % |  84.4 % |
water tap running  |  248 |   63 |  34.1 %    84.1 %     21.4 % | 0.83    0.79    0.04  |  21.4 %    99.4 %    60.4 % |  90.1 % |
```

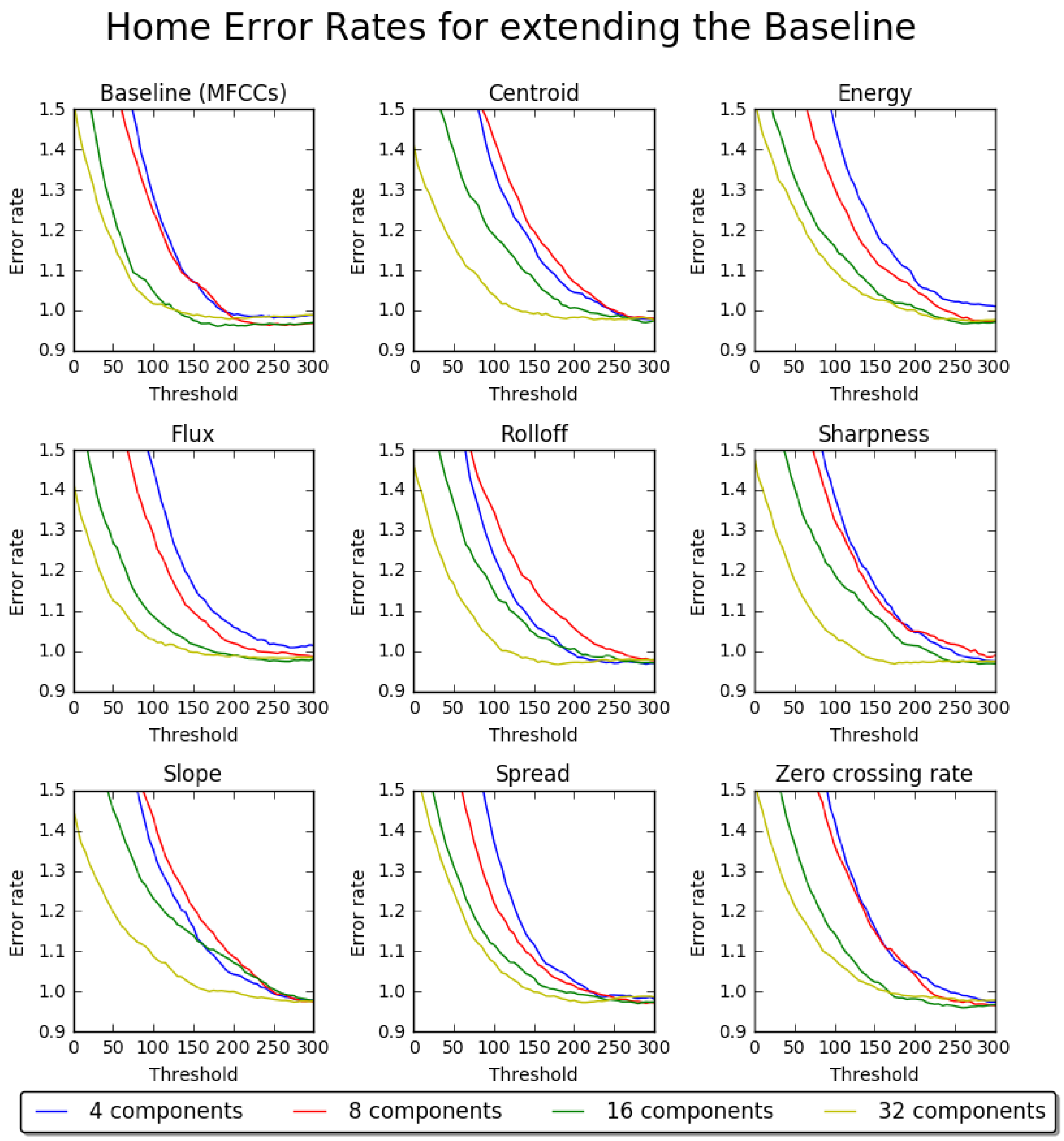
C.2.2 Residential Area

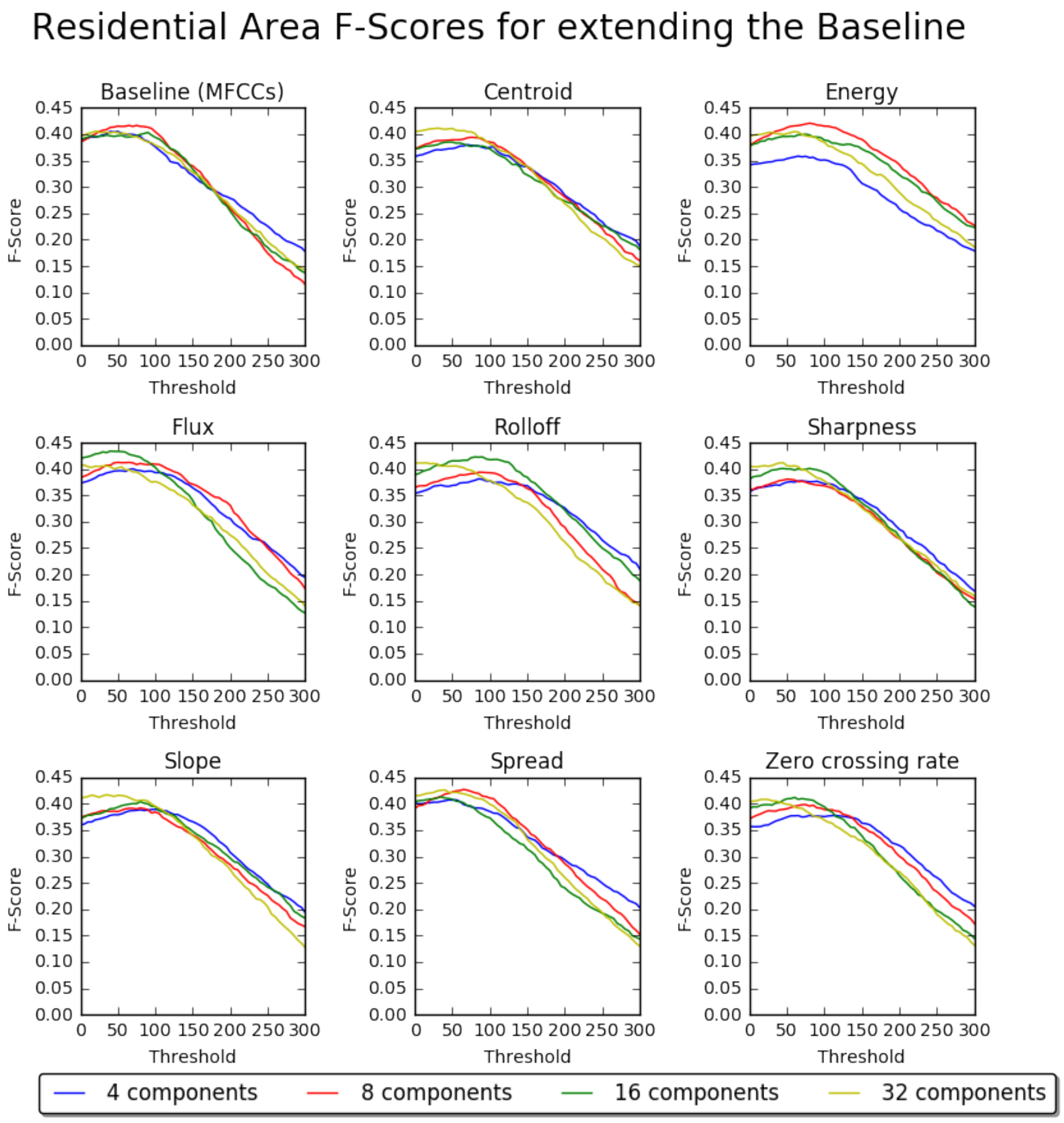
```
Segment based metrics
========================================
Evaluated length   : 2482.9 sec
Evaluated files    : 12    files
Segment length     : 1.00  sec

Overall metrics (micro-average)
===============
F-measure
F-measure (F)      :  31.5 %
Precision          :  63.9 %
Recall             :  20.9 %
Error rate
Error rate (ER)    :  0.86
Substitution rate  :  0.05
Deletion rate      :  0.74
Insertion rate     :  0.07
Accuracy
Sensitivity        :  20.9 %
Specificity        :  98.0 %
Balanced accuracy  :  59.5 %
Accuracy           :  87.1 %

Class-wise average metrics (macro-average)
===============
F-measure
F-measure (F)      :  20.5 %
Precision          :  41.9 %
Recall             :  12.4 %
Error rate
Error rate (ER)    :  1.03
Deletion rate      :  0.88
Insertion rate     :  0.16
Accuracy
Sensitivity        :  12.4 %
Specificity        :  97.8 %
Balanced accuracy  :  55.1 %
Accuracy           :  87.1 %


Class-wise metrics
===============
Event label       | Nref | Nsys |   F     :   Pre    :   Rec    |  ER   :  Del  :  Ins  |  Sens   :  Spec    :  Bacc   |  Acc    |
------------------+------+------+---------+----------+----------+-------+-------+-------+---------+----------+---------+---------+
(object) banging  |   26 |    0 |  nan %     nan %      0.0 %  | 1.00    1.00    0.00  |   0.0 %   100.0 %    50.0 % |  99.0 % |
bird singing      | 1095 |  275 | 30.1 %     74.9 %     18.8 %  | 0.87    0.81    0.06  |  18.8 %    95.1 %    57.0 % |  61.9 % |
car passing by    |  576 |  320 | 54.5 %     76.2 %     42.4 %  | 0.71    0.58    0.13  |  42.4 %    96.1 %    69.2 % |  83.8 % |
children shouting |   68 |    5 |  0.0 %      0.0 %      0.0 %  | 1.07    1.00    0.07  |   0.0 %    99.8 %    49.9 % |  97.1 % |
people speaking   |  365 |   67 | 25.0 %     80.6 %     14.8 %  | 0.89    0.85    0.04  |  14.8 %    99.4 %    57.1 % |  87.1 % |
people walking    |  207 |   35 |  1.7 %      5.7 %      1.0 %  | 1.15    0.99    0.16  |   1.0 %    98.6 %    49.8 % |  90.5 % |
wind blowing      |  157 |  115 | 11.8 %     13.9 %     10.2 %  | 1.53    0.90    0.63  |  10.2 %    95.8 %    53.0 % |  90.5 % |
```

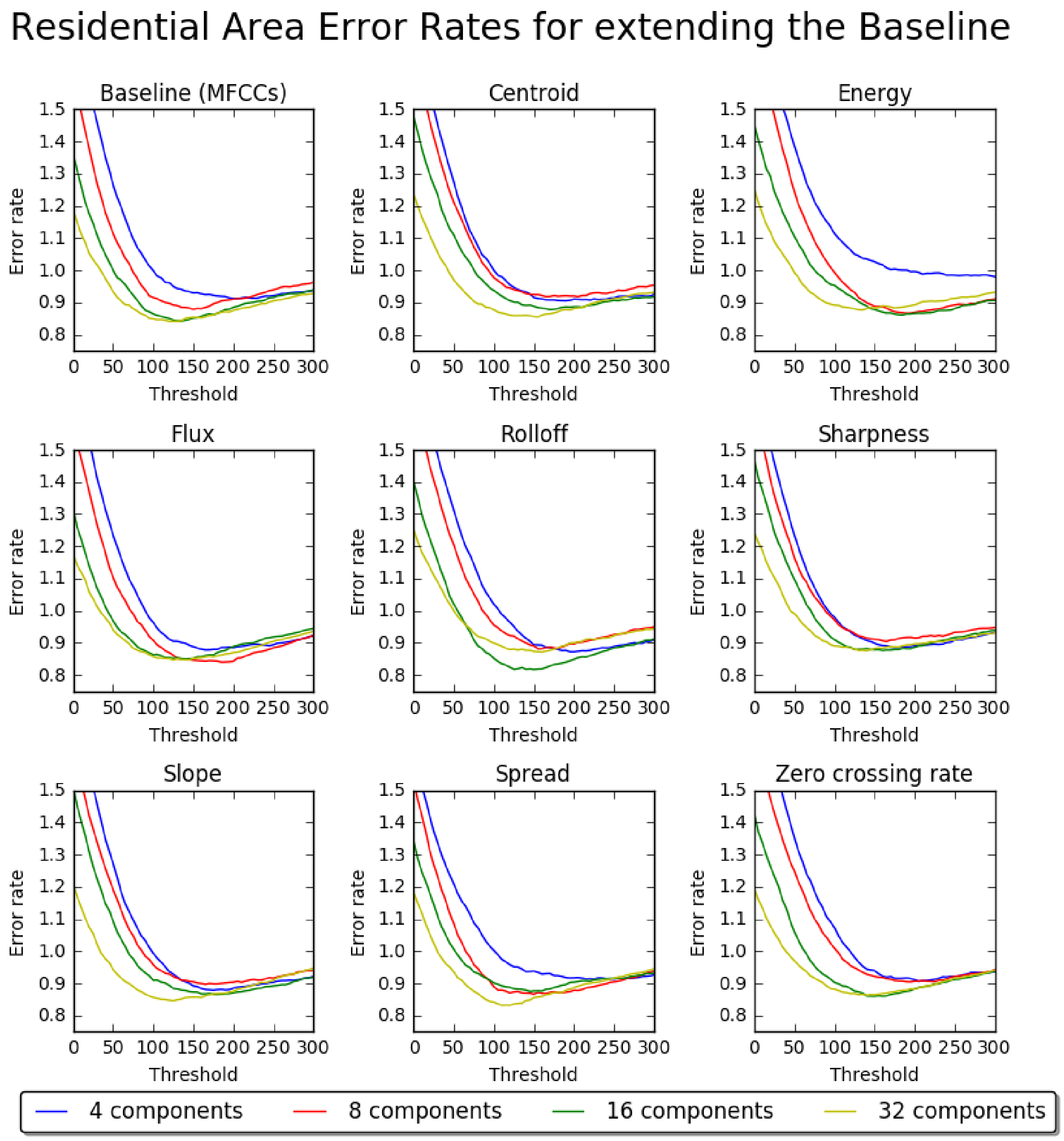
Appendix D

Experiments with multiple Components


Home F-Scores for extending the Baseline
Baseline (MFCCs)
Centroid
Energy
Flux
Rolloff
Sharpness
Slope
Spread
Zero crossing rate
F-Score
Threshold
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
50
100
150
200
250
300
4 components
8 components
16 components
32 components


Home Error Rates for extending the Baseline
Baseline (MFCCs)
Centroid
Energy
Flux
Rolloff
Sharpness
Slope
Spread
Zero crossing rate
Error rate
Threshold
0.9
1.0
1.1
1.2
1.3
1.4
1.5
0
50
100
150
200
250
300
4 components
8 components
16 components
32 components


Residential Area F-Scores for extending the Baseline
Baseline (MFCCs)
Centroid
Energy
Flux
Rolloff
Sharpness
Slope
Spread
Zero crossing rate
F-Score
Threshold
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0
50
100
150
200
250
300
4 components
8 components
16 components
32 components


Residential Area Error Rates for extending the Baseline
Baseline (MFCCs)
Centroid
Energy
Flux
Rolloff
Sharpness
Slope
Spread
Zero crossing rate
Error rate
Threshold
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
0
50
100
150
200
250
300
4 components
8 components
16 components
32 components