A Real-Time Sheep Vocalisation Detection System

By
James Colin Bishop

A thesis submitted for the degree of
Bachelor of Computing Science (Honours)
of The University of New England
January 2016

Acknowledgements

I would like to thank the University of New England for allowing me the opportunity to pursue my Honours degree. My gratitude to the Australian Wool Education Trust for awarding me a scholarship, which facilitated the purchase of equipment, without which this project would not have been possible. A huge thanks to my supervisors, Dr. Gregory Falzon, for his wealth of knowledge and limitless depths of patience, and Dr. Mark Trotter, for his infectious enthusiasm and expertise in all thing agriculture related. Finally, a special thank you to my family, for giving me a reason to push the limits of my knowledge, and for supporting me in the pursuit of my goals.

Abstract

The thesis aims to investigate the viability of a real-time sheep vocalisation detection system.

In order to achieve this objective, the different aspects of a real-time audio pattern recognition system were identified, and a thorough review of the literature was conducted. Based on this information, a number of novel approaches to audio detection were selected, including Discrete Wavelet Transforms, low-computational statistical features, and a Support Vector Data Description.

To truly test the objective, software was developed in the C++ programming language, theoretically capable of performing the intended function: the methods used to achieve this are outlined.

In order to test the detection accuracy of the system under different conditions, four experiments were conducted, and the results are presented.

The results obtained lend considerable strength to the assertion that real-time sheep vocalisation detection is possible: high detection accuracy was measured in 3 experiments. The performance of some of the key components was explored, with results indicating that they are all viable options for real-time audio analysis, mirroring the findings of other research found in the literature. In some areas, the system's performance was reduced, and these findings highlight areas for future research.

As this system also relates to livestock welfare and management, Precision Livestock Farming principles are discussed, with particular attention paid to topics pertaining to sheep, and to the benefits the proposed system would offer to industry.

Contents

Chapter 1

Introduction

1.1 Objectives

The objective of this thesis is to develop a real-time audio recognition system which can be applied to the field of Precision Livestock Farming (PLF), and in particular, the problem of sheep vocalisation detection. In order to achieve this aim, a number of technologies need to be surveyed and explored, as real-time audio pattern recognition is a broad and challenging field, encompassing aspects of computer science, digital signal processing (Digital Signal Processing (DSP)), machine learning, and engineering. Due to the breadth of the topic, and the intricacies of each step in the signal chain, there are a multitude of differing approaches to acoustic detection problems, with suitability determined both by the problem, and the theoretical space being explored. Although a large portion of research to date has focused on automatic speech recognition (ASR)[127, 228] and thus a great deal of amelioration has originated from this field, there are examples of similar approaches being applied to other acoustic detection problems, such as wild animal call detection and discrimination [11, 42, 108, 155, 182, 241, 245, 263], emotion recognition [310, 200], and PLF applications [125, 30, 107].

1.2 Overview of Audio Detection Systems

The functions of an audio detection system generally mirror the overview presented in (Fig 1.1): that of capturing the sound, transforming it to emphasise certain characteristics, extracting and compressing meaningful information, then feeding this data into a machine learning model, which designates the class of the sound, and thus facilitates decision making [65, 64, 58, 282]. The desire for real-time detection and feedback imposes computational constraints on the system, in that the entire signal chain must be traversed within a very small time frame so as to allow an immediate response to acoustic stimuli. As each step in the signal chain requires a knowledge of a completely different

discipline, each with its own approaches and nuances, covering a vast array of different problems and deployment spaces, the appropriateness and successful configuration of a particular combination of techniques can be difficult to ascertain; it is therefore necessary to focus on each stage before attempting to combine them into a functional system.

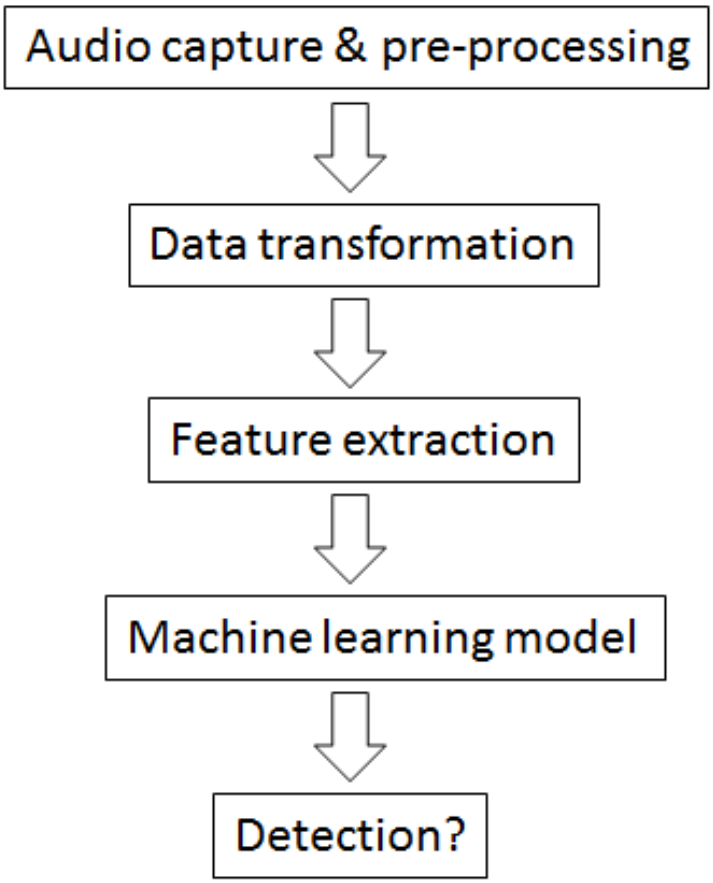

Figure 1.1: A broad overview of how audio pattern recognition systems operate.

1.3 Audio Capture and Pre-Processing

The audio capture component of the system is responsible for obtaining a window of the incoming digital audio stream, converted from the original analogue signal, and represented as a series of time-amplitude data [175] (Fig 1.2); the characteristics of digital audio are discussed in (Section 2.1.1). To accurately sample incoming sounds, it is standard practice to employ an overlapping or sliding window scheme (Fig 1.3), with the length and overlap being determined by the target sound of interest [214, 420]. Although time-amplitude information is helpful in audio analysis, sounds are characterised by their composition of frequency components (Fig 1.4) (measured in cycles per second), and for

this reason, there is arguably more useful information for discrimination contained within the spectral domain [38, 342].

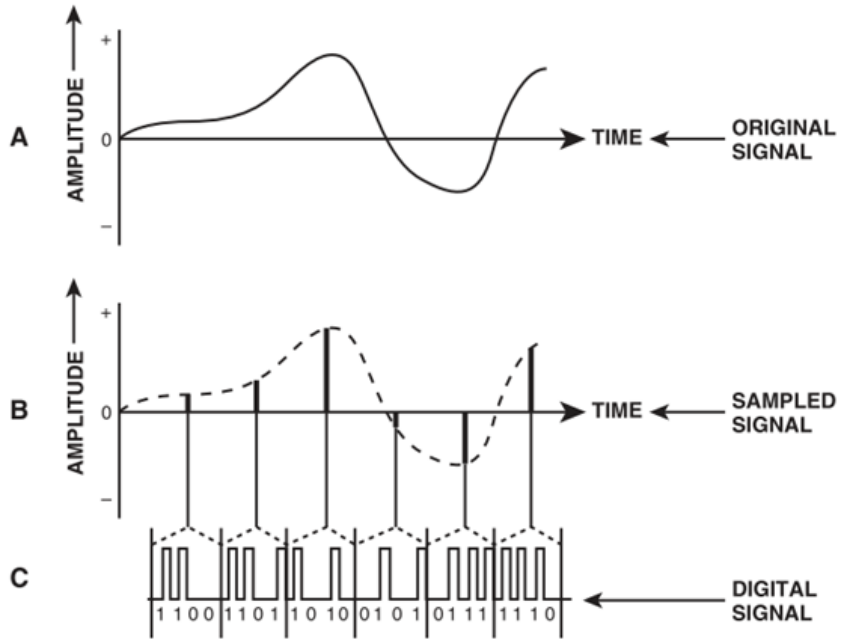

Figure 1.2: An overview of analogue to digital conversion [308]. The original signal is sampled, at a rate determined by the sample rate, and then quantised, with a range defined by the bit rate.

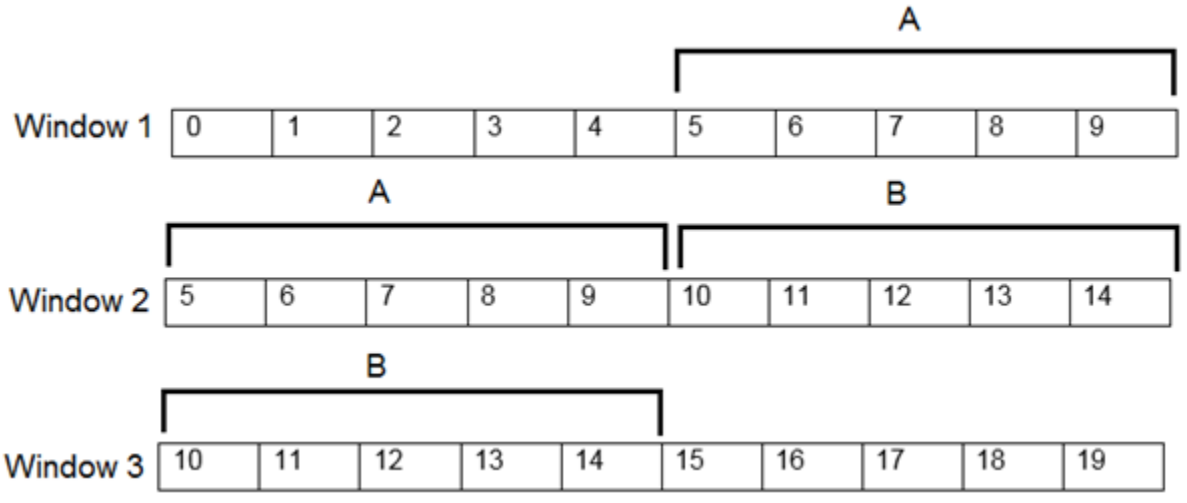

Figure 1.3: Overlapping / sliding window, where overlap is 2 (i.e. 2 sub-windows make 1 full window). Note how each subsequent window contains a portion of the previous window.

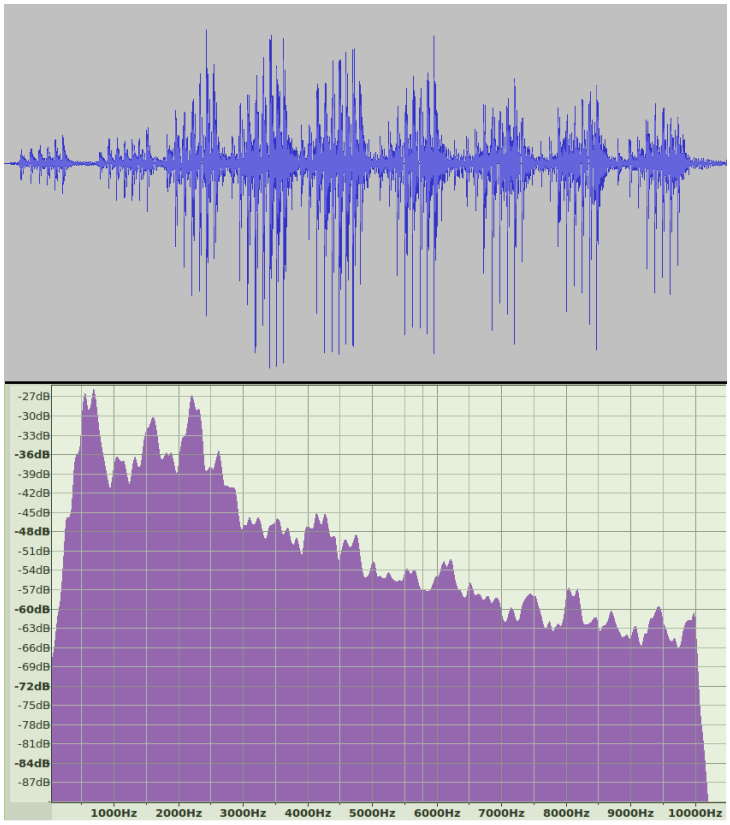

Figure 1.4: A sheep call. The top panel displays the call in the temporal domain (i.e. amplitude at a given time), whereas the bottom panel shows the call in the spectral domain (i.e. the amplitude of each frequency component, from low to high). These images were produced with Audacity sound editing software [12].

1.4 Data Transformation

1.4.1 Spectral Analysis

Once incoming audio data has been obtained, it is necessary to transform the data to extract information that will subsequently be used by the machine learning model [26, 37, 146]. There are myriad approaches to this problem, but they can be largely categorised by the space they operate in, namely temporal, spectral, or a combination of the two [390]. The majority of audio detection techniques use spectral information, but this approach is not without its drawbacks (see Section 2.1.3), with one of the most predominant being the loss of temporal information cause by conversion to the spectral domain [304].

1.4.2 The Discrete Wavelet Transform

An approach that incorporates both the temporal and spectral domains would be theoretically advantageous, and it is hypothesised that such a technique could improve detection accuracy and system performance [301]. One such approach, namely wavelets, has been an area of active inquiry, and the Discrete Wavelet Transform (Discrete Wavelet Transform (DWT)) [79, 80, 147], which has historically been applied mainly to image analysis problems [206, 204, 395, 266, 70, 82, 62, 385, 417, 205, 330, 301, 177], has begun to gain some traction within the audio analysis research community [2, 1, 38, 76, 326, 342, 420, 211, 295]. The DWT provides an output of coefficient sub-bands that describe amplitude simultaneously in both the temporal and spectral domains [301, 420], and in particular, the Daubechies Discrete Wavelet Transform (Daubechies Discrete Wavelet Transform (DDWT)) [79, 80] is a theoretically ideal candidate for acoustic dissection.

1.5 Feature Extraction

The output provided by the DWT tends to be dimensionally vast [1, 287, 463], something that is generally undesirable in a machine learning setting [94, 420], and as such, feature extraction techniques must be used to generalise and compress the information contained within the data [146, 181], thereby facilitating improved classification. The aim of feature extraction is to reduce the size of the data whilst simultaneously preserving the underlying patterns contained within, thereby producing a more compact and robust version of the original feature space [431]. Approaches vary greatly, from audio specific [269, 287, 65], to simple statistical measures [62, 63], with the appropriateness of a given technique being largely problem and data specific. This point is extremely relevant to features derived from DWT sub-bands, as their use in audio analysis is still in its preliminary phases, and unlike frequency-based approaches, audio specific features have not been thoroughly explored for DWTs [2, 1, 420] .

1.6 Machine Learning

1.6.1 Overview of Machine Learning

Machine learning, which encompasses aspects of statistical learning, data mining, and artificial intelligence (AI), is the process of discovering useful information, associations, or patterns in (generally) vast amounts of data [149], and producing models that can use this information to expedite meaningful decisions. There are a multitude of different machine learning algorithms demonstrated in the literature [180, 382, 94, 127, 88], and

they can be grouped first by their training method (supervised or unsupervised), and then by their output type (classification or prediction), making selection of a suitable paradigm almost entirely problem specific [180]. In supervised learning, the focus is on mapping the relationships between a dependent (output) variable and various independent (input) variables, with the aim of selecting a model that can accurately predict an output based on given inputs [180]. This is achieved by training the model with a set of labelled data, and then testing with new, unknown instances [458]. The output of a supervised learning model is intrinsically linked to the type of problem it solves, be it estimation (i.e. prediction of a numerical output value), or classification (i.e. assigning an instance to one of a set of classes) [302]. By contrast, unsupervised learning aims to unearth relationships in data that has not been labelled: it is concerned with knowledge and pattern discovery [382, 57].

1.6.2 Definition of the Problem

The case of identifying sheep vocalisations can be defined as a supervised learning, classification problem, with one clearly defined class (i.e. sheep vocalisations), and another ambiguous class that is not present in the training data (i.e. every other sound); this can be defined as a one class classification (One-Class Classification (OCC)) problem [406, 284]. One type of model that is suitable for this problem is a Support Vector Machine (Support Vector Machine (SVM)) [429, 427], in particular, an OCC-specific algorithm known as a Support Vector Data Description (Support Vector Data Description (SVDD)) [411, 408].

1.6.3 Phases of Implementing a Machine Learning Solution

The phases involved in implementing a machine learning solution can be described using the plan-do-check-act (PDCA) cycle [275], also known as the Deming Cycle [86] (Fig 1.5). During the planning phase, the scope of the problem is identified and data is collected [447]. In this case, our problem is identifying sheep vocalisations amongst a variety of other sounds, with the training data being collected from both field and experiment recordings, and new instances being obtained directly from the input stream at set intervals. The data is then processed and transformed to accentuate particular characteristics, features are extracted, and the model is trained [180]. The model is then tested with an unknown set of data instances, and evaluated using performance metrics applicable to the problem; this process is then repeated after adjusting parameters, features, models, or some combination thereof [382].

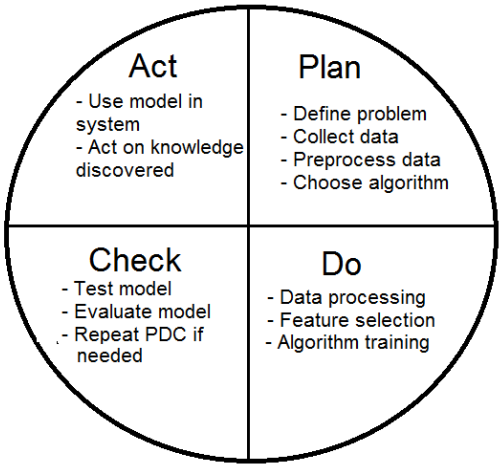

Figure 1.5: A sheep call. The top panel displays the call in the temporal domain (i.e. amplitude at a given time), whereas the bottom panel shows the call in the spectral domain (i.e. the amplitude of each frequency component, from low to high). These images were produced with Audacity sound editing software [12].

1.6.4 Model Evaluation

Finding a practical, optimal solution within such a vast possibility space can be time consuming, and therefore it is common to implement some form of optimisation routine, particularly in regards to model parameters [171]. The comparative difference in a models performance between training and testing data highlights an issue in machine learning known as the bias-variance trade-off [180], which states that the more bias a model possesses (i.e. how well it fits the training data), the less variance it tends to have (i.e. how flexible it is in accommodating new instances) [382]. When selecting and optimising model configurations it is imperative to keep this balance in mind, and techniques such as cross- validation [149] can be used to improve and maintain equity [94].

1.7 Precision Livestock Farming

1.7.1 Overview of PLF

PLF can be defined as the application of process engineering principles to livestock farming (Fig 1.6) [441], particularly the use of sensors to monitor development and welfare, and integrated management systems to aid producers in making informed decisions [442, 355]. In this instance, the aim is to monitor sheep vocalisations in order to detect when vocal

activity is occurring, as this can act as the first point in a wider audio-based surveillance and decision system. As an example, a system such as this could be used to detect distressed or abandoned lambs and send an alert to the producer, facilitating earlier intervention, and thereby reducing lamb losses. As starvation and mismothering is a leading cause of lamb deaths [400, 153, 315, 104], any system that could alleviate this problem would be most welcomed by industry.

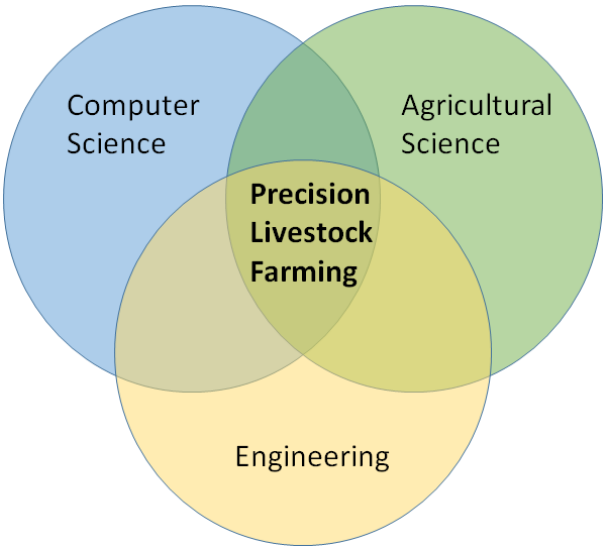

Figure 1.6: A Venn diagram illustrating the intersection of disciplines that encompass PLF.

1.7.2 Acoustic Monitoring in a PLF Setting

Acoustic monitoring provides an accurate and non-invasive way to measure various biological responses, and by extension, welfare states of livestock [125, 107]. Despite the fact that research has demonstrated that acoustic monitor systems can be viable for a number of specific applications in pigs [58, 282, 65, 106, 112, 123, 144, 183, 421], chickens [14, 105, 120, 243, 134, 320], and cows [64, 225], little investigation appears to have been conducted into the suitability of such systems to problems pertaining to sheep. The importance of sheep vocalisations has been an area of active research, and they have been shown to be a quantifiable biological signal [435, 436, 437, 438, 439, 98, 294, 329, 373, 363, 374, 364, 97, 412, 229, 162]; this is encouraging in terms of both the applicability and predicted success of a pattern recognition based approach.

1.8 Objectives of the Research

Fig 1.7 illustrates a simplified overview of the components in the proposed sheep vocalisation detection system (further detailed in Section 3), and whilst it bears at least a passing resemblance to other examples in the literature [64, 65] (Fig 1.8), it is quite innovative in its specific implementation. The use of either DWTs [420, 38, 211] or SVDDs [64, 65] for audio analysis has not been thoroughly explored, and the combination of the two, especially for a sheep-based PLF application, is entirely novel, as is shown in (Section 2).

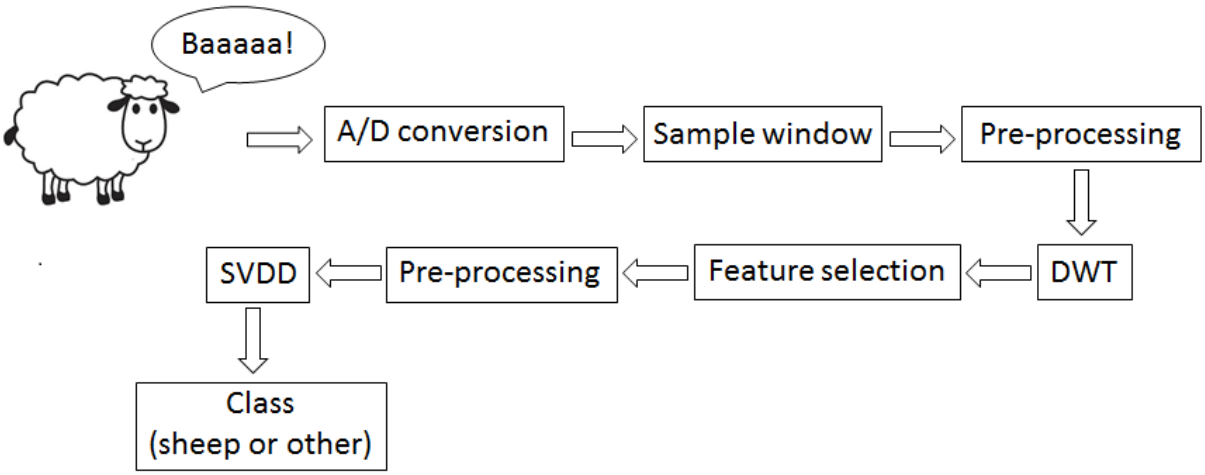

Figure 1.7: An overview of the components in the proposed sheep vocalisation detection system.

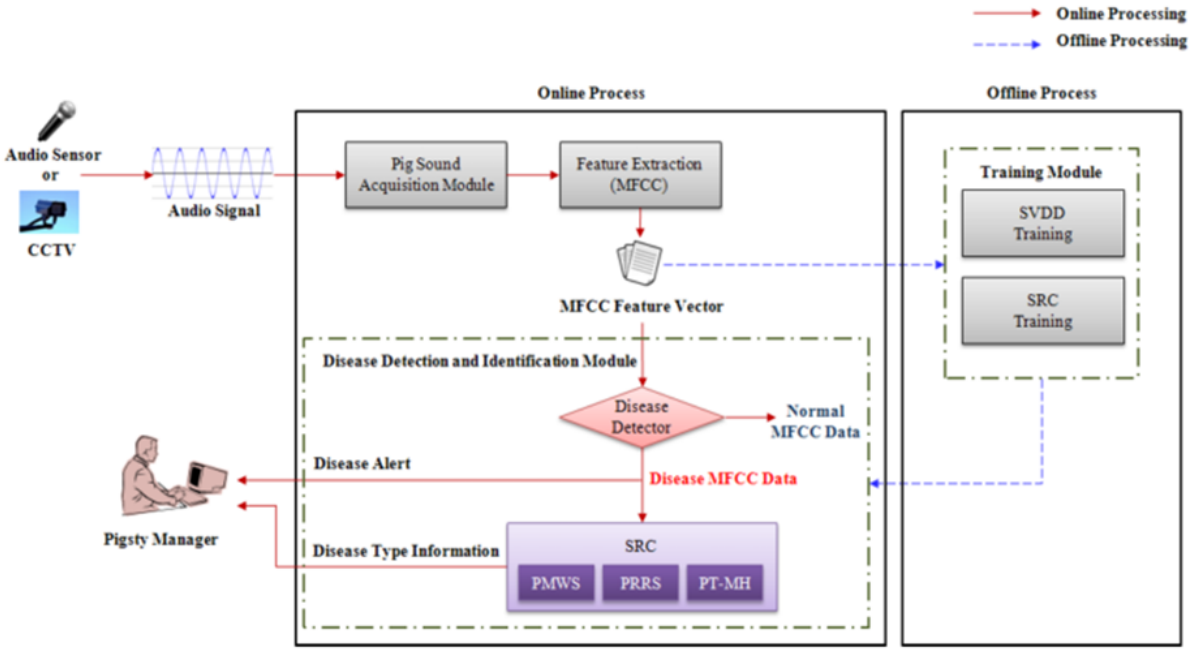

Figure 1.8: An example of a PLF acoustic detection system [65]. This system is used for detecting pig coughs, and diagnosing disease using an SVDD based on Mel-Frequency Cepstral Coefficient (MFCC) features.

The objectives of the project aim to answer several pertinent questions, namely:

1. Is it possible to create software capable of detecting sheep vocalisations in real-time?
2. Can DWTs be used to accurately analysis sheep vocalisations in order to facilitate discrimination by a machine learning model?
3. Can an SVDD be used to successfully detect sheep vocalisations, based on features derived from DWT coefficient sub-bands?
4. Can low-computational statistical features, suitable for a real-time system (e.g. Shannon entropy, mean, standard deviation), provide adequate features for precise discrimination by an SVDD?
5. What effect does distance from the sound source have on matching accuracy for this system?
6. Is it possible to run this application on a mini-computer, such as a Raspberry Pi [345] (Fig 1.9), to allow for affordable prototyping of field nodes (Fig 1.10), or on-collar monitoring devices (Fig 1.11)?

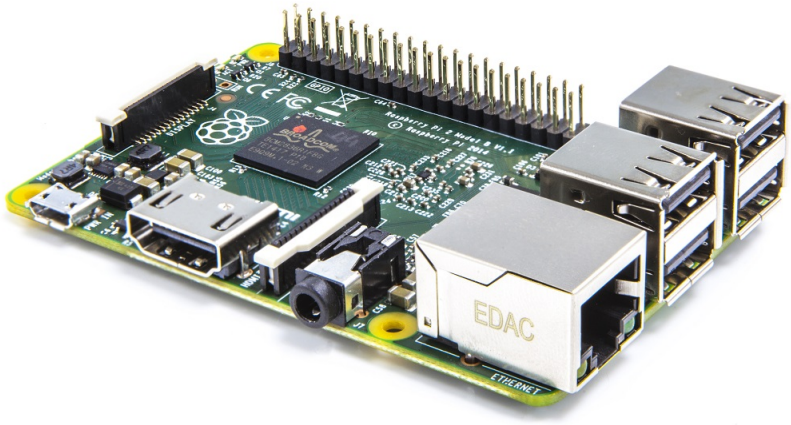

Figure 1.9: A Raspberry Pi mini-computer [345]. These units provide an affordable avenue for prototyping field nodes.

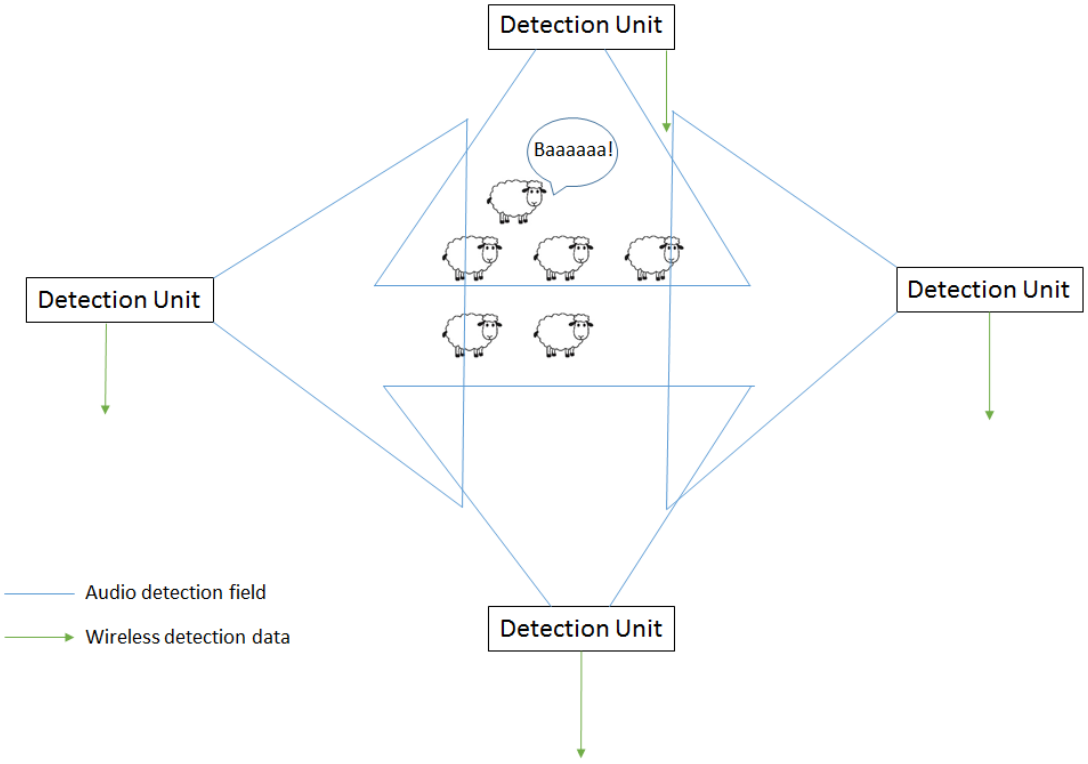

Figure 1.10: A network of field nodes, used to detect sheep vocalisations. Note the overlapping fields of detection (blue). Detection data is sent wireless to a central server, allowing notifications to be sent the producer.

Figure 1.11: A prototype of an on-collar audio monitoring device, developed using a Thingsee [413].

1.9 Structure of the Thesis

Section 2 is comprised of a thorough review of the literature, covering all research fields relevant to the project. Section 3 details the methods used to develop a prototype sheep vocalisation detection system, including the creation of original software. This section also describes several experiments that were conducted to answer the questions posed in (Section 1.8). Section 4 displays the results of the aforementioned experiments, using relevant performance metrics and comparison techniques. Section 5 discusses the results, their implications to the system, whether the objectives of the thesis have been met, and highlights possible future research directions. Section 2.4 summarises the findings of the thesis, and discusses the authors planned future work in this field.

Chapter 2

Literature Review

Given the breadth of the research space explored in order to develop the sheep vocalisation detection system, the literature review will be grouped into the following categories:

- 2.1: Digital Signal Processing
- 2.2: Machine Learning
- 2.3: Precision Livestock Farming
- 2.4: Conclusion

2.1 Digital Signal Processing

2.1.1 Digital Audio

Digital audio is produced by converting an analogue audio signal, using an A/D converter [175], discretising it as a series of data points, representing amplitude at a given point in time [273] (Fig 1.2). Digital audio signals possess two variable characteristics, sampling (i.e. the representation of time), and quantisation (i.e. the measure of amplitude) [308]. The sample rate of a signal is the number of measurements that are taken per second, which defines the bandwidth of the system [390]. The bit rate determines the size of the variable used to store a given amplitude measurement, and as such, the range of values that can be represented [273] (Fig 2.1), which in turn affects dynamic range and the signal to noise ratio [308].

Bit Rate	Range	SNR
8	[−128, +127]	48.16 dB
16	[−32,768, +32,767]	96.33 dB
24	[−8,388,608, +8,388,607]	144.49 dB
32	[−2,147,483,648, +2,147,483,647]	192.66 dB

Figure 2.1: The range of values represented, and the corresponding signal to noise ratio, for common digital audio bit rates [273].

2.1.2 Feature Extraction: Temporal Domain

Feature extraction is the process of generating meaningful, compact representations of the underlying patterns within a data set for use as predictors in a machine learning model [180]. As data may be dimensionally vast, it is often necessary to transform the data into a form that simultaneously reduces the number of features (i.e. the size of the data), whilst still preserving the underlying patterns and information necessary to allow for accurate discrimination [94]. As previously outlined in Section 1.3, digital audio stream data exists in the temporal domain, and whilst this type of information can be useful for generating features such as envelope [420, 333, 326, 304, 301, 238], energy [334, 333, 301] power [334, 333, 390], norm [333], and root mean square (RMS) [390, 238, 304, 334], it only provides an insight into the amplitude of a sound at a given point in time, and saying nothing of the frequencies characteristics that give the sound its character [38, 420].

2.1.3 Spectral Analysis of Audio Signals

In order to analysis the frequency components of an audio signal, it is necessary to apply an algorithm within the discrete Fourier transform family (FFT, STFT etc) [186, 331, 304], yielding a representation that describes the amplitude of frequency bands [186, 390, 331, 304] (Fig 2.2). The resolution of the transform is determined by the window length [390, 304, 238], 18], which also affects the ability to discern between closely spaced frequency components (Fig 2.3) [390]. The Nyquist-Shannon theorem [298, 378] states that for a frequency to be correctly described in the digital domain, the sample rate must be twice as high as the frequency itself, otherwise the frequency may be misconstrued, introducing alias frequencies and harmonic distortion [175] (Fig 2.4), and as such, the sample rate used will also affect the performance of spectral analysis. During analogue to digital conversion, a low-pass filter is applied to frequencies outside the

chosen sample rate, but as an infinitely attenuating filter is impossible to realise, there is a certain level of roll off after the sample rate [175]; for this reason it is advisable to oversample [273].

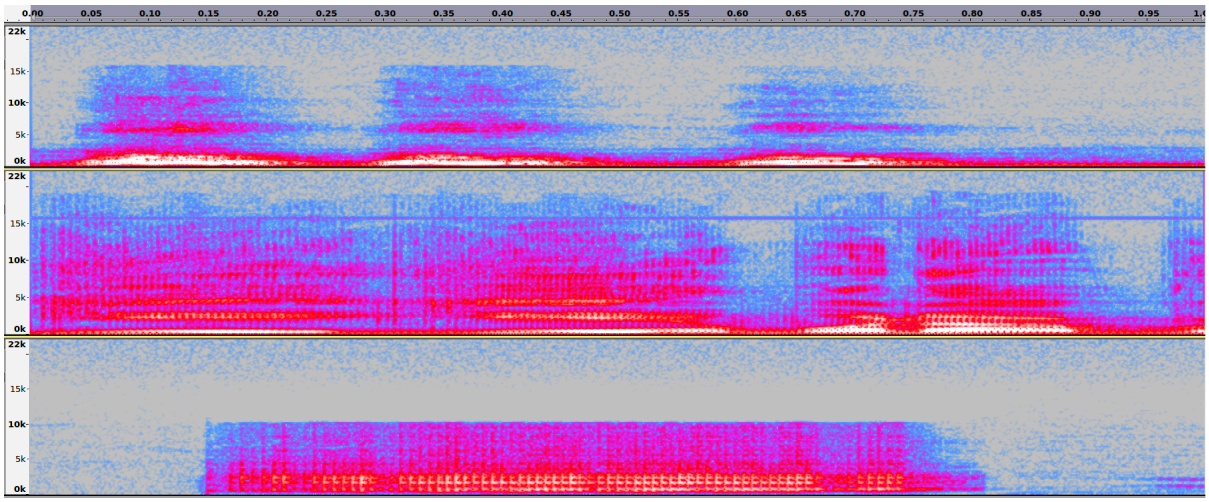

Figure 2.2: A spectrogram (frequency) view of three sounds; the x-axis is time, the y-axis is frequency. The top sound is a dog, the middle is talking, and the bottom is a sheep vocalisation. Note the similarity of activity in some frequency bands.

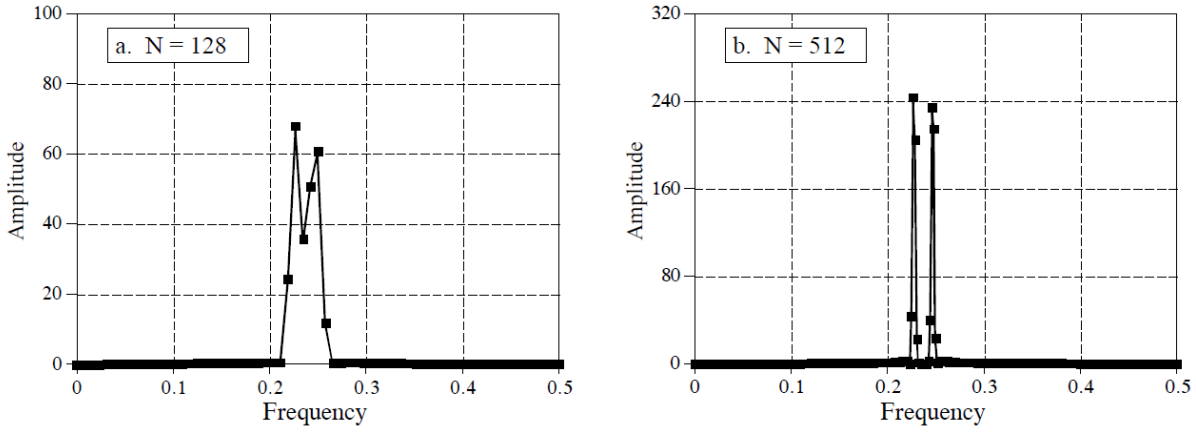

Figure 2.3: Effect of window length on frequency resolution, and the ability to discern closely spaced frequency components [390], where N = window length; note the improved component separation and representation.

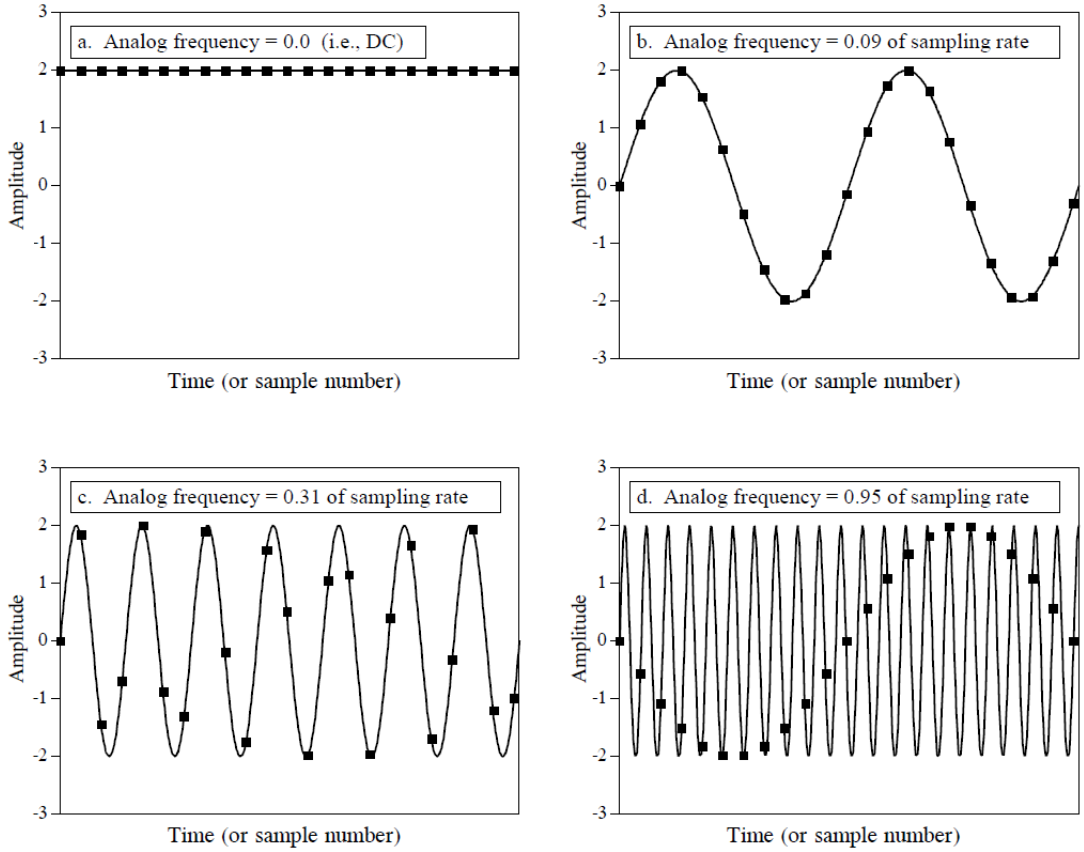

Figure 2.4: Sampling, and the Nyquist-Shannon theorem. Figures a, b, and c illustrate proper sampling. Figure d shows a frequency that is greater than half the sample rate (i.e. the Nyquist frequency), which is not correctly sampled, causing aliasing [390].

2.1.4 Window Functions

Conversion to the frequency spectrum is not without its pitfalls, the most predominant being spectral leakage inherent with applying an algorithm in the discrete Fourier transform family [283]. Essentially, spectral leakage is when information is present in the wrong frequency bins [165], and is caused by the nature of discretisation; by acquiring a finite set of values for a signal, the signal becomes distorted [128]. Spectral leakage cannot be eliminated entirely, but its effects can be minimised by using a window function before applying a discrete Fourier transform [347]. For a given interval, a window function returns finite non-zero values inside, and zero values outside [327]. When selecting a window function, there is always a trade-off between side lobe level reduction, side lobe roll-off rate, and overall frequency resolution [128] (Fig 2.5). Some common windowing function include the Hamming [148], Hann / Hanning [34], Blackman-Harris [152, 297], Dolph-Chebyshev [237], Kaiser-Bessel [192], Gaussian [128], Flat-top [158], and Nuttall [297]; each function has its own strengths and weaknesses (Fig 2.6) (Fig 2.7) [414, 327, 347, 152, 158, 152].

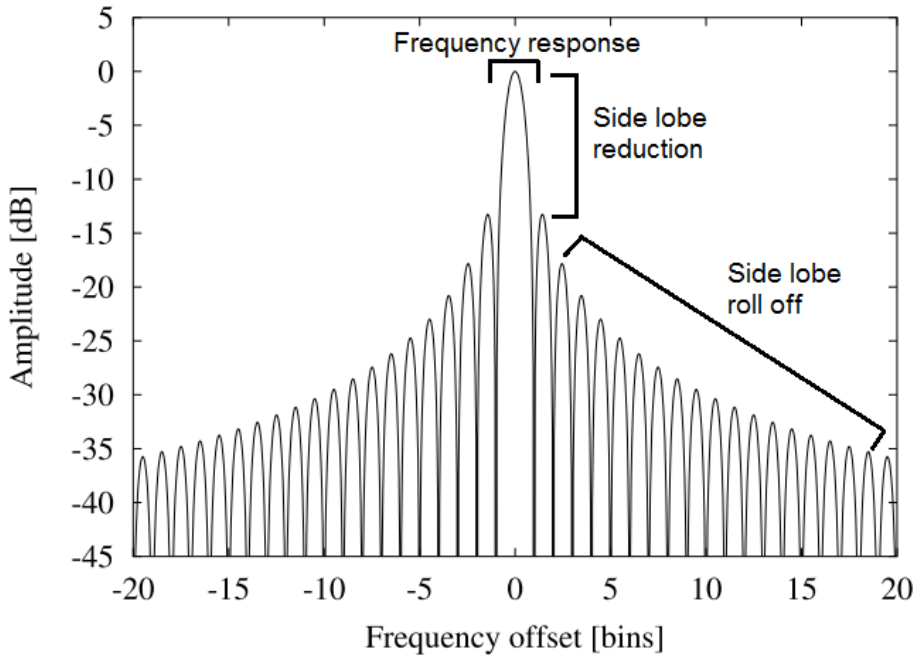

Figure 2.5: Trade-off between side lobe level reduction, side lobe roll-off rate, and overall frequency resolution [390].

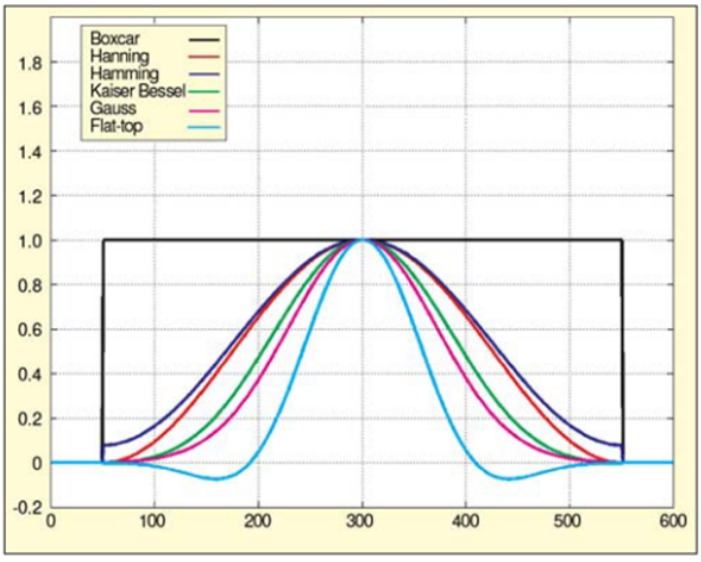

Figure 2.6: The shape of 6 common window functions. Note, Boxcar is the equivalent of no window function [128].

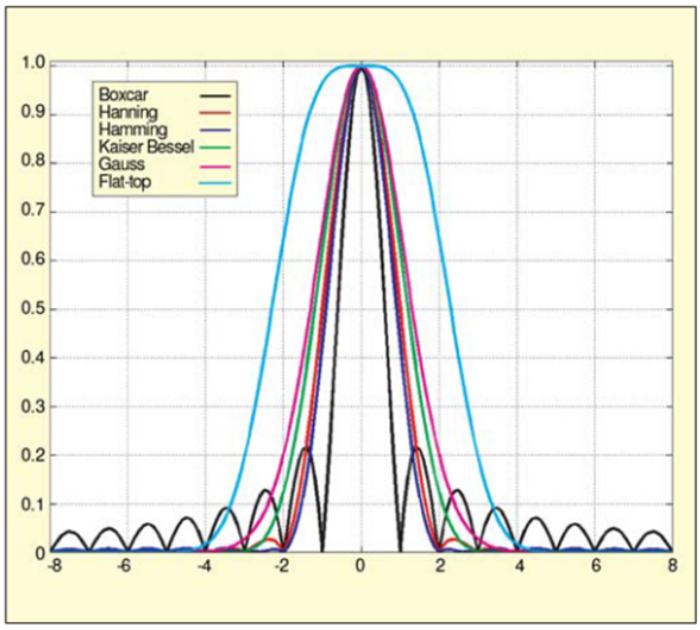

Figure 2.7: DFTs of 6 common window functions [128]. Note the difference in resolution and side lobe behaviour.

2.1.5 Feature Extraction: Spectral Domain

Frequency components generally contain more useful information relevant to audio pattern recognition [38, 420, 342], but as the output of a Fourier transform tends to be dimensionally large [431, 463], there is a need to apply further feature extraction and dimensionality reduction techniques [319, 127]. For this reason, an abundance of audio feature extraction methods have been proposed in the literature [228, 431, 319, 127], each with its own set of optimal applications. Some examples of frequency-based approaches include mel-frequency (MFCCs) [269, 287, 65], power-normalized (Power-Normalized Cepstral Coefficient (PNCC)s) [200, 201], and gammatone frequency (Gammatone Frequency Cepstral Coefficient (GFCC)s) [379, 3] cepstral coefficients, zero crossing peak amplitude (Zero Crossing Peak Amplitude (ZCPA)) [47, 48], perceptual linear prediction coefficients (Perceptual Linear Prediction Coefficient (PLP)s) [164, 163, 305], spectral centroids [142, 219, 319], and those based on human auditory physiology and perception [394]. Although spectral analysis is very useful in audio pattern recognition, it does suffer from a lack of temporal localisation [38, 420, 342], and as such, there has been sufficient interest in approaches that can incorporate information from both domains [301].

2.1.6 The Discrete Wavelet Transform

The DWT allows for the efficient computation of the amplitude of a signal that is simultaneously represented in both the time and frequency domains [301]. The most basic DWT, the Haar DWT, was developed by Alrd Haar in 1909 [147], but more recent developments by Ingrid Daubechies, beginning in 1988 [79, 80], have seen a resurgence in wavelet-based research. There are numerous DWT algorithms, most notably the Haar [147, 244, 330], Daubechies [79, 80], Dual-Tree Complex [242, 203, 376], Stationary [166, 122, 10], Harmonic [291, 50], and Wavelet Packet Tree [47, 9, 29] transform. DWTs have been successfully applied to problems in a diverse array of fields, including signal compression [454, 195, 63, 386, 16, 341, 434, 145], audio [2, 1, 38, 76, 326, 342, 338, 420, 211, 295], image [206, 204, 395, 266, 70, 82, 62, 385, 417, 205, 330, 301, 177], EEG [335, 289, 456, 311, 81], and MRI analysis [460, 461, 462], fault detection [166, 359, 224, 332, 216], and database approximate query processing [129, 130, 131, 52]. Wavelets have also been combined with frequency-based feature extraction techniques, such as wavelet-based MFCCs [1, 2], and Mel-scaled discrete wavelet coefficients (MFDWCs) [138, 109].

As an overview of how DWTs operate, a DWT decomposes a discrete signal into two sub-signals, or sub-bands, that are half the length of the original, one being a running average or trend, commonly referred to as approximation coefficients, and the other being a running difference or fluctuation [77], known as detail coefficients [434]. Approximation sub-bands are produced by a low-pass filter, whereas detail sub-bands are determined by a high-pass filter [301], together forming a quadrature mirror filter [133, 267] (Fig 2.8). The DWT provides high time and low frequency resolution for high frequencies, with the inverse being true for low frequencies, and in this way, it exhibits similar time-frequency resolution to that of human hearing [420], which is a theoretically advantageous trait for audio analysis.

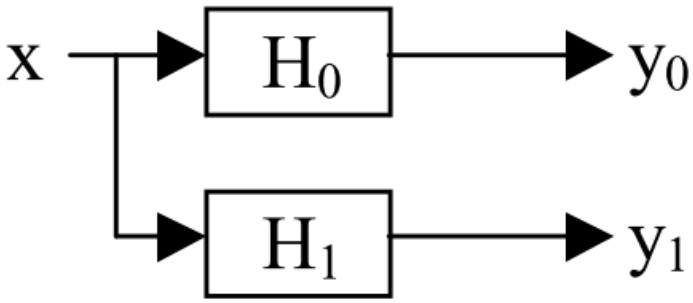

Figure 2.8: Quadrature mirror filter bank [38], where x = input signal, H0 = low-pass filter, H1= high-pass filter, y0 and y1 = output signals.

DWTs can be performed at multiple decomposition levels by successively applying the algorithm to the approximation coefficient sub-band calculated at each level, utilising a cascading filter bank scheme (Fig 2.9) [38], also known as Mallats algorithm [244, 397]. Each level will experience a down-sampling by 2, to produce sub-bands of half the length of the previous level (Fig 2.10) [454]: each level will also display a consecutively increasing loss of energy [434]. Due to the nature of the DWT, there is dyadic restriction on the length of the signal [301, 206, 326], with the number of decomposition levels also constrained to a length defined by the number of taps (i.e. the signal at a given level cannot be shorter than the number of taps) [434]. The output of a J-level DWT will be J+1 sub-bands, consisting of J-detail sub-bands, and a single approximation coefficient sub-band, produced by the last level of the DWT [434, 38, 179, 417].

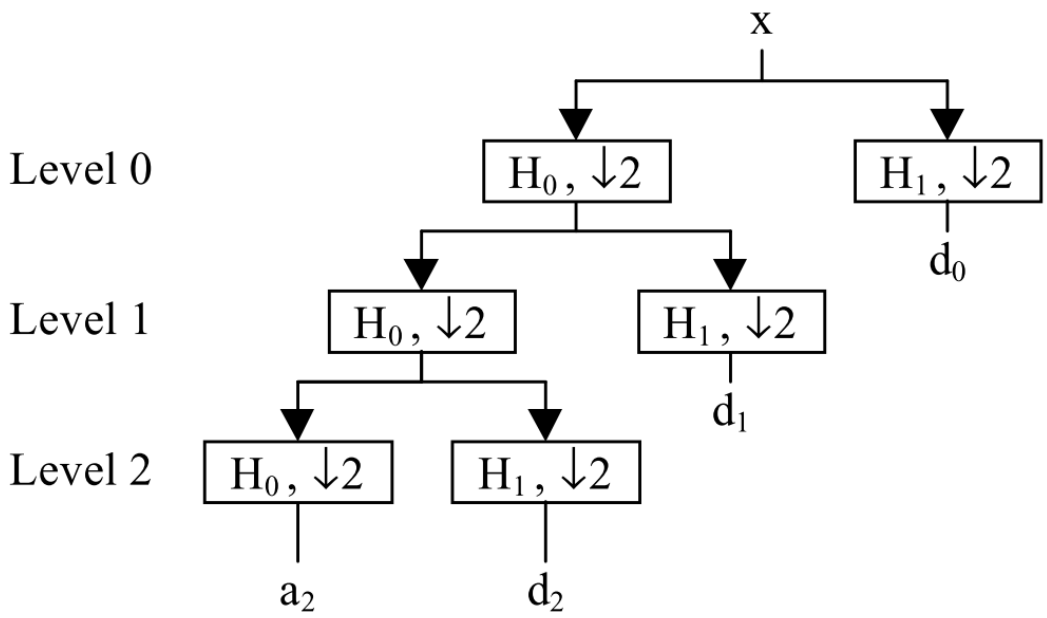

Figure 2.9: Quadrature mirror filter bank [38], where x = input signal, H0 = low-pass filter, H1= high-pass filter, y0 and y0 = output signals.

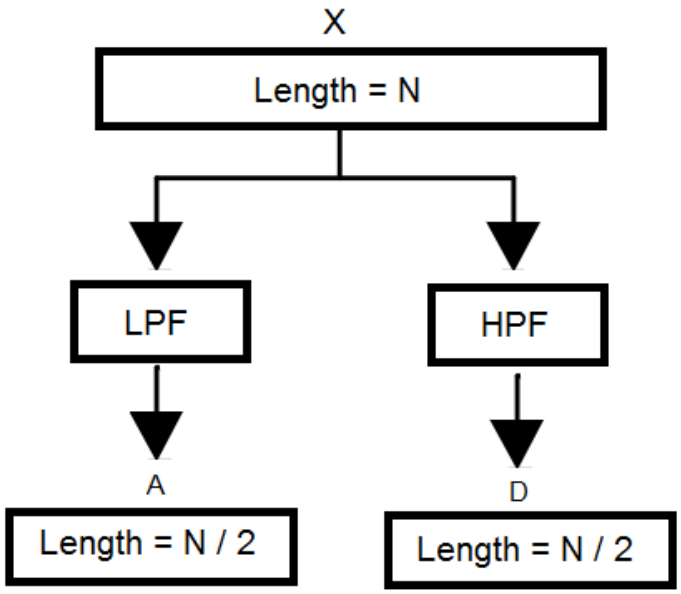

Figure 2.10: Down-sampling, where X = original signal, LPF = low-pass filter, HPF = high-pass filter, A = approximation coefficient sub-band, D = detail coefficient sub-band, and N = length of signal.

2.1.7 Daubechies Discrete Wavelet Transforms (DDWT)

DDWTs are a set of orthogonal wavelet transforms, each characterised by the number of vanishing moments they possess, with more moments allowing for the representation of complex functions using a sparser set of wavelet coefficients [79, 38, 354]. DDWT algorithms are commonly referred to by either the number of taps they possess (e.g. D4 / DAUB4 has 4 taps) [179, 326], or by the maximal number of vanishing moments (e.g. db2 has 2 vanishing moments) [145, 338] - the naming schemes are used interchangeably in the literature. The DDWT shares many similarities with the Haar (point in fact, the Haar is a D2 / db1 DDWT) [145], in that running averages and difference are computed using scalar products with scaling signals and wavelets [434], but the difference lies in how these scaling functions and wavelets are defined [145, 301]. Essentially, the DDWTs use a progressively increasing number of values from the signal, producing wavelets and scaling functions with longer supports, which in turn provide a substantial improvement to its capabilities [434]; (Fig 2.11, 2.12, 2.13, 2.14) show an example of a D4 DDWT [434]. The scaling functions for the standard DDWT algorithm are predefined, with the wavelet coefficients being derived by reversing the order of the scaling values, and then reversing the sign of every second value [38, 434]. Each successive algorithm in the DDWT family (e.g. D2-D20) uses more taps, has more vanishing moments, and therefore incurs higher computational overheads [38]. DDWTs have become increasingly popular since their

inception, and have been successfully applied to a variety of applications and problems [420, 76, 145, 301, 1, 300, 63, 326, 179, 434, 417, 205, 38].

$$\alpha_1 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad \alpha_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad \alpha_3 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad \alpha_4 = \frac{1-\sqrt{3}}{4\sqrt{2}}.$$

Figure 2.11: D4 scaling function values (i.e. taps) [434].

$$\beta_1 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \quad \beta_2 = \frac{\sqrt{3}-3}{4\sqrt{2}}, \quad \beta_3 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad \beta_4 = \frac{-1-\sqrt{3}}{4\sqrt{2}}.$$

Figure 2.12: D4 wavelet function values (i.e. taps) [434]. Note the relationship with the scaling values.

$$\mathbf{V}_m^1 = \alpha_1 \mathbf{V}_{2m-1}^0 + \alpha_2 \mathbf{V}_{2m}^0 + \alpha_3 \mathbf{V}_{2m+1}^0 + \alpha_4 \mathbf{V}_{2m+2}^0$$

Figure 2.13: D4 DDWT approximation coefficient formula [434].

$$\mathbf{W}_m^1 = \beta_1 \mathbf{V}_{2m-1}^0 + \beta_2 \mathbf{V}_{2m}^0 + \beta_3 \mathbf{V}_{2m+1}^0 + \beta_4 \mathbf{V}_{2m+2}^0.$$

Figure 2.14: D4 DDWT detail coefficient formula [434].

2.2 Machine Learning

2.2.1 Overview

Machine learning models describe the relationships between features present in a set of data [302], attempting to map and understand the patterns that designate instances into a particular class [94], or to predict a given value based on feature information [382]. Supervised learning models achieve this by studying the sequences present in a set of training data, then using this information to extrapolate to new instances [180]. In classification tasks, the goal is to accurately classify new instances into one or more predefined classes, based on the values of the feature vector [458]. There are many examples of supervised learning classification models in the literature, but some of the most notable and widely used families of algorithms include support vector machines (SVMs) [8, 39, 73, 430, 365], artificial neural networks (Artificial Neural Network (ANN)s) [325, 353, 356, 156, 110, 443, 272, 172], hidden Markov models (Hidden Markov Model (HMM)s) [398, 20, 21, 22, 23, 24, 127, 361], and Gaussian mixture models (Gaussian Mixture Model (GMM)s) [361, 254, 87, 127]. Each family of algorithms contain many variations of the overall model, for example, ANN can have different neurons (e.g. logistic sigmoid [75], hyperbolic tangent (tanh [27], radial basis function (Radial Basis Function (RBF)) [172], rectifier [221, 135], softplus [95, 290], hard tanh [69]), architectures (e.g. feed-forward [402, 280, 115], recurrent [360, 401], recursive [13, 268], convolutional [403, 257, 66, 217], counter propagation [349, 157, 55, 253]), and learning approaches (e.g. stochastic gradient descent [41, 40, 403], momentum-based gradient descent [339, 293], Hessian optimisation [293, 33], particle swarms [265, 459, 261], and deep learning [261, 27, 403, 404]); as the optimal configuration is highly problem dependent [309, 100, 194, 137, 384, 151, 324, 222, 154, 213, 168, 452, 399] a comprehensive review of the research is absolutely necessary.

2.2.2 Feature Generation and Dimensionality Reduction

Not all recorded data values prove to be useful in explaining the associations necessary for a model to classify new instances [180]. This is especially true when instances contain a high number of attributes, as is commonly the case in bioinformatics data [422], and acoustic signal processing applications [31, 235]. Most classification models do not perform well within a high dimensionality feature space [382], and even those that are better able to cope with this problem, such as SVMs [149], still tend to perform better within a reduced space [382]: this is known as the curse of dimensionality [94]. In order to reduce the feature space, two non-exclusive approaches can be used, namely feature generation [391, 31] and feature selection algorithms [26, 146, 181]. Depending on the structure of

the data, it may be possible to condense the features into a more succinct form, for example, the output produced by a DDWT [79, 80] can be generalised using measures such as Shannon entropy (Fig 2.15) [166, 359, 393, 179], mean value [420], energy [206, 300], standard deviation [420], MFCCs [1, 2, 314, 310, 278, 387], and percentile thresholding [177, 38]. Specific algorithms for feature selection can be used, such as filter methods (e.g. principal component analysis (PCA) [317, 448, 449, 169, 189], linear discriminant analysis (LDA) [117, 259], independent component analysis (ICA) [161, 191, 71, 72]), wrapper methods [208, 26], and weighting methods [37, 366, 232, 453]. The drawback of these algorithms is that because of their high computational time, implementation schemes require additional processes to be compatible with the restrictive environment of real-time applications.

$$E_j = -\sum_k E_{jk} \log E_{jk}$$

where E_{jk} is defined as:

$$E_{jk} = \left| D_j(k) \right|^2$$

Figure 2.15: Shannon entropy [359].

2.2.3 Data Normalisation

Before data can be provided to the machine learning algorithm, it is often necessary to normalize or scale the data. Data normalisation can be used to standardise the data set, reducing the significance of outliers, using such techniques as logarithmic transformation [382], z-scores [210], or zero-mean normalisation [382]. Various scaling approaches, such as autoscaling [178], range [388], pareto [102], and vast [197] scaling, may also be used to accentuate or diminish the relevance of outliers, based on the statistical distribution of the data [422]. Depending on the classifier used, it may be necessary to normalise the data values to conform to a certain range of values, for example [0, +1] or [-1, +1], and min-max normalisation (Fig 2.16) can be used to fulfil this purpose [382]. This type of normalisation is mandatory for some families of models, such as ANNs [382], and has been shown to improve the performance of others, particularly SVMs [377].

$$D' = \left(\frac{D(i) - min(D)}{max(D) - min(D)}\right)\big((U - L) + L\big)$$

Figure 2.16: Min-max normalisation, where D = normalised data set, D = original data set, min = the minimum value, max = the maximum value, U = upper normalisation bound, and L = lower normalisation bound [382].

2.2.4 Testing

Once a model has been built, it must be tested to determine its accuracy and overall performance, and the most reliable way to do this is to use data that was not present in the training data [302]. The reason for this, is that if the same data is used to both train and test a model, it will likely perform quite well, as the model has been fit to the data, and no unknown instances are present to truly test its performance [149]. To combat this problem, it is common to either keep a separate set of testing data when training, or to use a form of cross-validation [207], or bootstrapping [99]. There are numerous approaches to cross-validation, such as percentage splitting [382], leave-one-out cross-validation (acLOOCV) [149], and k-fold cross-validation [94]. The premise of each scheme remains the same: use a portion of the data to train the model, another to test it, and in the case of LOOCV and k-fold, repeat with a different subset of data, then average the results of the calculated performance metrics [382]. Bootstrapping operates in a similar fashion to cross-validation, except that subsets are formed through random sampling with replacement [207]. In OCC problems, care must be taken when using cross-validation, as only one class is present in the training data, it is easy to create a model that matches all instance, but it will most likely suffer from extremely high variance, and will perform poorly when encountering instances from the negative class [171].

2.2.5 Performance Metrics

Depending on the nature of the classification problem, it is common to look at numerous metrics when ascertaining the fitness of a model, the most basic of which are true positive (TPR), false positive (FPR), true negative (TNR), and false negative rates (FNR) [382]; this information can be captured by a confusion matrix (Fig 2.17) [121]. Other performance metrics can be calculated based on these rates, such as accuracy, precision, and geometric mean [212]; Fig 2.18 details the equations for all metrics presented. In tests of a predictive nature, performance metrics designed for classification problems, such as those

just outlined, are unsuitable for analysis. Metrics such as mean squared error (Mean Squared Error (MSE)), and mean absolute error (Mean Absolute Error (MAE))(Fig 2.19 measure the overall error produced by differences in prediction, and can be useful in assessing a model's fitness in prediction problems [180]. The performance of a model will change depending on the parameters chosen, and generally speaking, there is a trade-off between different measures that relate to the bias-variance trade-off (e.g. TPR and FPR), as model parameters are adjusted [180]; this can be captured by a receiver operating characteristic (ROC) curve [139] (Fig 2.20). A ROC is a graphical representation of the TPR / FPR trade-off, and can include multiple models, so as to allow direct comparison of these metrics between models with different settings [149]. The importance of each measure in determining the suitability of a model, and the balance required between opposing metrics, is problem dependent [180, 382]; there is no free lunch in statistics [450].

		Predicted	
		Negative	Positive
Actual	Negative	a	b
	Positive	c	d

Figure 2.17: A confusion matrix [382]. Many performance metrics are derived from these aggregate measures.

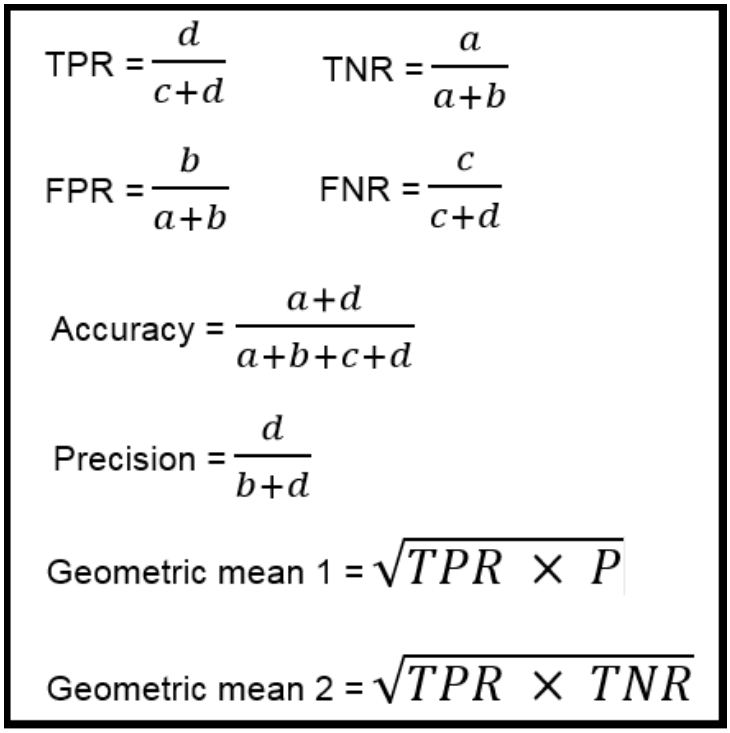

Figure 2.18: Equations for TPR, FPR, TNR, FNR, accuracy, precision, gMean1, and gMean2 [382].

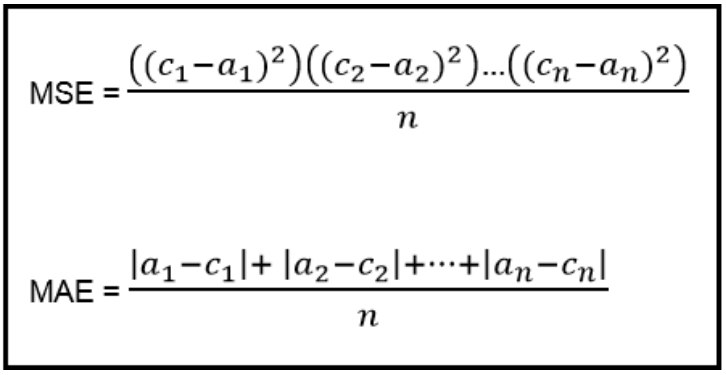

Figure 2.19: Equations for MSE and MAE, where a(i) = the actual value, c(i) = the predicted value, and n = number of test instances [382].

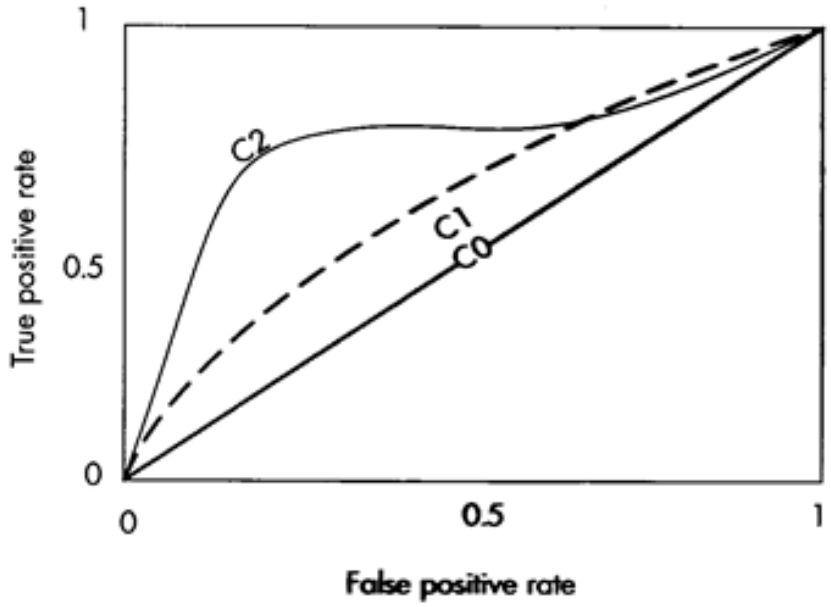

Figure 2.20: Receiver Operating Curve (ROC) [382], with the X-axis denoting FPR, and the y-axis showing TPR. C0 represents the results of random guessing, and therefore defines the baseline for model comparison. The best model possible would exist in the upper-left corner (i.e. TPR = 1 and FPR = 0). C1 and C2 are different parametric models, with C2 performing better than C1. Note the trade-off between TPR and FPR that exists for each model.

2.2.6 Novelty Detection

Novelty detection [32], also known as outlier detection [348], or concept learning [184], is concerned with recognising new data instances that do not fit into one of the classes defined in the training data (positive instances), and assigning them to a negative class [254]. Generally, the negative class is not present in the training data (positive-only), or is not properly sampled, meaning its boundaries are defined by the positive class [409]. Novelty detection approaches are used when a negative class is too broad or hard to define [199], such as in audio detection (i.e. the negative class is every other sound) [296, 262, 182], or when it would be prohibitively expensive to represent the negative class, such as in machine malfunction detection [185]. Other examples of novelty detection include radar target detection [48, 61], hand written digit recognition [407, 223, 231, 409], fault detection [77, 78, 202, 375], document classification [248, 249, 59], and credit card fraud transaction identification [111, 87, 198].

2.2.7 One-Class Classification

When the positive class is singular, and the classifier algorithm only uses examples of this class during training, the problem is said to be a one-class classification problem [284, 406, 410]. OCC aims to create a decision boundary that simultaneously accepts as many of the positive (or target) class instances as possible, whilst minimising the chance of accepting negative instances [199]. This is a challenging problem, as only one side of the decision boundary can be defined, and it is difficult to know which features will produce the optimal separation [199]; for this reason, OCC problems generally require more training data than multi-class problems [406]. Many OCC approaches have been proposed in the literature, particularly SVM-based algorithms, such as the SVDD by Tax and Duin [408, 411], the methods outlined by Scholkopf et al [371], the One-Class SVM (One-Class Support Vector Machine (OSVM)) [249, 410], Support Vector Mapping Convergence (Support Vector Mapping Convergence (SVMC)) [457], and Positive Sample Only Learning (Positive Sample Only Learning (PSoL)) [440] algorithms. Other techniques have been explored, such as Artificial Neural Networks (ANNs) [247, 249], modified C4.5 decision trees [340, 85, 227], hidden Markov models [266, 464], and Bayesian networks [323].

2.2.8 Support Vector Machines

The use of support vectors in machine learning were first introduced by Vapnik and Lerner in 1963 [429], and further defined in 1964 [427], sparking great interest in the research community. The basic model was improved by numerous developments, such as the introduction of large margin hyperplanes [74, 93], slack variables [389, 28, 73], and kernels [8, 328, 433], until a version close to the current form was introduced in 1992 [39]. The standard soft margin SVM was proposed by Cortes and Vapnik in 1995 [73], and extended by Vapnik the same year [430]. Few major improvements have been made to the model since then, but some notable developments include the first rigorous statistical bound on the generalisation of hard margin SVMs [17, 380], as well as statistical bounds on the generalisation of soft margin algorithms [381].

As an overview, SVMs operate by creating a separating hyperplane, a decision boundary that attempts to maximise the division between classes [180, 382]. The position of the hyperplane is defined by its margins, delineated by the closest instances to the boundary: the support vectors [428] (Fig 2.21). For some training data, a perfectly separating hyperplane may not exist, or if it does, it may overfit the data, leading to poor performance on new, unknown instance [302]. For this reason, a slack variable (C) is present in

SVMs, allowing the width of the margins to be adjusted, thereby allowing some control of the bias-variance trade-off of the model [180] (Fig 2.22). As many data sets will not be linearly separable, kernels are incorporated to allow for non-linear decision boundaries (Fig 2.23), whilst reducing the higher computational complexity produced by enlarging the feature space [302]. Kernels come in many forms, such as polynomial, sigmoidal, and Gaussian RBF [172] 2.24. SVMs have been shown to perform admirably when empirically compared to other supervised learning methods [49, 277], as well as when applied to real world problems, such as animal sound detection and species recognition [11, 108], freeway incident detection [60], object recognition [276, 306, 35, 45, 352], text recognition and categorisation [188, 187, 223, 96, 92], and bioinformatics applications [89, 126, 173, 174, 316, 465].

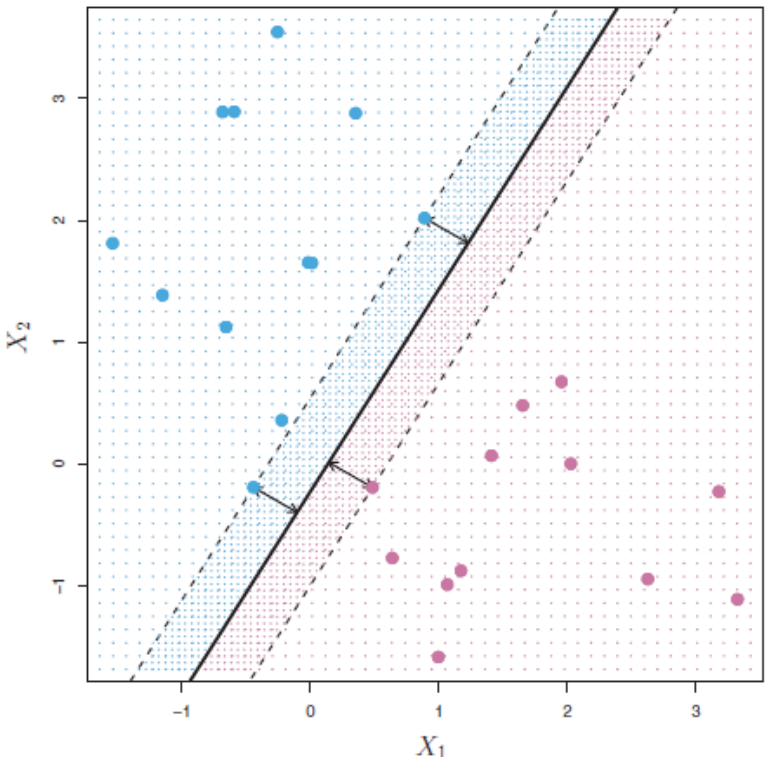

Figure 2.21: An example of a linear SVM, with each colour representing a different class present in the data. Note the linear hyperplane, and the position of the margins based on the nearest instances (support vectors) [180].

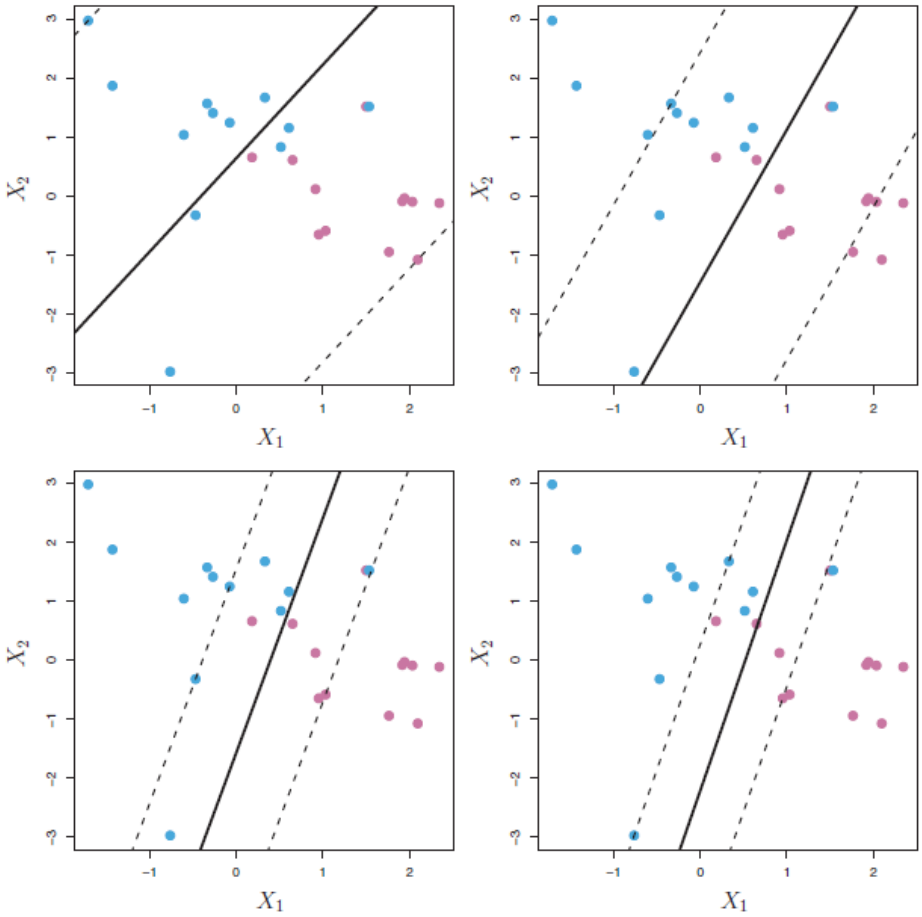

Figure 2.22: An SVM with different values for C. The largest value of C is in the top left panel, with descending values from the top right, bottom left, to bottom right. Note the difference in margin width and the number of allowed violations [180].

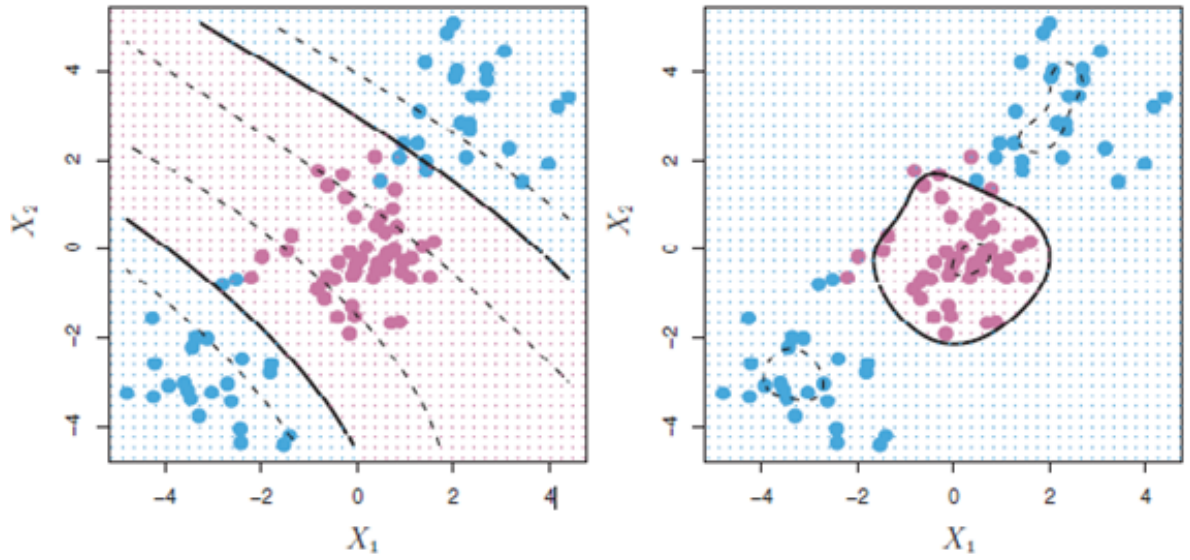

Figure 2.23: Non-linear decision boundaries, formed by an SVM. The left panel shows a polynomial kernel of degree 3, the right shows an RBF kernel; both are using the same data [180].

$$K(x_i x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

where d is a positive integer (i.e. polynomial of degree d).

$$K(x_i x_{i'}) = \tanh K(x_i\ x_j + c)$$

where $\kappa > 0$, $c < 0$.

$$K(x_i x_{i'}) = \exp(-y \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

where y is a positive integer.

Figure 2.24: Standard SVM kernels [171]. The top equation is polynomial , the middle is sigmoidal, and the bottom is RBF.

2.2.9 Support Vector Data Description

To achieve greater accuracy and efficiency using SVMs to solve problems of an OCC nature, various approaches to further redefine a non-linear decision boundary have been suggested in the literature [411, 371, 457, 248, 411, 408]. The SVDD [408, 411], an extension of the work in [367], aims to create a spherical decision boundary with minimal volume (or radius) around instances of the target class, encapsulating the class in the feature space [408] (Fig 2.25). Other methods tend to use a probability density estimation approach, whereby the distribution of the target class is extrapolated from the training data [371]. As the distribution of many data sets will not be perfectly spherical in nature, kernels can be incorporated into the algorithm, allowing for a less rigid hypersphere, and therefore improving accuracy and flexibility [408]. As highlighted by various authors [51, 54], optimisation can be an issue for SVDDs, leading to Chang and Lin, authors of the LIBSVM library [55], treating it as a convex optimisation problem [56]. SVDD has been shown to perform well under experimental conditions, producing commensurate results to other methods for OCC problems [411]. SVDD has been successfully applied to a wide range of real world OCC problems, such as the detection of pig wasting diseases [65], and cow oestrus [64], via acoustic analysis, face recognition [226], pattern denoising [312], and anomaly detection [15].

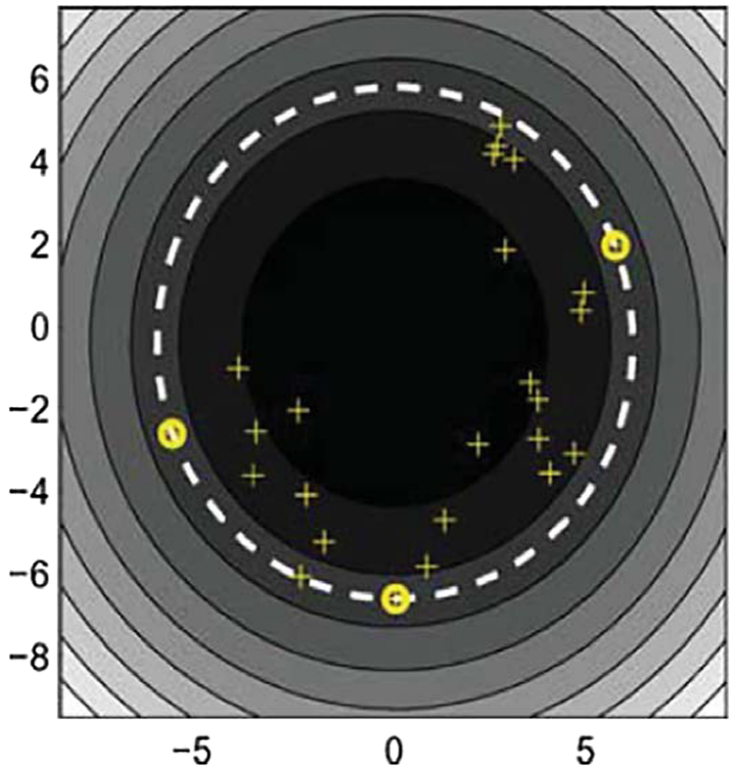

Figure 2.25: An example of an SVDD [411]. Note the tight spherical decision boundary around the positive class instances.

2.3 Precision Livestock Farming

2.3.1 Overview

The trend towards single species intensive farming [336, 36, 136, 346, 103], coupled with the refinement of consumer preferences, both in terms of welfare and consumption [236, 279, 358, 432, 446, 234, 281, 118, 119, 143, 418], has created a dilemma for the modern farmer; there is less time to observe individual animals, yet there are more stringent production targets for each animal [125]. Through the use of integrated systems, it is possible to gain a much more detailed overview of important health and economic indicators, leading to a more robust and efficient livestock enterprise [125, 441, 442, 30]. Such systems utilise sensors monitoring various biological processes, extraction of relevant features, statistical models to define input meaning, and decision support systems to provide feedback on traits and conditions of interest [6, 442, 7, 114, 392, 124] (Fig 2.26). PLF systems can employ sensors that discretise such inputs as image [271, 415, 190, 423, 251, 258, 370, 445, 444, 91], sound [451, 396, 167, 350, 123, 421, 112, 105, 64, 65, 282, 14, 107, 106, 243, 255, 134, 320, 252, 424, 426, 425, 58, 183, 144, 225, 120], odour [321, 383, 322, 299, 90, 46], temperature [6, 4, 215, 318, 196, 132, 170, 240], weight [419, 25, 113, 218], or magnetic resonance (MRI) [18, 19, 372, 209, 67, 68, 274, 405, 246]. Sensors have been used to monitor many different biological processes and production traits, including weight [419, 25, 368, 369, 271, 113, 218, 445, 120], oestrus [101, 83, 64, 239, 225], conformation [270, 423, 18, 116, 140, 141], disease [303, 240, 292, 83, 282, 112, 65, 58, 424, 426, 425, 183, 144, 239], movement [170, 256, 318, 357, 150], and other physiological factors [170, 196, 132, 252]. PLF should be viewed as an augmentation of the knowledge and experience of producers, rather than a replacement [30, 124, 125], providing reliable, up-to-date, quantifiable measures of characteristics and states [441].

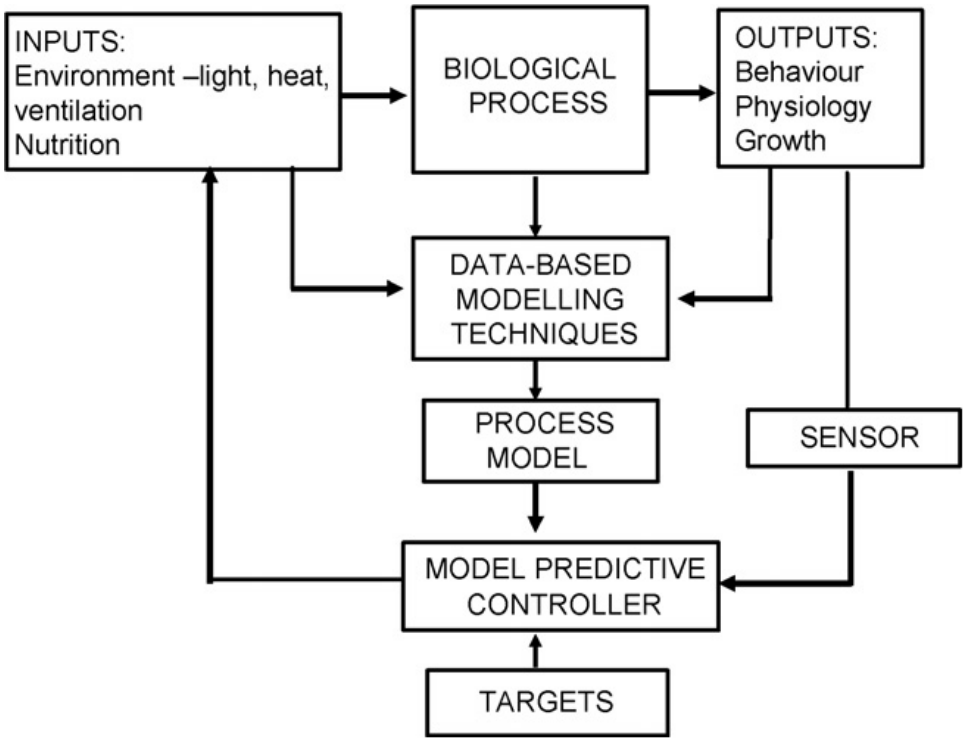

Figure 2.26: An overview of an integrated PLF system [5, 7].

2.3.2 Acoustic Monitoring

The monitoring and analysis of livestock vocalisations has been an area of extensive research [451, 396, 167, 350, 123, 421, 112, 105, 64, 65, 282, 14, 107, 106, 243, 134, 320, 252, 424, 426, 425, 58, 183, 144]. Acoustic monitoring provides an accurate and non-invasive way to measure biological responses, and by extension, welfare states, of livestock [125, 107, 30]. By continuously monitoring acoustic activity in real-time, PLF systems can detect sounds of interest as they occur, with modern systems generally using acoustic feature extraction techniques which feed into machine learning classification models [58, 282, 65, 64, 107]. The use of machine learning and data mining techniques has become quite common in agricultural applications [313, 343, 344, 260, 160, 250, 285, 286], and they are pivotal in creating a reliable automated surveillance system. By training components to detect different sounds, meaningful decisions can be made by the system, such as determining if livestock are suffering from a certain disease [350, 421, 112, 65]. In the livestock production field, acoustic detection research to date has mainly focused on pigs, chickens, and to a lesser extent, cows [30]; there is a distinct lack of sheep-based applications. It must also be noted that the vast majority of research has focused on indoor applications, whereas the target deployment for the system created for this thesis is outdoor: each domain has its challenges and considerations.

2.3.3 Pigs

There are numerous examples in the literature of applications involving the detection and diagnosis of respiratory infections in pigs [58, 282, 65, 425, 112, 106, 107, 350, 421, 183, 144], as not only are they both a major welfare and economic issue [65, 112], but the acoustic properties of pig coughs have been shown to contain information pertaining to emotional state [451, 167, 252] and the cause of the cough [350, 421, 112, 65]. Research has focused on detection and classification of coughs in an offline capacity [112, 58, 144, 183, 425], in real-time [65, 282, 106, 107], using DSP approaches [106, 107, 112, 183, 144, 425], utilising machine learning models (e.g. ANNs) [58, 282, 65], incorporating localisation algorithms [106], and using speech-recognition approaches, such as MFCCs and SVDDs [65]. Although there has been ample progress in this field, a demonstrable commercial system has yet to be developed, further illustrating the point raised by [107, 125, 441], that PLF has suffered from a disparity between research and producer amelioration; it can be hard to make the leap from the lab to the farm.

2.3.4 Chickens

Acoustic analysis in chicken production has been an active area of study, and can be loosely grouped into incubation / embryonic monitoring, and vocalisation monitoring [30]. In chicken production, eggs are hatched in incubators, but a large variation in hatching time is often observed, which can lead to poor health outcomes for early hatchers [416]. To solve this problem, a system was developed by [14] which could detect the sound of a chick pipping (i.e. breaking the shell), thereby providing feedback on hatching rates and progression. The authors of [105] developed a system to analyse chicken embryo sounds in real-time, allowing a producer to more accurately monitor different incubation stages without disturbing the process. The importance of chicken vocalisations has been well established [396, 44, 43, 84], and numerous studies have looked at detecting and quantifying vocalisations to categorise chicken vocalisations as an indicator of welfare and development [255, 243, 134, 320, 120].

2.3.5 Sheep

Although very little research appears to have been conducted into the development of a system capable of detecting sheep vocalisations or activity, considerable effort has been applied to quantify and understand vocalisations [435, 436, 439, 438, 437, 98, 294, 329, 373, 374, 363, 364, 97, 412, 229, 162]. The majority of research has focused on the

vocalisations between mothers and lambs [435, 436, 439, 438, 437, 98, 162], and this communication has been shown to be important in the establishment of the mother-young bond [294, 329, 374, 363, 364]. Furthermore, it has been shown that ewes will emit both high and low frequency bleats in response to their lambs, both of which have been quantified [364, 412, 229, 97] (Fig 2.27) (Fig 2.28). As it has been shown that there is a distinct pattern to various sheep vocalisations [373, 374, 364, 229], their detection by an automated, real-time system is plausible. Some work has been done to develop a predator detection system based on sheep responses [233], but acoustic activity detection and interpretation has yet to be explored in this area. As mismothering has been shown to be the leading cause of lamb losses (both a welfare and economic concern) [400, 153, 315, 104], automated detection of ewe-lamb vocal activity would be beneficial to both researchers and producers, as not only would it facilitate further study in the field, it may also lead to the ability to select for mothering aptitude, and could help mitigate losses due to early intervention of separated lambs.

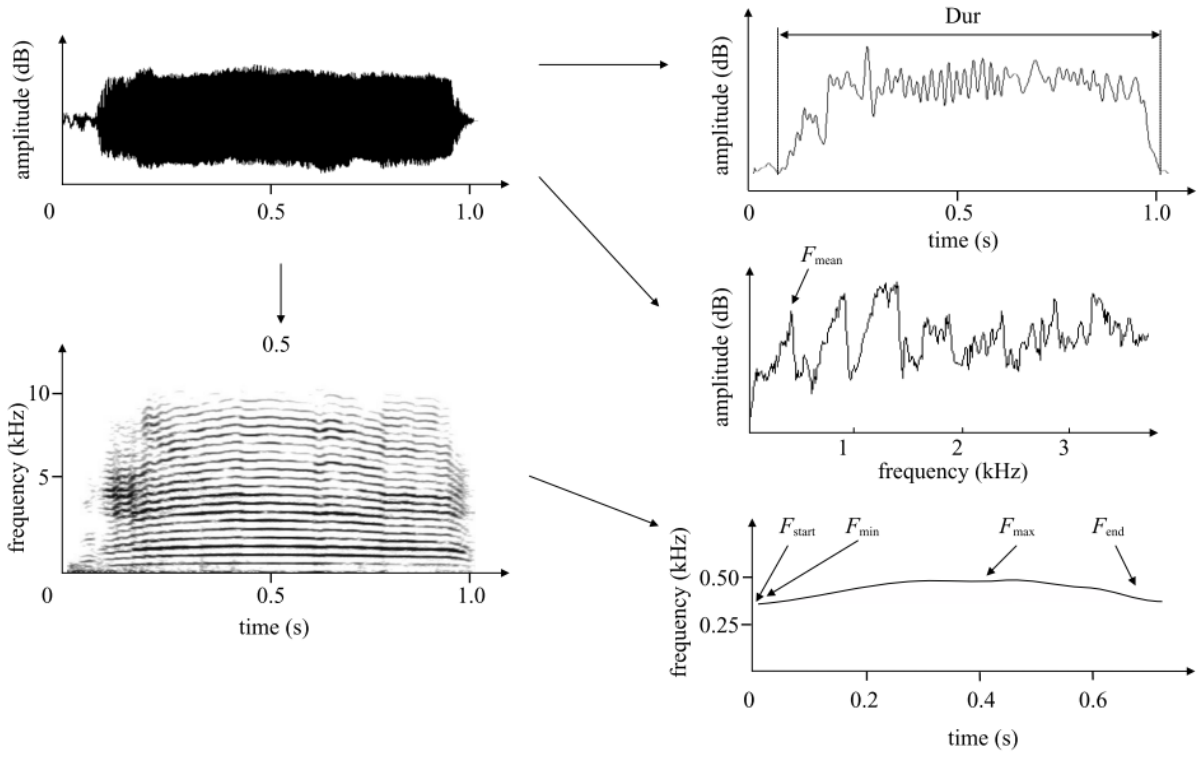

Figure 2.27: Spectral and temporal analysis of a lamb call [373].

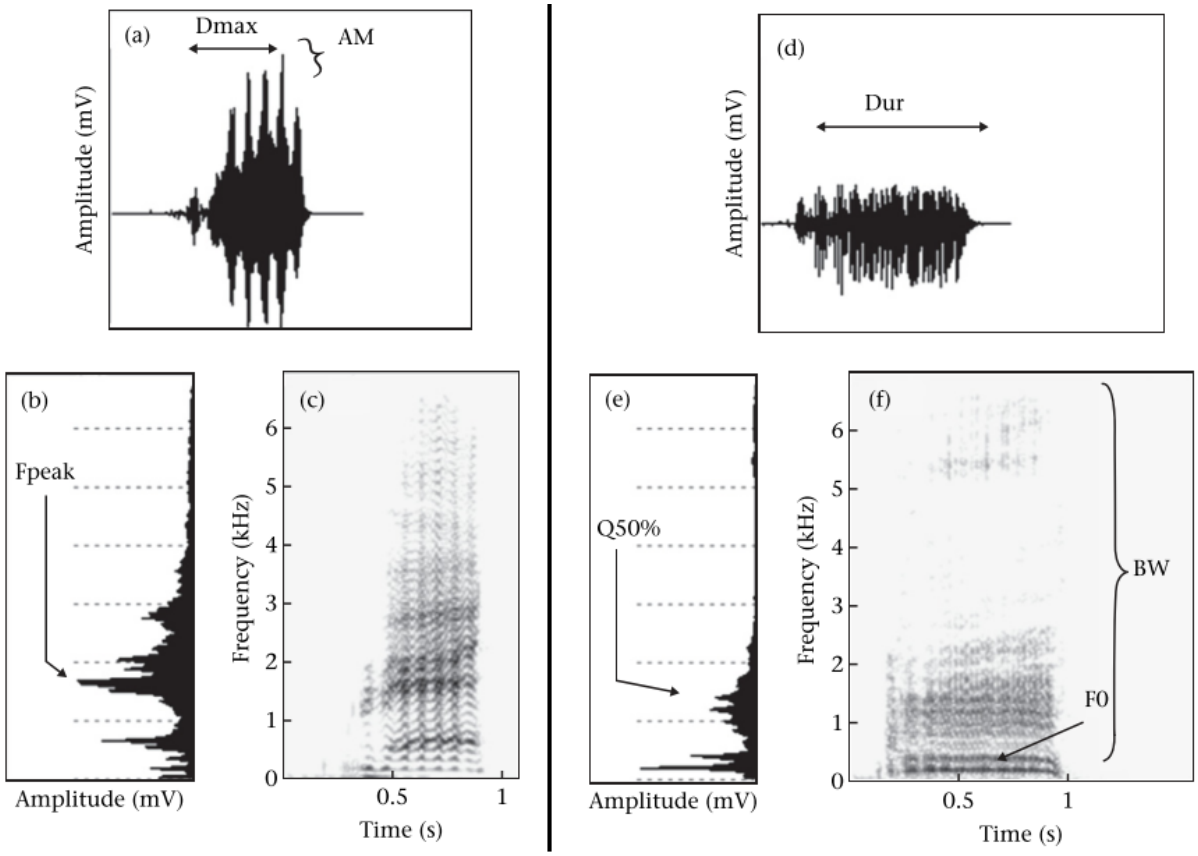

Figure 2.28: The two distinct vocalisations of a calling ewe [364]. The left side of the image shows the high-pitch call, whereas the right shows the low-pitch call.

2.4 Conclusion

After surveying the literature, it appears highly likely that the creation of a real-time system capable of detecting sheep vocalisations is possible. As has been shown, similar systems have been developed to address other PLF problems, and sheep vocalisations appear to have a distinct signature, which would theoretically allow them to be detected. Real-time audio detection systems contain numerous components, all working in concert, so attention must paid to the selection of each aspect of the signal chain. DWTs provide a novel approach to audio analysis, due to incorporating information from both the temporal and spectral domains. A window function will be applied, to assert whether this can improve DWT performance. As low computation times are key to a successful real-time system, simple statistical measures will be tested, providing features for the machine learning model. As the problem of sheep vocalisation detection has been defined as an OCC problem, an SVDD model will be trialled for classification. Section 3 describes the methods used to implement a system such as the one outlined, and the experiments used to test its performance.

Chapter 3

Methods

3.1 Objectives

The primary objective of this thesis is to ascertain whether it is possible to detect sheep vocalisations in real-time. Within this scope, the viability of DWTs in audio analysis, low-computational statistical features, and SVDDs for OCC, will also be explored. This will be achieved through a number of experiments that test the accuracy and performance of the real-time vocalisation detection system, as outlined in Section 3.6.

3.2 System Overview

3.2.1 Overview

The system developed has two distinct states, creating the model, and real-time operation, as illustrated in (Fig 3.1) and (Fig 3.2) respectively. Although these operations are performed separately, there are many functions which are used by both, and through the use of object oriented programming (OOP), many class functions are reused; code re-usability and low redundancy were key motivators during development. The flexible nature of class-based functions also facilitates the rapid comparison of variations in the signal chain, such as component order, and modification of individual parameters (e.g. DWT decomposition level, feature types, window padding schemes, etc).

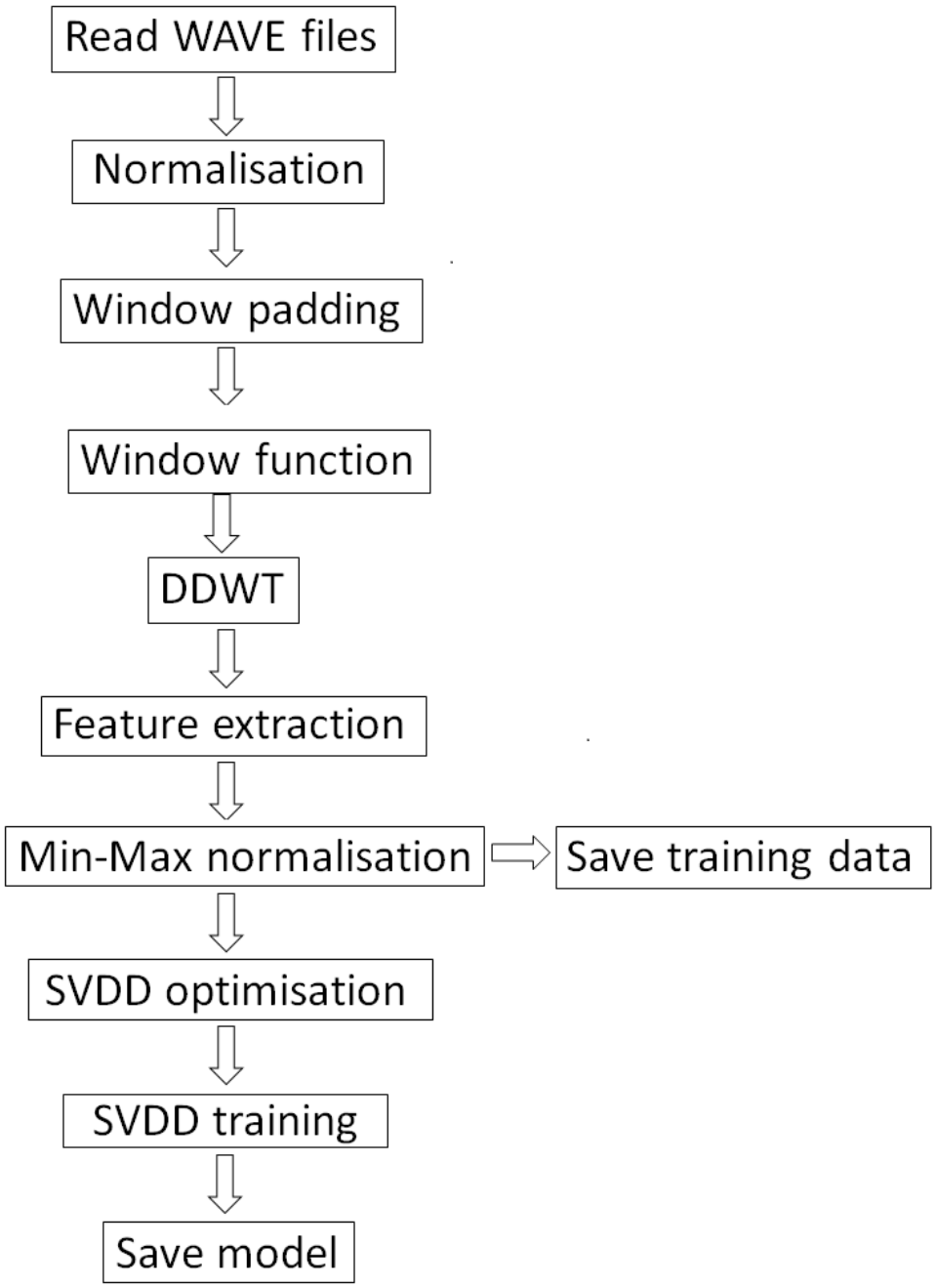

Figure 3.1: An overview of the model training and creation portion of the software.

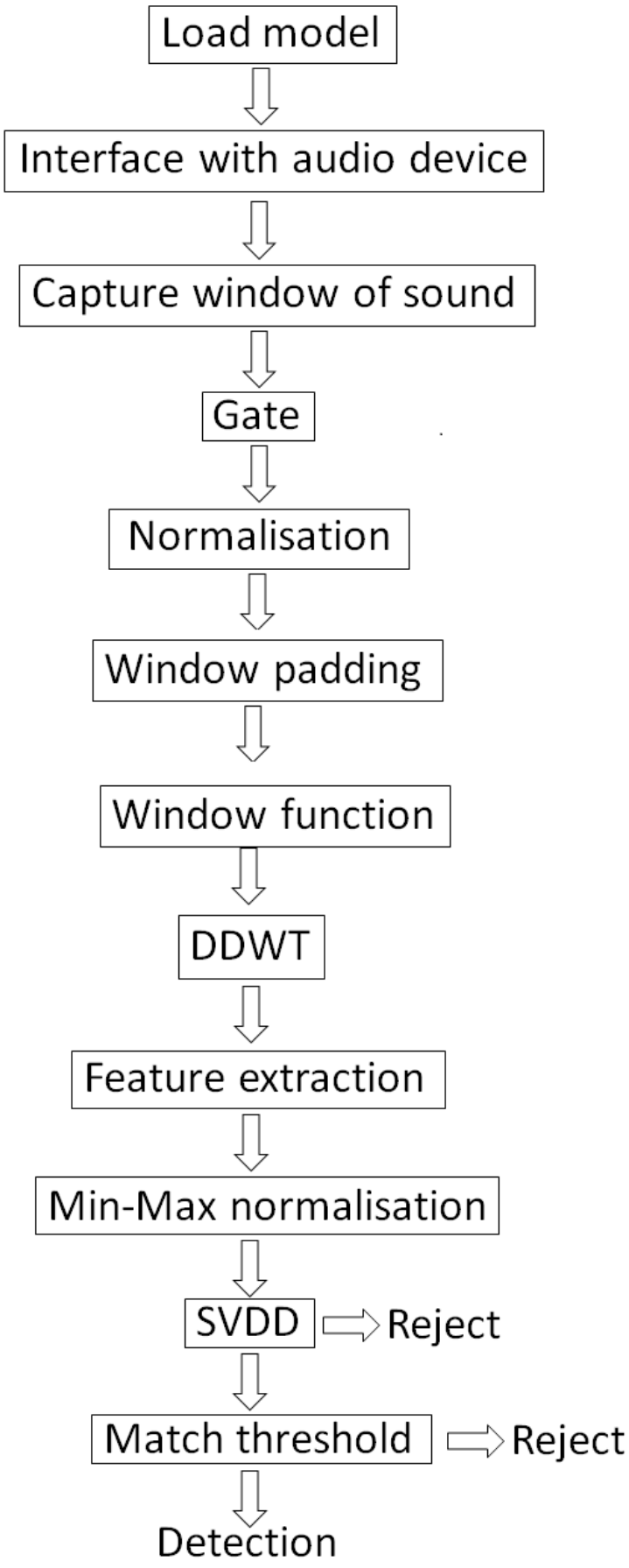

Figure 3.2: Overview of the real-time portion of the software. Note the similarity in some components as compared to (Fig 3.1).

The software has been written in the C++ programming language, and the only third-party dependency is the libSVM library [53], which handles the actual calculations required to create the SVDD model. (Fig 3.3) shows the classes used to build the system, and their interactions; the full code for this software is shown in Appendix 2.

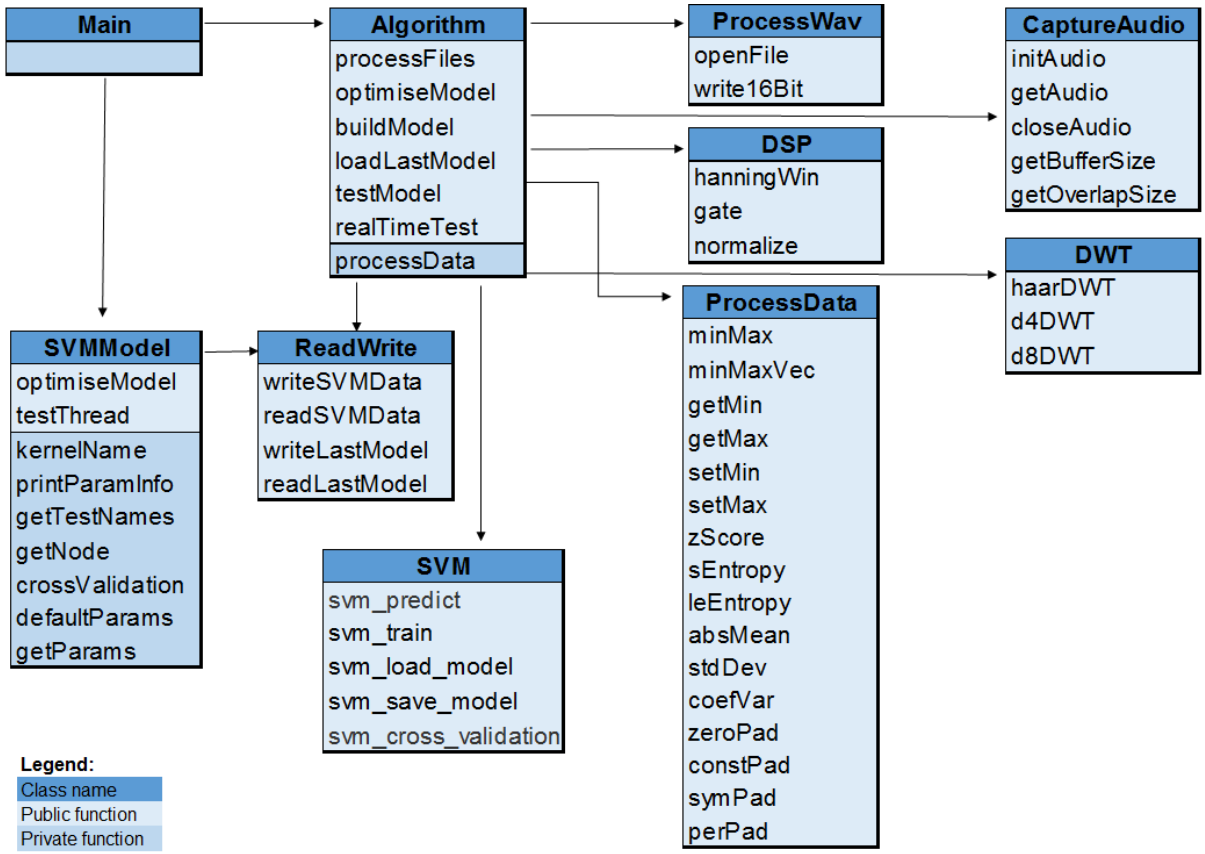

Figure 3.3: Class diagram for the sheep vocalisation detection system. Note that the SVM class is from the libSVM library, and only used functions are listed, for the sake of brevity.

3.2.2 Overview of the Training Mode

The training mode portion of the software starts by reading the training data, audio files in WAVE format, and normalising them, in terms of amplitude. The files are then extended in length to meet the dyadic restriction of the DWT, padded, a window function is applied, and this data is then transformed by the DDWT. Statistical features are extracted from the coefficient sub-bands produced by the DDWT, forming a data instance, which is aggregated with the whole training data set. The data set is normalised to ensure that all values comply with a minimum and maximum range (e.g. [0, +1], [-1, 1]) [382]. The

training data can now be used to create an SVDD model, using an optimisation routine in order to select appropriate parameters. Once a suitable model has been found, it is saved for use in the real-time system. The training mode algorithm can be summarised by the pseudo-code presented in (Fig 3.4).

```
1.  Run findWav.sh to get training data file names.
2.  While there are more audio files, continue processing:
3.      For each audio file:
4.          Normalise amplitude.
5.          Apply window padding.
6.          Apply window function.
7.          Perform DDWT, with a predefined type and decomposition level.
8.          Extract statistical features for each DDWT coefficient sub-band.
9.  When all files are processed:
10.     Perform min-max normalisation on training data.
11.     Save training data to file, in libSVM format.
12. Optimise SVDD parameters.
13. Train SVDD with training data.
14. Save SVDD model to file, in libSVM format.
```

Figure 3.4: Pseudo-code for the training mode algorithm.

3.2.3 Overview of the Real-Time Mode

The real-time system commences by interfacing with the audio device, and obtaining a window of incoming audio data. A noise gate [175] is used to test if the captured window contains acoustic activity loud enough to be of interest. If the window passes the gate, it goes through the same process as the training data file (i.e. normalisation, window extension, padding, window function, DDWT, feature extraction, min-max normalisation). As the data is now in the correct form, it can be tested for a match using the SVDD created in the training mode. If a sufficient amount of consecutive matches are recorded to pass the matching threshold, an alert is sent to indicate that sheep vocalisation activity has occurred. The real-time algorithm can be summarised by the pseudo-code presented in (Fig 3.5).

```
1.  Initiate hardware device.
2.  Get audio input, with a buffer length of (window length / overlap).
3.  When window is full with incoming audio data:
4.      Loop:
5.          Normalise amplitude
6.          Apply window padding
7.          Apply window function
8.          Perform DDWT, with a predefined type and decomposition level.
9.          Extract statistical features for each DWT coefficient sub-band.
10.         Min-max data, with values defined by training data.
11.         Test for match using SVDD
12.         If a match occurs:
13.             Increment match count
14.             If match threshold is reached:
15.                 Send alert of detection, with date and time
16.             Else:
17.                 Get new data, and go to start of loop.
18.         Else:
19.             Get new data, and go to start of loop.
```

Figure 3.5: Pseudo-code for the real-time algorithm.

3.3 System Components in Detail

3.3.1 Read WAVE Files

This component is used to read the audio data contained within WAVE files, by accessing the data block of the file header (Fig 3.6); 8, 16, and 32-bit quality files are supported. The data contained in the files is in time-amplitude format, at a resolution defined by the sample rate. The files read will be either training or testing data for the system (see Section 4.1 Training and Testing Data).

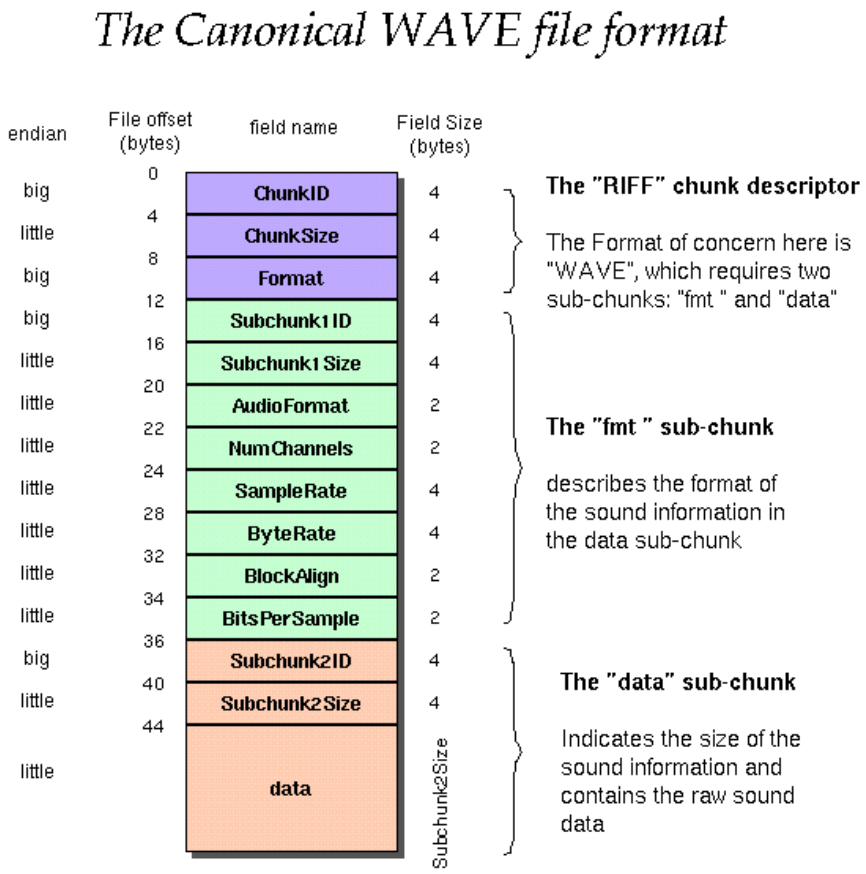

Figure 3.6: The header of a WAVE file [362]. Each piece of information is read a block at a time, with the last block containing the actual audio data.

3.3.2 Audio Normalisation

As each audio file or window of incoming data may contain sounds of vastly different amplitudes, it is necessary to normalise this data to allow for consistency between data instances. This ensures that all training and testing data is consistent in amplitude, whilst still preserving the original dynamics; this aids in reducing the influence that disparate levels may have on feature extraction measures, and in turn, machine learning accuracy. In this system, peak normalisation [220] was used, which involves adding a constant rate of amplitude to each data point, based on the difference between the waveform peak and the highest value supported by the bit rate (Fig 3.7) (Fig 3.8).

$$Y = Y\left(1 + \left(\frac{max\ value - peak}{peak}\right)\right)$$

Figure 3.7: Normalisation equation, where Y = the current data point, peak = the highest data point, and max value = the highest value the variable can hold.

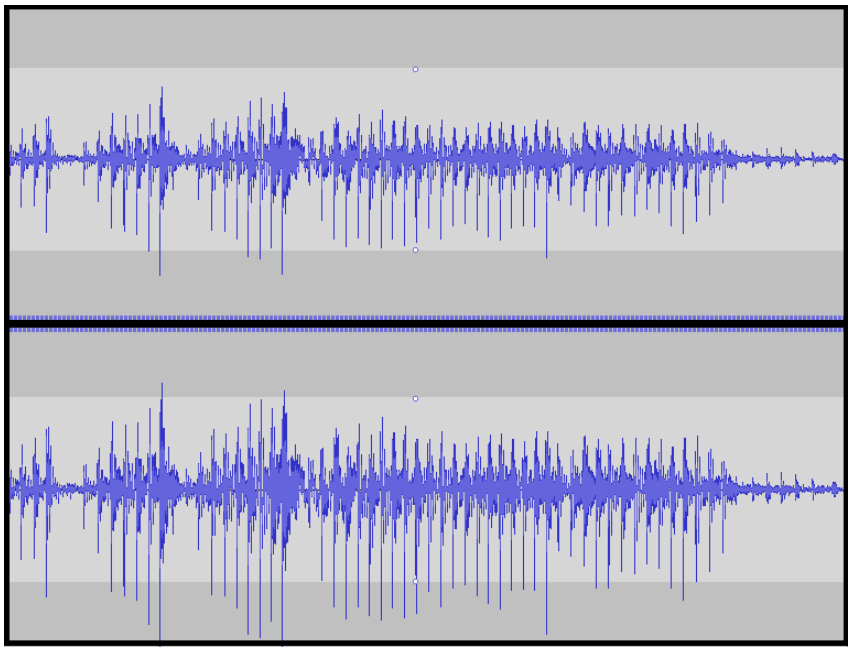

Figure 3.8: Peak normalisation. Each point of the sound is increased by a set rate, based on the distance between the highest data point, and the highest value supported by the bit rate.

3.3.3 Window Padding

As there is an inherent dyadic restriction for DWT window lengths, it may be necessary to extend the length of the original data by employing a window padding technique. The software supports zero [186, 347], periodic [176], and symmetric [38] padding (Fig 3.9). Through testing, it was found that periodic padding performed well for the current application.

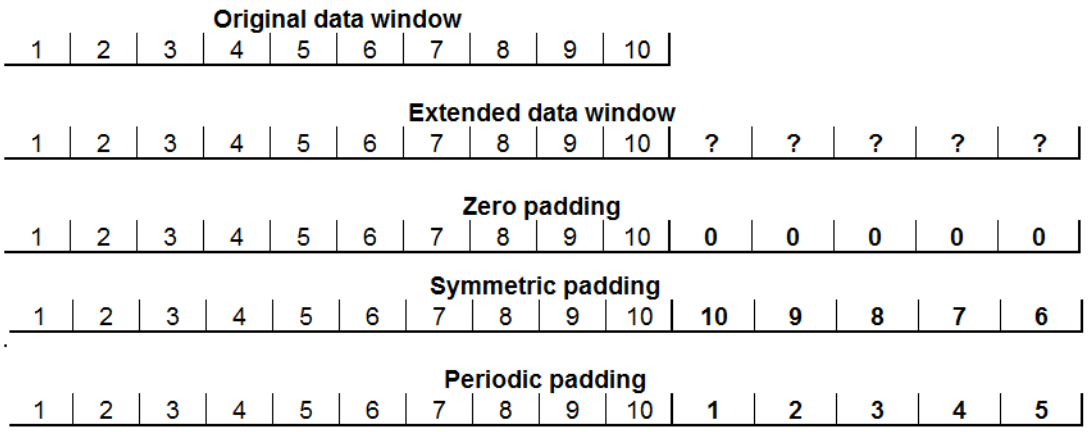

Figure 3.9: An example of the different window padding strategies available in the system; periodic padding is currently being used. In the example, the original window is extended by 5 data points, and each padding type uses a different method for populating these points.

3.3.4 Window Functions

As window functions are widely used in spectral analysis, it was hypothesised that they may be useful in improving the performance of DWT-based feature extraction methods. A Hann window function is used in the system, as this has been suggested as a balanced function [347, 128].

3.3.5 DDWT

The DDWT class supports [db2, db8] type DDWTs, at a decomposition level passed to the function; the code was developed from examples provided by [193]. The DDWT produces a number of coefficient sub-bands, namely detail sub-bands equal to the decomposition level, and one approximation sub-band. The first prototype of the system used a db4 DDWT with 3 decomposition levels, but it was found through testing that a db8 DDWT with 4 decomposition levels was necessary to facilitate accurate detection.

3.3.6 Feature Extraction

The feature extraction component is responsible for deriving statistical measures from the DDWT coefficient sub-bands in order to reduce their dimensionality. The software includes functions for Shannon entropy (Fig 2.15), mean, and standard deviation. Each of the selected measures is calculated for each sub-band, which are then assembled to form the data instance (Fig 3.10. Through testing, it was found that a feature set comprised of

Shannon entropy, mean, and standard deviation was sufficient to allow for discrimination by the SVDD model. Using a DDWT with 4 decomposition levels, the feature set has a dimensionality of 15, as compared to the unprocessed DDWT data that possessed a dimensionality in the 100,000s.

Approximation Sub-Band			Detail Sub-Band Level 1			Detail Sub-Band Level 2			Detail Sub-Band Level 3			Detail Sub-Band Level 4		
Sh. Entropy	Mean	Std	Sh. Entropy	Mean	Std	Sh. Entropy	Mean	Std	Sh. Entropy	Mean	Std	Sh. Entropy	Mean	Std
0.0149598	0.0252505	0.0782444	0.00059488	0.0205444	0.0218168	0.000148944	0.00392407	0.0101035	0.000685103	0.0106714	0.0209294	0.0004462	0.0063599	0.0179495

Figure 3.10:

3.3.7 Min-Max Normalisation

It has been demonstrated that the performance of SVM-based models can be improved through data normalisation (Section 2.2.3), and as such, a min-max normalisation function has been included in the system. The training data set is passed to the function in its entirety, and once the minimum and maximum values in the set have been ascertained, each value is normalised so that it complies to the range imposed (e.g. [0, +1], [-1, +1]). For example, if a range of [0, 1] was used, all data values would be normalised to take on values within this range, with the lowest recorded value being 0, and the highest being 1; (Fig 2.16) demonstrates how this is achieved mathematically. The un-normalised minimum and maximum values are recorded during processing, to ensure that new instances, such as test data, can be normalised effectively, as these values are based on the training data as a whole; issues pertaining to data normalisation are discussed further in (Section 5.5.1).

3.3.8 Save Training Data

To facilitate faster model creation and testing, the ability to save training data in libSVM format (Fig 3.11), has been included. This means that when adjusting model parameters and creating new models, training data only has to be processed once, rather than every time a new model is created. This feature also allowed for the timely comparison of other changes to the signal chain (e.g. DDWT decomposition level), as training data files could be easily accessed without system modification.

Format:	<index1>:<value1>	<index2>:<value2>	.....	<-1>:<?>

where:
index = feature position in data set,
value = the recorded value for the feature,
Note that all instances must end with -1.
Only non-zero values are recorded.

Sample Data					
Instance	**Label**	**Feature 1**	**Feature 2**	**Feature 3**	**Feature 4**
A	1	0.2	0.4	0.6	0.8
B	1	0	0.1	0.2	0.3
C	1	0.3	0.6	0.9	0

libSVM Format					
Instance	**1st Value**	**2nd Value**	**3rd Value**	**4th Value**	**5th Value**
A	1 : 0.2	2 : 0.4	3 : 0.6	4 : 0.8	-1 : ?
B	2 : 0.1	3 : 0.2	4 : 0.3	-1 : ?	
C	1: 0.3	2 : 0.6	3 : 0.9	-1 : ?	

Figure 3.11: An example of training data in libSVM format [53]. A data instance is described using index-value pairs, for non-zero values only. All instance must be ended with a -1 index value.

3.3.9 SVDD Optimisation

Overview

The SVDD optimisation routine has a number of stages (Fig 3.12), each designed to reduce the number of potential models based on certain criteria, thereby selecting appropriate model parameters for the current problem. This approach greatly reduced the time necessary to produce an acceptable model as compared to manually adjusting model parameters.

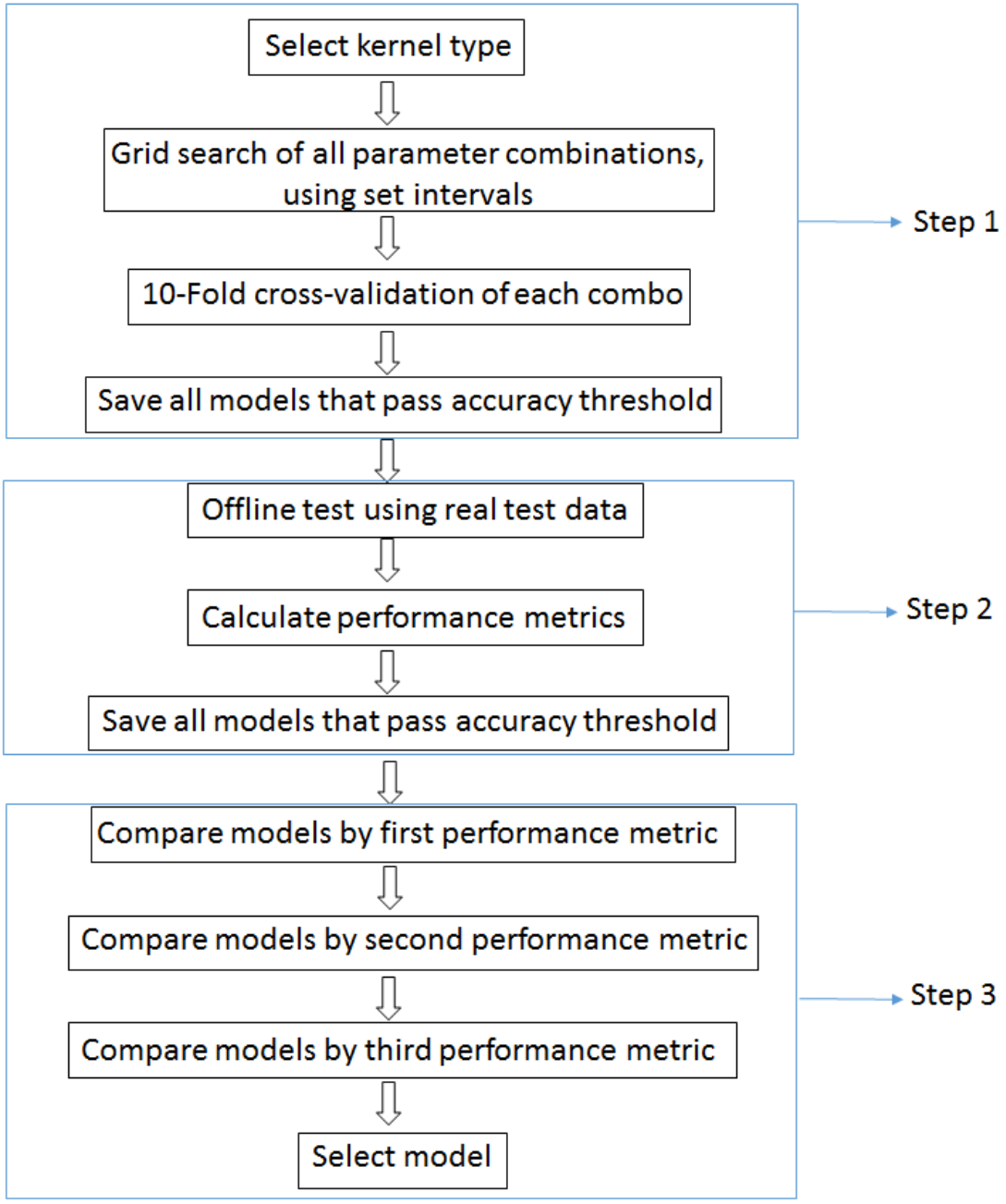

Figure 3.12: Optimisation routine for the SVDD model. Each major step in the routine, outlined in blue, filters the number of possible models into a smaller subset.

Step 1: Cross-Validation

Step 1 of the optimisation process involves preparing the libSVM environment, then performing 10-fold cross-validation on each combination of kernel, and its associated parameters; (Fig 3.13) illustrates the parameters used for each kernel. Essentially, a grid search algorithm [171] is used to incrementally change each parameter, within a predefined

range (Fig 3.14), testing each possible combination. As this is an OCC problem, a cross-validation threshold, rather than performance metrics (e.g. TPR, FPR, etc), was used to determine which models should continue to the next stage: 80% was found to be a sufficient threshold. In order to reduce runtime, threading was used to allow each parameter combination to be tested on its own thread; the performance of this approach is discussed in (Section 5.5.5). It was found that the RBF kernel was almost exclusively selected as the best kernel type, reiterating the conclusions of [171, 430, 230]; the RBF kernel was used in the final model. The pseudo-code displayed in (Fig 3.15) summarises the process of step 1 of the optimisation function.

Kernel type:	Linear	RBF	Sigmoid	Polynomial
Parameters:	C-value	C-value	C-value	C-value
		Gamma	Gamma	Gamma
			Coef0	Coef0
				Degree

Figure 3.13: The parameters associated with each kernel type.

	C-value	Gamma	Coef0	Degree
Implemenation:	2^X	2^X	X	X
Increment value:	0.125	0.125	1	1
Start value:	-10	-15	0	1
End value:	0	5	1	10

Figure 3.14: An overview of the implementation of kernel parameter adjustment, where X = the value to be incremented. Each parameter has its own form, range, and increment value.

```
1.  Create a libSVM problem, using the training data already created.
2.  Create a libSVM parameter model.
3.  Set model type to SVDD.
4.  Set cache size to 500.
5.  Set epsilon to 0.001.
6.  Disable shrinking heuristics.
7.  Disable probability estimation.
8.  For each kernel type, test each combination of parameters:
9.      If kernel is linear:
10.         Adjust C-value, using 2^X (X incremented by 0.125, range [-10, 0] ).
11.         Create thread, and perform 10-fold cross-validation.
12.         If result is >= cross-validation threshold:
13.             Save parameters.
14.     Else if kernel is RBF:
15.         Adjust C-value, using 2^X (X incremented by 0.125, range [-10, 0] ).
16.         Adjust gamma, using 2^X (X incremented by 0.125, range [-15, +5] ).
17.         Create thread, and perform 10-fold cross-validation.
18.         If result is >= cross-validation threshold:
19.             Save parameters.
20.     Else if kernel is sigmoid:
21.         Adjust C-value, using 2^X (X incremented by 0.125, range [-10, 0] ).
22.         Adjust gamma, using 2^X (X incremented by 0.125, range [-15, +5] ).
23.         Adjust coef0 (increment by 1, range [0, 1] ).
24.         Create thread, and perform 10-fold cross-validation.
25.         If result is >= cross-validation threshold:
26.             Save parameters.
27.     Else if kernel is polynomial:
28.         Adjust C-value, using 2^X (X incremented by 0.125, range [-10, 0] ).
29.         Adjust gamma, using 2^X (X incremented by 0.125, range [-15, +5] ).
30.         Adjust coef0 (increment by 1, range [0, 1] ).
31.         Adjust degree (increment by 1, range [1, 10] ).
32.         Create thread, and perform 10-fold cross-validation.
33.         If result is >= cross-validation threshold:
34.             Save parameters.
35. Once all combinations have been exhausted, wait for threads to complete.
36. Continue to step 2 of the optimisation process.
```

Figure 3.15: Pseudo-code for Step 1: Cross-Validation, of the SVDD optimisation routine.

Step 2: Offline Testing

Step 2 of the optimisation process tests all the models obtained from step 1 using real test data (Fig 3.16). Tests are performed in an offline capacity, in that test data instances are read from file, rather than being captured in real-time; the test data used is shown in (Fig 4.15). As can be seen, a higher percentage of negative class instances, as compared to positive, are used in this step, as step 1 only included instances of the positive class, due to the OCC nature of the problem. The instances of the positive class are not present in the training data, so as to provide a more accurate test of the models performance (see Section 2.2.4). For each model, performance metrics are calculated based on the results of the test, and these include TPR, FPR, TNR, FNR, accuracy, precision, gMean1, and gMean2 (Fig 2.18.Threading was also used in this step of the optimisation process, again, in an effort to reduce runtime duration. Once all potential models have been tested, those that pass an initial performance threshold are saved, for use in step 3. The performance

threshold in this step helps to quickly narrow down the models of interest, and in the final implementation, a combination of gMean1 (threshold of 90%) and FPR (threshold of 5%) were used, as these were found to emphasize the desired traits of the system (e.g. low false-positives, and high overall accuracy); pseudo-code for this step is presented in (Fig 3.17).

Sound Type	Quantity	Bit Rate	Sample Rate	Class
Sheep vocalisation	10	16-bit	44.1 kHz	Positive
Dog barking	5	16-bit	44.1 kHz	Negative
Gun shot	5	16-bit	44.1 kHz	Negative
Music	5	16-bit	44.1 kHz	Negative
Noise / static	5	16-bit	44.1 kHz	Negative
Rain	5	16-bit	44.1 kHz	Negative
Talking	5	16-bit	44.1 kHz	Negative
Thunder	5	16-bit	44.1 kHz	Negative
Tractor	5	16-bit	44.1 kHz	Negative
Total:	50			
Postive:	20%			
Negative:	80%			

Figure 3.16: The composition of the test data set used for step 2 of the SVDD optimisation process. Note the higher percentage of negative class instances, as compared to positive. See (Section 3.4 for more information about the test data).

```
1.  For each model saved in Step 1:
2.      Create new thread.
3.      Run offline test, using test data set.
4.      Based on class labels:
5.          Calculate TPR.
6.          Calculate FPR.
7.          Calculate TNR.
8.          Calculate FNR.
9.          Calculate accuracy.
10.         Calculate precision.
11.         Calculate  gmean1.
12.         Calculate gmean2.
13.     If gMean1 >= threshold (90%) and FPR <= threshold (5%):
14.         Save model.
```

Figure 3.17: Pseudo-code for Step 2: Offline Testing, of the SVDD optimisation routine.

Step 3: Select Top Model

Step 3 of the optimisation process ranks the set of potential models produced by step 2, based on a 3-tiered system of performance metrics. Essentially, each model is iteratively compared to the current top rated model, one metric at a time, and if it is found to surpass the top model, it replaces it. Once model comparison is complete, the parameters of the top model are selected for use in the subsequent SVDD model. The selection of performance metrics, and the order in which they appear, can produce significantly different models, and some testing was required to find a suitable set of metrics for the given problem. As in step 2, the metrics were chosen to accentuate the desired traits of the system, in particular, emphasis on a low rate of false positives, as it was deemed more important to be sure of vocalisation activity, rather than producing numerous false alerts. In the final system, the first metric used was precision, followed by TPR, and finally, gmean2; pseudo-code for step 3 is presented in (Fig 3.18).

```
1.  If this is the first model:
2.      Set as top model.
3.  Else:
4.      Compare the precision of the current model, to the top model.
5.      If precision is >= the top model:
6.          Compare the TPR of the current model, to the top model.
7.          If TPR is >= the top model:
8.              Compare the gMean2 of the current model, to the top model.
9.              If gMean2 is > the top model:
10.                 Set current model as the new top model.
11.             Else:
12.                 Move onto the next model in the set.
13.         Else:
14.             Move onto the next model in the set.
15. If the end of the set is reach:
16.     End.
```

Figure 3.18: Pseudo-code for Step 3: Select Top Model, of the SVDD optimisation process.

3.3.10 SVDD Training

The parameters discovered using the optimisation component are set for the SVDD, and it is trained using the training data prepared by previous components. Through testing during development, it was found that an SVDD model using an RBF kernel, with a C-value of 0.5, and a gamma value of 0.25, was found to be sufficient to conduct further testing in order to meet the objectives of the system.

3.3.11 Interface with Audio Device

To access incoming real-time audio data, the software must interface with the drivers used by the audio device in the system. The specific drivers used will be dependent on the audio device selected for the system, but in the current deployment, ALSA drivers were used. Pseudo-code to explain the process of interfacing with ALSA drivers is provided in (Fig 3.19).

1.	Open audio device
2.	Allocate hardware parameter structure
3.	Initialise parameter structure with device capabilities
4.	Set access mode (regular or mmap), and interleave
5.	Set bit rate, endianess, and signed
6.	Set sample rate
7.	Set number of channels (e.g. 1 = mono, 2 = stereo)
8.	Set buffer size
9.	Set period size
10.	Apply hardware parameters to device
11.	Prepare audio device for use

Figure 3.19: The necessary steps to setup and interface with the audio device, using ALSA driver function calls.

3.3.12 Capture Window

Once access to the audio device has been enabled, incoming audio data can be obtained at set window lengths, and the software does this by using an overlapping, or sliding, window approach (Fig 1.3). The system uses an overlap of 10, which equates to 0.1 second of new content per window; this is illustrated in (Fig 3.20). The premise for the use of a relatively high rate of overlap was that this would enable the system to more precisely capture a whole vocalisation sound, thereby facilitating more accurate matching.

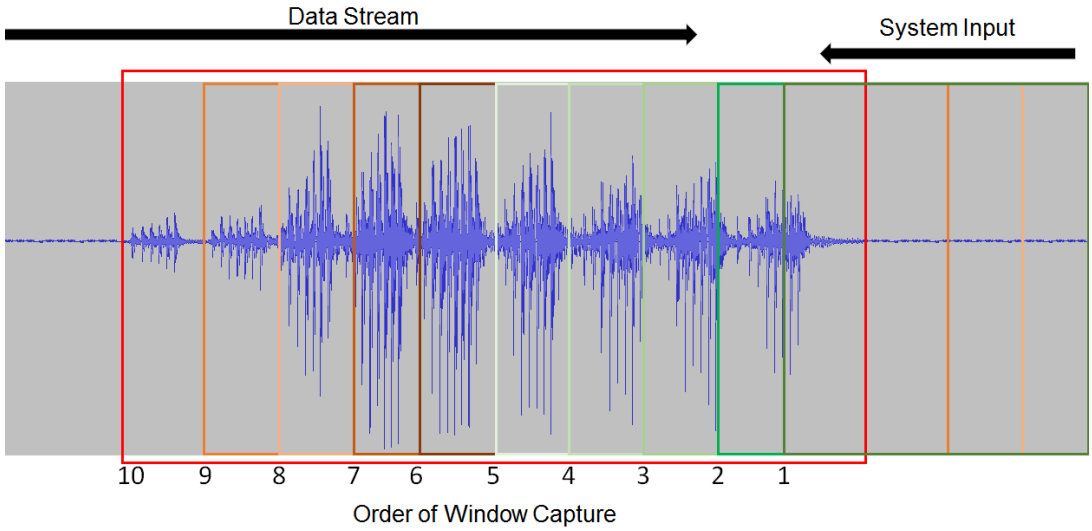

Figure 3.20: An example of an overlapping window strategy, with a window length of 1 second, and an overlap rate of 10. As the audio stream containing the sheep vocalisation comes into the system, set windows of data are captured, each containing a large portion of the last window.

3.3.13 Noise Gate

To reduce the amount of unnecessary computation, a noise gate [175] is employed to test whether a window of incoming data contains any components that are of a significantly high enough amplitude to be of interest. This is done by testing whether or not a window contains three consecutive data points that have an absolute value greater than or equal to the predefined threshold (Fig 3.21). The reason that three points must pass the threshold is to reduce the effect of transient peaks or system interference that may cause an otherwise quiet, uninteresting window to be processed; any sound that is loud enough and of even minor duration will have at least 3 consecutive data points that pass the threshold. It is generally futile to attempt to match sounds that are very low in amplitude (e.g. far away, very quiet), as there is generally too much noise, and not enough signal information to provide an adequate feature set for detection.

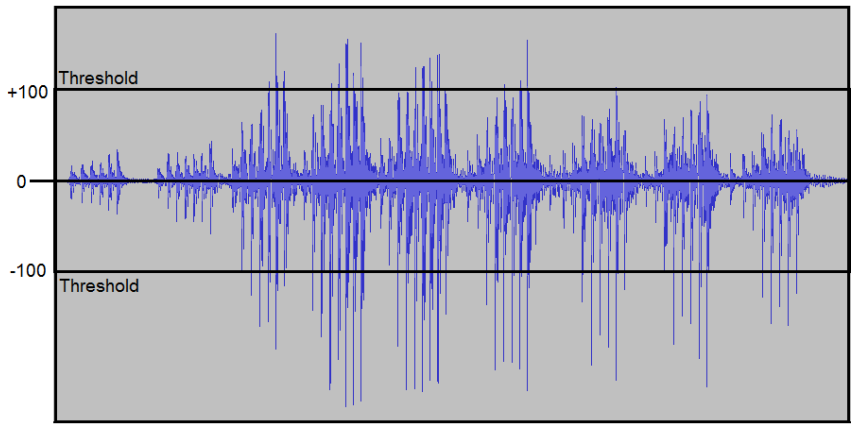

Figure 3.21: An example of how a gate is used to test for sufficient acoustic activity to warrant further processing. This example shows a sheep vocalisation passing the set threshold. Note how both positive and negative values are taken into account.

3.3.14 Match Threshold

It was discovered during development that by using a relatively high overlap rate (e.g. 10 in the final system), a successful match will frequently occur in a number of consecutive windows; when a vocalisation takes place, it is present in numerous windows. To decrease the number of superfluous match alerts, a matching threshold was implemented, whereby a set number of sequential windows must record a match in order for a detection alert to be sent. This feature was also found to reduce the rate of false positives, and therefore increase overall accuracy. A match threshold of T = 4 was used for the final system, as this was found to be a sufficient level to reduce multiple alerts and false positives.

3.4 Training and Testing Data

3.4.1 Data Acquisition

Two unedited recordings of sheep vocalisations were acquired for this project, to be used as training and testing data in the system. The first was provided by Dr. Rachelle Hergenhan, and was an indoor recording of a ewe calling her lamb, produced to aid in experiments conducted as part of her PhD thesis [162]. The second file was a field recording captured by a listening station as part of research undertaken by the University of New Englands (UNE) Precision Agriculture Research (Precision Agriculture Research Group (PARG)) team. Instances of sheep vocalisations were manually extracted from each recording, at 16-bit / 44.1 kHz quality, with a one second window length used to ensure uniformity.

From the first recording, 93 instances of sheep vocalisations were found, all of which were reasonably clean and free of substantial noise. The field recording yielded 71 instances, but many contained a high level of noise (thunder, line hiss, birds, etc), were too far from the sound source, or both. This set of files was then manually categorised by quality, producing 15 clean, 13 distant, 7 noisy (mainly thunder), and 35 unusable instances. Unfortunately, the field recording proved largely unusable, due to the low number of clean instances, but will still assist in providing a challenging test for the system (see Section 3.6.4).

3.4.2 Training Data

The main training data set is comprised of 83 sheep vocalisations, taken from the recording containing the calling ewe; there is little to no background noise present in the samples.

3.4.3 Test Data

The main test data set is composed of 10 instances of sheep vocalisations, randomly selected from the calling ewe data, and 40 instances of other sounds, together forming an example of the negative class (i.e. any other sound). (Fig 3.22) shows the quantity and types of sounds used to represent the negative class. As can be seen, some of these sounds are quite different to the positive class (e.g. gun shots, music), whereas others are included because of the similarity they possess, in terms of where in the spectrum spectral activity is occurring (e.g. dog barks, humans talking) (Fig 2.2). The negative class is more heavily represented in the test data, as in a real-world setting, there would be far more examples of the negative class.

Sound Type	Quantity	Bit Rate	Sample Rate	Class
Sheep vocalisation	10	16-bit	44.1 kHz	Positive
Dog barking	5	16-bit	44.1 kHz	Negative
Gun shot	5	16-bit	44.1 kHz	Negative
Music	5	16-bit	44.1 kHz	Negative
Noise / static	5	16-bit	44.1 kHz	Negative
Rain	5	16-bit	44.1 kHz	Negative
Talking	5	16-bit	44.1 kHz	Negative
Thunder	5	16-bit	44.1 kHz	Negative
Tractor	5	16-bit	44.1 kHz	Negative
Total:	50			
Postive:	20%			
Negative:	80%			

Figure 3.22: The main test data set used for the system. Note the high percentage of negative class instances.

3.5 Equipment

(Fig 3.23) illustrates the configuration of the hardware that comprises the system. As a major component of the system involves real-time audio capture, careful attention was paid to the selection of audio equipment; accuracy in both playback and capture was essential.

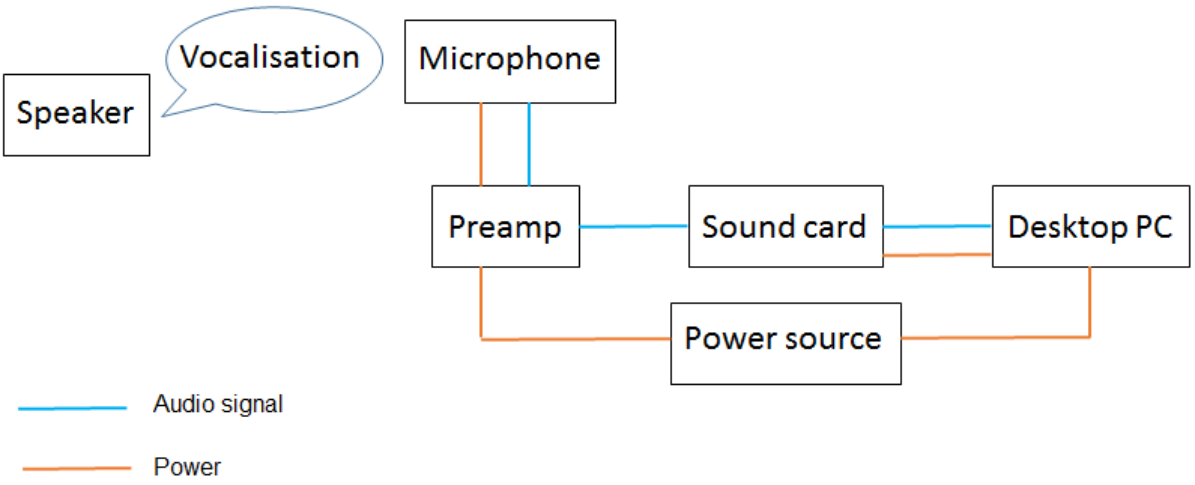

Figure 3.23: The main test data set used for the system. Note the high percentage of negative class instances.

3.5.1 Playback

To allow for accurate playback of test sounds, speakers with a flat frequency response were required. Yamaha HS8 studio monitors were selected for their frequency response, which was both flat (Fig 3.24), and paired well with the frequency range inhabited by sheep vocalisations (Fig 2.2).

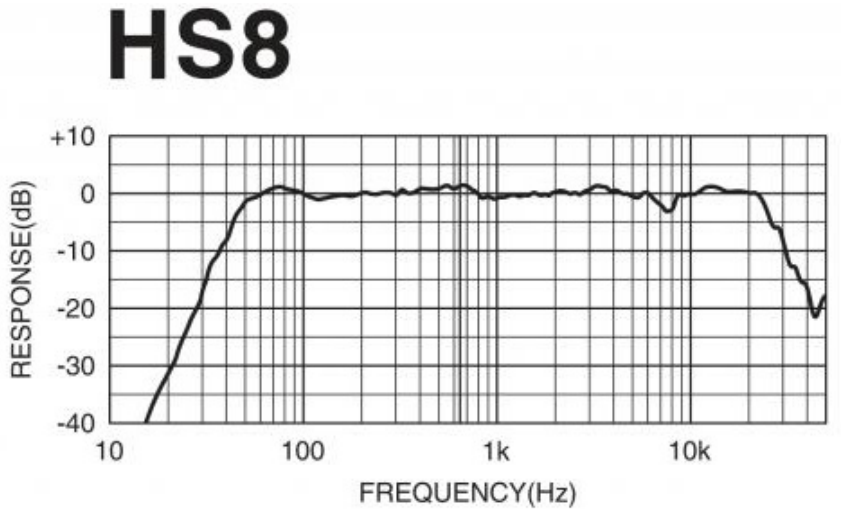

Figure 3.24: The frequency response of a Yamaha HS8 studio monitor [455], as used during testing.

3.5.2 Capture

As microphones are designed for many different applications [175, 273, 307], it is crucial to choose a microphone that meets the requirements of the system, namely, flat frequency response, and high sensitivity. Condenser microphones fulfil this role, and the Rode NT5 [351] was selected, due to its performance (Fig 3.25), its cardioid polar pattern (Fig 3.26) (high front / low back sensitivity) (Fig 3.27), and because of its small size (as would be used in a real-world system) (Fig 3.28). One drawback of condenser microphones is that they require external power to function, which is typically delivered down the signal lead, using phantom power [273]. As this feature was not available on the sound card used for testing, an external preamp was used to deliver power, while also adding the benefit of increased amplification; a McLelland MQS-51 Mini-Mic Preamplifier was used for this purpose.

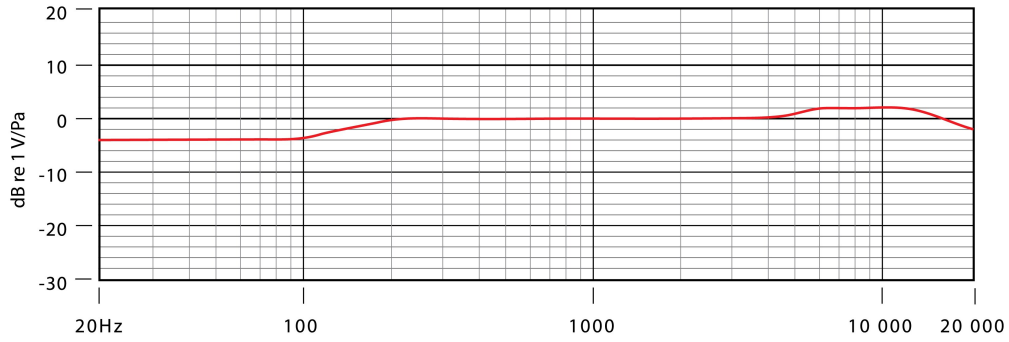

Figure 3.25: The frequency response of the Rode NT-5 condenser microphone, as used during testing [351]. Note the flat frequency response between 200Hz and 5 KHz.

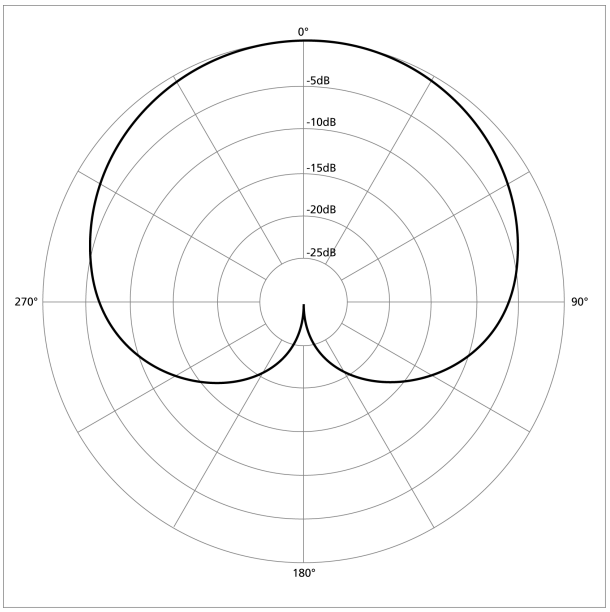

Figure 3.26: A cardioid polar pattern [175], as used by the Rode NT5. Note the high level of rear rejection, moderate side rejection, and full frontal pickup.

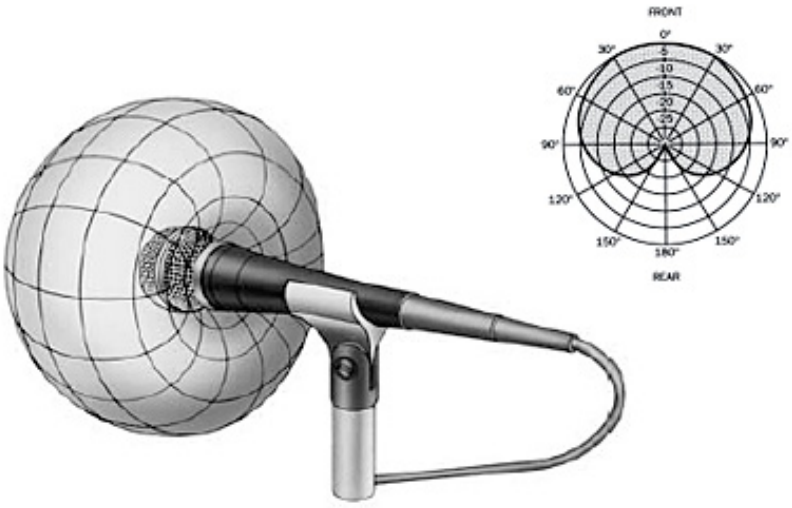

Figure 3.27: A 3-dimensional view of the cardioid polar pattern [337], which clearly illustrated the rear rejection.

Figure 3.28: A comparison of microphone sizes. The top mic is the ubiquitous Shure SM58, the middle is a Behringer ECM8000 reference mic, and the bottom is the mic used in the system, a Rode NT5. Note the small size of the NT5.

3.5.3 Training and Testing Machine

For training and testing the system, a desktop PC with the following specifications was used:

- Intel Core i7-4790K CPU @ 4.00 GHz.
- 16GB DDR-3 RAM.
- Samsung 850 EVO SSD, with SATA III.
- Asus Sabertooth Z-97 Mark 1 motherboard, with SATA express.
- On-board Realtek ALC1150 8-Channel sound card, supporting 24-bit / 192kHz quality, with a 104 dB SNR recording input.
- Linux 3.11-2-amd64 x86_64 operating system.

As one of the target deployments for this system is a Raspberry Pi [345], the resources of this device were considered when developing the real-time component of the software. A Pi-based system would have the following specifications:

- Raspberry Pi 2 Model B.
- 900MHz quad-core ARM Cortex-A7 CPU.
- 1 GB of RAM.
- Cirrus Logic sound card, which supports 24-bit / 192kHz quality sound.

3.6 Experiment Methodology

In pursuance of the thesis objectives, four experiments are proposed to test the capabilities of the system:

- 3.6.1: Real-Time Detection of Sheep Vocalisations Using a Mixed Data Set
- 3.6.2: The Effect of Distance on Real-Time Sheep Vocalisation Detection Accuracy
- 3.6.3: Real-Time Segmentation of Sheep Vocalisations
- 3.6.4: Real-Time Detection of Sheep Vocalisations Using Noisy Data

The aim is to test the systems ability to detect sheep vocalisations in real-time, and to ascertain if this is indeed possible. These experiments will also explore the use of DDWT for audio analysis, low-computational statistical measures as features, and SVDDs for classification of audio data. It should be noted that in sound analysis, it is often easier to test new methodologies offline, without the complex issues of acoustics, noise, phase, accurate sampling, and other sound related phenomenon interfering with results, and it is for this reason that all experiments will be conducted in real-time, using a live system (Fig 3.29), as this reflects the true nature of the systems performance.

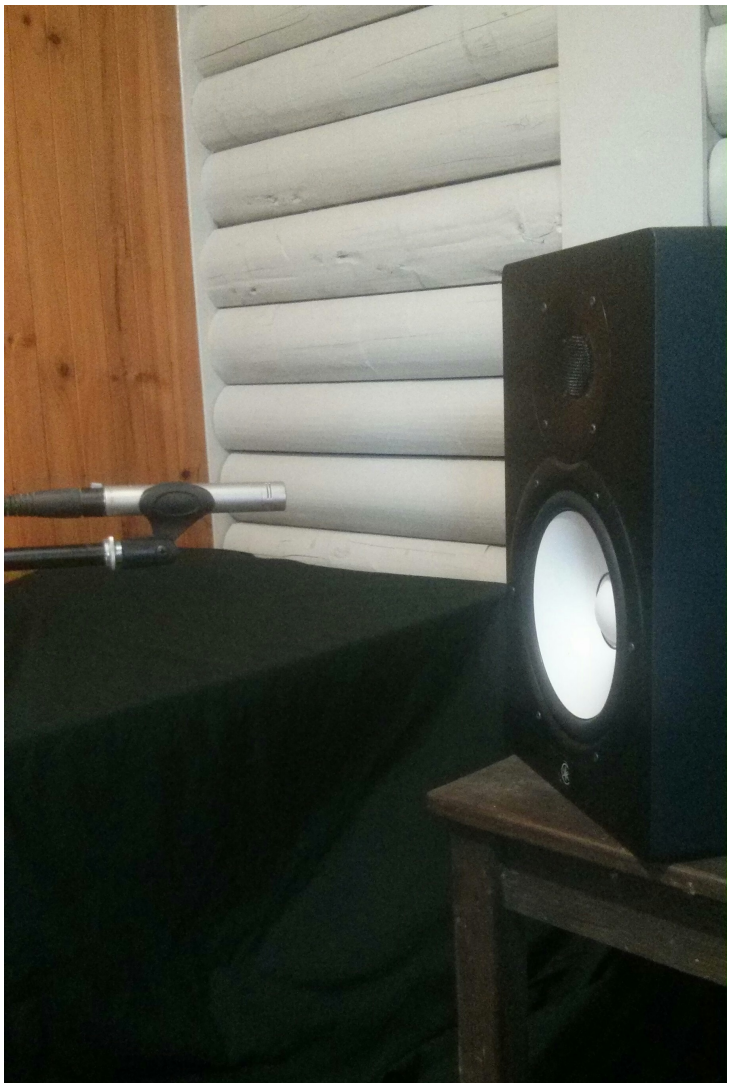

Figure 3.29: The standard testing set up used during experimentation. This image shows the microphone placed 20cm away from the sound source.

3.6.1 Real-Time Detection of Sheep Vocalisations Using a Mixed Data Set

The aim of this experiment was to test the overall detection accuracy of the system, taking into account performance on instances of both the positive and negative class.

- The standard training data set was used, as described in (Section 3.4).
- The test data set presented in (Section 3.4) was used, which contains a variety of sounds that may be encountered by the system.
- The test data was combined into a continuous file, with each instance occurring at a random interval from the last.
- The continuous test data file was played through the speakers.
- The microphone was placed 0.2m away from the sound source, with an average digital input level of [-12, -6]. These represent good conditions for the system.
- Two tests will be run during this experiment, each using a different matching threshold, in order to separate the effect this feature may have on perceived detection accuracy (i.e. a relatively high threshold may appear to produce lower accuracy).
- Each test was conducted 3 times, to ensure consistency in results.
- The results of this experiment are presented in (Section 4.2), and discussed in (Section 5.2.1).

3.6.2 The Effect of Distance on Real-Time Sheep Vocalisation Detection Accuracy

As one of the hypothetical deployments for the system may be as a network of field nodes (Fig 1.10), it is necessary to ascertain the effect distance, and in turn, reduced input level, may have on system accuracy. This experiment is intended to test the effect of distance on the systems detection accuracy, taking into account instances of both the positive and negative class. The speaker will be moved away from the microphone, to simulate a dynamic sound source, reducing system input levels, and introducing more acoustic noise. Input levels will be adjusted at further distances, in order to determine the effect of distance irrespective of input level. Input level may affect system performance, and although reduced level is related to distance, it is not the only factor that may cause distance to reduce detection accuracy (e.g. phase, acoustics, frequency response, etc).

- Both the training and test data are the same as the experiment outlined in (Section 3.4).
- The speaker will be placed at 4 set intervals: 0.2m, 0.4m, 1m, and 1.5m; these distances are progressively further away from the microphone.
- Input levels at distances (1m and 1.5m) will be adjusted, using the input gain on the preamp, to demonstrate the effect of distance, irrespective of input level.
- Each test was conducted 3 times, to ensure consistency in results.
- The results of this experiment are presented in (Section 4.3), and discussed in (Section 5.2.2).

3.6.3 Real-Time Segmentation of Sheep Vocalisations

As the system uses an overlapping window approach for audio capture, it is possible to determine the duration of detected sheep vocalisations by calculating the number of sequential windows that record a positive detection. Due to the OCC nature of the problem, the system is effectively a binary classifier, possessing two states: 0 (not detection) and 1 (detection). To gain insight into the true performance of the system, it is necessary to determine exactly how many sequential windows are registering a positive detection, and in turn, how long the system stays in a 'positive' state.

- The standard training data set was used, as described in (Section 3.4).
- The microphone was placed 0.2m away from the sound source, with an average digital input level of [-12, -6]; these represent good conditions for the system.
- A continuous test file was produced, containing 20 ewe call vocalisation instances, each with a random interval between them. The instances used were the 10 instances present in the test data set, and 10 instances chosen randomly from the training data.
- The test file was annotated to denote the start and end times of each vocalisation, thereby determining the duration of each call; this information was considered the 'gold standard' for system performance comparison.
- The software was adjusted to allow the number of sequential positive detections to be displayed; detections must first meet the matching threshold to be considered.
- The predicted and actual durations of each call will be compared to determine error, with MSE and MAE used to demonstrate the system's overall duration error.

- The test was conducted 3 times, to ensure consistency in results.
- The results of this experiment are presented in (Section 4.4), and discussed in (Section 5.2.3).

3.6.4 Real-Time Detection of Sheep Vocalisations Using Noisy Data

The data procured from the field recordings was deemed too noisy, and too low in quantity, to be represented in the primary training and test data sets (Section 3.4). However, this data does pose a significant challenge of the system's performance, and its variance in terms of matching new data (i.e. does the model possess enough variance to match sheep calls from other data sources?). Two tests are proposed as part of this experiment, one in which the 'cleanest' data instances from the field recordings is used as a test data set for the existing model, and one where the same 'clean' field recording instances are also used to train a new model.
For test 1:

- The microphone was placed 0.2m away from the sound source, with an average digital input level of [-12, -6]. These represent good conditions for the system.
- A continuous test file was produced, containing the 15 'clean' instances from the field recording data with intervals of a random duration present between each instance.
- The first test used the standard training data set, as described in (Section 3.4).
- The model produced was tested for accuracy using the continuous test file.
- Each test was conducted 3 times, to ensure consistency in results.

For test 2:

- The microphone was placed 0.2m away from the sound source, with an average digital input level of [-12, -6]; these represent good conditions for the system.
- A continuous test file was produced, containing the 15 'clean' instances from the field recording data, with intervals of a random duration present between each instance.
- The second test used the standard training data set, with the inclusion of the 15 'clean' field recording instances; a new model was produced based on this training data set.

- The model produced was tested for accuracy using the continuous test file.
- Each test was conducted 3 times, to ensure consistency in results.

The results of this experiment are presented in (Section 4.5), and discussed in (Section 6.5.2.4).

3.7 Conclusion

As part of this thesis, original software has been developed, which is theoretically capable of detecting sheep vocalisations in real-time. In order to test the main objective of the thesis (Section 1.8), a series of experiments have been conducted, aimed at investigating different aspects of system performance, and range from moderate difficulty (e.g. experiment 1) to very challenging (e.g. experiment 4). Through these experiments, the secondary objectives of the thesis will also be explored, as the results will go some way to determining the viability of these novel approaches. (Section 4) will provide the results of the aforementioned experiments, and (Section 5.2) will discuss the implications of these results for the system's future development.

Chapter 4

Results

4.1 Overview

The results are presented for each of the four experiments outlined in (Section 3.6). For each experiment, the model parameters, test data, and preliminary offline testing performance are included for reference. Performance metrics and statistical measures will be explained where appropriate, but further consideration of the implication of the results will be saved for (Section 5).

4.2 Experiment 1: Real-Time Detection of Sheep Vocalisations Using a Mixed Data Set

The aim of this experiment was to test whether or not it was possible to detection sheep vocalisations in real-time, whilst simultaneously rejecting instances of the negative class. Two individual tests were conducted as part of this experiment, each with a different match threshold, in order to separate the effect this feature may have on perceived performance accuracy. (Fig 4.1) shows the results obtained from this experiment.

Model Information		
Kernel	C-value	Gamma
RBF	0.5	0.25

Test Data	
Positive	Negative
10	40

Offline Test Results										
TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
10	0	40	0	100	0	100	0	100	100	100	100

Experiment 1 - Test 1													
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	0.2m	[-12. -6]	9	1	39	1	90	2.5	97.5	10	90	96	90	93.67497
Experiment 1 - Test 2														
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
6	0.2m	[-12. -6]	8	0	40	2	80	0	100	20	100	96	89.442719	89.442719

Figure 4.1: Classifier performance from experiment 1. Two separate tests were run, each with a different matching threshold. Distance from the sound source, and input level, reflect good conditions for the system.

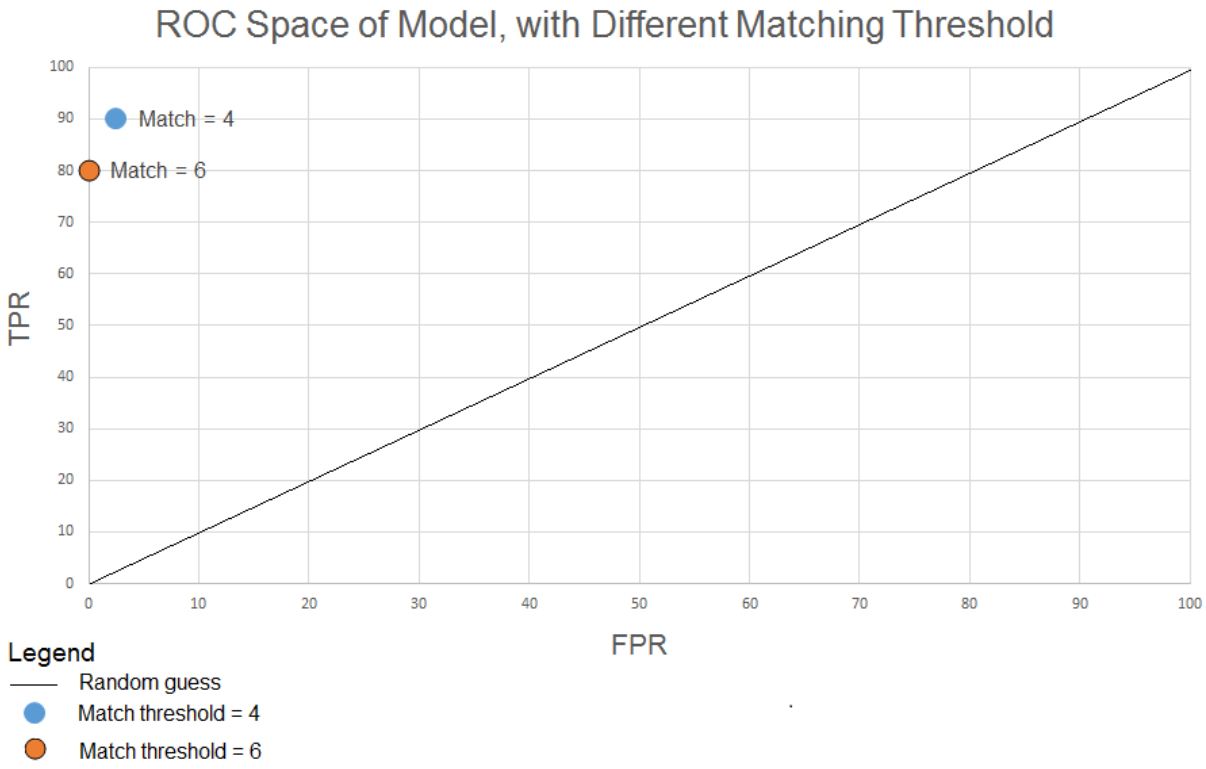

Figure 4.2: ROC space for the model used in the final system, with different matching threshold. The black line represents the result of a random guess. As can be seen, the model performs far better than a random guess at both matching thresholds. The number of sequential matches needed to designate a detection appears to affect accuracy, both in terms of TPR and FPR; more thorough experimentation is needed to find the optimal match threshold for a given model. There was insufficient variation of the model to facilitate the construction of a ROC curve.

4.3 Experiment 2: The Effect of Distance on Real-Time Sheep Vocalisation Detection Accuracy

As a network of field nodes is one of the possible deployments for this system, it is necessary to test the effect distance, and reduced input levels, have on system accuracy. Livestock are dynamic by nature, and therefore it is highly likely that vocalisations will take place at less than ideal distances from the sound source. As an increased distance tends to also reduce input level, this experiment aims to test the effect of both these variables.(Fig 4.3) shows the results obtained from this experiment.

Model Information		
Kernel	C-value	Gamma
RBF	0.5	0.25

Test Data	
Positive	Negative
10	40

Offline Test Results											
TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
10	0	40	0	100	0	100	0	100	100	100	100

Experiment 2 - Test 1													
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	0.2m	[-12. -6]	9	1	39	1	90	2.5	97.5	10	90	96	90	93.67497
Experiment 2 - Test 2														
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	0.4m	[-18, -12]	8	0	40	2	80	0	100	20	100	96	89.442719	89.442719
Experiment 2 - Test 3														
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	1m	[-24, -18]	3	0	40	7	30	0	100	70	100	86	54.772256	54.772256
Experiment 2 - Test 4														
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	1m	[-18, -12]	6	1	39	4	60	2.5	97.5	40	85.714286	90	71.713717	76.485293
Experiment 2 - Test 5														
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	1.5m	[-24, -18]	2	0	40	8	20	0	100	80	100	84	44.72136	44.72136
Experiment 2 - Test 6														
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	1.5m	[-18, -12]	5	0	40	5	50	0	100	50	100	90	70.710678	70.710678

Figure 4.3: Classification performance from experiment 2. Tests are performed at different distances, with the furtherest two also being tested at different input levels, in an attempt to separate each effect. Performance metrics are given for each test to facilitate comparison.

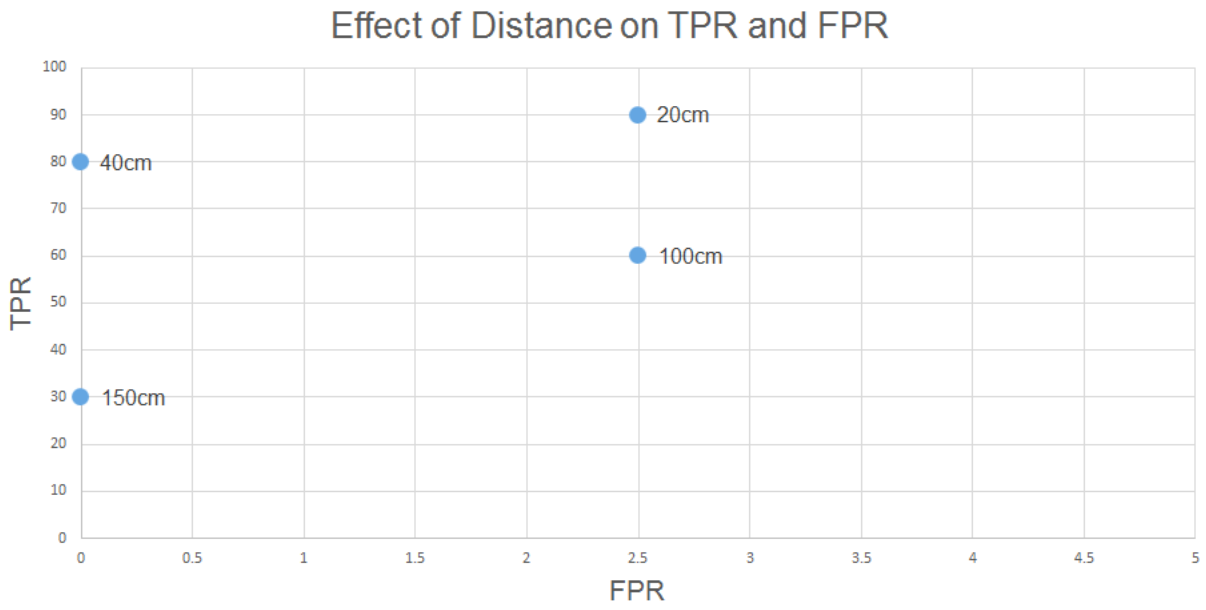

Figure 4.4: The effect of distance on TPR and FPR. As can be seen, as distance from the sound source increases, TPR decreases, and in one instance, FPR increases. It appears distance has a negative effect on TPR. Distance may have a negative effect on FPR, although their is insufficient evidence to make a definitive statement; more testing is necessary to prove the existence of this effect.

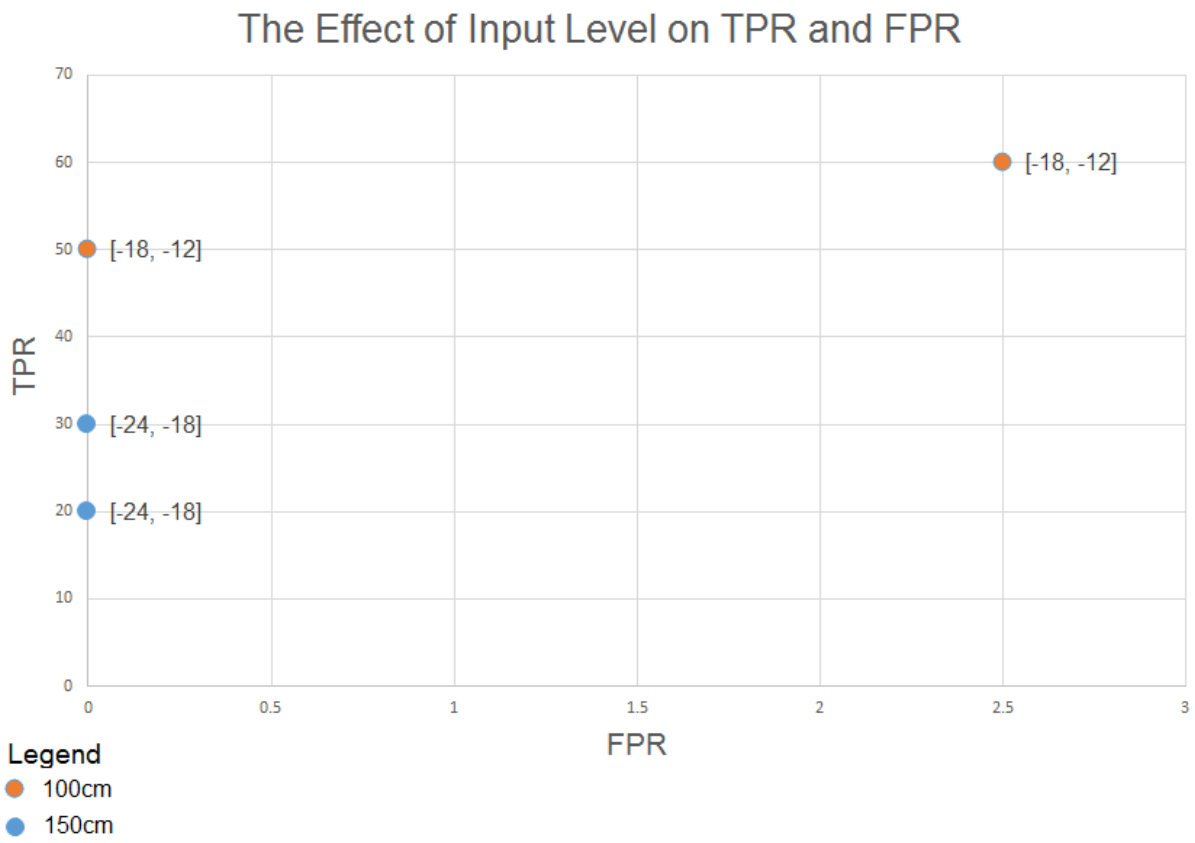

Figure 4.5: The effect of input level on TPR and FPR, for both the 100cm and 150cm tests. In order to separate the effect of input level from distance, the input level was adjusted at both distances. As can be seen, reduced input level does appear to negatively affect TPR. The effect on FPR is not clear, and further testing would be needed to demonstrate a link.

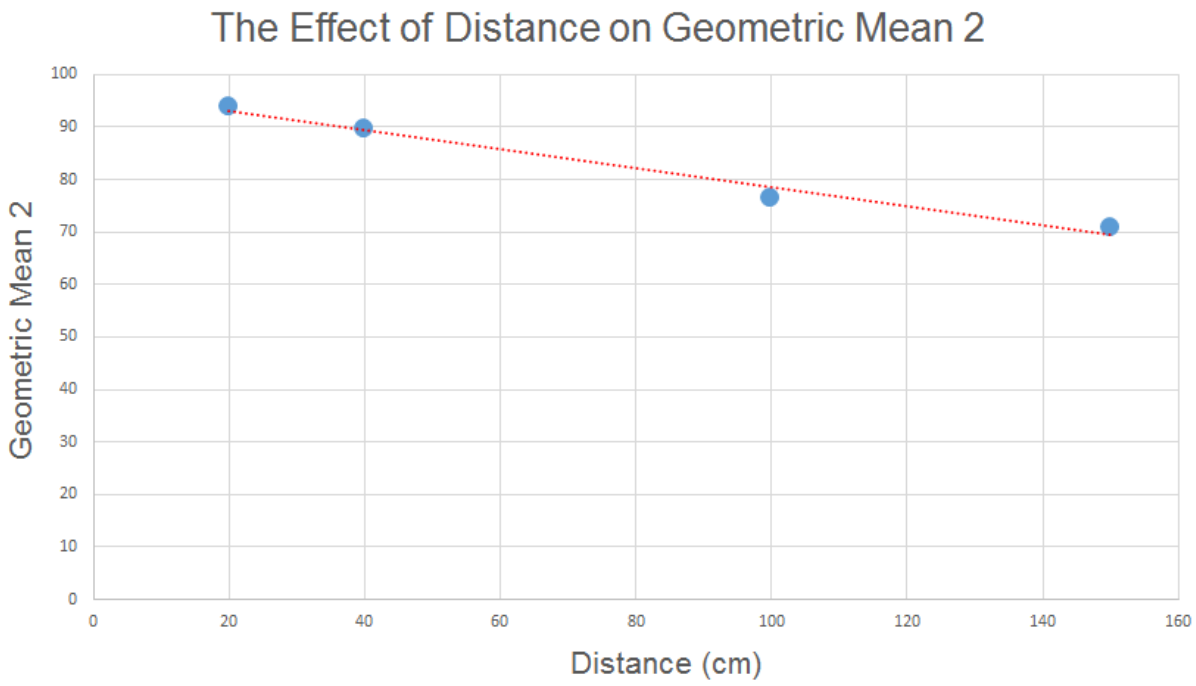

Figure 4.6: The effect of distance on geometric mean 2. This metric takes into account both TPR and TNR. The trend line shows that as distance increases, gMean2 decreases, further reinforcing the findings displayed in (Fig 4.4).

4.4 Experiment 3: Real-Time Segmentation of Sheep Vocalisations

The system is effectively a binary classifier, with only two states: detection or rejection. As subsequent windows frequently record a detection, and indeed must to pass the match threshold, it is possible to ascertain the duration of vocalisation activity, from the perspective of the system. Each window contains 0.1 second of new information (i.e. 1 second window / 10 overlap rate), and this information can be used to determine duration. The system's results are compared to a human performing the same task, as this can be considered the 'gold standard' of duration detection. Although the system was not designed with segmentation in mind, this experiment does give an insight into how accurately the system is able to discern when a vocalisation is taking place, and how many subsequent windows are recording a detection.

As this is a prediction problem, different performance metrics were needed to determine the rate of error for the system, and as such, MSE and MAE (see Section 2.2.5) were calculated based on the test results.

Model Information		
Kernel	C-value	Gamma
RBF	0.5	0.25

Test Data	
Positive	Negative
10	40

Offline Test Results											
TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
10	0	40	0	100	0	100	0	100	100	100	100

Experiment 3 - Test 1											
Matches	Distance	Level									
4	0.2m	[-12. -6]	VOCAL 1	VOCAL 2	VOCAL 3	VOCAL 4	VOCAL 5	VOCAL 6	VOCAL 7	VOCAL 8	VOCAL 9	VOCAL 10
Sytem Durations:			1	1	1.3	1.3	1.4	0.4	1.3	0.5	1.1	miss
Actual Durations:			0.63	0.4	0.66	0.45	0.75	0.75	0.65	0.6	0.7	0.72
Difference:			0.37	0.6	0.64	0.85	0.65	-0.35	0.65	-0.1	0.4	N\A
			VOCAL 11	VOCAL 12	VOCAL 13	VOCAL 14	VOCAL 15	VOCAL 16	VOCAL 17	VOCAL 18	VOCAL 19	VOCAL 20
Sytem Durations:			1.3	1.2	1.1	1.2	miss	miss	1.3	1.1	1.1	1.3
Actual Durations:			0.78	0.5	0.54	0.49	0.66	0.6	0.7	0.47	0.33	0.59
Difference:			0.52	0.7	0.56	0.71	N\A	N\A	0.6	0.63	0.77	0.71

TP	FN	TPR	FNR	MEAN SQR	MAE
17	3	85	15	0.36461765	0.57705882

Figure 4.7: Classifier performance for experiment 3, showing the duration determined for each vocalisation detected. The true durations are shown for comparison. As can be seen, the system consistently over-estimates the duration of vocalisation activity, and the relatively high MSE and MAE (0 = no error, 1 = all error) reflect this.

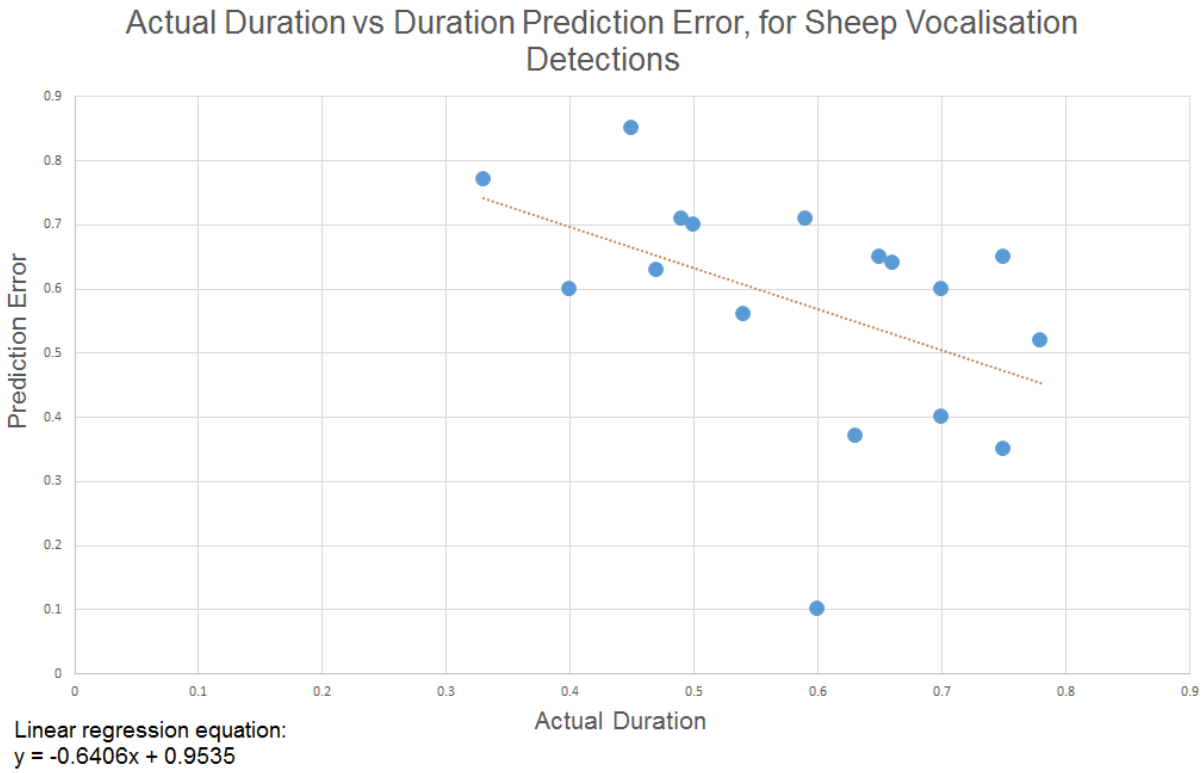

Figure 4.8: The actual duration vs the duration prediction error. The purpose of this figure is to determine if the actual duration of the vocalisation has some effect on the size of prediction error. From the composition of the data points, and the fact that the slope for the linear regression fit is significantly different from 0, the duration of the call appears to have no significant effect on the size of the prediction error, with the system performing relatively poorly on vocalisations of different durations.

4.5 Experiment 4: Real-Time Detection of Sheep Vocalisations Using Noisy Data

Experiment 4 poses the biggest challenge to the system, in that it tests if the system can detect sheep vocalisations from a noisy field recording previously deemed to be unsuitable for development. This experiment uses both the standard model from the final system, as well as a new model created from a training data set that includes the ‘clean‘ instances from the field recording data (see Section 3.4).

Model Information		
Kernel	C-value	Gamma
RBF	0.5	0.25

Test Data	
Positive	Negative
10	40

Offline Test Results											
TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
10	0	40	0	100	0	100	0	100	100	100	100

Experiment 4 - Test 1													
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
4	0.2m	[-12. -6]	0	0	0	0	0	0	0	0	0	0	0	0

Model Information		
Kernel	C-value	Gamma
RBF	0.5	8

Test Data	
Positive	Negative
25	40

Offline Test Results											
TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
21	1	39	4	84	2.5	97.5	16	95.4545	92.3077	89.5443	90.4986

Experiment 4 - Test 2													
Matches	Distance	Level	TP	FP	TN	FN	TPR	FPR	TNR	FNR	P	ACC	GMEAN1	GMEAN2
2	0.2m	[-12. -6]	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.9: Classifier performance from experiment 4. At no point was the system able to positively identify vocalisations from the field recording data; there are a number of possible reasons for this result (see 5.2.4).

Chapter 5

Discussion

5.1 Overview

In this section, the results presented for each conducted experiment (Section 4) will be interpreted, gleaning insight into aspects of the software‘s performance. Possible reasons for the results will be hypothesised, shortcomings in the experiments will be identified, and potential solutions will be explored. The ramifications of these results will be discussed as they related to each system component, and how this information will be used to inform future work.

5.2 Discussion of Results

5.2.1 Experiment 1:

Real-Time Detection of Sheep Vocalisations Using a Mixed Data Set

Experiment 1 was designed to test the average performance of the system, under ideal conditions, using a data set containing both positive and negative examples. The system performed admirably during this experiment, producing a high TPR and low FPR for both matching thresholds used. Even though the positive class test instances were few in quantity, the examples used in the test data set were not included in the training data, which adds some credibility to the findings. The results of this experiment indicate that the system performs well when detecting sheep vocalisations from the same source as the training data, but it does not indicate how it will perform on vocalisations from a different source; see (Section 5.4.1) for discussion concerning the limitations of the training data.

The trade-off between TPR and FPR is always a consideration in software of this kind,

and the nature of the problem dictated that a low FPR was the most important metric, provided a reasonable TPR could be obtained. In this system, it would be advantageous to provide less positive alerts overall, rather than more positive alerts amalgamated with a number of false alerts. It is better to be certain of vocalisation activity, rather than overestimating, or wasting the user‘s time with false alerts. With these objectives in mind, the results of experiment 1 have shown that, under test conditions, the system possesses an appropriate level of negative class rejection.

As part of this experiment, the match threshold was adjusted to illustrate its effect on performance metrics, particularly TPR and FPR. As was seen, as the match threshold was increased, both TPR and FPR decreased. It was observed that a higher match threshold, particularly a threshold that was larger than half the overlap rate (i.e. 6), was effective in suppressing the number of multiple detection alerts. The variation of match threshold level was something of an afterthought for this experiment, and further testing is needed to completely understand the effect that the matching threshold has on detection accuracy and alert frequency, and to find the optimal threshold level for a particular model; see (Section 5.4.4) for further discussion.

Overall, this experiment showed that it is possible to detect sheep vocalisations in real-time, using a combination of a DDWT, low-computational statistical features, and an SVDD. The conditions for this experiment were considered good, from the perspective of the system, and in many ways reflect a quiet indoor (e.g. barn) setting; success in this domain is certainly advantageous for possible future deployments. Further testing is necessary to ascertain the viability of an outdoor system, although these preliminary findings are certainly encouraging.

5.2.2 Experiment 2:

The Effect of Distance on Real-Time Sheep Vocalisation Detection Accuracy

Experiment 2 aimed to test the effect of distance on detection accuracy, but it was difficult to separate this effect from the many others it is associated with, particularly that of reduced input level, as the two tend to be linked. The effect distance has on a sound source may vary, and is affected by variables such as topology and weather. Changes in distance from the sound source may also produce changes in other sound-related variables, such as phase, frequency colouration, increased background noise, acoustics, and other sounds related phenomenon, which can all affect the characteristics of the capture sound, thereby clouding the true effect distance has on detection accuracy. In summary,

it is very difficult to isolate distance as a single variable, and due to this fact, the experiment must be viewed as preliminary in nature; far more robust testing would be needed to quantify the effect of distance.

One of the most immediate effect of an increase in distance from the source is a reduction in input level. In most circumstances, as the sound source moves away from the capture device, the input level received by the system decreases, as the increased distance reduces the sounds perceived amplitude; this factor may also affect detection accuracy. A decrease in input level may also have additional effects, such as an increase in background noise, as the drop in level allows other sounds to become relatively louder by comparison.

The first test in this experiment was the same as experiment 1, and formed a baseline reading from which to make comparisons. The second test, from 0.4m away, also reduced the input level, and it was found that this marginally reduced TPR and FPR. For tests 3 and 4, care was taken to try to separate the effect of input level from that of distance, and the each test was conducted using two different input levels; this was achieved by increasing the gain on the preamp (i.e. overall input is higher). In regards to distance, it was observed that an increase in distance consistently produced a decrease in TPR, regardless of input level; it can be asserted, albeit in a preliminary capacity, that distance does affect detection accuracy.

As tests 3 and 4 used two different input levels, the effect of reduced input level could be observed, admittedly in a preparatory manner. At both distances, a reduction of input level had a negative effect on TPR, and so it can be hypothesised that a reduction in input level also affects detection accuracy; a more thorough examination is needed to prove and quantify this effect.

In some examples, namely test 1 and 4, FPR was affected by a change in distance, but it opposing ways, in that an increase in distance between tests 1 and 2 lowered FPR, whereas this produced a higher FPR between tests 3 and 4. As this phenomenon was only observed in a low percentage of cases, it cannot be asserted that the effect exists, and further experiments are needed to understand the relationship between FPR and distance, for software of this kind.

There are a number of reasons that both distance and input level may have an effect of system performance, and additional testing is needed to isolate which components in the signal chain are responsible. As a preliminary hypothesis, the DDWT may be sensitive

to changes in amplitude and phase, due to its ability to take into account the temporal domain. As this feature was originally viewed as advantageous, and indeed was one of the deciding factors in selecting this approach, it is hoped that further research can aid in understanding the effects distance and input level have on the DDWTs output. Another possibility is that the features used to generalise the DDWTs coefficient sub-bands could be sensitive to changes in distance and input level, or that they are generally too coarse to fully explain these changes. Again, an empirical comparison of DDWT-based feature extraction methods, tested under different conditions, is needed. A final consideration is that the training and testing data used were not representative of the differences in distance and input level that would arise in a real-world setting, particularly as all the training data instances were recorded at the same distance; see (Section 5.4.1) for further discussion.

Experiment 2 demonstrated that the system is sensitive to changes in both distance and input level, and that these factors can negatively affect detection accuracy. Which components are responsible for this sensitivity, and how this can be addressed in the future, will form the focus of future work; see (Section 6.1).

5.2.3 Experiment 3:

Real-Time Segmentation of Sheep Vocalisations

This experimented investigated the systems ability to determine the duration of a vocalisation event, and although duration calculation was not an intended feature of the software, the experiment gave an insight into exactly how long the classifier remained in a positive state. Due to the OCC nature of the problem, the SVDD classifier is essentially a binary classifier, with only a positive (detection) and negative (rejection) state. Therefore, the number of sequential windows that record a detection is of interest, as this illustrates the duration of positive states. Also, as the window overlap rate was relatively high (i.e. overlap = 10), the amount of signal information needed to determine a match can be determined (i.e. what percentage of a sound needs to be captured to identify a vocalisation).

It was found that the system generally overestimated the duration of vocalisation events, and in the majority of cases, did not accurately calculate duration. From this observation, some insight was gleaned into the average time the classifier remains in a positive state, demonstrating that it tends to linger in this state for longer than is necessary. The resulting MSE and MAE were relatively high, but a comparison to other systems is

needed to truly understand the meaning of these measures.

An interesting by-product of this experiment is that the TPR of the system was calculated under different conditions, producing a rate of 85%; this is in line with the results of experiment 1. This result further reinforces the performance observed during experiment 1, adding weight to the hypothesis that it is possible to detect sheep vocalisations in real-time using the methods outlined.

There are numerous possible reasons for the systems relatively poor performance in duration calculation. The high window overlap rate may be a factor, due to the low percentage of new information that is contained in each window, making consecutive detections too frequent. The classifier, or the features it has been trained on, may lack discriminative ability, allowing an undesirable amount of sequential detections; this may also be due to issues surrounding the training data (Section 5.4.1).

Similarly to experiments 1 and 2, future work should focus on attempting to isolate the components responsible for the less than ideal performance observed in the results from this experiment; the inclusion of a dedicated duration calculation feature would also be beneficial.

5.2.4 Experiment 4:

Real-Time Detection of Sheep Vocalisations Using Noisy Data

Experiment 4 posed the greatest challenge to the system, as it involved the detection of sheep vocalisations obtained from field recordings containing a high level of noise. In the first test, none of these instances were included in the training data set for the model, whereas in test 2 they were. The results from both tests were underwhelming, with no successful detections recorded.

In many ways, the results were to be expected, as the test data instances were very demanding for software at the current level of development. Although the test instances were of sheep vocalisations, they possessed substantially different characteristics, in that they were recorded in a different location (i.e. outdoors), contained far more noise (e.g. background sounds, microphone hiss, etc), were of different individuals (gender, age, and breed are unknown), and were generally at a relatively lower amplitude. Test 1, using the standard training data set, produced no detections, which is unsurprising given that no examples of the field data were included, and the issues surrounding the training data

in general (see Section 5.4.1).

In test 2, it was hoped that by incorporating the field recording instances into the training data, accuracy could be improved, but this was not the case; TPR remained at 0. The low number of instances (15) may be the cause for the ineffective incorporation of these instances, as there are not enough to allow the SVDD to truly understand the relationships contained in the features. Also, the features extraction approaches used are most likely not noise-robust, and it is hypothesised that further noise-reduction techniques, such as spectral filtering, would be necessary to improve performance in such challenging environments.

The results from experiment 4 were disappointing, in terms of the objectives of the thesis, but completely expected and reasonable. Experiment 4 served the purpose of testing the system‘s performance in a challenging scenario, thereby highlighting potential improvements and research directions that may be pursed.

5.3 Implications of Results on Thesis Objectives

5.3.1 Main Objective: The viability of a system capable of detecting sheep vocalisations in real-time

At this point in research, it appears that a system capable of detecting sheep vocalisations in real-time is viable. The results from experiments 1, 2, and 3 demonstrate this fact, as significantly high levels of TPR, and low levels of FPR, were achieved. Taking into account latency (i.e. a delay in processing) caused by the sound card [175], alerts were provided by the system in an immediate

The results have indicated a number of areas for improvement, and further research is needed to fully develop this system; it is believed that none of these issues present insurmountable barriers to successful development. Despite the fact there were a number of limitations within the experiments, it was still demonstrated that real-time detection of sheep vocalisations is possible.

5.3.2 Secondary Objectives

The use of DWTs in Audio Analysis

Although it has not been unequivocally demonstrated that DDWTs are suitable for real-time vocalisation detection tasks, it has certainly been shown that their use remains plausible in audio analysis, reinforcing the findings of other studies in the literature (see Section 2.1.6). The results from experiment 1 showed that features derived from a DDWT could be utilised by a machine learning model to discriminate between sound events of differing different types. Preliminary findings from experiment 2 suggest the DDWT may be sensitive to changes in input level and distance, or the subsequent effects caused by changes to these variables, but this merely highlights the need for further research. More sophisticated approaches could be explored, such as adding more taps, or using different algorithms (Section 2.1.6), to determine if this could improve detection accuracy. An empirical comparison of DWT techniques and feature extraction approaches, for use in audio analysis, is greatly needed, as this has not been adequately addressed in the literature.

The use of SVDDs to Classify Sheep Vocalisations

The SVDD appears to be a robust OCC approach, producing statistically significant results for all performance metrics used, particularly in experiment 1. The SVDD displayed an acceptable level of negative class rejection, a key trait when assessing the overall performance of an algorithm such as this. The process for selecting optimal model parameters was relatively straight forward, and was greatly aided by the inclusion of an optimisation process. The ideal features for classifying audio data with an SVDD remain predominately unknown, and requires further inquiry. The libSVM library was found to be an easy to use implementation of the SVDD, and is recommended for future use.

The Use of Low-Computational Statistical Features for DWT Sub-Band Dimensionality Reduction

Even though the results from the experiments were encouraging, the suitability of the statistical features used in the software, for audio-based discrimination applications, is still undecided. Preliminary findings suggest that these features can be used for accurate discrimination, as was demonstrated in experiments 1, 2, and 3, which reflects the results of other studies in the literature (see Section 2.2.2). To fully understand the performance of features of this type, an audio analysis benchmarking study is needed, comparing them

to other approaches found in the literature; this would allow the most appropriate strategy to be adopted.

The contribution of feature selection to the poor performance in experiment 4 can only be hypothesised, due to the difficulty in isolating the effect each component has on performance, in a system at this stage of development. It is possible that the features used in the software were too coarse to provide adequate discrimination potential in the challenging environment posed by experiment 4, particularly as they reduced the original, dimensionally vast sub-bands (e.g. 22050 data points in size) to a single measure, and only three measures were considered per sub-band. If this were the case, these features would still hold potential for indoor, or on-collar monitoring device applications.

As the features outlined were not specifically designed for audio-based applications, there is the possibility that domain-specific approaches, particularly methods based on biological structures, such as MFCC-based approaches [2, 1, 287], may be better suited to this application. Future work should focus on comparing different DWT feature extraction approaches (see Section 2.1.6), to each other, as well as to spectral-based approaches (see Section 2.1.5), to truly assert the suitability of low-computational statistical measures for use in the software.

Overall, the low-computational statistical features used in the software performed admirably, but further study is need to truly understand their performance potential.

The Effect of Distance on Matching Accuracy

Despite the limitations of the experiment design, experiment 2 demonstrated that distance has an effect on the detection accuracy of the system. The possible reasons for this effect have already been discussed at length in (Section 5.2.3), but it can be stated that this effect is real, but how much of the observed accuracy reduction is primarily due to the change in distance alone remains unknown. More comprehensive statistically designed tests can be implemented in the future, and forthcoming development will focus on distance and noise robust approaches. This is in line with the thesis objectives, as the focus of this work was the development of a prototype system, to assert the viability of a real-time vocalisation detection system.

The Potential to Run the Software on Low-Resource Systems
While conducting the experiments, resource usage was monitored, and it was found that the software could safely run on a low-resource system, such as a Raspberry Pi. In fact, there was still considerable resource headroom, which would allow further complexity to be added to the signal chain, something that will be mandatory if performance is to be increased.

During real-time operation, the system used approximately the following resources:

- CPU: 320 MHz
- Memory: 1.7 M

There is little to no disk access required during real-time operation, so measuring this resource was deemed irrelevant.

5.4 Implications of Results on Other Aspects of the System

5.4.1 Training Data

The main training data set used during experimentation was derived from a single indoor recording of a ewe calling her lamb, and was recorded indoors. This provides a very limited representation of sheep vocalisation activity, and creates a scenario that encourages the model to have a high level of bias, in that the model is learning to detect one source of vocalisation (i.e. one individual, indoors), rather than any source (i.e. different individuals, in different settings). The training data was also quite small in size, again, not providing enough information for the model to adequately understand the patterns that have been shown to underlie all sheep vocalisations (see Section 2.3.5); the training data is not representative of the entire class, but rather one individual.

At this stage of development, the training data set provides a ‘proof of concept‘, but a much more substantial data set will be needed for future work. To truly represent the sheep vocalisation class, it would be necessary to acquire instances of vocalisations in a variety of settings (e.g. on pasture, indoors, different weather, etc), as well as distances from the capture device (e.g. close, medium, far, etc). These instances could be extracted and categorised from field recordings, rather than setting up ‘recording sessions‘ for each domain; this would save time and be more realistic overall. Vocalisations from a number

of different individuals, breeds, genders, and ages would also be advantageous. Essentially, a highly inclusive sheep vocalisation data set needs to be produced to truly test the viability and capabilities of a real-time vocalisation detection system.

5.4.2 Test Data

The main test data set suffered from the same problems as the training data, in that it was not particularly representative of the real positive and negative classes that will be encountered by a system in the field; the size of the test data set was also small. It is true that the test vocalisation instances used were not part of the training data, and this is certainly beneficial when testing the performance of the system, but the quantity of these instances was very low (i.e. 10), meaning that performance metrics could be easily skewed by even a single false negative (e.g. 1 false negative results in a 10% decrease in the TPR). Also, even though test instances were randomly removed from the training data set, there is no guarantee that they are representative of the target class, or that they provide an adequate challenge to the system; there are too few instances to provide reliable sampling. The reality is that obtaining instances of sheep vocalisations, in a 'natural' setting, is very difficult, due to the low quantity of examples present in most field recordings, and because of the substantial processing time required to extract instances from these recordings: a large amount of time and effort is needed to build an adequate test (and training) set.

Even though this is an OCC problem, in which the negative class is not represented in the training data, it became apparent during development that it was essential to include examples of the negative class during real-time performance testing. The reason for this, is that it was easy to create a model that appeared to perform well on vocalisation instances, only to find that it considered most sounds to be positive; the model possessed far too much variance. Only by observing the system when it encountered other sounds did its true performance begin to become apparent. The issue surrounding the negative instances in the test data set is that there was little consistency between them, in that they were derived from multiple sources, with no knowledge of the conditions in which they were recorded; most of the sounds were taken from sound libraries and field recordings. In order to truly test how the system responds to examples of the negative class, a consistent test data set needs to be created, with location-based recordings of sounds that will likely be encountered by the system. The test data set created for the experiments conducted as part of this thesis certainly served their purpose, in that they did contain some challenging sounds (e.g. dog bark, rain, talking), and therefore provided a good

preliminary test for the software, but a more robust test data set will have to be created for future testing.

5.4.3 Digital Audio Quality

The sample rate chosen is theoretically high enough to eliminate any problems associated with aliasing, particularly as most of the important frequency activity being detected exists within a much lower range than the Nyquist frequency (Fig 1.4). To be absolutely sure of system integrity, a higher sample rate could be used, in line with the principle of oversampling (see Section 2.1.3), but the perceived increase in quality may not warrant the increase in memory usage and computation time associated with higher sample rates.

The bit rate used during development and testing is arguably too low, especially considering 32-bit systems are now possible. A higher bit rate would allow the utilisation of more precise variable types, thereby facilitating more accurate measurements. As with the sample rate, there would be higher resource and computational overheads, but it is believed that the benefit of improved accuracy would justify these costs.

5.4.4 Capture Window and Match Threshold

The effect of overlap rate on performance was highlighted during experimentation, and further research is necessary to fully understand this. It was common for the system to register multiple sequential detections, but the results of experiment 3 highlighted the redundancy and inaccuracy of the current approach. At present, too many windows are recording a detection, resulting in multiple detection alerts being produced, and inaccurate vocalisation duration calculations.

Match threshold and window overlap rate may be intrinsically linked, as more windows per second will likely produce more consecutive detections, therefore forcing the match threshold to raise in order to compensate. The overlap rate should inform the selection of the match threshold in a much more automated way; the current rates were found manually during development. Further testing is needed to determine the ideal strategy for nominating overlap rate, window length, and match threshold.

5.5 Other Considerations

5.5.1 Min-Max Normalisation

It is important to consider how the minimum and maximum values are decided when normalising new data instances, as some instances may contain values that supersede the values calculated during training. For the sake of simplicity, the system merely uses the original min and max values calculated during training, and any values that fall outside of the range are forced back in (i.e. assuming the range is [0, +1], if a new max value is encountered, it is just set to 1). A differing approach was tested during development, whereby a new min or max could be set if it was exceeded, allowing subsequent windows to fall within the new range, but this produced subpar results, mainly due to the tendency of the range to move or grow, shifting the actual values obtained. A more advanced approach to min-max normalisation is required, as this may have been a source of error in the signal chain.

5.5.2 Audio Normalisation

The current system uses a basic peak normalisation approach, which admittedly can be sensitive to transient peaks (i.e. one loud peak in an otherwise quiet file will diminish the amplification applied to it), and does not guarantee homogeneous amplitudes between data instances. To ensure that captured instances are being properly normalised, a more progressive technique that takes into account the possible presence of uncharacteristic peaks is needed.

5.5.3 Window Padding

Although four window padding schemes were implemented, they were not empirically compared for performance. It is unknown whether the window padding strategy employed has a significant effect on performance when applying a DWT. The padding types implemented, and others in the literature [38] should be analytically compared to ensure signal chain integrity.

5.5.4 Window Functions

From anecdotal observations during development, the addition of a window function before applying a DDWT did not seem to improve performance, but this effect has not been empirically proven. Also, only one window function was used in the software, which says nothing of the effect the multitude of other functions illustrated in the literature may

have on performance (see Section 2.1.4); more research is needed in this area.

5.5.5 SVDD Optimisation

Although the optimisation routine greatly reduced the time taken to select appropriate model parameters, it is quite simple in nature, and the use of threads to reduce runtime is questionable. In the current implementation, a new thread is created to test each model, quickly taking up system resources and creating additional overheads due to thread management; future implementations should employ a thread pooling strategy. The filtering and ranking of models based on performance metrics is acceptable as a quick method of parameter selection, but is susceptible to bias, in that the metrics are based on how well a model performs with a given test set.

5.5.6 Equipment

The frequency response of both speakers and microphones is extremely important in audio detection analysis, and is often quite overlooked when reporting on experiment results. It is standard practice for audio manufacturing companies to produce speakers that deliberately colour the sound by adding more of certain frequency ranges, usually to add bass boost or a clearer vocal range, and to make microphones that are designed to accentuate particular frequency ranges, for use on particular instruments, but these types of features are the antithesis of audio analysis requirements. The HS8 speakers and NT5 microphone appeared to perform impeccably, but some response testing would be needed to be completely sure of their integrity.

The weakest link in the systems signal chain was certainly the sound card, as this was an on-board model, which produced quite high levels of digital noise, and suffered from unacceptable latency. For deployment, a dedicated device would be designed, and a much better sound card could be implemented; the Raspberry Pi system outlined in (Section 3.5.3) already contains a better card than the one used during testing. For future work, a more suitable sound card will need to be obtained to ensure integrity in the signal chain, as well as facilitating the recording of training and testing data.

5.6 Conclusion

As been shown, the software developed, and experiments conducted, have met the objectives of the thesis, and have added evidence to the hypothesises proposed. The direction of future work, as well as a full conclusion, are given in (Section 6).

Chapter 6

Conclusion

6.1 Future Work

As the feasibility of a real-time vocalisation detection system has now been demonstrated, future work will focus on refining the system and improving overall performance, and the author plans to conduct this research in a PhD capacity.

It will be necessary to survey and revise each component in the signal chain, isolating each one in order to understand how it affects both overall performance, and other components it interacts with. Particular addition will be paid to data transformation algorithms, with further exploration into both Wavelet-based (see Section 2.1.6), as well as spectral (see Section 2.1.5) approaches. The use of DWTs in audio analysis will likely remain a research focus, especially considering the encourages results obtained during experimentation.

The comparison of feature extraction methods specifically designed for the transformation approach selected will need to be conducted, with further investigation into the robustness of the low-computational features examined in this thesis. Features, and other DSP techniques, designed to improve performance under noisy conditions will also be pursued.

Although the SVDD performed well during experimentation, other machine learning models should be researched, particularly if the system is to move away from strictly binary classification and into multi-class problems. For example, the system could be extended to classify sheep vocalisations by class, such as age groups (e.g. lamb, adult), call type (e.g. ewe calling lamb, distressed lamb, alert call, etc), or distance from the device. Although the SVDD is designed for OCC problems, is also possible to combine OCC classifiers in order to facilitate multi-class classification [409, 159, 264, 288]. Other

architectures will also be reviewed, particularly ANNs (see Section 2.2), as they are currently receiving a lot of attention in the literature.

As has been thoroughly conveyed in the thesis, there is an immediate need for a more comprehensive and robust set of training and testing data, and this will be procured from the many field recordings undertaken by PARG. The process of extracting instances, and the large size of the data files (prohibitive with slow internet speeds), was the main barrier to creating a better data set, but this can be easily surmounted in an on-campus PhD capacity.

The need for power-efficient algorithms will inform future development, as regardless of the deployment, there is a high probability that dedicated units will need to be developed in order to run the software, and they will need to run for as long as possible on the limited power provided.

The end goal is to create a fully functional acoustic livestock welfare monitoring system, which can be demonstrated to both researchers and industry. It is hoped that stakeholders will then be able to realise the full benefits such a system has to offer.

6.2 Final Conclusions

The software developed, and the subsequent experiments that were conducted to test its capabilities, were successful in meeting the objectives defined in this thesis (see Section 1.8). Although these questions have not been definitively answered, due to the limitations of the software and the nature of the experiments, evidence has been produced that justifies further development of a real-time sheep vocalisation detection system. Future areas of research have been highlighted, and possible strategies for the improvement of different aspects of the system have been suggested.

It appears a real-time sheep vocalisation detection system is feasible, and it is hypothesised that a demonstrable, 'real-world' system will be produced in the future. The software created acts as a 'proof of concept', clearly demonstrating that it is possible to detect sheep vocalisations in real-time, but that further refinement is necessary to develop a more robust system. Evidence for the successful use of DDWTs in audio analysis has been acquired, providing a novel area of research for sound detection-related problems. The employment of low-computation statistical measures in deriving features

from DDWT coefficient sub-bands has been explored, and whilst reasonable results were obtained, further experimentation is needed to understand the performance, and possible limitations, associated with this approach. The use of an SVDD model to discriminate audio-based DWT-derived features was examined, and from preliminary experiments, the model produced satisfactory results. Although the effect of distance was hard to isolate from other related variables, it appears that distance, as well as input level, do have an effect on the overall detection accuracy of the system. Due to the efficient and minimalist signal chain, and from monitoring resource usage during operation, the software should be capable of running on a low-resource device, such as a Raspberry Pi, and in fact, it may be possible to add additional complexity to the signal chain.

The need for a real-time sheep vocalisation detection system, which could form part of wider acoustic farm welfare monitoring system, has been demonstrated (see Section 2.3.5), and it is hoped that the research conducted as part of this thesis provides sufficient evidence and justification for future development, in order to make this system a reality.

Appendix A

Acronyms

ANN Artificial Neural Network

DSP Digital Signal Processing

DDWT Daubechies Discrete Wavelet Transform

DWT Discrete Wavelet Transform

GFCC Gammatone Frequency Cepstral Coefficient

GMM Gaussian Mixture Model

HMM Hidden Markov Model

LOOCV Leave-One-Out Cross-Validation

MAE Mean Absolute Error

MFCC Mel-Frequency Cepstral Coefficient

MSE Mean Squared Error

OCC One-Class Classification

OSVM One-Class Support Vector Machine

PARG Precision Agriculture Research Group

PLF Precision Livestock Farming

PLP Perceptual Linear Prediction Coefficient

PNCC Power-Normalized Cepstral Coefficient

PSoL Positive Sample Only Learning

RBF Radial Basis Function

SVDD Support Vector Data Description

SVM Support Vector Machine

SVMC Support Vector Mapping Convergence

ZCPA Zero Crossing Peak Amplitude

Appendix B

Software Code

B.1 main.cpp

```
// Main driver for Audio Detector

#include <stdio.h>
#include <vector>
#include <string>

#include "Algorithm.h"
#include "CaptureAudio.h"



int menu() {
    std::cout << "-----------------------------------" << std::endl;
    std::cout << "Select:\n";
    std::cout << "1. Process audio files for SVM model" << std::endl;
    std::cout << "2. Optimise and build SVM model from last audio files"
              << std::endl;
    std::cout << "3. Build default SVM model from last audio files"
              << std::endl;
    std::cout << "4. Run offline SVM test, using last build"
              << std::endl;
    std::cout << "5. Run real-time SVM test, using last build"
              << std::endl;
    std::cout << "0. Quit" << std::endl;
```

```
    std::cout << "------------------------------------" << std::endl;

    int c;
    std::cin >> c;

    return c;

}

// main
int main(int argc, char** argv) {
  Algorithm alg;
  int c = menu();
  std::vector<std::string> testNames;
  alg.getTestNames(testNames);
  std::vector<double> stats;

  while ( c != 0) {
    switch (c) {
      case 1:
        alg.processFiles("svm");
        break;
      case 2:
        alg.buildModel(true);
        break;
      case 3:
        alg.buildModel(false);
        break;
      case 4:
        alg.testModel(NULL, testNames, stats);
        alg.printStats(stats);
        break;
      case 5:
        alg.realTimeTest();
        break;
      case 0:
        return 0;
```

```
            break;
        default:
            std::cout << "Make a valid selection from the menu."
<< std::endl;
            menu();
            break;
    }
    c = menu();
  }
  return 0;
}
```

B.2 ProcessWav.h

```
#ifndef PROCESSWAV_H
#define PROCESSWAV_H

#include <cstdio>
#include <string>
#include <iostream>
#include <fstream>
#include <cstring>
#include <cstdlib>
#include <vector>
#include <stdint.h>


class ProcessWav {
  public:

    std::vector<double> openFile(std::string filename);
    int write16Bit(std::vector<double> dataChunk, std::string filename);

private:
    const char riff[5] = "RIFF", wave[5] = "WAVE",
fmt[5] = "fmt ", data[5] = "data";
```

```
    FILE *fp;
  char fileName;
                char chunkID[5];
                char format[5];
                char subchunk1ID[5];
                char subchunk2ID[5];
                char *soundBuffer8;
                short *soundBuffer16;
                int *soundBuffer32;
                int chunkSize, subchunk1Size, sampleRate,
byteRate, subchunk2Size;
                short audioFormat, numChannels,
blockAlign, bitsPerSample;
                char *writeFile;
                std::string dataDir  = "Data/";
                bool fileRead = false;
                // functions
                int compare(char *a, const char *b, int size);

                void fileInfo();

};

#endif
```

B.3 ProcessWav.cpp

```
#include "ProcessWav.h"


int ProcessWav::compare(char *a, const char *b, int size) {
  while(size--> 0) {
    if (*a != *b) {
      return -1;
    }
```

```
    a++;
    b++;
  }
  return 0;
}

// open a WAVE file and get the data block
std::vector<double> ProcessWav::openFile(std::string filename) {
  std::vector<double> answ;
  const char * openName = filename.c_str();
  fp = fopen(openName, "r");

  if(fp != NULL) {
    // read first 4 bytes, to check its RIFF
    fread(chunkID, 1, 4, fp);
    chunkID[4] = '\0';

    // check header is RIFF
    if(compare(chunkID, riff, sizeof(chunkID)) == 0) {
      // read chunk size
      fread(&chunkSize, 1, 4, fp);
      // read file format type
      fread(format, 1, 4, fp);
      format[4] = '\0';

      // check header is wave
      if (compare(format, wave, sizeof(format)) == 0) {
        // read subchunk1 ID
        fread(subchunk1ID, 1, 4, fp);

        // check header is fmt
        if (compare(subchunk1ID, fmt, sizeof(subchunk1ID)) == 0) {
          // read subchunk1 size
          fread(&subchunk1Size, 1, 4, fp);
          // read audio format (i.e. compression type)
          fread(&audioFormat, 1, 2, fp);
          // read number of channels
```

```
fread(&numChannels, 1, 2, fp);
// read sample rate
fread(&sampleRate, 1, 4, fp);
// read byte rate
fread(&byteRate, 1, 4, fp);
// read block align
fread(&blockAlign, 1, 2, fp);
// read bits per sample
fread(&bitsPerSample, 1, 2, fp);
// read subchunk2 ID
fread(subchunk2ID, 1, 4, fp);
subchunk2ID[4] = '\0';

// check header is data chunk
if (compare(subchunk2ID, data, sizeof(subchunk2ID)) == 0) {
  // read subchunk2 size
  fread(&subchunk2Size, 1, 4, fp);

  // read data section to buffer
  if (bitsPerSample == 8) {
    soundBuffer8 = (char*)malloc(sizeof(char)*subchunk2Size);
    fread(soundBuffer8, 1, subchunk2Size, fp);
  }
  else if (bitsPerSample == 16) {
    soundBuffer16 = (short*)malloc(subchunk2Size);
    subchunk2Size /= sizeof(short);

    fread(soundBuffer16, 2, subchunk2Size, fp);

    for (int i = 0; i < subchunk2Size; i++)
        answ.push_back(soundBuffer16[i]);
  }
  else if (bitsPerSample == 32) {
    soundBuffer32 = (int*)malloc(sizeof(int)*subchunk2Size);
    fread(soundBuffer32, 4, subchunk2Size, fp);
  }
  fileRead = true;
```

```
            }
            else {
              // ERROR - DATA CHUNK NOT FOUND
              std::cout << "Error: Data chunk not found"
                        << std::endl;
            }
          }
          else {
            // EROR - FORMAT SECTION ID IS WRONG
            std::cout << "Error - Format section header wrong or not found"
                      << std::endl;
          }
        }
        else {
          // ERROR - NOT A WAV FILE
          std::cout << "Error: Not a .wav file" << std::endl;
        }
      } else {
        // ERROR - FILE HEADER DOES NOT CONTAIN RIFF
        std::cout << "Error: Not a RIFF file" << std::endl;
      }
    }
    else {
      // ERROR - COULD NOT OPEN FILE
      std::cout << "Error: Could not open file" << std::endl;
    }
    // close file stream
    fclose(fp);
    // success
    return answ;
}

// write a 16-bit WAVE file
int ProcessWav::write16Bit(std::vector<double> dataChunk,
std::string filename) {
    // size of subchunk 1 (FMT chunk)
    subchunk1Size = 16;
```

```
// compression format (1 = PCM)
audioFormat = 1;
// number of channels (1 = mono)
numChannels = 1;
// sample rate (44100 KHz)
sampleRate = 44100;
// bit rate (16 bit)
bitsPerSample = 16;
// byte rate
byteRate = (sampleRate * bitsPerSample * numChannels) / 8;
// block align
blockAlign = (numChannels * bitsPerSample) / 8;
// size of subchunk 2 (data chunk)
subchunk2Size = (dataChunk.size() * numChannels * bitsPerSample) / 8;
// chunk size
chunkSize = 4 + (8 + subchunk1Size) + (8 + subchunk2Size);
//chunkSize = subchunk2Size + 36;


short writeData[dataChunk.size()];

for (int i = 0; i < dataChunk.size(); i++)
    writeData[i] = dataChunk[i];

// set text blocks for header
for (int i = 0; i < 4; i++) {
    chunkID[i] = riff[i];
    format[i] = wave[i];
    subchunk1ID[i] = fmt[i];
    subchunk2ID[i] = data[i];
}

filename.append(".wav");
const char * cFilename = filename.c_str();

// write header
fp = fopen(cFilename, "wb");
```

```
  if(fp != NULL) {
    fwrite(chunkID, 1, 4, fp);
    fwrite(&chunkSize, 4, 1, fp);
    fwrite(format, 1, 4, fp);
    fwrite(subchunk1ID, 1, 4, fp);
    fwrite(&subchunk1Size, 4, 1, fp);
    fwrite(&audioFormat, 2, 1, fp);
    fwrite(&numChannels, 2, 1, fp);
    fwrite(&sampleRate, 4, 1, fp);
    fwrite(&byteRate, 4, 1, fp);
    fwrite(&blockAlign, 2, 1, fp);
    fwrite(&bitsPerSample, 2, 1, fp);
    fwrite(subchunk2ID, 1, 4, fp);
    fwrite(&subchunk2Size, 4, 1, fp);
    // write data
    fwrite(writeData, (bitsPerSample/8),
(subchunk2Size / (bitsPerSample/8)), fp);
  }
  else {
    std::cout << "Error: Could not create .wav file to write to"
<< std::endl;
    return -1;
  }
  return 0;
}

// print file information
void ProcessWav::fileInfo() {
  // check that a file has been read
  if (fileRead) {
    // chunk ID (this should be "RIFF")
    std::cout << "Chunk ID is: " << chunkID <<"\n";
    // chunk size (i.e. file size)
    printf("Chunk Size is: %d\n", chunkSize);
    // file format (this should be "WAVE")
    printf("File format is: %s\n", format);
    // subchunk1 ID (this should be "fmt ")
```

```
    printf("Subchunk1 ID is: %s\n", subchunk1ID);
    // subchunk1 Size
    printf("Subchunk1 Size is: %d\n", subchunk1Size);
    // Audio compression format (1 = PCM)
    printf("Audio compression format is: %d\n", audioFormat);
    // Number of channels (1 = mono, 2 = stereo)
    printf("Number of channels is: %d\n", numChannels);
    // Sample Rate (e.g. 44.1KHz)
    printf("Sample Rate is: %d\n", sampleRate);
    // Byte Rate
    printf("Byte Rate is: %d\n", byteRate);
    // Block Align
    printf("Block Align is: %d\n", blockAlign);
    // Bits Per Sample (i.e. bit rate)
    printf("Bits Per Sample is: %d\n", bitsPerSample);
    // Subchunk2 ID (this should be "data")
    printf("Subchunk2 ID is: %s\n", subchunk2ID);
    // Subchunk 2 Size (i.e. size of data section)
    printf("Subchunk2 Size is: %d\n\n", subchunk2Size);
  }
  else {
    printf("Error: File not read\n");
  }
}
```

B.4 ProcessData.h

```
#ifndef PROCESSDATA_H
#define PROCESSDATA_H


#include <vector>
#include <math.h>
#include <cmath>
#include <limits>
#include <iostream>
```

```
class ProcessData {
public:
    ProcessData ();
    void minMax (std::vector< std::vector<double> > &data,
                 int min, int max, int features);
    void minMaxVec (std::vector<double> & data, int min,
                    int max);
    double* getMin();
    double* getMax();
    void setMin(double min[], int size);
    void setMax(double max[], int size);
    std::vector<double> zScore (std::vector<double> data);
    double sEntropy(std::vector<double> data);
    double absMean(std::vector<double> data);
    double stdDev(std::vector<double> data);

private:
    double *dataMin;
    double *dataMax;
};

#endif
```

B.5 ProcessData.cpp

```
#include "ProcessData.h"
#include "Algorithm.h"

ProcessData::ProcessData() {
    dataMin = NULL;
    dataMax = NULL;
}

// perform min max on data set
```

```
// min max is calculated separately for each feature vector
void ProcessData::minMax (std::vector< std::vector<double> >
&data, int min, int max, int features) {
  double dmax = 0, dmin = 0;

  dataMax = new double[features];
  dataMin = new double[features];

  for (int i = 0; i < features; i++) {
    for (int j = 0; j < data.size(); j++) {
      if (j == 0) {
        dmin = data[j][i];
        dmax = data[j][i];
      }
      else if (data[j][i] < dmin)
        dmin = data[j][i];
      else if (data[j][i] > dmax)
        dmax = data[j][i];
    }
    dataMax[i] = dmax;
    dataMin[i] = dmin;

    double result;

    for (int j = 0; j < data.size(); j++) {
       result = (((data[j][i]-dmin) / (dmax-dmin))
* (max-min)+min);

       if (!isnan(result))
         data[j][i] = result;
       else
           data[j][i] = 0;
    }
  }
}

// min-max a vector
```

```
// used for normalising incoming windows
void ProcessData::minMaxVec (std::vector<double> & data,
int min, int max) {
  double result;

  for (int i = 0; i < data.size(); i++) {
    result = (((data[i]-dataMin[i]) / (dataMax[i]-dataMin[i]))
* (max-min)+min);

    if (!isnan(result))
      data[i] = result;
    else
      data[i] = 0;
  }
}

// shannon entropy
double ProcessData::sEntropy(std::vector<double> data) {
  double total = 0;

  for (int i = 0; i < data.size(); i++) {
    if (data[i] != 0)
       total += pow(data[i], 2) * log10(pow(data[i], 2));
  }

  return total;
}

double ProcessData::absMean(std::vector<double> data) {
    double mean = 0;

    for (int i = 0; i < data.size(); i++)
        mean += std::abs(data[i]);

    mean /= data.size();

    return mean;
```

```
}

double ProcessData::stdDev(std::vector<double> data) {
  double var = 0, mean = 0;

  // calc mean
  for (int i = 0; i < data.size(); i++)
      mean += data[i];
  mean /= data.size();

  // calc variance
  for (int i = 0; i < data.size(); i++)
      var += pow((data[i] - mean), 2);
  var /= data.size();

  var = sqrt(var);

  return var;
}

double* ProcessData::getMin() {
  if (dataMin != NULL)
    return dataMin;
  else
    return NULL;
}

double* ProcessData::getMax() {
  if (dataMax != NULL)
      return dataMax;
  else
    return NULL;
}

void ProcessData::setMin(double min[], int size) {
  dataMin = new double[size];
```

```
  for (int i = 0; i < size; i++)
      dataMin[i] = min[i];
}

void ProcessData::setMax(double max[], int size) {
  dataMax = new double[size];

  for (int i = 0; i < size; i++)
      dataMax[i] = max[i];
}
```

B.6 Algorithm.h

```
#ifndef ALGORITHM_H
#define ALGORITHM_H

#include <vector>
#include <iostream>
#include <ctime>
#include <map>
#include <string>
#include <sstream>
#include <thread>
#include <mutex>

#include "ProcessWav.h"
#include "ProcessData.h"
#include "DWT.h"
#include "ReadWrite.h"
#include "CaptureAudio.h"
#include "DSP.h"
#include "svm.h"

class Algorithm {
public:
  Algorithm();
```

```
  Algorithm(const Algorithm& orig);
  virtual ~Algorithm();
  int processFiles(std::string modelType);
  int buildModel(bool opt);
  int testModel(const svm_model *testModel,
                std::vector<std::string> testNames,
                std::vector<double> &stats);
  int getTestNames(std::vector<std::string> &tests);
  int printStats(std::vector<double> stats);
  int loadLastModel(std::string type);
  int realTimeTest();
  int buildANN();
  int testANNOffline();
private:
  ProcessWav pw;
  ProcessData pd;
  DWT dwt;
  ReadWrite rw;
  CaptureAudio ca;
  DSP dsp;
  int gateThreshold;
  int level;
  int normMin;
  int normMax;
  int neuronIn, neuronOut, neuronHidden, numLayers;
  double *dataMin;
  double *dataMax;
  int features;
  char *lastModel;
  const svm_model *model;
  int kernelMin, kernelMax, cMin, cMax, gammaMin, gammaMax,
      coef0Min, coef0Max, degreeMin, degreeMax;
  double cvThres, optIncr, optIncrFine, accThres, fprThres;
  std::vector<std::map<std::string, double> > topOpt;
  std::vector<std::map<std::string, double> > topTest;
  std::mutex barrier;
  bool writeWav;
```

```
    void processData(std::vector<double> &data);
    void getNode(std::vector<double> data, svm_node node[]);
    int getModel(svm_model &model, std::string fileName);
    void getDate(std::string &date);
    int optimiseModel(svm_parameter &param, svm_problem prob);
    int crossValidation(svm_parameter param, svm_problem prob);
    void defaultParams(svm_parameter &param);
    int testThread(svm_parameter param,
                   svm_problem prob,
                   std::vector<std::string> testNames);
    int getHash(svm_parameter param,
                std::map<std::string, double> &map);
    int kernelName(int kernel, std::string &name);
    int printParamInfo(std::map<std::string, double> model);
    int getParams(svm_parameter &param,
                  std::map<std::string, double> map);
};

#endif
```

B.7 Algorithm.cpp

```
#include "Algorithm.h"
#include "DSP.h"

Algorithm::Algorithm() {
    // data min max values for SVM
    normMin = 0;
    normMax = 1;

    // threshold for gate
    gateThreshold = 20;

    // DWT level
    level = 4;
```

```
  features = level * 4;

  // classwide svm model
  model = NULL;

  // accuracy thresholds for optimisation routine
  cvThres = 80;
  fprThres = 10;
  accThres = 70;

  // optimisation settings
  optIncr = 0.25;
  kernelMin = 2;
  kernelMax = 2;
  cMin = -10;
  cMax = 0;
  gammaMin = -15;
  gammaMax = 5;
  coef0Min = 0;
  coef0Max = 1;
  degreeMin = 1;
  degreeMax = 10;

  // set to true to write wav files of captured audio
  writeWav = false;
}

Algorithm::Algorithm(const Algorithm& orig) {
}

Algorithm::~Algorithm() {
}


// process files, for use in machine learning model
int Algorithm::processFiles(std::string modelType) {
  // check audio directory
```

```
  system("/home/bishop/Dev/projects/AudioDetector/findWav.sh");
  std::ifstream audioFiles ("files.txt");
  std::string filename;
  // data stored as [instance][features]
  std::vector< std::vector<double> > data;
  std::vector<double> instance;

  // open files, normalize (in terms of volume), and add to data
  if(audioFiles.is_open()) {
    while(std::getline(audioFiles, filename)) {
      instance = pw.openFile(filename);

      processData(instance);

      data.push_back(instance);

      instance.clear();
    }
  }

  // min-max data
  features = data[0].size();
  pd.minMax(data, normMin, normMax, features);

  // write data to file, in libsvm format
  if (modelType.compare("svm") == 0)
    rw.writeSVMData(data);

  return 0;
}


// extract features and create data instance
void Algorithm::processData(std::vector<double> &instance) {
  // normalize sound volume
  dsp.normalize(instance);
```

```
  // pad window
  dsp.perPad(instance, (ca.getBufferSize()
            * ca.getOverlapSize()));

  // dwt coefficients as [level][coefficients]
  std::vector< std::vector<double> > dwtCoef;

  // perform DWT
  dwt.d8DWT(instance, level, dwtCoef);

  instance.clear();

  // extract features
  for (int i = 0; i < dwtCoef.size(); i++) {
    instance.push_back(pd.sEntropy(dwtCoef[i]));
    instance.push_back(pd.absMean(dwtCoef[i]));
    instance.push_back(pd.stdDev(dwtCoef[i]));
  }
}


// default SVM parameters
void Algorithm::defaultParams(svm_parameter &param) {
  // set SVM type
  // 5 = SVDD
  param.svm_type = 5;
  // set kernel type
  // 0 = linear, 1 = polynomial, 2 = RBF, 3 = sigmoid
  param.kernel_type = 2;
  // set cache size
  param.cache_size = 500;
  param.shrinking = 0;
  param.probability = 0;
  param.eps = 0.0001;
  param.C = 0.4;
  param.gamma = (1 / features);
  param.degree = 1;
```

```
  param.coef0 = 1;
}


// cross-validation portion of the optimisation routine
int Algorithm::crossValidation(svm_parameter param,
                               svm_problem prob) {
  const char *errorMsg;
  double *cv = (double *)malloc((prob.l)*sizeof(double));
  double correct = 0;

  // check parameters
  errorMsg = svm_check_parameter(&prob, &param);
  if (errorMsg != NULL) {
    // ERROR
    std::cout << "libSVM parameter error: " << errorMsg
              << std::cout;
    return -1;
  }

  svm_cross_validation(&prob, &param, 10, cv);

  for (int i = 0; i < prob.l; i++) {
    if (cv[i] == 1)
      correct++;
  }

  correct = (correct / prob.l)*100;

  if (correct >= cvThres) {
    std::map<std::string, double> paramMap;
    paramMap["kernel"] = param.kernel_type;
    paramMap["c"] = param.C;
    paramMap["gamma"] = param.gamma;
    paramMap["degree"] = param.degree;
    paramMap["coef0"] = param.coef0;
    paramMap["cv"] = correct;
```

```
    barrier.lock();
      topOpt.push_back(paramMap);
    barrier.unlock();
  }

  return 0;
}


// optimisation routine
int Algorithm::optimiseModel(svm_parameter &param,
                             svm_problem prob) {
  std::cout << "Beginning optimisation process..." << std::endl;

  // search for suitable parameters using cross-validation
  double cExp = cMin, gammaExp = gammaMin, coef0 = coef0Min,
         degree = degreeMin;

  // start with defaul params
  defaultParams(param);

  std::vector<std::thread> threads;

  // test each kernel type
  for (int kernel = kernelMin; kernel <= kernelMax; kernel++) {
    param.kernel_type = kernel;

    // test C value in range
    while (cExp <= cMax) {

      param.C = pow(2, cExp);

      // linear kernel
      if (kernel == 0)
        threads.push_back(std::thread
(&Algorithm::crossValidation, this, param, prob));
```

```
        else {
            // test gamma value in range
            while (gammaExp <= gammaMax) {
              param.gamma = pow(2, gammaExp);

              // rbf kernel
              if (kernel == 2)
                threads.push_back(std::thread
(&Algorithm::crossValidation, this, param, prob));

              else {
                while (coef0 <= coef0Max) {
                    param.coef0 = coef0;

                    // sigmoid kernel
                    if (kernel == 3)
                      threads.push_back(std::thread
(&Algorithm::crossValidation, this, param, prob));

                    else {
                      // polynomial kernel
                      if (kernel == 1) {
                        while (degree <= degreeMax) {
                          param.degree = degree;
                          threads.push_back(std::thread
(&Algorithm::crossValidation, this, param, prob));
                          degree++;
                        }
                      }
                      else {
                        // ERROR
                        std::cout << "Error: unknown kernel type "
                        << "in optimisation test" << std::endl;
                        return -1;
                      }
                    }
```

```
                degree = degreeMin;
                coef0++;
              }
            }

            coef0 = coef0Min;
            gammaExp += optIncr;
          }
      }
      gammaExp = gammaMin;
      cExp += optIncr;
    }
    cExp = cMin;
}

// wait for all threads to finish cross-validation
for (int i = 0; i < threads.size(); i++) {
    threads[i].join();
}

threads.clear();

std::cout << "After cross-validation, " << topOpt.size()
          << " models will "
          << "now be tested on real data" << std::endl;

topTest.clear();
std::vector<std::string> testNames;
getTestNames(testNames);


// run an offline test for each of the 'best' value combos
for (int i = 0; i < topOpt.size(); i++) {
  // set all parameters
  param.kernel_type = topOpt[i]["kernel"];
  param.C = topOpt[i]["c"];
```

```
    param.gamma = topOpt[i]["gamma"];
    param.coef0 = topOpt[i]["coef0"];
    param.degree = topOpt[i]["degree"];

    threads.push_back(std::thread(&Algorithm::testThread,
                      this, param, prob, testNames));
  }

  // wait for all threads to finish real data test
  for (int i = 0; i < threads.size(); i++) {
      threads[i].join();
  }

  topOpt.clear();

  std::map<std::string, double> top;

  // get models with p that meets threshold, and FPR
  for (int i = 0; i < topTest.size(); i++) {
    if (topTest[i]["gmean1"] >= accThres && topTest[i]["fpr"]
        <= fprThres) {
      topOpt.push_back(topTest[i]);
    }
  }

  // from subset of models, get model with best gmean2
  if (topOpt.size() > 1) {
    top = topOpt[0];

    for (int i = 1; i < topOpt.size(); i++) {
      if (topOpt[i]["p"] >= top["p"])
        if (topOpt[i]["tpr"] >= top["tpr"])
          if (topOpt[i]["gmean2"] > top["gmean2"])
            top = topOpt[i];
    }
  }
  // output model info
```

```
  std::cout << "model selected for use is: " << std::endl;
  printParamInfo(top);

  //  set params for best model found
  getParams(param, top);

  return 0;
}


// print parameter info about a test model (used in testing)
int Algorithm::printParamInfo(std::map<std::string, double>
                              model) {
    std::string name;
    kernelName(model["kernel"], name);
  std::cout << "Test model info: " << std::endl
            << "kernel: " << name
            << ", C: " << model["c"]
            << ", gamma: " << model["gamma"]
            << ", degree: " << model["degree"]
            << ", coef0: " << model["coef0"];

  std::map<std::string, double>::iterator it = model.find("correct");
  if (it != model.end())
      std::cout << ", Correct: " << model["correct"];

  it = model.find("acc");
  if (it != model.end())
      std::cout << ", ACC: " << model["acc"]
                << ", P: " << model["p"]
                << ", gmean1: " << model["gmean1"]
                << ", gmean2: " << model["gmean2"];

   it = model.find("tpr");
  if (it != model.end())
      std::cout << ", TPR: " << model["tpr"]
              << ", FPR: " << model["fpr"]
```

```
                << ", TNR: " << model["tnr"]
                << ", FNR: " << model["fnr"];


  std::cout << std::endl;

 return 0;
}


// return name of libSVM kernel name
int Algorithm::kernelName(int kernel, std::string &name) {
  switch(kernel) {
    case 0:
      name = "linear";
      break;
    case 1:
      name = "polynomial";
      break;
    case 2:
      name = "rbf";
      break;
    case 3:
      name = "sigmoid";
      break;
    default:
      std::cout << "Error: Could not discern kernel used in model build"
                << std::endl;
      name = "unknown";
      break;
    }

  return 0;
}


// use to pass param and return map
```

```
int Algorithm::getHash(svm_parameter param,
                       std::map<std::string, double> &map) {
  map["kernel"] = param.kernel_type;
  map["c"] = param.C;
  map["gamma"] = param.gamma;
  map["degree"] = param.degree;
  map["coef0"] = param.coef0;

  return 0;
}


// get params from map
int Algorithm::getParams(svm_parameter &param,
                         std::map<std::string, double> map) {
  param.kernel_type = map["kernel"];
  param.C = map["c"];
  param.gamma = map["gamma"];
  param.degree = map["degree"];
  param.coef0 = map["coef0"];

  return 0;
}


// build an SVM model
int Algorithm::buildModel(bool opt){
 // get training data
 std::vector< std::vector<svm_node> > data;
 rw.readSVMData(data);

 // create libsvm problem
  svm_problem prob;
  // number of training data
  prob.l = data.size();
  // class labels
  prob.y = (double*)malloc((prob.l)*sizeof(double));
```

```
svm_node *trainData[prob.l];

svm_node *instance;

for (int i = 0; i < data.size(); i++) {
  // first node is class label
  prob.y[i] = data[i][0].index;

  instance = new svm_node[data[i].size()];

  // skip first node, as it is class label
  for (int j = 1; j < data[i].size(); j++)
    instance[j-1] = data[i][j];
  trainData[i] = instance;
}
prob.x = trainData;

const char *errorMsg;
svm_parameter param;

if (opt)
  optimiseModel(param, prob);

else
  defaultParams(param);

// check parameters
errorMsg = svm_check_parameter(&prob, &param);
  if (errorMsg != NULL) {
    // ERROR
    std::cout << "libSVM parameter error: " << errorMsg
              << std::endl;
    return -1;
  }

// train model
```

```
model = svm_train(&prob, &param);

// save model
time_t now = time(&now);
std::string dataDir = "model/";
dataDir += ctime(&now);
dataDir.erase(dataDir.find_last_not_of(" \t\f\v\n\r")+1);

int error = svm_save_model(dataDir.c_str(), model);

if (error != 0) {
    std::cout << "Cannot save SVM model to file.\n";
    exit(1);
}

// write name of last model, and min and max values
dataMin = new double[features];
dataMax = new double[features];
dataMin = pd.getMin();
dataMax = pd.getMax();

if (dataMin == NULL) {
  // ERROR
  std::cout << "Error: Min value not set; last model filename
          << "and min-max values not written." << std::endl;
  return -1;
} if (dataMax == NULL) {
  // ERROR
  std::cout << "Error: Max value not set; last model filename
          << "and min-max values not written." << std::endl;

  return -1;
} else {
    std::string type = "svm_model";
  rw.writeLastModel(dataDir, dataMin, dataMax, features, type);
}
```

```
  return 0;
}


// load the last model built
int Algorithm::loadLastModel(std::string type) {
  // read name, and min-max values, of last model built
  std::map<std::string, std::string> dataMap;
  rw.readLastModel(dataMap, type);

  double dataMin[(dataMap.size() - 1) / 2];
  double dataMax[(dataMap.size() - 1) / 2];

  std::string minKey, maxKey;

  // set min value for training data range
  for (int i = 0; i < (dataMap.size() - 1) / 2; i++) {
    std::ostringstream ss;
    ss << i;
    minKey = "min" + ss.str();
    maxKey = "max" + ss.str();

    if (dataMap.count(minKey) == 1) {
      dataMin[i] = atof(dataMap[minKey].c_str());
    } else {
      // ERROR
      std::cout << "Error: Could not find min value of training"
                << "data for loaded model." << std::endl;
    }

    // set max value for training data range
    if (dataMap.count(maxKey) == 1) {
      dataMax[i] = atof(dataMap[maxKey].c_str());
    } else {
      // ERROR
      std::cout << "Error: Could not find max value of training"
                << "data for loaded model." << std::endl;
```

```
    }
  }
  pd.setMin(dataMin, (dataMap.size() - 1) / 2);
  pd.setMax(dataMax, (dataMap.size() - 1) / 2);

  // load SVM or ANN model from file
  if (dataMap.count("filename") == 1) {
    const char *filename = dataMap["filename"].c_str();

    if (type.compare("svm_model") == 0) {
      // load model
      model = svm_load_model(filename);

      if (model == NULL) {
        std::cout << "Error: Model failed to load."
                  << std::endl;
        return -1;
      }

      // output name of model
      std::cout << "model loaded is: " << filename << std::endl;
    }

  } else {
    // ERROR
    std::cout << "Error: Could not find filename of model to load."
              << std::endl;
    return -1;
  }

  return 0;
}


// return a list of the test files names
int Algorithm::getTestNames(std::vector<std::string> &tests) {
  tests.clear();
```

```
  system("/home/bishop/Dev/projects/AudioDetector/findTest.sh");
  std::ifstream audioFiles ("testfiles.txt");
  std::string filename;

  if (audioFiles.is_open()) {
   while(std::getline(audioFiles, filename))
     tests.push_back(filename);
  }
  else {
    // ERROR
      std::cout << "Error: cannot open file containing names of test files"
                << std::endl;
      return -1;
  }

  return 0;
}


// test a model during setp 2 of the optimisation routine
int Algorithm::testThread(svm_parameter param,
                          svm_problem prob,
                          std::vector<std::string> testNames) {
  const char *errorMsg;
    // check parameters
    errorMsg = svm_check_parameter(&prob, &param);
    if (errorMsg != NULL) {
      // ERROR
      std::cout << "libSVM parameter error: " << errorMsg
              << std::endl;
      return -1;
    }
    const svm_model *test;

    // train model
    test = svm_train(&prob, &param);
```

```
    std::vector<double> stats;

   testModel(test, testNames, stats);

  // save params
  if (stats.size() > 0) {
    std::map<std::string, double> result;
    getHash(param, result);
    result["acc"] = stats[0];
    result["p"] = stats[1];
    result["gmean1"] = stats[2];
    result["gmean2"] = stats[3];
    result["tpr"] = stats[4];
    result["fpr"] = stats[5];
    result["tnr"] = stats[6];
    result["fnr"] = stats[7];

    barrier.lock();
      topTest.push_back(result);
    barrier.unlock();
  }

  return 0;
}


// print performance metrics
int Algorithm::printStats(std::vector<double> stats) {
    std::cout << "acc: " <<  stats[0] << "%, p: " << stats[1]
              << "%, gmean1: " << stats[2] << "%, gmean2"
              << stats[3] << "%, tpr: " << stats[4] << "fpr: "
              << stats[5] << "%, tnr: " << stats[6]
              << "%, fnr: " << stats[7];
}


// test a model using offline test data
```

```
int Algorithm::testModel(const svm_model *testModel,
                         std::vector<std::string> testNames,
                         std::vector<double> &stats) {
 // test that model is loaded
  if (testModel == NULL) {
      loadLastModel("svm");
  }

  // a data instance
  std::vector<double> instance;
  // class label
  double predictClass;
  // true and false totals in test
  double tpr, fpr, tnr, fnr, p, acc;
  double a = 0, b = 0, c = 0, d = 0;

  ProcessWav process;

  // read, process, then test each file
  for (int i = 0; i < testNames.size(); i++) {
    // get data from file
    instance = process.openFile(testNames[i]);

    // extract features
    processData(instance);

    // min-max
    pd.minMaxVec(instance, normMin, normMax);

    // format for libsvm
    svm_node test[instance.size()+1];
    for (int i = 0; i < instance.size(); i++) {
      if (instance[i] != 0) {
        test[i].index = i+1;
        test[i].value = instance[i];
      }
    }
```

```
    // last node is -1, for libsvm
    test[instance.size()].index = -1;
    test[instance.size()].value = 0;

    // test instance using model
    predictClass = svm_predict(testModel, test);

    // determine TP, FP, TN, FN
    std::size_t start, pos;
    std::string className;

    start = testNames[i].find_last_of("/", testNames[i].size()-1);
    pos = testNames[i].find_last_of(".", testNames[i].size()-1);
    className = testNames[i].substr(start+1, (pos - start)-1);

    // if the test filename is numeric only, it's positive
    if (className.find_first_of("1234567890") >= 0
        && className.find_first_of("abcdefghijklmnopqrstuvwxyz")
        == std::string::npos) {

      if (predictClass == 1)
        d++;

      else if (predictClass == -1)
        c++;
    }
    // if the test file contains letters, it's negative
    else if (className.find_first_of("abcdefghijklmnopqrstuvwxyz")
>= 0
            && className.find_first_of("1234567890") >= 0) {

      if (predictClass == 1)
        b++;
      else if (predictClass == -1)
        a++;
    }
    else {
```

```
        std::cout << "Error: could not discern class label "
                  << "name in offline test" << std::endl;
    }

  instance.clear();
}
// calculate TPR, FPR, TNR, FNR
tpr = d / (c + d);
fpr = b / (a + b);
tnr = a / (a + b);
fnr = c / (c + d);

// precision
p = d / (b + d);

// accuracy
acc = ((a + d) / (a + b + c + d)) * 100;

// gMean1 and gMean2
double gMean1 = sqrt(tpr * p);
double gMean2 = sqrt(tpr * tnr);

p *= 100;
tpr *= 100;
fpr *= 100;
tnr *= 100;
fnr *= 100;
gMean1 *= 100;
gMean2 *= 100;

// test for NaN
if (isnan(p))
    p = 0;
if (isnan(gMean1))
  gMean1 = 0;
if (isnan(gMean2))
  gMean2 = 0;
```

```
  // store results
  stats.push_back(acc);
  stats.push_back(p);
  stats.push_back(gMean1);
  stats.push_back(gMean2);
  stats.push_back(tpr);
  stats.push_back(fpr);
  stats.push_back(tnr);
  stats.push_back(fnr);;

  return 0;
}


// run the software in real-time mode
int Algorithm::realTimeTest() {
  std::string type = "svm_model";
  loadLastModel(type);

  // intiate hardware
  ca.initAudio();

  short buf[ca.getBufferSize()];

  std::vector<double> buffer, data;
  double predictClass;

  int last, matchThres, seq = 0;

  // get audio input
  for (int i = 0; i < 2000; i++) {
    ca.getAudio(buf);

    // fill the buffer first
    if (i < ca.getOverlapSize()) {
      for (int j = 0; j < ca.getBufferSize(); j++)
```

```
        buffer.push_back(buf[j]);

      matchThres = 0;
    }
    // get data, process it, then test classification
    else {
      // move data forward, then add new window at the end
      for (int j = 0; j < (ca.getBufferSize()
          * (ca.getOverlapSize() - 1)); j++)
        buffer[j] = buffer[(ca.getBufferSize() + j) - 1];
      for (int j = 0; j < ca.getBufferSize(); j++)
        buffer[((ca.getBufferSize() * (ca.getOverlapSize() - 1))
            + j) - 1] = buf[j];

      // if writeWav is true, write captured audio to wav file
      if (writeWav) {
        std::string num;
        std::ostringstream os;
        os << i;
        num = os.str();
        std::string writeName = "audio/check/rt" + num;
        pw.write16Bit(buffer, writeName);
      }
      // copy data from buffer
      data = buffer;

      // send to gate to test amplitude
      if(dsp.gate(data, gateThreshold, seq)) {

        // extract features
        processData(data);

        // min-max
        pd.minMaxVec(data, normMin, normMax);

        // convert to libsvm format
        svm_node instance[data.size()+1];
```

```
        getNode(data, instance);

        // test classification
        predictClass = svm_predict(model, instance);


        // check if window is a sequential match
        if (predictClass == 1) {
          if (seq - last == 1) {
            matchThres++;

           // if match threshold met, send alert
           if (matchThres == 4) {

              std::string date;
              getDate(date);
              std::cout << "DETECTED: " << date << std::endl;
            }
          }
          last = seq;
        }
        else{

          matchThres = 0;
        }
      }

        data.clear();
    }
    seq++;
  }
  // close audio stream
  ca.closeAudio();

  return 0;
}
```

```
// return the passed vector in svm_node format
void Algorithm::getNode(std::vector<double> data, svm_node node[]) {
  for (int i = 0; i < data.size(); i++) {
    node[i].index = i+1;
    node[i].value = data[i];
  }
  node[data.size()].index = -1;
  node[data.size()].value = 0;
}


// return data and time
void Algorithm::getDate(std::string &date) {
  time_t now = time(&now);
  date += ctime(&now);
  date.erase(date.find_last_not_of(" \t\f\v\n\r")+1);
}
```

B.8 DSP.h

```
#ifndef DSP_H
#define DSP_H

#include <math.h>
#include <fftw3.h>
#include <vector>
#include <stdlib.h>
#include <cmath>
#include <limits>

class DSP {
public:
  DSP();
  void hannWin(short buf[], int len);
```

```
  bool gate(std::vector<double> data, int threshold, int seq);
  void zeroPad(std::vector<double> &data, int len);
  void constPad(std::vector<double> &data, int len);
  void symPad(std::vector<double> &data, int len);
  void perPad(std::vector<double> &data, int len);
  void normalize(std::vector<double> &data);
private:
    int gateLast;
};

#endif
```

B.9 DSP.cpp

```
#include "DSP.h"

DSP::DSP() {
}

// noise gate
bool DSP::gate(std::vector<double> data, int threshold,
               int seq) {
  if (seq - gateLast == 1)
    return true;

  for (int i = 0; i < data.size()-2; i++) {
    if (abs(data[i]) >= threshold && abs(data[i+1])
        >= threshold && abs(data[i+2]) >= threshold) {

      gateLast = seq;
      return true;
    }
    else
      return false;
  }
}
```

```
// pad window with zeroes
void DSP::zeroPad(std::vector<double> &data, int len) {
  len -= data.size();

  for (int i = 0; i < len; i++)
    data.push_back(0);
}

// pad window by repeating last value
void DSP::constPad(std::vector<double> &data, int len) {
  len -= data.size();

  for (int i = 0; i < len; i++)
    data.push_back(data.back());

}

// pad window by reversing the order of values (i.e. symetric)
void DSP::symPad(std::vector<double> &data, int len) {
  len -= data.size();

  int pos = data.size()-1;

  for (int i = 0; i < len; i++) {
    data.push_back(data[pos]);

    if (pos > 0)
      pos--;
    else
      pos = data.size()-1;
  }
}

// pad window by repeating values sequencially
void DSP::perPad(std::vector<double> &data, int len) {
  len -= data.size();
```

```
    for (int i = 0; i < len; i++) {
        data.push_back(data[i]);
    }
}

// normalize the sound data, in terms of amplitude
// applying a set gain to the window, based on the peak
void DSP::normalize (std::vector<double> &data) {
  short peak = 0, diff;

  for (int i = 0; i < data.size(); i++) {
    if (abs(data[i] > peak))
      peak = abs(data[i]);
  }

  diff = abs((std::numeric_limits<short>::max() - peak));
  peak = 1 + (diff / peak);

  for (int i = 0; i < data.size(); i++)
    data[i] *= peak;
}

// Hann window
void DSP::hannWin(short buf[], int len) {
  double multi;

  for (int i = 0; i < len; i++) {
    multi = 0.5 * (1-cos((2*M_PI*i) / (len-1)));
    buf[i] *= multi;
  }
}
```

B.10 DWT.h

```
#ifndef DWT_H
```

```
#define DWT_H

#include <iostream>
#include <vector>
#include <math.h>

class DWT {
public:
    DWT();
    std::vector<double> haarDWT(std::vector<double> input);
    void d4DWT(std::vector<double> input, int levels,
               std::vector< std::vector<double> > &data);
    void d8DWT(std::vector<double> input, int levels,
               std::vector< std::vector<double> > &data);
};

#endif
```

B.11 DWT.cpp

```
#include "DWT.h"
#include <math.h>

DWT::DWT() {
}

// Haar wavelet transform.
// Input vector must be 2^n length!
std::vector<double> DWT::haarDWT(std::vector<double> input) {
  std::vector<double> temp(input.size(), 0);

  // test n !=0 and n is power of 2
  if (input.size() > 0 && !(input.size() & (input.size()-1)) ) {
    int w = input.size();

    for (int i = 0; i < input.size();i++)
```

```
                temp[i] = 0;

    while (w > 1) {
      w /= 2;

      for (int i = 0; i < w; i++) {
        temp[i] = (input[2*i] + input[2*i+1]) / sqrt(2.0);
        temp[i+w] = (input[2*i] - input[2*i+1]) / sqrt(2.0);
      }

      for(int i=0;i<(w*2);i++)
        input[i] = temp[i];
    }
  }
  else {
    // ERROR
      std::cout << "DWT only accepts inputs of length 2^n"
                << std::endl;
  }
  return input;
}


// Daubechies D4 wavelet transform
void DWT::d4DWT(std::vector<double> input, int levels,
                std::vector< std::vector<double> > &data) {
  // scaling function coefficients
  double h0 = 0.6830127, h1 = 1.1830127,
         h2 = 0.3169873, h3 = -0.1830127;

  // wavelet function coefficients
  double g0 = h3, g1 = -h2, g2 = h1, g3 = -h0;

  int n = input.size();
  std::vector<double> output(n);

  std::vector<double> temp;
```

```
int i = 0, j = 0;

for (int k = 0; k < levels; k++) {
  if (n >= 4) {


    while (j < n-3) {
      // high-pass filter - detail coefficients
      output[i+(n/2)] = input[j]*h0 + input[j+1]*h1
                      + input[j+2]*h2 + input[j+3]*h3;

      // low-pass filter - approximation coefficients
      output[i] = input[j]*g0 + input[j+1]*g1
                + input[j+2]*g2 + input[j+3]*g3;

      i++;
      j += 2;
    }
    // high-pass filter - last in sequence
    // (periodic treatment)
    output[i+(n/2)] = input[n-2]*h0 + input[n-1]*h1
                    + input[0]*h2 + input[1]*h3;

    // low-pass filter - last in sequence
    // (periodic treatment)
    output[i] = input[n-2]*g0 + input[n-1]*g1 + input[0]*g2
              + input[1]*g3;
  }
  i = 0;
  j = 0;

  // n is now half
  n /= 2;

  // save the detail coefficients
  for (int l = n-1; l < output.size(); l++) {
```

```
      temp.push_back(output[l]);
     // std::cout << output[l] << ", ";
    }
    data.push_back(temp);
    temp.clear();

    if (k == levels-1) {
      for (int l = 0; l < n; l++)
        temp.push_back(output[l]);
      data.push_back(temp);
      temp.clear();
    }
    // copy approximation coefficients to input
    for (int l = 0; l < n; l++)
      input[l] = output[l];

    input.resize(n);
    output.resize(n);
  }
}

// Daubechies D8 wavelet transform
void DWT::d8DWT(std::vector<double> input, int levels,
                std::vector< std::vector<double> > &data) {
   // scaling function coefficients
  double h0 = 0.32580343, h1 = 1.01094572,
         h2 = 0.8922014, h3 = -0.03957503,
         h4 = -0.26450717, h5 = 0.0436163,
         h6 = 0.0465036, h7 = -0.01498699;

  // wavelet function coefficients
  double g0 = -h7, g1 = h6, g2 = -h5, g3 = h4, g4 = -h3,
         g5 = h2, g6 = -h1, g7 = h0;

  int n = input.size();
  std::vector<double> output(n);
```

```
std::vector<double> temp;

int i = 0, j = 0;

for (int k = 0; k < levels; k++) {
  if (n >= 8) {
       while (j < n-7) {
      // high-pass filter - detail coefficients
      output[i+(n/2)] = input[j]*h0 + input[j+1]*h1
                        + input[j+2]*h2 + input[j+3]*h3
                        + input[j+4]*h4 + input[j+5]*h5
                        + input[j+6]*h6 + input[j+7]*h7;

      // low-pass filter - approximation coefficients
      output[i] = input[j]*g0 + input[j+1]*g1 + input[j+2]*g2
                  + input[j+3]*g3 + input[j+4]*g4 + input[j+5]*g5
                  + input[j+6]*g6 + input[j+7]*g7;

      i++;
      j += 2;
    }

     // high-pass filter - last in sequence
     // (periodic treatment)
    output[i+(n/2)] = input[n-2]*h0 + input[n-1]*h1 + input[0]*h2
                      + input[1]*h3 + input[2]*h4 + input[3]*h5
                      + input[4]*h6 + input[5]*h7;

    // low-pass filter - last in sequence
    // (periodic treatment)
    output[i] = input[n-2]*g0 + input[n-1]*g1 + input[0]*g2
                + input[1]*g3 + input[2]*g4 + input[3]*g5
                + input[4]*g6 + input[5]*g7;

  }

  i = 0;
```

```
    j = 0;

    // n is now half
    n /= 2;

    // save the detail coefficients
    for (int l = n-1; l < output.size(); l++)
      temp.push_back(output[l]);


    data.push_back(temp);
    temp.clear();

    if (k == levels-1) {
      for (int l = 0; l < n; l++)
        temp.push_back(output[l]);
      data.push_back(temp);
      temp.clear();
    }
    // copy approximation coefficients to input
    for (int l = 0; l < n; l++)
      input[l] = output[l];

    input.resize(n);
    output.resize(n);
  }
}
```

B.12 CaptureAudio.h

```
#ifndef CAPTUREAUDIO_H
#define CAPTUREAUDIO_H

#include <alsa/asoundlib.h>
#include <iostream>
#include <math.h>
```

```
class CaptureAudio {
public:
    CaptureAudio();
    CaptureAudio(const CaptureAudio& orig);
    virtual ~CaptureAudio();
    int initAudio();
    int getAudio(short buf[]);
    int closeAudio();
    int getBufferSize();
    int getOverlapSize();
private:
    snd_pcm_t *captureHandle;
    snd_pcm_hw_params_t *hwParams;
    snd_pcm_uframes_t bufferSize;
    snd_pcm_uframes_t periodSize;
    unsigned int sampleRate;
    int channels;
    int overlapSize;
    int dir;
    bool ready;
};

#endif
```

B.13 CaptureAudio.cpp

```
#include "CaptureAudio.h"

CaptureAudio::CaptureAudio() {
    sampleRate = 44100;
    channels = 1;

    overlapSize = 10;

    bufferSize = sampleRate / overlapSize;
```

```
    periodSize = bufferSize / 2;
    ready = false;
}

CaptureAudio::CaptureAudio(const CaptureAudio& orig) {
}

CaptureAudio::~CaptureAudio() {
}

int CaptureAudio::initAudio() {
  int error;

  // may need to change from default to actual device name
  const char *deviceName = "default";

  // open device
  if ((error = snd_pcm_open(&captureHandle, deviceName,
       SND_PCM_STREAM_CAPTURE, 0)) < 0) {
    std::cout << "Cannot open audio device: "
              << snd_strerror(error) << "\n";
    return -1;
  }

  // allocate hardware parameter structure
  if ((error = snd_pcm_hw_params_malloc(&hwParams)) < 0) {
    std::cout << "Cannot allocate hardware parameter structure: "
              << snd_strerror(error) << "\n";
    return -1;
  }

  // initialise parameter structure with device capabilities
  if ((error = snd_pcm_hw_params_any(captureHandle, hwParams)) < 0) {
    std::cout << "Cannot initialise hardware parameter structure"
              << " with device capabilities: "
              << snd_strerror(error) << "\n";
    return -1;
```

```
}

// set access mode (e.g. regular or mmap'd) and interleave
// note, interleave doesn't affect mono inputs
if ((error = snd_pcm_hw_params_set_access(captureHandle,
     hwParams, SND_PCM_ACCESS_RW_INTERLEAVED)) < 0) {
  std::cout << "Cannot set access type: "
            << snd_strerror(error) << "\n";
  return -1;
}

// set bit rate, endianess, and signed
if ((error = snd_pcm_hw_params_set_format(captureHandle,
     hwParams, SND_PCM_FORMAT_S16_LE)) < 0) {
  std::cout << "Cannot set bit rate: " << snd_strerror(error)
            << std::endl;
  return -1;
}

// set sample rate
if ((error = snd_pcm_hw_params_set_rate_near(captureHandle,
     hwParams, &sampleRate, &dir)) < 0) {
  std::cout << "Cannot set sample rate: " << snd_strerror(error)
            << std::endl;

  return -1;
}

// set number of channels (i.e. 1 = mono, 2 = stereo)
if ((error = snd_pcm_hw_params_set_channels(captureHandle,
     hwParams, channels)) < 0) {
  std::cout << "Cannot set number of channels: "
            << snd_strerror(error) << std::endl;

  return -1;
}
```

```
  // set buffer size
  if ((error = snd_pcm_hw_params_set_buffer_size_near
       (captureHandle, hwParams, &bufferSize)) < 0) {
    std::cout << "Cannot set buffer size: "
              << snd_strerror(error) << std::endl;
    return -1;
  }

  // set period size
  if ((error = snd_pcm_hw_params_set_period_size_near
       (captureHandle, hwParams, &periodSize, &dir)) < 0) {
    std::cout << "Cannot set period size: "
              << snd_strerror(error) << std::endl;
    return -1;
  }

  // apply hardware parameters to device
  if ((error = snd_pcm_hw_params(captureHandle, hwParams)) < 0) {
    std::cout << "Cannot set hardware parameters to device: "
              << snd_strerror(error) << std::endl;
    return -1;
  }

  // free structure from memory
  snd_pcm_hw_params_free(hwParams);

  // prepare audio interface for use
  if ((error = snd_pcm_prepare(captureHandle)) < 0) {
    std::cout << "Cannot prepare audio interface for use: "
              << snd_strerror(error) << std::endl;
    return -1;
  }
  ready = true;

  return 0;
}
```

```
int CaptureAudio::getAudio(short buf[]) {
  int error;

  if (ready) {
    if ((error = snd_pcm_readi(captureHandle, buf, bufferSize)) < 0) {
      std::cout << "Cannot read from audio interface: "
                << snd_strerror(error) << std::endl;
      return -1;
    }
    // check for overrun
    if (error == -EPIPE) {
      std::cout << "Buffer overrun has occurred" << std::endl;

      // prepare audio interface for use
      if ((error = snd_pcm_prepare(captureHandle)) < 0) {
        std::cout << "Cannot prepare audio interface for use: "
                  << snd_strerror(error) << std::endl;
        ready = false;
        return -1;
      }
      else
        ready = true;
    }
  }

  return 0;
}


int CaptureAudio::closeAudio() {
  if (ready) {
    snd_pcm_drain(captureHandle);
    snd_pcm_close(captureHandle);
    ready = false;
  }
  else {
```

```
    std::cout << "Cannot close audio interface stream\n";
    return -1;
  }

  return 0;
}


int CaptureAudio::getBufferSize() {
    return bufferSize;
}

int CaptureAudio::getOverlapSize() {
    return overlapSize;
}
```

vspace5mm

B.14 findTest.sh

```
#!/bin/bash
echo `ls audio/test/*.wav > testfiles.txt`
```

B.15 findWav.sh

```
#!/bin/bash
echo `ls audio/*.wav > files.txt`
```

Bibliography

[1] M. I. Abdalla, H. M. Abobakr, and T. S. Gaafar. Dwt and mfccs based feature extraction methods for isolated word recognition. *International Journal of Computer Applications*, 69:21–26, 2013.

[2] M. I. Abdalla and H. S. Ali. Wavelet-based mel-frequency cepstral coefficients for speaker identification using hidden markov models. *Journal of Telecommunications*, 1(2):16–21, 2010.

[3] A. Adiga, M. Magimai-Doss, and C. S. Seelamantula. Gammatone wavelet cepstral coefficients for robust speech recognition. In *TENCON 2013 - 2013 IEEE Region 10 Conference (31194)*, pages 1–4, 2013.

[4] J. M. Aerts and D. Berckmans. A virtual chicken for climate control design: static and dynamic simulations of heat losses. *Transactions of the ASAE*, 47(5):1765–1772, 2004.

[5] J. M. Aerts, D. Berckmans, E. Decuypere, J. Buyse, and V. Goedseels. Modelling growth responses of broiler chickens to variations of the control input feed supply. In *ASAE Annual International Meeting*, volume 15, 1998.

[6] J. M. Aerts, D. Berckmans, P. Saevels, E. Decuypere, and J. Buyse. Modelling the static and dynamic responses of total heat production of broiler chickens to step changes in air temperature and light intensity. *British Poultry Science*, 41:651–659, 2000.

[7] J. M. Aerts, C. M. Wathes, and D. Berckmans. Dynamic data-based modelling of heat production and growth of broiler chickens: Development of an integrated management system. *Biosystems Engineering*, 84:257266, 2003.

[8] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[9] A.N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*. Boston, MA: Academic Press, Boston, MA, USA, 1992.

[10] A.N. Akansu and Y. Liu. On signal decomposition techniques. *Optical Engineering*, 30(7):912–920, 1991.

[11] R. Akmelawati and A. G. Muthalif. Animal sound activity detection using multiclass support vector machines. In *4th International Conference on Mechatronics*, pages 1–5, 2011.

[12] Audacity. Audacity: free, open source, cross-platform software for recording and editing sounds, 2015.

[13] P. Baldi and G. Pollastri. The principled design of large-scale recursive neural network architectures - dag-rnns and the protein structure prediction problem. *Journal of Machine Learning Research*, 4:575–602, 2003.

[14] F. Bamelis, B. Kemps, K. Mertens, B. De Ketelaere, E. Decuypere, and J. De-Baerdemaeker. An automatic monitoring of the hatching process based on the noise of the hatching chicks. *Poultry Science*, 84:1101–1107, 2005.

[15] A. Banerjee, P. Burlina, and R. Meth. Fast hyperspectral anomaly detection via svdd. In *Proceedings of IEEE International Conference on Image Processing*, 2007.

[16] R. J. Barsanti and A. Athanason. Signal compression using the discrete wavelet transform and the discrete cosine transform. In *Southeastcon*, pages 1–5, 2013.

[17] P. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

[18] U. Baulain. Magnetic resonance imaging for the in vivo determination of body composition in animal science. *Computers and Electronics in Agriculture*, 17:189–203, 1997.

[19] U. Baulain. *Body composition of farm animals by MRI*, pages 16–19. Quality Meat Scotland, Ingliston, UK, Dublin, 2013.

[20] L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8, 1972.

[21] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967.

[22] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[23] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[24] L. E. Baum and G. R. Sell. Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2):211–227, 1968.

[25] C. Belyavin and D. G. Filmer. Computer assisted control of growth in poultry houses. In *Proceedings from the VIII European Poultry Conference*, volume 25-28 June, 1990.

[26] S. Bengio. An introduction to statistical machine learning: Feature selection, 2003.

[27] Y. Bengio, I. J. Goodfellow, and A. Courville. *Deep Learning*. MIT Press, 2015.

[28] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[29] A. Benyassine and A.N. Akansu. Performance analysis and optimal structuring of subchannels for discrete multitone transceivers. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2, pages 1456–1459, 1995.

[30] D. Berckmans. Automatic on-line monitoring of animals by precision livestock farming. In *International Society for Animal Hygiene*, pages 27–30, 2004.

[31] N. V. Bhimani. Speaker recognition system based on mfcc and vq algorithms. *International Journal of Engineering Research & Technology*, 2(3):772–774, 2014.

[32] C. Bishop. Novelty detection and neural network validation. *IEEE Proceedings on Vision, Image and Signal Processing. Special Issue on Applications of Neural Networks*, 141(4):217–222, 1994.

[33] C. M. Bishop. Exact calculation of the hessian matrix for the multi-layer perceptron. *Neural Computation*, 4(4):494–501, 1992.

[34] R. B. Blackman and J. W. Tukey. The measurement of power spectra, from the point of view of communications engineering - part i. *Bell System Technical Journal*, 37(1):185–282, 1959.

[35] V. Blanz, B. Scholkopf, B.kopf, H.. Bulthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3d models. In *International Conference on Artificial Neural Networks*, page 251256, 1996.

[36] H. Blatt. *America's Food - What you don't know about what you eat.* The MIT Press, Boston, 2008.

[37] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[38] F. Bomers. *Wavelets in real time digital audio processing: Analysis and sample implementations.* Thesis, University of Mannheim, 2000.

[39] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[40] L. Bottou. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nimes*, 1991.

[41] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *19th International Conference on Computational Statistics*, pages 177–187. Springer, 2010.

[42] J. Buck and P. Tyack. A quantitative measure of similarity for tursiops truncatus signature whistles. *Journal of the Acoustical Society of America*, 94(5):2497–2506, 1993.

[43] S. C. Bugden and R. M. Evans. Vocal solicitation of heat as an integral component of the developing thermoregulatory system in young domestic chickens. *Canadian Journal of Zoology*, 75(12):1949–1954, 1997.

[44] S. C. Bugden and R. M. Evans. The development of vocal thermoregulatory response to temperature in embryos of the domestic chicken. *Wilson Bulletin*, 111(2):188–194, 1999.

[45] C. Burges and B. Schoelkopf. *Improving the accuracy and speed of support vector learning machines*, volume 9, pages 375–381. MIT Press, Cambridge, MA, 1997.

[46] H. G. Byun, S. M. Khaffaf, and K. C. Persaud. Application of unsupervised clustering methods to assessment of malodour in agriculture using an array of conducting polymer sensors. *Computers and Electronics in Agriculture*, 17(2):233–247, 1997.

[47] H. Caglar, Y. Liu, and A.N. Akansu. Statistically optimized pr-qmf design. *Proceedings of SPIE Visual Communications and Image Processing*, 1605:86–94, 1991.

[48] G. A. Carpenter, M. A. Rubin, and W. W. Streilein. Artmap-fd: familiarity discrimination applied to radar target recognition. In *International Conference on Neural Networks*, volume 3, pages 1459–1464, 1997.

[49] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *23rd International Conference on Machine Learning*, 2006.

[50] C. Cattani and A. Kudreyko. On the discrete harmonic wavelet transform. *Mathematical Problems in Engineering*, 2008:1–7, 2008.

[51] H. Cevikalp and B. Triggs. Efficient object detection using cascades of nearest convex model classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3138–3145, 2012.

[52] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. In *26th International Conference on Very Large Data Bases*, volume VLDB'00, pages 111–122, 2001.

[53] C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

[54] C. C. Chang, H. C. Tsai, and Y. J. Lee. A minimum enclosing balls labeling method for support vector clustering. Report, National Taiwan University of Science and Technology, 2007.

[55] F. J. Chang and Y. C. Chen. A counterpropagation fuzzy-neural network modeling approach to real time streamflow prediction. *Journal of Hydrology*, 245(1-4):153–164, 2001.

[56] W. Chang, C. Lee, and C. Lin. A revisit to support vector data description. Unpublished Report, 2013.

[57] S. Chaudhuri, M. Harvilla, and B. Raj. Unsupervised learning of acoustic unit descriptors for audio content. In *INTERSPEECH*, pages 2265–2268, 2011.

[58] A. Chedad, D. Moshou, J. M. Aerts, A. H. Van Hirtum, D. Ramon, and D. Berckmans. Recognition system for pig cough based on probabilistic neural networks. *Journal of Agricultural Engineering Research*, 79(4):449–457, 2001.

[59] B. Chen, B. Li, Z. Pan, and A. Feng. Document classification with one-class multiview learning. In *IIS International Conference on Industrial and Information Systems*, pages 289–292, 2009.

[60] R. L. Cheu, D. Srinivasan, and Teh Eng Tian. Support vector machine models for freeway incident detection. In *Intelligent Transportation Systems*, volume 1, pages 238–243, 2003.

[61] H. G. Chew, D. J. Crisp, R. E. Bogner, and C. C. Lim. Target detection in radar imagery using support vector machines with training size biasing. In *Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision*, 200.

[62] H. Choi, B. Hendricks, and R. Baraniuk. Analysis of multiscale texture segmentation using wavelet-domain hidden markov models. *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers*, 2:1287–1291, 1999.

[63] M. Chowdhury and A. Khatun. Image compression using discrete wavelet transform. *IJCSI International Journal of Computer Science Issues*, 9(4):327–330, 2012.

[64] Y. Chung, J. Lee, S. Oh, D. Park, H. H. Chang, and S. Kim. Automatic detection of cow's oestrus in audio surveillance system. *Asian-Australas J Anim Sci*, 26(7):1030–7, 2013.

[65] Y. Chung, S. Oh, J. Lee, D. Park, H. H. Chang, and S. Kim. Automatic detection and recognition of pig wasting diseases using sound data in audio surveillance systems. *Sensors (Basel)*, 13(10):12929–42, 2013.

[66] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, volume 2, pages 1237–1242, 2011.

[67] G. Collewet, P. Bogner, P. Allen, H. Busk, A. Dobrowolski, E. Olsen, and A. Davenel. Determination of the lean meat percentage of pig carcasses using magnetic resonance imaging of lean and obese pigs. *Meat Science*, 70(4):563–572, 2005.

[68] G. Collewet, J. Bugeon, J. Idier, S. Quellec, B. Quittet, M. Cambert, and P. Haffray. Rapid quantification of muscle fat content and subcutaneous adipose tissue in fish using mri. *Food Chemistry*, 138:2008–2015, 2013.

[69] R. Collobert. *Large Scale Machine Learning*. Thesis, University of Paris, 2004.

[70] A. J. Colmenarez and T. S. Huang. Face detection with information-based maximum discrimination. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 782–787, 1997.

[71] P. Comon. Independent component analysis: A new concept? *Signal Processing*, 36:287–314, 1994.

[72] P. Comon, C. Jutten, and J. Herault. Blind separation of sources, part ii: Problem statement. *Signal Processing*, 24:11–20, 1991.

[73] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273, 1995.

[74] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14(3):326–334, 1965.

[75] G. V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[76] D. Darlington, L. Daudet, and M. Sandler. Digital audio effects in the wavelet domain. In *5th International Conference on Digital Audio Effects (DAFX-02)*, pages 7–12, 2002.

[77] D. Dasgupta and S. Forrest. Novelty-detection in time series data using ideas from immunology. In *International Conference on Intelligent Systems*, 1996.

[78] D. Dasgupta and F. Nino. A comparison of negative and positive selection algorithms in novel pattern detection. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 125–130, 2000.

[79] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.

[80] I. Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[81] C. D'Avanzoa, V. Tarantinob, P. Bisiacchib, and G. Sparacino. A wavelet methodology for eeg time-frequency analysis in a time discrimination task. *International Journal of Bioelectromagnetism*, 11(4):185–188, 2009.

[82] J. S. De Bonet and P. Viola. Texture recognition using a non-parametric multi-scale statistical model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 641–647, 1998.

[83] R. M. de Mol, A. Keen, G. H. Kroeze, and J. M. Achten. A detection model for heat and diseases of dairy cattle based on time series analysis combined with a kalman filter. In *Congress of ICT Applications in Agriculture*, 1996.

[84] J. De Moura, I. Naas, E. Alves, T. Crvalho, M. Vale, and K. Lima. Noise analysis to evaluate chick thermal comfort. *Scientia Agricola*, 65(4):438–443, 2008.

[85] F. DeComite, F. Denis, R. Gillerson, and F. Letouzey. Positive and unlabeled examples help learning. In *10th International Conference on Algorithmic Learning*, volume 1720, pages 219–230, 1999.

[86] W. E. Deming. *Elementary Principles of the Statistical Control of Quality.* JUSE, 1950.

[87] V. Dheepa and R. Dhanapal. Analysis of credit card fraud detection methods. *International Journal of Recent Trends in Engineering*, 2(3):126–128, 2009.

[88] T. G. Dietterich. Machine learning research - four current directions. *AI Magazine*, 18:97–136, 1997.

[89] C. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17:349–358, 2001.

[90] P. Dobbelaar, T. T. Mottram, C. Nyabadza, P. J. Hobbs, R. J. Elliott-Martin, and Y. H. Schukken. Sampling and calibration methods to detect analytes in cows' breath. In J. A. van Arendonk, editor, *Book of Abstracts of the 46th Annual Meetings of the European Association for Animal Production, Prague*, pages 157–166, Wageningen, Netherlands, 1995.

[91] A. B. Doeschl-Wilson, C. T. Whittemore, P. W. Knap, and C. P. Schofield. Using visual image analysis to describe pig growth in terms of size and shape. *Animal Science*, 79:415–427, 2004.

[92] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for span categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[93] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc, New York. NY, 1973.

[94] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, second edition, 2000.

[95] C. Dugas, Y. Bengio, F. Belisle, C. Nadeau, and R. Garcia. Incorporating second-order functional knowledge for better option pricing. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, pages 472–478. The MIT Press, 2001.

[96] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Management*, 1998.

[97] C. M. Dwyer. Genetic and physiological determinants of maternal behavior and lamb survival: implications for low-input sheep management. *Journal of Animal Science*, 86:246–258, 2008.

[98] C. M. Dwyer, K. A. McLean, L. A. Deans, J. Chirnside, S. K. Calvert, and A. B. Lawrence. Vocalisations between mother and young in sheep: effects of breed and maternal experience. *Applied Animal Behaviour Science*, 58(1):105–119, 1998.

[99] B. Efron and R. Tibshirani. *An introduction to the bootstrap*. Chapman & Hall, 1993.

[100] A. M. Emad and A. Shenouda. A quantitative comparison of different mlp activation functions in classification. In J. Wang, Z. Yi, J. M. Zurada, B. L. Lu, and H. Yin, editors, *Advances in Neural Networks ISNN 2006*, Lecture Notes in Computer Science, pages 849–857, 2006.

[101] W. J. Eradus, W. Rossing, P. H. Hogwerf, and E. Benders. Signal processing activity data for oestrus detection in dairy cattle. In *Proceeding of the International Symposium on Prospects for Automatic Miling*, pages 23–25, 1994.

[102] L. Eriksson, E. Johansson, N. Kettaneh-Wold, and S. Wold. *Scaling*, pages 213–225. Umetrics, 1999.

[103] European-Commission. Structure and dynamics of eu farms: changes, trends and policy relevance. Technical report, European Commission: EU Agricultural Economics Briefs, 2013.

[104] Evergraze. Lamb survival - turning reproductive potential into reality, 2013.

[105] V. Exadaktylos, M. Silva, and D. Berckmans. Real-time analysis of chicken embryo sounds to monitor different incubation stages. *Computers and Electronics in Agriculture*, 75:321–326, 2011.

[106] V. Exadaktylos, M. Silva, S. Ferrari, M. Guarino, and D. Berckmans. *Sound Localisation in Practice: An Application in Localisation of Sick Animals in Commercial Piggeries*, pages 575–590. InTech, 2011.

[107] Vasileios Exadaktylos, Mitchell Silva, and Daniel Berckmans. *Automatic Identification and Interpretation of Animal Sounds, Application to Livestock Production Optimisation*, journal article 4, pages 65–81. In, 2014.

[108] S. Fagerlund. Bird species recognition using support vector machines. In *EURASIP Journal on Advances in Signal Processing*, page 8, 2007.

[109] M. Farhid and M. A. Tinati. Robust voice conversion systems using mfdwc. In *IST 2008 International Symposium on Telecommunications*, pages 778–781, 2008.

[110] B. G. Farley and W. A. Clark. Simulation of self-organizing systems by digital computer. *IRE Transactions on Information Theory*, 4(4):76–84, 1954.

[111] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Fifth ACM SIGKDD Internataional Conference on Knowledge Discovery and Data Mining*, pages 53–62, 1999.

[112] S. Ferrari, M. Silva, M. Guarino, and D. Berckmans. Analysis of cough sounds for diagnosis of respiratory infections in intensive pig farming. *American Society of Agricultural and Biological Engineers*, 51(3):1051–1055, 2008.

[113] D. E. Filby, M. J. Turner, and M. J. Street. A walk-through weigher for dairy cows. *Journal of Agricultural Engineering Research*, 24:67–78, 1979.

[114] D. Filmer. *Nutritional management of meat poultry*, pages 133–146. Cambridge: British Society of Animal Science, 2001.

[115] T. L. Fine. *Feedforward Neural Network Methodology*. Statistics for Engineering and Information Science. Springer, 1999.

[116] A. V. Fisher. A review of the technique of estimating the composition of livestock using the velocity of ultrasound. *Computers and Electronics in Agriculture*, 17:217–231, 1997.

[117] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[118] M. Font i Furnols, C. E. Realini, L. Guerrero, M. A. Oliver, C. Sanudo, M. M. Campo, G. R. Nute, V. Canque, I. Alvarez, R. San Julian, S. Luzardo, G. Brito, and F. Montossi. Acceptability of lamb fed on pasture, concentrate or combinations of both systems by european consumers. *Meat Science*, 81:196–202, 2009.

[119] M. Font i Furnols, R. San Julian, L. Guerrero, C. Sanudo, M. M. Campo, J. L. Olleta, M. A. Oliver, V. Canque, I. lvarez, M. T. Diaz, W. Branscheid, M. Wicke, G. R. Nute, and F. Montossi. Acceptability of lamb meat from different producing systems and aging time to german, spanish and british consumers. *Meat Science*, 72:545–554, 2006.

[120] I. Fontana, E. Tullo, A. Butterworth, and M. Guarino. Broiler vocalisation to predict the growth. In *Measuring Behavior*, volume 27-29 August, 2014.

[121] A. D. Forbes. Classification-algorithm evaluation: five performance measures based on confusion matrices. *J. Clin. Monit*, 11:189–206, 1995.

[122] J. E. Fowler. The redundant discrete wavelet transform and additive noise. *IEEE Signal Processing Letters*, 12(9):629–632, 2005.

[123] T. H. Friend, L. A. OConnor, D. A. Knabe, and G. R. Dellmeier. Preliminary trials of a sound-activated device to reduce crushing of piglets by sows. *Applications in Animal Behavioural Science*, 24:23–29, 1989.

[124] A. R. Frost, D. J. Parsons, K. F. Stacey, A. P. Robertson, S. K. Welch, D. Filmer, and A. Fothergill. Progress towards the development of an integrated management system for broiler chicken production. *Computers and Electronics in Agriculture*, 39:227–240, 2003.

[125] A. R. Frost, C. P. Schofield, S. A. Beaulah, T. T. Mottram, J. A. Lines, and C. M. Wathes. A review of livestock monitoring and the need for integrated systems. *Computers and Electronics in Agriculture*, 17:139–159, 1997.

[126] T. S. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[127] S. Furui, L. Deng, M. Gales, H. Ney, and K. Tokuda. Fundamental technologies in modern speech recognition. *IEEE Signal Processing Magazine*, 29(6):16–17, 2012.

[128] H. Gaberson. A comprehensive windows tutorial. In *Sound and Vibration*, pages 14–23, 2006.

[129] M. Garofalakis. The discrete wavelet transform and wavelet synopses. Report, Department of Electronics and Computer Engineering, 2009.

[130] M. Garofalakis and P. B. Gibbons. Approximate query processing: Taming the terabytes. In *27th International Conference on Very Large Data Bases*, 2001.

[131] M. Garofalakis and P. B. Gibbons. Probabilistic wavelet synopses. *ACM Transactions on Database Systems*, 29(1):43–90, 2004.

[132] R. Geers, R. Puers, and V. Goedseels. Electronic identification and monitoring of pigs during housing and transport. *Journal of Agricultural Engineering Research*, 17(2):205–215, 1997.

[133] N. Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, Cambridge, England, 1998.

[134] K. Ghesquire, A. Van Hirtum, J. Buyse, and D. Berckmans. Non-invasive quantification of stress in the laying hen: a vocalization analysis approach. *Faculty of Agricultural and Applied Biology, University Wetenshcapen Ghent*, 67(4):55–58, 2002.

[135] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pages 315–323, 2011.

[136] A. Goldblatt. Agriculture in south africa: Facts & trends, 2009.

[137] G. S. Gomes, T. B. Ludermir, and L. M. Lima. Comparison of new activation functions in neural network for forecasting financial time series. *Neural Computing and Applications*, 20(3):417–439, 2010.

[138] J. Gowdy and Z. Tufekci. Mel-scaled discrete wavelet coefficients for speech recognition. *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3:1351–1354, 2000.

[139] D. M. Green and J. A. Swets. *Signal detection theory and psychophysics*. John Wiley and Sons, Inc, New York, NY, 1966.

[140] D. M. Green and C. T. Whittemore. Architecture of a harmonized model of the growing pig for the determination of dietary net energy and protein requirements and of excretions into the environment. *Animal Science*, 77:113–130, 2003.

[141] D. M. Green and C. T. Whittemore. Calibration and sensitivity analysis of a model of the growing pig for weight gain and composition. *Agricultural Systems*, 84:279–295, 2005.

[142] J. M. Grey and J. W. Gordon. Perceptual effects of spectral modifications on musical timbres. *Journal of the Acoustical Society of America*, 63(5):1493–1500, 1978.

[143] K. G. Grunert. Future trends and consumer lifestyles with regard to meat consumption. *Meat Science*, 74(1):149–160, 2006.

[144] M. Guarino, P. Jans, A. Costa, J. M. Aerts, and D. Berckmans. Field test of algorithm for automatic cough detection in pig houses. *Computers and Electronics in Agriculture*, 62(1):22–28, 2008.

[145] D. Gupta and S. Choubey. Discrete wavelet transform for image processing. *International Journal of Emerging Technology and Advanced Engineering*, 4(3):598–602, 2015.

[146] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[147] A. Haar. On the theory of orthogonal function systems. *Mathematical Annals*, 69(3):331–371, 1910.

[148] R. W. Hamming. *Digital filters*. Prentice Hall International, Hertfordshire, UK, third edition, 1989.

[149] J. Han, M. Kamber, and P. Pei. *Data Mining: Concepts and Techniques*. Morgan Kauffmann Publishers, third edition, 2012.

[150] R. N. Handcock, D. L. Swain, G. J. Bishop-Hurley, K. P. Patison, T. Wark, P. Valencia, P. Corke, and C. J. O'Neill. Monitoring animal behaviour and environmental interactions using wireless sensor networks, gps collars and satellite remote sensing. *Sensors (Basel)*, 9(5):3586–603, 2009.

[151] K. Hara and K. Nakayamma. Comparison of activation functions in multilayer neural network for pattern classification. In *IEEE International Conference on Neural Networks*, volume 5, pages 2997–3002. IEEE, 1994.

[152] F. J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[153] K. G. Haughey. Perinatal lamb mortality - its investigation, causes and control. *Journal of the South African Veterinary Association*, 62(2):78–91, 1991.

[154] B. A. Hawickhorst and S. A. Zahorian. A comparison of three neural network architectures for automatic speech recognition. *Intelligent Engineering Systems Through Artificial Neural Networks*, 5:221–226, 1995.

[155] S. A. Hayes, D. K. Mellinger, D. A. Croll, D. P. Costa, and J. F. Borsani. An inexpensive passive acoustic system for recording and localizing wild animal sounds. *Journal of Acoustic Society of America*, 107(6):3552–3555, 2000.

[156] D. Hebb. *The Organization of Behavior*. Wiley, New York, NY, 1949.

[157] R. Hecht-Nielsen. Counterpropagation networks. *Applied Optics*, 26(23):4979–4984, 1987.

[158] G. Heinzel, A. Rudiger, and R. Schilling. Spectrum and spectral density estimation by the discrete fourier transform (dft), including a comprehensive list of window functions and some new flat-top windows. Report, Max Planck Institute for Gravitational Physics (Albert Einstein Institute), 2002.

[159] K. Hempstalk, E. Frank, and I. H. Witten. One-class classification by combining density and class probability estimation. *Machine Learning and Knowledge Discovery in Databases*, 5211(1):505–519, 2008.

[160] P. J. Hepworth, A. V. Nefedov, I. B. Muchnik, and K. L. Morgan. Broiler chickens can benefit from machine learning: support vector machine analysis of observational epidemiological data. *Journal of the Royal Society - Interface*, 9(73):1934–1942, 2012.

[161] J. Herault and C. Jutten. Space or time adaptive signal processing by neural models. In *AIP Conference: Neural Networks for Computing*, volume 151, pages 206–211, 1986.

[162] R. Hergenhan. *Assessing neonatal lamb vigour*. Thesis, University of New England, 2011.

[163] H. Hermansky. Perceptual linear predictive (plp) analysis of speech. *Journal of Acoustic Society of America*, 87(4):1738–1752, 1990.

[164] H. Hermansky, B. A. Hanson, and H. Wakita. Perceptually based linear predictive analysis of speech. *Proceedings of 1985 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:509–512, 1985.

[165] Hewlett-Packard. A refresher course on windowing and measurements. Magazine article, Hewlett-Packard, 1996.

[166] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. *A real-time algorithm for signal analysis with the help of the wavelet transform*, pages 289–297. Springer-Verlag, 1989.

[167] X. Hongwei, J. A. DeShazer, and D. W. Leger. Swine voclations under selected husbandry practices. In *3rd Livestock Environment Symposium*, 1988.

[168] H. G. Hosseini, D. Luo, and K. J. Reynolds. The comparison of different feed forward neural network architectures for ecg signal diagnosis. *Medical Engineering & Physics*, 28(4):372–378, 2006.

[169] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[170] P. J. Howell and M. E. Paice. An adaptive data logging system for animal power studies. *journal of Agricultural Engineering Research*, 42:111–121, 1989.

[171] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Report, National Taiwan University, 2010.

[172] Y. H. Hu and J. N. Hwang. *Handbook of Neural Network Signal Processing*. CRC Press LLC, 2002.

[173] S. Hua and Z. Sun. A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *Journal of Molecular Biology*, 308:397–407, 2001.

[174] S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. In *Bioinformatics*, pages 721–728, 2001.

[175] D. M. Huber and R. E. Runstein. *Modern Recording Techniques*. Elsevier, sixth edition, 2005.

[176] D. Iatsenko. *Nonlinear Mode Decomposition: Theory and Applications*. Springer, 2015.

[177] H. Imtiaz and S. A. Fattah. A face recognition scheme using wavelet-based dominant features. *Signal & Image Processing : An International Journal (SIPIJ)*, 2(3):69–80, 2011.

[178] J. E. Jackson. *A user's guide to principal components*. John Wiley & Sons, 1991.

[179] S. Jain and A. K. Wadhwani. Preliminary detection of bearing faults using shannon entropy of wavelet coefficients. In *International Conference On Emerging Trends in Mechanical and Electrical Engineering*, volume 13th-14th March. International Journal of Engineering Research and Applications, 2014.

[180] G. James, W. Witten, T. Hastie, and R. Tibshirani. *Introduction to Statistical Learning with applications in R.* Wiley, second edition, 2014.

[181] A. G. Janecek, W. N. Gansterer, M. A. Demel, and G. F. Ecker. On the relationship between feature selection and classification accuracy. In *FSDM*, pages 90–105, 2008.

[182] Peter Janovi and Mnevver Kker. Automatic detection and recognition of tonal bird sounds in noisy environments. *EURASIP Journal on Advances in Signal Processing*, 2011(1):936–982, 2011.

[183] P. Jans, M. Guarino, A. Costa, J. M. Aerts, and D. Berckmans. Evaluation of an algorithm for cough detection in pig houses. In *16th IFAC World Congress*, 2005.

[184] N. Japkowicz. *Concept-Learning in the absence of counterexamples: An autoassociation-based approach to classification.* Thesis, The State University of New Jersey, 1999.

[185] N. Japkowicz, C. Myers, and M. Gluck. A novelty detection approach to classification. In *Fourteenth Joint Conference on Artificial Intelligence*, pages 518–523. Connectionist Models, 1995.

[186] W. K. Jenkins. *Fourier Series, Fourier Transforms, and the DFT*, book section 1, pages 6–36. CRC Press LLC, 1999.

[187] T. Joachims. *Learning to Classify Text Using Support Vector Machines.* Kluwer Academic Publishers / Springer, Kluwer, Boston, 2002.

[188] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *European Conference on Machine Learning*, pages 137–142, 2002.

[189] I. T. Jolliffe. *Principal Component Analysis.* Springer Series in Statistics. Springer, New York, NY, second edition, 2002.

[190] P. T. Jones, S. A. Shearer, and R. S. Gates. Edge extraction algorithm for feather sexing poultry chicks. *Transcripts from the American Society of Agricultural Engineering*, 34:635–640, 1991.

[191] C. Jutten and J. Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

[192] J. F. Kaiser. *Digital Filters*, book section 7. Wiley, New York, NY, USA, 1966.

[193] I. Kaplan. The daubechies d4 wavelet transform, 2001.

[194] B. Karlik and A. V. Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence And Expert Systems (IJAE)*, 1(4):111–122, 2010.

[195] D. A. Karras. Compression of mri images using the discrete wavelet transform and improved parameter free bayesian restoration techniques. In *IEEE International Workshop on Imaging Systems and Techniques*, pages 173–178, 2009.

[196] P. J. Kettlewell, M. A. Mitchell, and I. R. Meeks. An implantable radio-telemetry system for remote monitoring of heart rate and deep body temperature in poultry. *Computers and Electronics in Agriculture*, 17(2):161–175, 1997.

[197] H. C. Keun, T. D. Ebbels, H. Antti, M. E. Bollard, O. Beckonert, E. Holmes, J. C. Lindon, J. K. Nicholson, and Anal Chim Acta. Improved analysis of multivariate data by variable stability scaling: application to nmr-based metabolic profiling. *Analytica Chimica Acta*, 490(1-2):265–276, 2003.

[198] R. Khan, A. Clark, and G. Mohay. Fraud detection using transaction data in erp systems. In *9th International Conference on Information Technology and Applications*, 2014.

[199] S. Khan and G. Madden. A survey of recent trends in one class classification. *Artificial Intelligence and Cognitive Science:*, 6206:188–197, 2010.

[200] C. Kim and R. M. Stern. Feature extraction for robust speech recognition based on maximizing the sharpness of the power distribution and on power flooring. In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 4574–4577, 2010.

[201] C. Kim and R. M. Stern. Power-normalized cepstral coefficients (pncc) for robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4101–4104, 2012.

[202] S.P. King, D.M. King, P. Anuzis, K. Astley, L. Tarassenko, P. Hayton, and S. Utete. The use of novelty detection techniques for monitoring high-integrity plant. In *International Conference on Control Applications*, volume 1, pages 221–226, 2002.

[203] N. Kingsbury. The dual-tree complex wavelet transform: A new technique for shift invariance and directional filters. In *IEEE Digital Signal Processing Workshop*, 1998.

[204] N. Kingsbury. Image processing with complex wavelets. *Philosophical Transactions of the Royal Society London*, 357(1760):2543–2560, 1999.

[205] N. Kingsbury and J. Magarey. Wavelet transforms in image processing. In *First European Conference on Signal Analysis and Prediction*, pages 23–34, 1997.

[206] M. Kociolek, A. Materka, M. Strzelecki, and P. Szczypinski. Discrete wavelet transform - derived features for digital image texture analysis. In *Interational Conference on Signals and Electronic Systems*, pages 163–168, 2001.

[207] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *14th International Joint Conference on Artificial Intelligence*, 2:1137–1143, 1995.

[208] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[209] P. V. Kremer, M. Forster, and A. M. Scholz. Use of magnetic resonance imaging to predict the body composition of pigs in vivo. *Animal*, 7:879–884, 2013.

[210] Kreyszig. *Advanced Engineering Mathematics*. Wiley, 10th edition, 2011.

[211] R. Kronland-Martinet, J. Morlet, and A. Grossman. Analysis of sound patterns through wavelet transforms. *International Journal of Pattern Recognition and Artificial Intelligence*, 1(2):273–302, 1987.

[212] M. Kubat, R. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2/3):195–215, 1998.

[213] P. Kukielka and Z. Kotulski. Analysis of different architectures of neural networks for application in intrusion detection systems. In *International Multiconference on Computer Science and Information Technology*, pages 807–811, 2008.

[214] S. M. Kuo and B. H. Lee. *Real-Time Digital Signal Processing: Implementations, Applications, and Experiments with the TMS320C55X*. John Wiley & Sons Ltd, West Sussex, England, 2001.

[215] B. Lacey, T. K. Hamrita, M .P. Lacy, G. L. Van Wicklen, and M. Czarick. Monitoring deep body temperature responses of broilers using biotelemetry. *Journal of Applied Poultry Research*, 9(1):6–12, 2000.

[216] T. M. Lai, L. A. Snider, and D. Sutanto. High-impedance fault detection using discrete wavelet transform and frequency range and rms conversion. *IEEE Transactions on Power Delivery*, 20(1):397–407, 2005.

[217] S. Lawrence, C.L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.

[218] J. L. Laycock and M. J. Street. Development and use of automated management system for a large dairy herd. *Journal of Agricultural Engineering Research*, 30:265–273, 1984.

[219] P. N. Le, E. Ambikairajah, and J. Epps. Investigation of spectral centroid features for cognitive load classification. *Speech Communication*, 53(4):540–551, 2011.

[220] LearnDigitalAudio. How to normalize audio why do it? everything you need to know, 2016.

[221] Y. LeCun, S. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[222] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Mller. *Efficient BackProp*, book section 2, pages 9–48. Springer, 1998.

[223] Y. LeCun, L. Jackel, L. Bottou, A Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*, pages 53–60, 1995.

[224] B. Lee and Y. S. Tarng. Application of the discrete wavelet transform to the monitoring of tool failure in end milling using the spindle motor current. *International Journal of Advanced Manufacturing Technology*, 15(4):238–243, 1999.

[225] J. Lee, S. Zuo, Y. Chung, D. Park, H. H. Chang, and S. Kim. Formant-based acoustic features for cow's estrus detection in audio surveillance system. In *11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 236–240, 2014.

[226] S. W. Lee, J. Park, and S. W. Lee. Low resolution face recognition based on support vector data description. *Pattern Recognition*, 39(9):1809–1812, 2006.

[227] F.D. Letouzey and R. Gilleron. Learning from positive and unlabeled examples. In *11th International Conference on Algorithmic Learning Theory*, 2000.

[228] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach. An overview of noise-robust automatic speech recognition. *IEEE /ACM Transactions on audio, speech, and language processing*, 22(4):745–777, 2014.

[229] S. Ligout, F. Sbe, and R. H. Porter. Vocal discrimination of kin and non-kin agemates among lambs. *Behaviour*, 141:355–369, 2004.

[230] H. T. Lin and C. J. Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. Report, National Taiwan University, 2003.

[231] C. L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36:2271–2285, 2003.

[232] X. Liu and T. Yu. Gradient feature selection for online boosting. In *IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[233] F. Llario, S. Sendra, L. Parra, and J. Lloret. Detection and protection of the attacks to the sheep and goats using an intelligent wireless sensor network. In *IEEE International Conference on Communications Workshops (ICC)*, pages 1015–1019. IEEE, 2013.

[234] R. N. Lobo, I. D. Pereirab, O. Facao, and C. M. McManusc. Economic values for production traits of morada nova meat sheep in a pasture based production system in semi-arid brazil. *Small Ruminant Research*, 96:93–100, 2011.

[235] B. Logan. Mel frequency cepstral coefficients for music modelling. In *International Symposium on Music Information Retrieval*, pages 1–11, 2000.

[236] J. L. Lusk, F. B. Norwood, and R. W. Prickett. Consumer preferences for farm animal welfare: results of a natiowide telephone survey. Report, Department of Agricultrual Economics - Oklahoma State University, 2007.

[237] P. Lynch. The dolphchebyshev window: A simple optimal filter. *Monthly Weather Review - Notes and Correspondence*, 125:655–660, 1997.

[238] R. G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall, New Jersey, USA, third edition, 2011.

[239] K. Maatje, R. M. de Mol, and W. Rossing. Cow status monitoring (health and oestrus) using detection sensors. *Computers and Electronics in Agriculture*, 16(3):245–254, 1997.

[240] K. Maatje, P. H. Hogwerf, W. Rossing, and R. T. van Zonneveld. Measuring quarter milk conductivity, milk yield and milk temperature for detectioin of mastitis. In *International Symposium on Prospects for Automatic Milking*, pages 119–125, 1992.

[241] S. K. Madhusudhana, M. A Roch, E. M. Oleson, M. S. Soldevilla, and J. A. Hildebrand. Blue whale b and d call classification using a frequency domain based robust contour extractor. In *OCEANS 2009 - Europe*, pages 1–7, 2009.

[242] J. F. Magarey and N. Kingsbury. Motion estimation using complex-valued wavelet transform. *IEEE Trans. on Signal Processing*, 46(4):1069–1084, 1998.

[243] J. Mair, G. Marx, J. Petersen, and L. Mennicken. Development of multi parametric sound analysis parameters as welfare indicator in chicken flocks. *Proceedings of the International Seminar on Modal Analysis*, 3:1523–1528, 2001.

[244] S. G. Mallat. A theory for multiresolution signal decomposition : the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[245] A. Mallawaarachchi, S. H. Ong, M. Chitre, and E. Taylor. Spectrogram denoising and automated extraction of the fundamental frequency variation of dolphin whistles. *J Acoust Soc Am*, 124(2):1159–70, 2008.

[246] C. Maltin, C. Craigie, and L. Bunger. *FAIM: Farm Animal Imaging*. FAIM, Dublin, 2012.

[247] L. Manevitz and M. Yousef. Document classification on neural networks using only positive examples. In *23rd annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 304–306, 2000.

[248] L. Manevitz and M. Yousef. One class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[249] Larry Manevitz and Malik Yousef. One-class document classification via neural networks. *Neurocomputing*, 70(7-9):1466–1481, 2007.

[250] A. B. Mankar and M. S. Burange. Data mining - an evolutionary view of agriculture. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, 3(3):102–105, 2014.

[251] J. A. Marchant and C. P. Schofield. Extending the snake image processing algorithm for outlining pigs in scenes. *Computers and Electronics in Agriculture*, 8:261–275, 1993.

[252] J. N. Marchant, X. Whittaker, and D. M. Broom. Vocalisations of the adult female domestic pig during a standard human approach test and their relationships with behavioural and heart rate measures. *Applied Animal Behaviour Science*, 72(1):23–39, 2001.

[253] V. Marjan. Kohonen artificial neural network and counter propagation neural network in molecular structure-toxicity studies. *Current Computer - Aided Drug Design*, 1(1):73–78, 2005.

[254] M. Markou and S Singh. Novelty detection: a review - part 1: statistical approaches. *Signal Processing*, 83:2481–2497, 2003.

[255] G. Marx, J. Leppet, and F. Ellendorff. Vocalisation in chicks (galus galus dom.) during stepwise social isolation. *Applied Animal Behaviour Science*, 75(1):61–74, 2001.

[256] K Matsui. An ambulatory data logger for long-term determination of grazing and rumination behaviour of free-raning cattle, sheep and goats. *Applications in Animal Behavioural Science*, 39:123–130, 1994.

[257] M. Matusugu, K. Mori, Y. Mitari, and Y. Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5):555–559, 2003.

[258] N. J. McFarlane and C. P. Schofield. Segmentation and tracking of piglets. *Machine Visual Applications*, 8:187–193, 1995.

[259] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Interscience, 2004.

[260] R. McQueen, S. R. Garner, C. G. Nevill-manning, and I. H. Witten. Applying machine learning to agricultural data. *Computers and Electronics in Agriculture*, 12(4):275–293, 1995.

[261] M. Meissner, M. Schmuker, and G. Schneider. Optimized particle swarm optimization (opso) and its application to artificial neural network training. *BMC Bioinformatics*, 7(125):1–11, 2006.

[262] D. K. Mellinger. A comparison of methods for detecting right whale calls. *Canadian Acoustics*, 33(2):55–65, 2004.

[263] D. K. Mellinger, S. W. Martin, R. P. Morrissey, L. Thomas, and J. J. Yosco. A method for detecting whistles, moans, and other frequency contour sounds. *J Acoust Soc Am*, 129(6):4055–4061, 2011.

[264] E. Menahem, L. Rokach, and Y. Elovici. Combining one-class classifiers via meta learning. In *22nd ACM International Conference on Information & Knowledge Management*, 2013.

[265] R. Mendes, P. Cortez, M. Rocha, and J. Neves. Particle swarms for feedforward neural network training. In *International Joint Conference on Neural Networks (IJCNN '02)*, volume 2, pages 1895–1899. IEEE, 2002.

[266] L. M. Meng. An image-based bayesian framework for face detection. In *IEEE International Conference On Computer Vision and Pattern Recognition*, 2000.

[267] Y. Meyer. *Wavelets: Algorithms and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1993.

[268] A. Micheli, A. Sperduti, and A. Starita. An introduction to recursive neural networks and kernel methods for cheminformatics. *Current Pharmaceutical Design*, 13(14):1469–95, 2007.

[269] B. Milner and X. Shao. Speech reconstruction from mel-frequency cepstral coefficients using a source-filter model. In *7th International Conference on Spoken Language Processing*, pages 2421–2424, 2002.

[270] H. Minagawa. Measurement of cattle body surface area by stereo photogrammetry. In *3rd Livestock Environment Symposium*, pages 179–185, 1988.

[271] H. Minagawa, S. Saito, and T. Ichicawa. Determining the weight of pigs with image analysis system. In *Proceedings of the 4th Livestock Environment Symposium*, pages 528–535, 1993.

[272] M. Minsky and S. Papert. *An Introduction to Computational Geometry*. MIT Press, 1969.

[273] T. Misner. *Practical Studio Techniques*. SAE Institute Publication, third edition, 2001.

[274] A. D. Mitchell, T. G. Ramsay, and A. M. Scholz. Measurement of changes in body composition of piglets from birth to 4 kg using quantitative magnetic resonance (qmr). *Archiv Tierzucht*, 55:64–67, 2012.

[275] R. Moen and C. Norman. Evolution of the pdca cycle. Unpublished work, 2011.

[276] B. Moghaddam and M. H. Yang. Learning gender with support faces. *IEEE Transcripts on Pattern Analysis and Machine Intelligence*, 24:707–711, 2002.

[277] J. M. Moguerza and A. Munoz. Support vector machines with applications. *Statistical Science*, 21(3):322–336, 2006.

[278] S. Molau, M. Pitz, R. Schluter, and H. Ney. Computing mel-frequency cepstral coefficients on the power spectrum. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 73–76, 2001.

[279] F. Montossia, M. Font-i Furnolsb, M. del Campoa, R. San Juliana, G. Britoa, and C. Sanudoc. Sustainable sheep production and consumer preference trends: Compatibilities, contradictions, and unresolved dilemmas. *Meat Science*, 95(4):772–789, 2013.

[280] R.J.T. Morris and L.D. Rubin. Technique for object orientation detection using a feed-forward neural network, 1991.

[281] S. I. Mortimer, R. D. Warner, G. H. Geesink, J. E. Hocking Edwards, E. N. Ponnampalami, A. J. Ball, A. R. Gilmour, and D. W. Pethick. Genetic parameters for meat quality traits of australian lamb meat. *Meat Science*, 96:1016–1024, 2014.

[282] D. Moshou, A. Chedad, A. Van Hirtum, J. De Baerdemaeker, D. Berckmans, and H. Ramon. An intelligent alarm for early detection of swine epidemics based on neural networks. *American Society of Agricultural Engineers*, 44(1):167–174, 2001.

[283] M. Mottaghi-Kashtiban and M. G. Shayesteh. New efficient window function, replacement for the hamming window. *IET Signal Processing*, 5(5):499, 2011.

[284] M. Moya, M. Koch, and L. Hostetler. One-class classifer networks for target recgonition applications. In *Processings World Congress on Neural Networks*, pages 797–801, 1993.

[285] A. Mucherino and Rub. G. Recent developments in data mining and agriculture. In *11th ICDM Conference on Advances in Data Mining*. IBaI Publishing, 2011.

[286] A. Mucherino, P. J. Papajorgji, and P. Pardalos. *Data Mining in Agriculture*. Springer, 2009.

[287] L. Muda, M. Begam, and I. Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *Journal of Computing*, 2(3):138–143, 2010.

[288] J. Munoz-Mari, G. Camps-Valls, L. Gomez-Chova, and J. Calpe-Maravilla. Combination of one-class remote sensing image classifiers. In *IEEE International Geoscience and Remote Sensing Symposium*, 2007.

[289] M. Murugappan, N. Ramachandran, and Y. Sazali. Classification of human emotion from eeg using discrete wavelet transform. *Journal of Biomedical Science and Engineering*, 3:390–396, 2010.

[290] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[291] D. E. Newland. Harmonic wavelet analysis. *Proceedings of the Royal Society of London, Series A (Mathematical and Physical Sciences)*, 443(1917):203–225, 1993.

[292] M. Nielen, M. N. Spigt, and K. Maatje. Detecting mastitis with a neural network using electrical conductivity data. In *International Symposium on Prospects for Automatic Milking*, pages 370–376, 1992.

[293] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[294] R. Nowak. Lamb's bleats: Important for the establishment of the mother-young bond? *Behaviour*, 115(1):14–29, 1990.

[295] S. Ntalampiras and N. Fakotakis. *Speech/Music Discrimination Based on Discrete Wavelet Transform*, book section 18, pages 205–211. Springer, 2008.

[296] S. Ntalampiras, I. Potamitis, and N. Fakotakis. Acoustic detection of human activities in natural environments. *Journal of the Audio Engineering Society*, 60(9):1–10, 2012.

[297] A. H. Nuttall. Some windows with very good sidelobe behavior. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(1):84–91, 1981.

[298] H. Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

[299] R. Olafsson, E. Martindottir, G. Olafsdottir, P. I. Sigfusson, and J. W. Gardner. *Monitoring of fish freshness using tin oxide sensors*, volume 212 of *Sensors and Sensory Systems for an Electronic Nose*, book section 16, pages 257–272. Spring, Kluwer, Netherlands, 1992.

[300] E. F. Oliveira, A. G. Bianchi, L. S. Martins-Filho, and R. F. Machado. Granulometric analysis based on the energy of wavelet transform coefficients. *Revista Escola de Minas*, 63(2):347–354, 2010.

[301] H. Olkkonen. *Discrete Wavelet Transforms: Algorithms and Applications*. InTech, Rijeka, Croatia, 2011.

[302] D. L. Olson and D. Delen. *Advanced Data Mining Techniques*. Springer, 2008.

[303] C. M. Onyango, J. A. Marchant, J. R. Lake, and D. A. Stanbridge. A low maintenance conductivity sensor for detecting mastitis. *Journal of Agricultural Engineering Research*, 40:215–224, 1988.

[304] S. J. Orfanidis. *Introduction to Signal Processing*. Prentice Hall, Inc, 2010.

[305] J. R. Orozco-Arroyave, J. D. Arias-Londo, J. F. Vargas-Bonilla, and E. Noth. *Perceptual Analysis of Speech Signals from People with Parkinson's Disease*, volume 7930 of *Lecture Notes in Computer Science*, book section 21, pages 201–211. Springer, 2013.

[306] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[307] B. Owsinski. *The Recording Engineer's Handbook*. ArtistPro Publishing, Boston, MA, USA, 2005.

[308] B. Owsinski. *The Mixing Engineer's Handbook*. Thomson Course Technology, Boston, MA, USA, second edition, 2006.

[309] C. Ozkan and F. S. Erbek. The comparison of activation functions for multispectral landsat tm image classification. *Photographic Engineering and Remote Sensing*, 69(11):1225–1234, 2003.

[310] Y. Pan, P. Shen, and L. Shen. Speech emotion recognition using support vector machine. *International Journal of Smart Home*, 6(2):101–108, 2012.

[311] R. Panda, P. S. Khobragade, P. D. Jambhule, S. N. Jengthe, P. R. Pal, and T. K. Gandhi. Classification of eeg signal using wavelet transform and support vector machine for epileptic seizure diction. In *International Conference on Systems in Medicine and Biology (ICSMB)*, pages 405–408. IEEE, 2010.

[312] J. Park, D. Kang, J. Kim, J. T. Kwok, and I. W. Tsang. Svdd-based pattern denoising. *Neural Computation*, 19(7):1919–1938, 2007.

[313] H. Patel and D. Patel. A brief survey of data mining techniques applied to agricultural data. *International Journal of Computer Applications*, 95:6–8, 2014.

[314] K. Patel. Speech recognition and verification using mfcc and vq. *International Journal of Emerging Science and Engineering (IJESE)*, 1(7):33–37, 2013.

[315] C. Paterson. Identifying causes for lamb losses in low rainfall mixed farming regions. In *Eyre Peninsula Farming Systems*, pages 120–122, 2012.

[316] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy. Gene functional classification from heterogeneous data. In *Fifth Annual International Conference on Computational Biology*, pages 249–255. ACM Press, New York, 2001.

[317] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[318] S. Pedersen and C. B. Perdersen. Animal activity measured by infrared detectors. *journal of Agricultural Engineering Research*, 61:239–246, 1995.

[319] G. Peeters. A large set of audio features for sound description (similarity and classifcation) in the cuidado project. Report, IRCAM, 2004.

[320] E. M. Pereira, I. A. Naas, and F. C. Jacob. Using vocalization pattern to assess broilers well-being. *Precision Livestock Farming*, 11:11–14, 2011.

[321] K. C. Persaud and G. Dodd. Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose. *Nature*, 299:352–355, 1982.

[322] K. C. Persaud and P. Travers. Multielement arrays for sensing volatile chemicals. In *Intelligent Instruments and Computers*, pages 147–153, 1991.

[323] T. V. Pham, M. W. Arnold, and W. M. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23(4):451–461, 2002.

[324] F. Piekniewski and L. Rybicki. Visual comparison of performance for different activation functions in mlp networks. In *IEEE International Joint Conference on Neural Networks*, volume 4, pages 2947–2952. IEEE, 2004.

[325] W. Pitts. Some observations on the simple neuron circuit. *The bulletin of mathematical biophysics*, 4(3):121–129, 1942.

[326] R. D. Poblete. *Manipulation of Audio in the Wavelet Domain Processing a Wavelet Stream Using PD*. Institut fr Elektronische Musik (IEM), Graz, Austria, 2006.

[327] P. Podder, T. V. Khan, and M. H. Khan. Comparative performance analysis of hamming, hanning, and blackman window. *International Journal of Computer Applications*, 96(18):1–7, 2014.

[328] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[329] J. C. Pollard. Effects of litter size on the vocal behaviour of ewes. *Applied Animal Behaviour Science*, 34:75–84, 1992.

[330] P. Porwik and A. Lisowska. The haarwavelet transform in digital image processing: Its status and achievements. *Machine Graphics and Vision*, 12(1/2):79–98, 2004.

[331] A. D. Poularikas. *The Handbook of Formulas and Tables for Signal Processing*. CRC Press LLC, 1999.

[332] S. Prabhakar, A. R. Mohanty, and A. S. Sekhar. Application of discrete wavelet transform for detection of ball bearing race faults. *Tribology International*, 35(12):793–800, 2002.

[333] P. Prandoni and M. Vetterli. *Signal Processing for Communications*. EPFL Press, Lausanne, Switzerland, 2008.

[334] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice-Hall International, Inc, Upper Saddle River, New Jersey, third edition, 1996.

[335] A. Prochazka, J. Kukal, and O. Vysata. Wavelet transform use for feature extraction and eeg signal segments classification. In *3rd International Symposium on Communications, Control, and Signal Processing*, pages 719–722. IEEE, 1999.

[336] Productivity-Commission. Trends in australian agriculture. Technical report, Australian Government: Productivity Commission, 2005.

[337] Prosoundweb.com. Microphone characteristics - cardioid polar pattern, 2016.

[338] I. Z. Prusa. *Segmentwise Discrete Wavelet Transform*. Thesis, Brno University of Technolgy, 2012.

[339] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12:145–151, 1999.

[340] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[341] V. Rajasekhar, V. Vaishnavi, J. Koushik, and M. Thamarai. An efficient image compression technique using discrete wavelet transform (dwt). In *International Conference on Electronics and Communication Systems (ICECS)*, pages 1–4, 2014.

[342] P. Rajmic and J. Vlach. Real-time audio processing via segmented wavelet transform. In *10th International Conference on Digital Audio Effects (DAFX-07)*, pages 1–4, 2007.

[343] D. Ramash and B. Vishnu Vardhan. Data mining techniques in agricultural and environmental sciences. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(9):3477–3480, 2013.

[344] D. Ramesh and B. V. Vardhan. Data mining techniques and applications to agricultural yield data. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(9):3477–3480, 2013.

[345] RaspberryPi. Raspberry pi official web page, 2016.

[346] V. Rayner, N. Tan, and N. Ward. Trends in farm sector output and exports. Report, Reserve Bank of Australia, 2010.

[347] L. Richard. Windowing functions to improve fft results - part 1. *Test & Measurement World*, June 1998:50–58, 1998.

[348] G. Ritter, M. Gallegos, and (). Outliers in statistical pattern recognition and an application to automatic chromosome classification. *Pattern Recognition Letters*, 18:525–539, 1997.

[349] P. J. Roberts and R. A. Walker. Application of a counter propagation neural network for star identification. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2005.

[350] J. F. Robertson and J. Benzie. Assessment of a cough counter for pigs. *Farm Build*, 95:25–28, 1989.

[351] Rode. Rode microphones - nt5 condenser microphone, 2016.

[352] D. Roobaert and M. M. Van Hulle. *View-based 3d object recognition with support vector machines*, book section 9, pages 77–86. IEEE, 1999.

[353] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[354] O. A. Rosso, S. Blanco, J. Yordanova, V. Kolev, A. Figliola, M. Schurmann, and E. Basar. Wavelet entropy: a new tool for analysis of short duration brain electrical signals. *Journal of Neuroscience Methods*, 105:65–75, 2001.

[355] L. Ruiz-Garcia, L. Lunadei, P. Barreiro, and J. I. Robla. A review of wireless sensor technologies and applications in agriculture and food industry: state of the art and current trends. *Sensors (Basel)*, 9(6):4728–4750, 2009.

[356] D. E. Rumelhart and J. L. MacClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA, 1986.

[357] S. M. Rutter, N. A. Beresford, and G. Roberts. Use of gps to identify the grazing areas of hill sheep. *Computers and Electronics in Agriculture*, 17(2):177–188, 1997.

[358] A. Safari and N. M. Fogarty. *Genetic Parameters for Sheep Production Traits: Estimates from the Literature.* Australian Sheep Industry Cooperative Research Centre, 2003.

[359] S. E. Safty and A. El-Zonkoly. Applying wavelet entropy principle in fault classification. *World Academy of Science, Engineering and Technology*, 40:133–135, 2008.

[360] H. Sak, A. W. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech*, pages 338–342, 2014.

[361] G. Saon and J. T. Chien. Large-vocabulary continuous speech recognition systems. *IEEE Signal Processing Magazine*, 29(6):18–33, 2012.

[362] C. S. Sapp. Wave pcm soundfile format, 2005.

[363] F. Sbe, T. Aubin, R. Nowak, and P. Poindron. Establishment of vocal communication and discrimination between ewes and their lamb in the first two days after parturition. *Developmental Psychobiology*, 49:375–386, 2007.

[364] F. Sbe, J. Duboscq, T. Aubin, S. Ligout, and P. Poindron. Early vocal recognition of mother by lambs: contribution of low- and high-frequency vocalizations. *Animal Behaviour*, 79(5):1055–1066, 2010.

[365] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[366] M. Scherf and W. Brauer. Feature selection by means of a feature weighting approach. Report, University of Munchen - Institute fur Informatik, 1997.

[367] B. Schoelkopf. *Support Vector Learning*. Thesis, Technischen University, Berlin, 1997.

[368] C. P. Schofield. Evaluation of image analysis as a means of estimating the weight of pigs. *Journal of Agricultural Engineering Research*, 47:287–296, 1990.

[369] C. P. Schofield and J. A. Marchant. Image analysis for estimating the weight of live animals. *International Society of Optical Engineering*, 1379:209–219, 1991.

[370] C. P. Schofield, J. A. Marchant, R. P. White, N. Brandl, and M. Wilson. Monitoring pig growth using a prototype imaging system. *Journal of Agricultural Engineering Research*, 72:205–210, 1999.

[371] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12*, pages 582–588, 2000.

[372] A. M. Scholz, L. Bnger, J. Kongsro, U. Baulain, and A. D. Mitchell. Non-invasive methods for the determination of body and carcass composition in livestock: dual-energy x-ray absorptiometry, computed tomography, magnetic resonance imaging and ultrasound: invited review. *Animal*, 9(7):1250–1264, 2015.

[373] A. Searby and P. Jouventin. Mother-lamb acoustic recognition in sheep: a frequency coding. *Proceedings. Biological sciences / The Royal Society*, 270(1526):1765–71, 2003.

[374] F. Sebe, T. Aubin, A. Boue, and P. Poindron. Mother-young vocal communication and acoustic recognition promote preferential nursing in sheep. *Journal of Experimental Biology*, 211(Pt 22):3554–3562, 2008.

[375] T. Sekozawa. Dynamic programming matching for detecting abnormalities in machines emitting intermittent sounds. In *Recent Advances in Applied & Biomedical Informatics and Computational Engineering in Systems Applications*. World Scientific and Engineering Academy and Scoiety (WSEAS), 2011.

[376] I. W. Selesnick, R. G. Baraniuk, and N. C. Kingsbury. The dual-tree complex wavelet transform. *IEEE Signal Processing Magazine*, 22(6):123–151, 2005.

[377] L. A. Shalabi, Z. Shaaban, and B. Kasabeh. Data mining: A preprocessing engine. *Journal of Computer Science*, 2(9):735–739, 2006.

[378] C. E. Shannon. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers*, 37(1):10–21, 1949.

[379] Y. Shao, S. Srinivasan, Z. Jin, and D. Wang. A computational auditory scene analysis system for speech segregation and robust speech recognition. *Computer Speech and Language*, 24(1):77–93, 2010.

[380] J. Shawe-Taylor. Structural risk minimization over datadependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

[381] J. Shawe-Taylor and N. Cristianini. *Margin distribution and soft margin*, page 349358. MIT Press, 2000.

[382] A. B. Shawkat Ali and S. A. Wasimi. *Data Mining: Methods and Techniques*. Thomson, 2007.

[383] H. V. Shurmer. An electronic nose: a sensitive and discriminating substitue for a mammalian olfactory system. *Circuits, Devices and Systems, IEE Proceedings G*, 137(3):197–204, 1990.

[384] P. Sibi, S. Allwyn Jones, and P. Siddarth. Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, 47(3):1264–1268, 2013.

[385] H. K. Singh, S. K. Tomar, and P. Singh. Analysis of multispectral image using discrete wavelet transform. In *Third International Conference on Advanced Computing and Communication Technologies (ACCT)*, pages 59–62. IEEE, 2013.

[386] V. K. Singh. Discrete wavelet transform based image compression. *International Journal of Remote Sensing*, 20(17):3399–3405, 1999.

[387] N. Singh-Miller, M. Collins, and T. J. Hazen. Dimensionality reduction for speech recognition using neighborhood components analysis. In *Conference of the International Speech Communication Association*, page 4, 2007.

[388] A. K. Smilde, M. J. van der Werf, S. Bijlsma, B. van der Werff-van der Vat, and R. H. Jellema. Fusion of mass-spectrometry-based metabolomics data. *Annuals of Chemistry*, 77:6729–6736, 2005.

[389] F. W. Smith. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, C-17(4):367–372, 1968.

[390] S. W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Calfornia Technical Publishing, San Diego, CA, 1997.

[391] Y. Souissi, M. Abdelaziz El Aabid, N. Debande, S. Guilley, and J. Danger. Novel applications of wavelet transforms based side-channel analysis. In *Non-Invasive Attacking Testing Workshop*, 2011.

[392] K. F. Stacey, D. J. Parsons, A. R. Frost, C. Fisher, D. Filmer, and A. Fothergill. An automatic growth and nutrition control system for broiler production. *Biosystems Engineering*, 89(3):363–371, 2004.

[393] J. L. Starck, F. Murtagh, and R. Gastaud. A new entropy measure based on the wavelet transform and noise modeling. *IEEE Transactions on Circuits and Systems - II Analog and Digital Signal Processing*, 45(8):1118–1124, 1998.

[394] R. Stern and N. Morgan. *Features based on auditory physiology and perception*, pages 193–222. Wiley, New York, NY, USA, 2012.

[395] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics Theory and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[396] N. D. Stone, P. B. Siegel, C. S. Adkisson, and W. B. Grass. Vocalisations and behaviour of two commercial stocks of chickens. *Poultry Science*, 63:616–619, 1984.

[397] G. Strang and T. Nguyen. *Wavelets and filter banks*. Wellesley-Cambridge Press, 1996.

[398] R. L. Stratonovich. Conditional markov processes. *Theory of Probability and its Applications*, 5(2):156–178, 1960.

[399] M. Sundermeyer, I. Oparin, J. L. Gauvain, B. Freiberg, R. Schluter, and H. Ney. Comparison of feedforward and recurrent neural network language models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8430–8434, 2013.

[400] R. Suter. Improving lamb survival: case studies from the sentinel flock project. *Sheep Notes*, Autumn 2013:12–13, 2013.

[401] K. Suzuki. *Artificial Neural Networks - Methodological Advances and Biomedical Applications*. InTech, Croatia, 2011.

[402] D. Svozil, V. Kvasnicka, and J. Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39:43–62, 1997.

[403] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. Report, Google, 2015.

[404] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems 26*, pages 2553–2561, 2013.

[405] E. Szentirmai, G. Milisits, T. Donk, Z. Budai, J. Ujvri, T. Flp, I. Repa, and Z. St. Comparison of changes in production and egg composition in relation to in vivo estimates of body weight and composition of brown and white egg layers during the first egg-laying period. *British Poultry Science*, 54:587–593, 2013.

[406] D. Tax. *One Class Classification*. Phd, Delft University of Technology, 2001.

[407] D. Tax and R. Duin. Outlier detection using classifier instability. In *Advances in Pattern Recognition, the Joint IAPR International Workshops*, pages 593–601, 1998.

[408] D. Tax and R. Duin. Data domain description using support vectors. In *European Symposium on Artificial Neural Networks*, pages 251–256, 1999.

[409] D. Tax and R. Duin. Combining one-class classifiers. *Lecture Notes in Computer Science*, 2096:299–308, 2001.

[410] D. Tax and R. Duin. Uniform object generation for optimizing one-class classifier. *Journal of Machine Learning Research*, 2:155–173, 2001.

[411] D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004.

[412] A. Terrazas, R. Nowak, N. Serafin, A. Ferreira, F. Lvy, and P Poindron. Twenty-four-hour-old lambs rely more on maternal behavior than on the learning of individual characteristics to discriminate between their own and an alien mother. *Developmental Psychobiology*, 40:408–418, 2002.

[413] Thingsee. Thingsee: The iot device platform, 2016.

[414] M. Thompson. A comparison of fft window functions. Magazine article, Agilent Technologies, 1993.

[415] R. D. Tillett. Model-based image processing to locate pigs within images. *Computers and Electronics in Agriculture*, 6:51–61, 1991.

[416] K. Tona, F. Bamelis, B. De Ketelaere, V. Bruggeman, V. M. Moraes, J. Buyse, O. Onagbesan, and E. Decuypere. Effects of egg storage time on spread of hatch, chick quality and chick juvenile growth. *Poultry Science*, 82:736–741, 2003.

[417] B. Toufik and N. Mokhtar. *The Wavelet Transform for Image Processing Applications*, pages 395–423. InTech, 2012.

[418] D. J. Troy and J. P. Kerry. Consumer perceptions and the role of science in the meat industry. *Meat Science*, 86:214–226, 2010.

[419] M. J. Turner, P. Gurney, J. S. Crowther, and J. R. Sharp. An automatic weighing system for poultry. *Journal of Agricultural Engineering Research*, 29:17–24, 1984.

[420] G. Tzanetakis, G. Essl, and P. Cook. Audio analysis using the discrete wavelet transform. In *WSES International Conference of Acoustics and Music: Theory and Applications*, pages 1–6, 2002.

[421] S. Van Compernolle, S. Janssens, R. Geers, H. Ville, A. Oosterlinck, V. Goedseels, K. Goossens, G. Arduyns, J. Van Bael, and L. Bosschaerts. Welfare monitoring of pigs by automatic speech processing. In *12th International Pig Veterinary Society Conference*, 1992.

[422] R. A. van den Berg, H. C. Hoefsloot, J. A. Westerhuis, A. K. Smilde, and M. J. van der Werf. Centering, scaling, and transformations: improving the biological information content of metabolomics data. *BMC Genomics*, 7(142):15, 2006.

[423] E. Van der Stuyft, C. P. Schofield, J. M. Randall, P. Wambacq, and V. Goedseels. Development and application of computer vision systems for use in livestock production. *Computers and Electronics in Agriculture*, 6:243–265, 1991.

[424] A. Van Hirtum, J. M. Aerts, D. Berckmans, B. Moreaux, and P. Gustin. On-line cough recognizer system. *Journal of Acousticla Society of America*, 106:4, 1999.

[425] A. Van Hirtum and D. Berckmans. Fuzzy approach for improved recognition of citric acid induced piglet coughing from continuous registration. *Journal of Sound and Vibration*, 266(3):677–686, 2003.

[426] A. Van Hirtum and D. Berckmans. Objective recognition of cough sound as a biomarker for aerial pollutants. *Indoor Air*, 14:10–15, 2004.

[427] V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25:., 1964.

[428] V. Vapnik, S. E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems*, 9:281–287, 1997.

[429] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[430] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, USA, first edition, 1995.

[431] T. Virtanen, R. Singh, and B. Raj. *Techniques for Noise Robustness in Automatic Speech Recognition*. Wiley, New York, NY, USA, 2012.

[432] P. von Rohr, A. Hofer, and N. Kunzi. Economic values for meat quality traits in pigs. *Journal of Animal Science*, 77:2633–2640, 1999.

[433] G. Wahba. Spline models for observational data. In *CBMS-NSF Regional Conference Series in Applied Mathematics*, volume 59. SIAM: Society for Industrial and Applied Mathematics, 1990.

[434] J. S. Walker. *A Primer on Wavelets and their Scientific Applications*. Chapman & Hall / CRC Press LLC, 1999.

[435] E. S. Walser and P. Hague. Variations in the structure of bleats from sheep of four different breeds. *Behaviour*, 75(1):22–35, 1980.

[436] E. S. Walser, P. Hague, and E. Walters. Vocal recognition of recorded voices by lambs ewes of three breeds of sheep. *Behaviour*, 78(3):260–272, 1981.

[437] E. S. Walser, E. Walters, and P. Hague. A statistical analysis of the structure of bleats from sheep of four different breeds. *Behaviour*, 77(1):67–76, 1981.

[438] E. S. Walser, E. Walters, and P. Hague. Vocal communication between ewes and their own and alien lambs. *Behaviour*, 81(2):140–151, 1982.

[439] E. S. Walser, E. Walters, and P. Hague. A statistical analysis of vocal communication between ewes and lambs. *Behaviour*, 85(1):146–156, 1983.

[440] C. Wang, C. Ding, R. F. Meraz, and S. R. Holbrook. Psol: a positive sample only learning algorithm for finding non-coding rna genes. *Bioinformatics*, 22(21):2590–6, 2006.

[441] C. M. Wathes. *Precision Livestock Farming for Animal Health, Welfare, and Production*, pages 397–404. Wageningen Academic Publishers, Wageningen, the Netherlands, 2007.

[442] C. M. Wathes, H. H. Kristensen, J. M. Aerts, and D. Berckmans. Is precision livestock farming an engineer's daydream or nightmare, an animal's friend or foe, and a farmer's panacea or pitfall? *Computers and Electronics in Agriculture*, 64(1):2–10, 2008.

[443] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Thesis, Harvard University, 1974.

[444] R. P. White, C. P. Schofield, D. M. Green, D. J. Parsons, and C. T. Whittemore. The effectiveness of a visual image analysis (via) system for monitoring the performance of growing / finishing pigs. *Animal Science*, 78:409–418, 2004.

[445] C. T. Whittemore and C. P. Schofield. A case for size and shape scaling for understanding nutrient use in breeding sows and growing pigs. *Livestock Production Science*, 65:203–208, 2000.

[446] J. L. Williams. *Genetic Control of Meat Quality Traits*, book section 2, pages 21–60. Springer Science and Business, 2008.

[447] Ian H. Witten, Eibe Frank, and Mark A. Hall. Data mining : Practical machine learning tools and techniques, 2011.

[448] H. Wold. *Nonlinear estimation by iterative least squares procedures*, pages 411–444. Wiley, New York, NY, 1966.

[449] S. Wold. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2:37–52, 1987.

[450] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[451] D. G. Wood-Gush. *Elements of Ethology*. Chapman and Hall, New York, The Edinburgh School of Agriculture, University of Edinburgh, 1983.

[452] T. Xie, H. Yu, and B. Wilamowski. Comparison between traditional neural networks and radial basis function networks. In *IEEE International Symposium on Industrial Electronics (ISIE)*, pages 1194–1199, 2011.

[453] Z. Xu, G. Huang, K. Weinberger, and A. Zheng. Gradient boosted feature selection. In *20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014.

[454] V. K. Yadav, A. Jain, and L. Bhargav. Analysis and comparison of audio compression using discrete wavelet transform. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(1):310–313, 2015.

[455] Yamaha. Yamaha hs8 powered studio monitors, 2016.

[456] R. E. Yohanes, W. Ser, and G. B. Huang. Discrete wavelet transform coefficients for emotion recognition from eeg signals. In *Conference Proceedings of the IEEE Engineering in Medicine and Biology Society*, pages 2251–2254, 2012.

[457] H. J. Yu. Svmc - single-class classification with support vector machines. In *18th International Joint Conference on Artificial Intelligence*, pages 567–572. Morgan Kaufmann Publishers, 2003.

[458] M. J. Zaki and L. Wong. Data mining techniques. *WSPC Lecture Notes Series*, 9(6):1–40, 2003.

[459] J. R. Zhang, J. Zhang, T. M. Lok, and M. R. Lyu. A hybrid particle swarm optimizationback-propagation algorithm for feedforward neural network training. *Applied Mathematics and Computation*, 185(2):1026–1037, 2007.

[460] Y. Zhang, Z. Dong, A. Liu, S. Wang, G. Ji, Z. Zhang, and J. Yang. Magnetic resonance brain image classification via stationary wavelet transform and generalized eigenvalue proximal svm. *Journal of Medical Imaging and Health Informatics*, 5(7):1395–1403, 2015.

[461] Y. Zhang, Z. Dong, S. Wang, G. Ji, and J. Yang. Preclinical diagnosis of magnetic resonance (mr) brain images via discrete wavelet packet transform with tsallis entropy and generalized eigenvalue proximal support vector machine (gepsvm). *Entropy*, 17(4):1795–1813, 2015.

[462] Y. Zhang, S. Wang, Y. Huo, L. Wu, and A. Liu. Feature extraction of brain mri by stationary wavelet transform and its applications. *Journal of Biological Systems*, 18(s1):115–132, 2010.

[463] Y. Zheng and E. A. Essock. A novel feature extraction method wavelet-fourier analysis and its application to glaucoma classification. In *7th Joint Conference on Information Sciences*, volume September, pages 672–675, 2003.

[464] Y. Zhu and S. Schwartz. Efficient face detection with multiscale sequential classification. In *IEEE International Conference on Image Processing*, volume 121, 2002.

[465] A. Zien, G. Rtsch, S. Mika, B Schlkopf, T. Lengauer, and K. Mller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.