UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

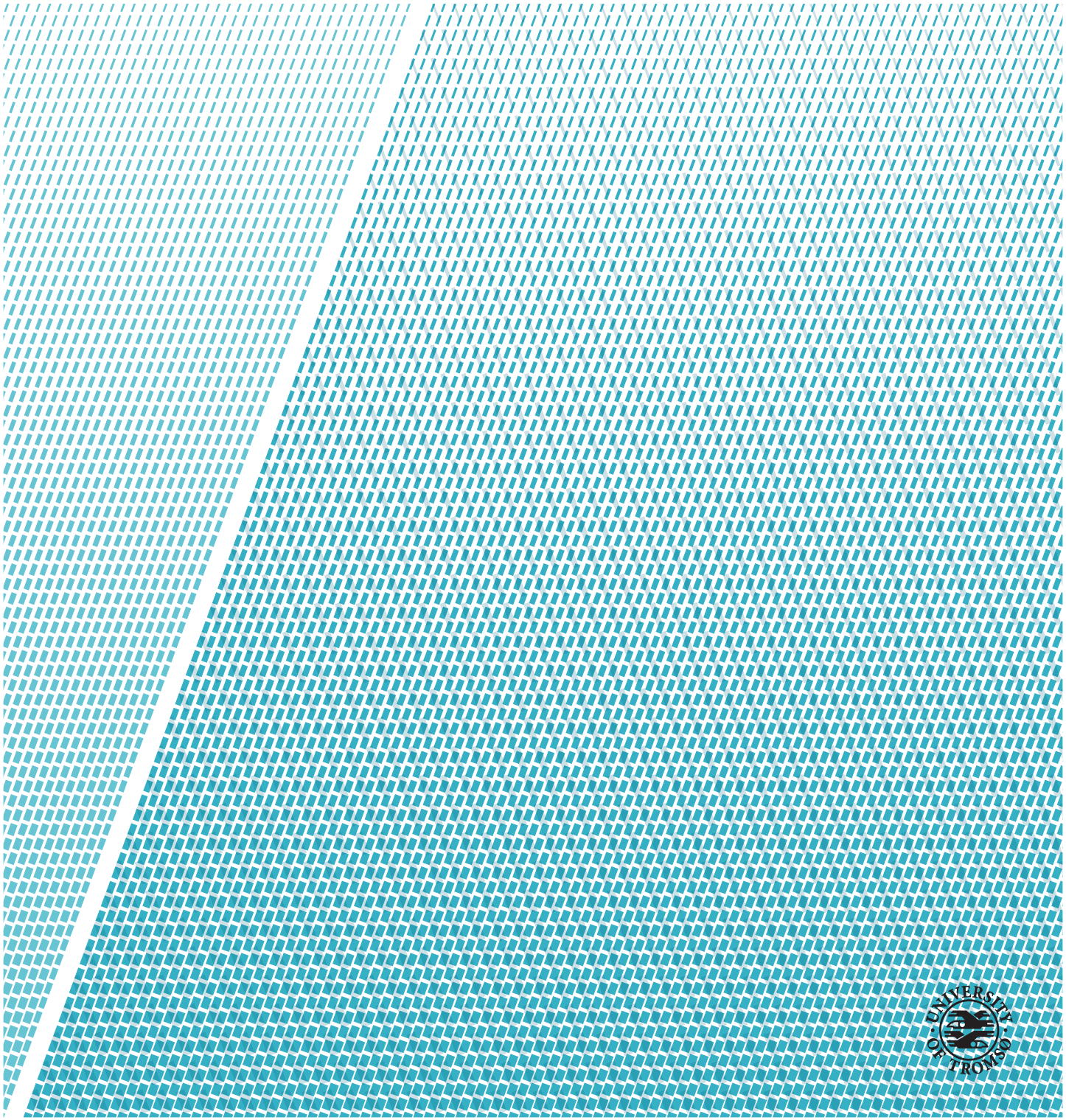
Faculty of Science and Technology
Department of Computer Science

# Pulmonary Crackle Detection Using Signal Processing and Machine Learning

—

**Morten Grønnesby**

*Capstone Project in Computer Science, November 2015*



# Abstract

In listening to lung sounds with stethoscopes or *lung auscultation*, pulmonary crackles are used as an indicator of pulmonary disease. General Practitioners often use the stethoscope when referring patients to pulmonologists, but there is a lot of subjectivity in what each general practitioner hears and classifies as a crackle. We propose an approach to do a computerized analysis to detect crackles in recordings from stethoscopes, through using signal processing and machine learning. We created an analysis pipeline that first pre-processes each audio file, dividing the audio signal into smaller windows and extracting features from these windows. Second our pipeline classifies each of these windows, we use the features extracted with a hierarchy of Support Vector Machines. They cooperate using a probability measure for whether there is an abnormal sound or normal sound in the current window. Finally we present the findings of each window to the user as an annotated waveform.

We evaluated the approach using cross validation when training our classifiers and also running experiments with our classifiers on data from the biggest dataset of lung sounds to date. Our results show that our pipeline picks up a rate of 1.3 crackles in each Normal audio file, which may be false positives, but crackles might occur naturally in lungs without further significance. More interesting, our pipeline found a rate of 5.3 crackles per crackle signal. We also evaluated our training accuracy of our classifiers, and we got 87% accuracy. Our approach, realized through our pipeline, can be used by medical staff both as a training tool or as an aid in diagnosis when listening to lungs.

# Acknowledgements

I would like to thank *Lars Ailo*, *Hasse Melbye* and *Juan Carlos Aviles* for allowing me to work as a summer intern which lead to this project, and thanks to *Ida Jacklin Johansen* for organizing communication between me and *Lars Ailo*.

I would also like to thank *Bjørn Fjukstad* and *Einar Holsbø* for helping me throughout the project and also welcoming me into their office space, and thanks to the whole *BDPS* research group for input in the project. In addition I would like to thank *Robert Jenssen* and *Michael Kampffmeyer* for their support in the Machine Learning field.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

In medicine there are many classification problems, such as discriminating between positive or negative findings, or different types of findings. This is true in Pulmonary[1] medicine as well. Often these tasks are prone to subjectivity of the observer, and may therefore be unreliable diagnostic indicators. Therefore there is a need for a gold standard that people in the medical field can use in both training, and as aid in practice.

In this report we propose a system for doing a computerized analysis and classification of one of these diagnostic indicators, namely *Lung Sounds*. We use signal processing and machine learning techniques to perform this classification task. Our goal is to create an approach for training medical students, and also to help medical professionals in classifying of *Lung Sounds*.

This project is part of the study *Inter-observer variation in categorizing lung sounds. A comparison between experts, lung specialists and general practitioners* (from now abbreviated the *Inter-observer Study*). The study is done by Juan Carlos Aviles Solis, Prof. Hasse Melbye and Prof. Peder Halvorsen. They will record the biggest repository of expert curated lung sounds in the world. The study is a part of *Tromsøundersøkelsen* [UiT, ], which is a long term epidemiological study conducted periodically in Tromsø since 1974.

1. Relating to, associated with, or affecting the lungs

## 1.1 Gold Standard

Our main goal is to develop an approach to classify sounds into different types, or classes, of lung sounds. As a part of the *Inter-observer Study*, the investigators will record lung sound samples from close to 3,000 patients, each session consisting of 6 audio recordings per patient, which culminates to around 18,000 audio samples. Out of these samples, we estimate it to be abnormal sounds present in about 10% of the recordings. The purposes of the *Inter-observer Study* will be to create this gold standard for lung sounds.

To create the gold standard, this large dataset has to be analysed manually and each sound file has to be classified as either normal or containing a type of abnormal sounds. In *lung auscultation* there are three different types of abnormal sounds that are relevant: *Crackles*, *Wheezes* and *Rhonchi*. All findings has to be agreed upon by three different doctors with experience in *pulmonology*[2]. This is a time consuming procedure, and possible disagreement adds even more time complexity to the process.

The resulting gold standard helps with the main challenge that projects dealing with automatic lung sound classification; the lack of large, diverse and representative datasets.

## 1.2 Related Work

The two most important fields of related work are machine learning and automatic crackle detection. Previous studies have applied both machine learning and rule-based decision systems to problems in automatic crackle detection.

Machine learning techniques applied to audio tend to often be in speech recognition[Deng and Li, 2013], music recommendation[van den Oord et al., 2013] and more specialized fields such as dental drill sound recognition[Zakeri et al., 2015].

We have seen that techniques for detecting abnormal lung sounds that are rule based, using a set of parameters extracted using signal processing[Pinho et al., 2015]. Some studies have also applied machine learning algorithms to these problems such as *SVMs and Neural Networks*[Serbes et al., 2013][Göğüş et al., 2015] and with very good results, but a common problem is the lack a large, diverse dataset. We believe that if the approach should have a real world application, it must also be able to deal with diverse data recorded under normal clinical

2. Medical specialization involving lung conditions and diseases

circumstances.

In addition, the *Inter-observer study* repository lung sounds provides the needed large multi-annotator gold standard, and also what sets this project apart from previous studies. There have also been few projects to apply machine learning to this type of data, and often the input audio are already divided into windows. Other repositories have audio that are recorded under special circumstances not available to general practitioners.

## 1.3 Automatic Classification

Manual classification can be a time consuming task. We therefore propose to use a computerized analysis in order to automatically classify the mentioned dataset.

Our solution is to develop an approach in collaboration with the Department of Public Health and General Practice to automatically classify these Pulmonary Sounds collected in *Tromsøundersøkelsen*. Our approach uses signal processing techniques to generate features that can later be given to our classifiers. There are several types of features that might be relevant to classification of pulmonary audio, especially from the areas of speech recognition and music information retrieval. After feature extraction, we need to employ *Machine Learning* techniques in order to learn a model from the manually classified data, that we can apply to previously unseen data.

Computers can analyse huge amounts of data in only a fraction of the time it would take humans to manually analyse data. We use Machine Learning because it is desirable to have a system that automatically perform classification tasks, for large amounts of data, without needing intervention of a human expert. As Pedro Domingos puts it in his paper *A Few Useful Things to Know about Machine Learning*:

"*Machine learning algorithms can figure out how to perform important tasks by generalizing from examples. This is often feasible and cost-effective where manual programming is not.*"[Domingos, 2012]

One of the main uses of Machine Learning is clustering, which entails finding specific patterns in a medium of data, be it text documents, pictures, music files or even electrical signals. It is a way of discovering correlations in immense amounts of data. An equally important aspect of Machine Learning is classification. Classification requires some prior knowledge about data to formulate a set of classes that correspond to certain patterns. Classification

requires a training set, which is a pre-classified set of data that can be used to learn these patterns and which classes they should adhere to. Learning can be done through several different methods dependent on the type of classifier. In this study the classifiers used are Support Vector Machines, from now abbreviated SVM, which learns patterns by finding a separating hyperplane between features that is plotted in a certain space.

Implementing our approach in a software tool and making it available to researchers in *pulmonology* can be of benefit in classifying and managing large datasets, simplifying data gathering and analysis.

## 1.4 Our Approach

Our approach requires a pipline with the following steps:

*Filtering* is done for all audio files before analysis. Filtering in this context means that any given audio file must be band pass filtered (excluding frequencies above or below a certain threshold) in order to keep the relevant frequency spectrum and discard an frequencies that may be of disturbance to the analysis, for example frequencies that are below 50 Hz are generally not of interest to this study. All audio files must be divided into smaller windows in order to gain a more fine grain analysis in the time scale, meaning that every time an abnormal sound is detected, it should be able to trace it back to its location in time. Dividing the audio also serves the purpose of limiting the amount of data that has to be analysed at a time, making it easier for a machine learning algorithm to distinguish patterns. Lastly this makes each data point (collection of an audio signal) conform to a set size, eliminating the chance that data points may be misclassified due to a lack of standardization of data length and shape.

*Feature Extraction* is important for any machine learning program, as it further decreases the size of the data without discarding the properties of the data. It can be seen as a summary of the data's contents. Our pipeline extracts a certain set of features in order to learn patterns that adhere to each of the different classes.

*Classification* is performed based on the previous steps on audio files of any given size and shape. The classifier employs machine learning techniques and algorithms in order to achieve this. The classifier is trainable (and re-trainable) with different training data. After we classify each audio file, we represent the findings in a way that is easy to understand, but also customizable (visual images, timestamps, boolean values etc.).

*Persistence* of results is important for users of the solution, and export any results into Excel or CSV.

*A User Interface* is provided in order for people not familiar with concepts of programming to be able to interact with proposed solution. This way it is useful for medical research and training, which is one of the main goals of this study. The solution is available through a simple web interface.

## 1.5 Contributions

The contributions of our work are:

- A novel approach to detect crackles in lung sounds, using machine learning and signal processing.
- Implementation and Design of our approach as a automatic pipeline that detects abnormal pulmonary sounds in audio recordings. Our pipeline performs the analysis without requiring any input or preprocessing from a user as this is part of our analysis.
- An experimental evaluation of our classifiers' accuracy through cross validation while training a hierarchy of Support Vector Machines, using the largest database of its kind in the world.

Through our experiments we found that we are able to achieve a cross-validated accuracy of 0.87. Running our analysis pipeline on samples of audio files of varying length we found that it can detect crackles of varying loudness and duration, in files containing ambient (extravascular) noise. We found some problems with these ambient noise sounds as the tubing of the stethoscope can produce crackle-like noises, which leads to a rate of false positives in our normal files. 25% of the normal files produce more than 4 crackles (that are false positives), but many of these are due to the patient talking during examination. 62.5% of the normal files are classified as true negative, which means the remaining 12.5% of our false negatives can be eliminated in post-processing. We found that the distribution of crackles in crackle files were at 60% containing more than 4 crackles. Which means most of the crackle files would not be affected in post-processing.

# 2 Methods

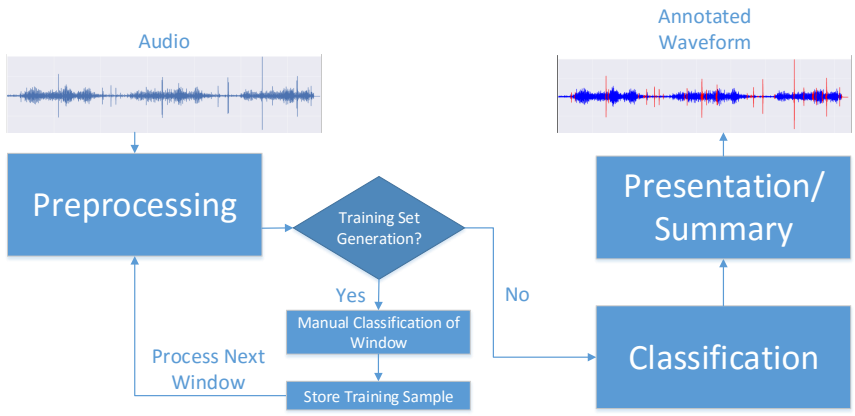



**Figure 2.1:** Overview of the different main steps in analysing audio

This chapter we will present the methods we used and how we arrived at a solution using Machine Learning methods. Our solution can be broken down into three main steps: *Preprocessing*, *Classification* and *Presentation*. In our *Preprocessing* we divide a signal into smaller chunks, as we found that looking at the signal as a whole gave a low accuracy close random guess. In addition the audio files were of varying length. After dividing the signal we do *feature extraction* from each of the smaller chunks before handing the set of features to the *Classifiers*. The classifiers will then label each of the chunks as either

a normal sample or a crackle sample and push the now labeled data to the presentation step. In *Presenting* the now labeled chunks, we rebuild the signal into a waveform representation of the original signal, and annotate each crackle. Further in this chapter we will look at each of these three steps in detail.

## 2.1 Preprocessing

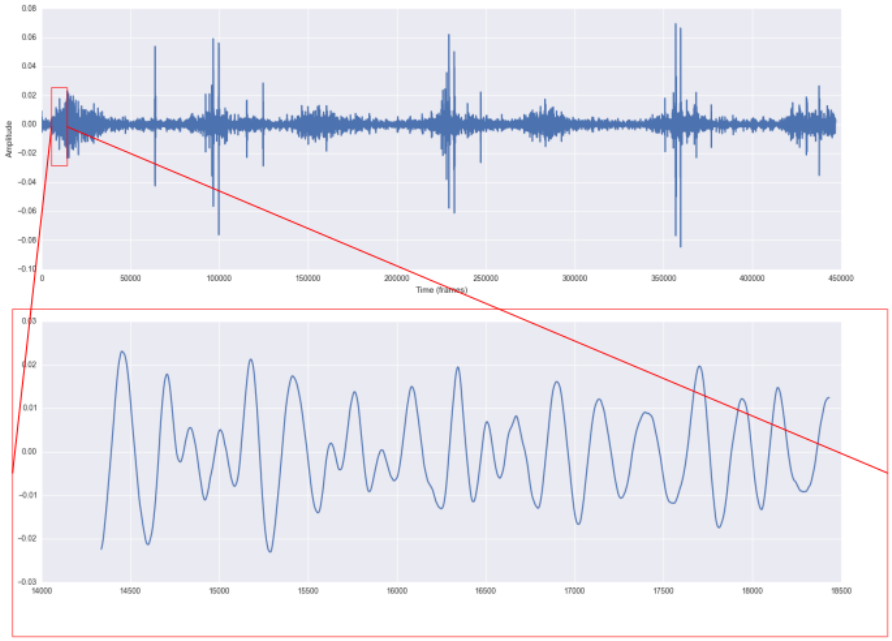


**Figure 2.2:** Dividing a signal into smaller chunks. Each window can either be examined individually to produce a training set, or sent to the classifier for classification.

First we have used a Butterworth bandpass to pass through frequencies between 150 Hz and 2400 Hz. The Butterworth bandpass is an electronic filter that rejects unwanted frequencies and has a uniform sensitivity for the wanted frequencies[Butterworth, 1930]. Frequencies above this range will most likely not be related to crackle sounds and frequencies below will most likely contain cardiovascular sounds that might interfere with the classification.

Further when investigating approaches to learn from the audio data, we found that using smaller chunks, of fixed size windows, produced a higher accuracy than looking at the signal as a whole. We believe that having less data per classification task will help to successfully find all crackles, and also their

location in time. So to achieve this chunking we divide the signal by sliding a window of 4096-points across the signal, with a 50% overlapping to the previous window (Figure 2.2). This is to avoid losing potential crackles between two windows. The last window of a signal is zero-padded if it is too short to fit into a 4096-point window. When we do this modification to the signal, the relation between windows will be unknown to the classifier, and each window will be treated as an isolated event. By extracting these windows and labelling them as either a window of normal breathing or crackle.

As the last step in preprocessing we extract features from each of these windows. Christopher M. Bishop states the goal of feature extraction in his book *Pattern Recognition and Machine Learning*: "*For most practical applications, the original input variables are typically preprocessed to transform them into some new space of variables where, it is hoped, the pattern recognition problem will be easier to solve*" [Bishop, 2006, p. 2] The three types of features we extract are *Wavelet Statistical Features*, *Short Time Fourier Transform Coefficients* and *Spectral Flux*. Although the aforementioned chunking of the signal can also be considered a part of feature extraction, the aforementioned chunks are further transformed before they are given to the classifier.

## 2.2 Features

When we extract features from the audio files, all windows are decimated by a factor of $M$, in our case $M = 4$. This means the windows go from 4096 samples down to 1024, by only keeping every $4^{th}$ sample and throwing away M-1 samples in between [Crochiere and Rabiner, 1981]. The reason for this is to further try to reduce the amount of data, while still keeping the features of interest intact.

One of the problems we have seen is that there is no one-fits-all type of feature. Some feature types, such as *Spectral Flux*, may be good at identifying the abnormal sound, but may also make the classifier very sensitive to additive noise. Since the information in the *Spectral Flux* feature is only concerned with rate of change of a signal. Other feature types have shown to make the classifier less sensitive to additive noise, but also very insensitive to the actual abnormal sounds, such as the *Statistical Wavelet Coefficients*. They have more detailed information about different sub-bands of the spectrum, but might miss more subtle crackles with less prominent characteristics (the ones that are harder to detect both by computers and humans). We also observed that the tubing of the stethoscope produced additive noise sounds that mimic the signal of a crackle very closely, and therefore they are often mistaken for crackles. Because of this we have chosen to include three different types of features in order for

our pipeline to have a balance between.

The original signal is kept as well, but not used as a feature. The raw data is only used to construct a waveform of the original signal after classification.

### 2.2.1 Short Time Fourier Transform

*The Fourier Transform* is often a central part of signal processing and especially in audio. While a *Fourier Transform* will give a summary of the frequencies in the audio, but it discards the time domain in order to do so. One of the most important uses of Fourier Transforms is the ability to shift time-dependent data into the frequency domain. So for a signal $S_{Time}^{Amplitude}$ the Fourier Transform translates this data into $\hat{S}_{Frequency(Hz)}^{Amplitude}$, which gives an overview of the spectral contents of the signal. Now the *Short Time Fourier Transform* uses a series of consecutive Fourier Transforms to compute a *Spectrogram* that retains the time scale.

Since the Crackle sound is an explosive sound which keeps its power through frequencies up to 1000.0 Hz, the fourier calculated spectrogram will give some indication of crackles that might be located in the audio.

### 2.2.2 Daubechies Discrete Wavelet Decomposition

*Discrete Wavelet Decomposition* is an alternative to the Fourier Transform, but it automatically retains the time scale. It also uses a less naive approach to different frequencies where the higher frequencies are sampled at a higher rate than lower frequencies. A multilevel discrete wavelet decomposition means that a signal will be decomposed several times, down-sampling and band passing the signal at each level creating a binary tree of coefficients, also known as a filter bank. Each of these levels of decomposition produces a set of *Detail Coefficients*, while the last decomposition is named the *Approximation Coefficients*.

After performing the DWT on a signal of a window, we summarize all of the coefficients by calculating the *Maximum frequency* and *Average Power* of the absolute values in each individual coefficient set.

### 2.2.3 Spectral Flux

*Spectral Flux* is a measure of how much a signal changes over time with regards to the power spectrum. Often used in *Music Information Retrieval* it is

an indication of the *timbre* of a signal. In this project the spectral flux is an indicator of how much the signal fills each window. Finding the *Spectral Flux* coefficients of a given signal usually involves calculating the Euclidian distance between two normalized spectra. One of the keywords here is the normalization step. All data has to be normalized, otherwise a window containing weaker amplitudes of normal breathing might have a very different coefficient than windows containing stronger amplitudes, e.g. a higher loudness in the sound. The result of this analysis means that a high coefficient indicates that the signal is changing a lot over the course of time in the window. A small coefficient would mean there are little change over time. This translates to that all windows that has a high peak, or a high spike in the amplitude would have a low coefficient, while a signal that contains few spikes would have a high coefficient.

We calculated the spectral flux by computing the change between windows of 256 samples with each of the 4096 sample window. This gives a 3-dimensional vector for each window, which is significantly lower than the other features. We have chosen to keep this feature in the feature set, but it will most probably be replaced in the future.

## 2.3 Classifiers

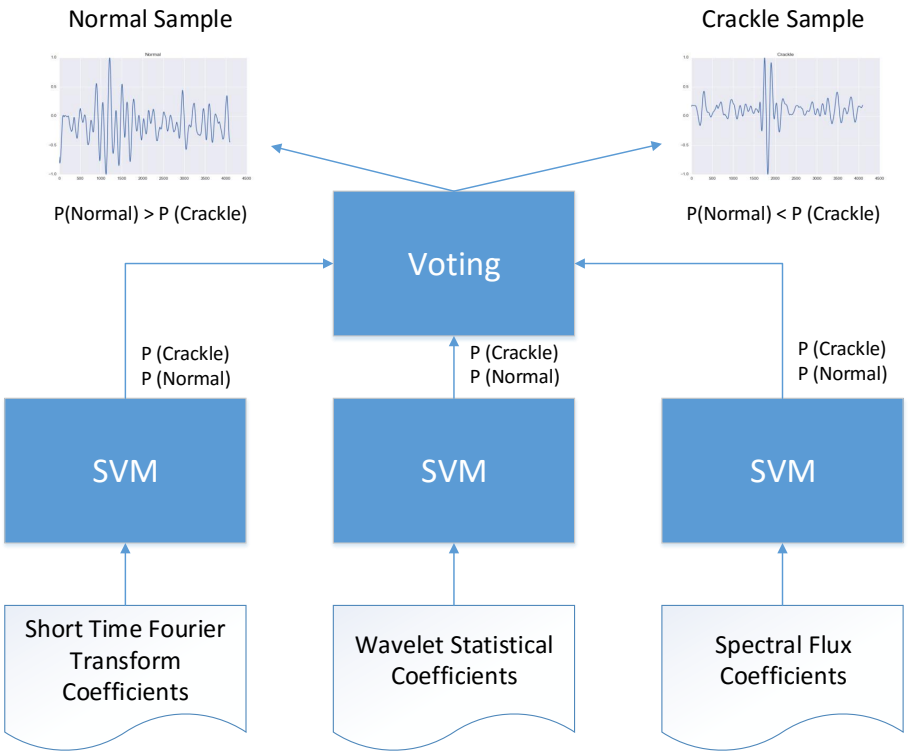



**Figure 2.3:** The classifiers hierarchy and voting scheme for determining the class of a given window.

For this project, we chose to use SVMs[1] as classifiers. One of the reasons for this is that SVMs are very flexible in terms of the decision boundary because of the kernel type. Using a non-linear kernel, the SVM is able to represent non-linear relations. Since the data is non-stationary, and none of the features selected works exclusively well for all samples, the SVMs are structured into a hierarchy inspired *Serbes et al.* [Serbes et al., 2013]. These SVMs will try to classify one window at a time, where each of the classifiers outputs a probability for each class using Platt Scaling[Platt, 1999]. The Platt Scaling formula solves the problem that SVMs does not output any degree of certainty about an answer. Platt scaling works by fitting a logistic regression model to the classifiers scores and the formula is given by:

1. Support Vector Machine

$$P(y = 1|x) = \frac{1}{1+exp(Af(x)+B)'}$$

Then using a simple voting scheme the result of the classification will be two probabilities for each class:

$$P(Crackle) = \sum_{i=0}^{n} P_i(Crackle) * W \qquad P(Normal) = \sum_{i=0}^{n} P_i(Normal) * W$$

Where $n$ is the number of classifiers and $W$ is the weight, the weight is just assigned to be $\frac{1}{n}$ and $i$ is the $i^{th}$ classifier. The class that has the highest probability will be the final class of the window. This can be seen in Figure 2.3.

## 2.4 Parameter Tuning

Support Vector Machines are dependent on several parameters, and for the SVM to perform well in classification, we need to chose optimal parameters for our classification problem. We can either choose these parameters manually, or search for the optimal combination using *Grid Search*. When using a *Grid Search*, we define a set of test parameters, and then systematically build a model using each of these parameters, and cross validating the models score. If the new model has a better accuracy than our previous try, we chose the new models parameters as the optimal ones, and if not we discard the new model. The advantage of SVMs is that the kernel is also interchangeable, which means that if any of the features are not linearly separable (Figure 2.5), we can just choose a non-linear kernel for that feature type. The wavelet decomposition features have 14 dimensions per vector (each window being a single vector), while the short time fourier transform coefficients have 2145 dimensions. This might be a case of the *Curse of Dimensionality*, where we may need to further reduce the dimensions of the latter vector.

The two first Principal Components of the Wavelet Decomposition Features (Figure 2.4) are almost linearly separable in a 2-dimensional space. Comparing to the two first Principal Components of the Short Time Fourier Transform features (Figure 2.5), are not separable in 2 dimensions. It is important to note that even though features are not linearly separable in 2 dimensions, they may be in higher dimensions, but it is not possible to show this separation on a 2 dimensional plane.

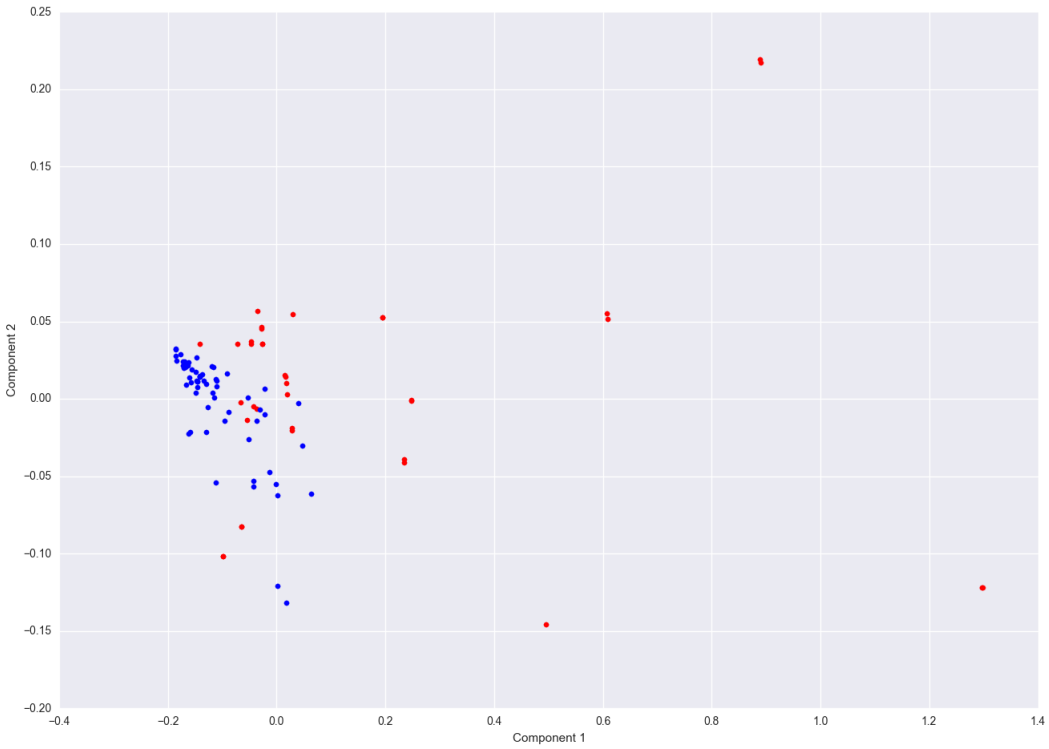



**Figure 2.4:** Principal 2-component analysis of Wavelet Decomposition Features. Red points are crackle features, while blue are normal breathing features.

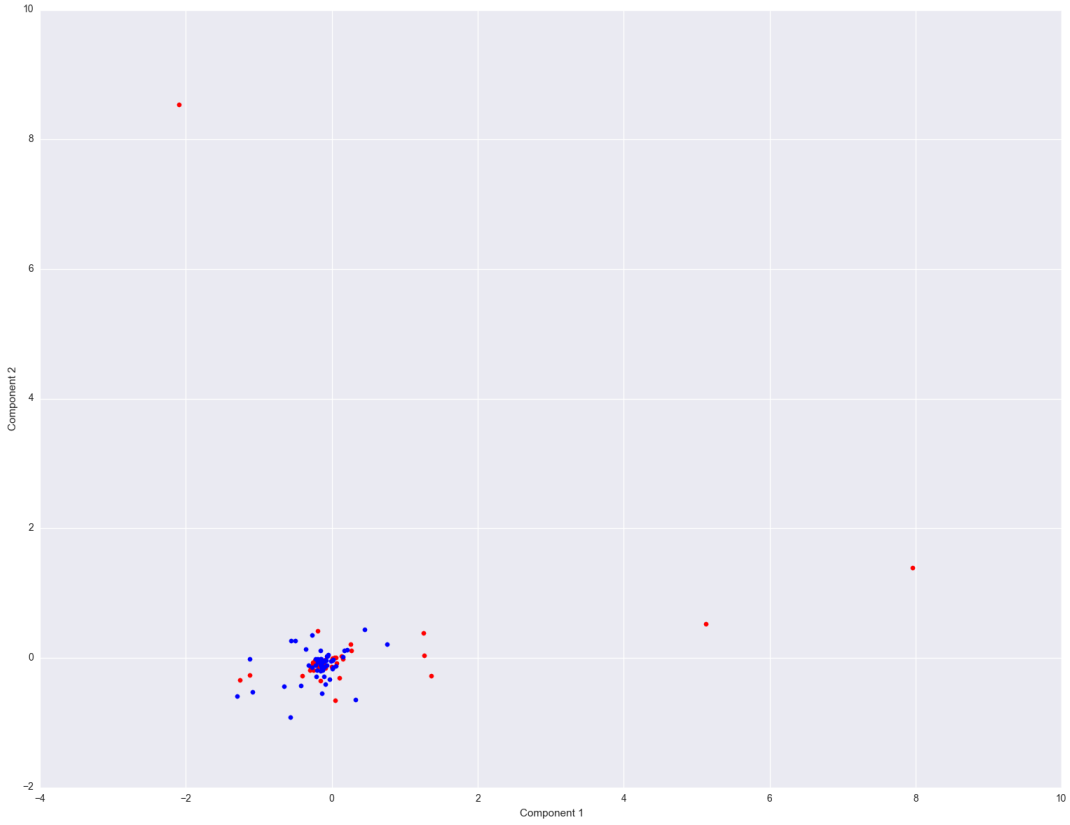



**Figure 2.5:** Principal 2-component analysis of Short Time Fourier Transform Features. Red points are crackle features, while blue are normal breathing features. This feature type is inseparable in 2 dimensions.

# 3

# Implementation

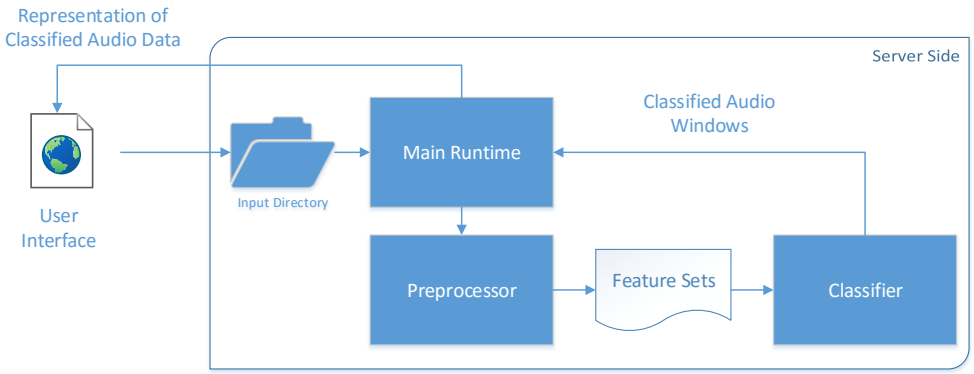



**Figure 3.1:** The overall Architecture of the solution

The general architecture of the current pipeline is given above in Figure 3.1 have three main components: An input unit, the core program and an output unit. The input unit can be just a folder, a web interface or a full desktop client that supplies audio files to the core main runtime, in our approach the Input Unit will take the form of a web interface that supplies audio files to a folder that can be reached by the core main runtime.

The core program consist of three specific units: *Main Runtime*, a *Preprocessor* and a *Classifier*. The *Main Runtimes* purpose is to handle input and output

for all other units, with the exception of the training set, which has its own utility module for storing and loading audio waveforms as binary data. It is also responsible for formatting data that can be understood by the different units. We have also tasked it with annotating the *Raw Audio Waveform* that is supplied by the *Classifier*. This correspond to the summary/presentation step in Figure 2.1. This is also the reason that the raw audio data is included in the feature set.

The pipeline is implemented in Python 2.7 using *Scikit-learn* for machine learning functionality, *Numpy* for array and matrix operations, *Scipy.signal* for interaction with and filtering of audio files, *librosa* and *pywt* for the *Short Time Fourier Transform* and *Wavelet Decomposition*.

Our experiments were run on a machine with the following specs:

Intel Core i5-4570S Quad Core CPU @ 2.90 GHz
6,00 GB RAM @ 1600 MHz
Gigabyte G1.Sniper B5 Motherboard
Windows 10 Pro, 64-bit Operating System

## 3.1 Training Set

As we have mentioned earlier, the preprocessor is able to create a training set by manual interaction. Each of the chunks collected from a signal can be saved to respective directories dependent on what class the user inputs. All chunks are then saved to either directory as binary files for optimal performance with Numpy. We have generated the current training set consisting of 98 samples, of which 37 are crackles and 61 is normal. It is a very small training set in comparison to how many audio files are available, and this will need to be expanded as the repository of lung sounds grows in order to utilize it to its full potential. In our experiments using grid search for parameter tuning, the optimal parameters depends a lot on the fold of the training data. This will be further discussed in the evaluation section.

## 3.2 Interface

Our intention is that the analysis pipeline we have implemented will run as a web server available within the university network. The web server uses a Python Flask back-end and runs within a Windows Server 2012 Virtual Machine. Figure 3.2 is the current web interface, we have used Bootstrap for visual

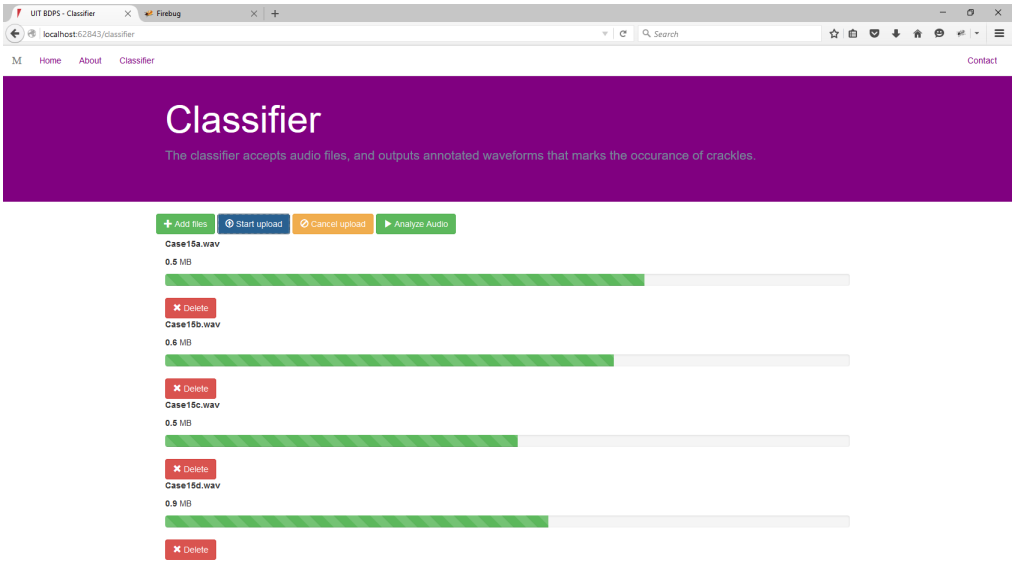



**Figure 3.2:** Web interface to the analysis pipeline, a user can upload audio files to the server for analysis.

components and Dropzone.js to handle file upload to the server. We have not yet connected the server interface to the actual implementation of the analysis pipeline, so the server is currently not able to analyse audio files.

We plan to have users upload audio files to the server for analysis and present the results to the user in two different stages. The first stage is a summary of whether there have been any abnormal sounds detected or not and second a detailed overview of each audio files with annotated waveforms.

# 4

# Evaluation

When evaluating our pipeline, we look at the results from our classifiers. We evaluated our classifiers with regards to: *Precision, Recall and F1-score*. *Precision* is the number of *Predicted Positives* that was actually *True Positives*, or the number of correct hits compared to the number of false hits. *Recall* is the number of *Predicted Positives* that was actually in the total amount of *True Positives*, or how many that are true did the classifiers find. The F1-Score is a weighted combination of the *Precision* and *Recall*. The accuracy measurement is a combination of precision and recall, much in the way of the F1-score, but it is not weighted. The accuracy will give an indication of how well the classifiers are classifying every sample correctly and the standard deviation of accuracy between training cycles shows that the classification accuracy is not dependent on certain samples of training data.

## 4.1 Methodology

When training the classifier we can validate the accuracy using a Stratified K-fold. We divide the training set into three equally sized sets (and with the same distribution of classes in each). We have used this to validate the performance of our classifier, and running a training - validation cycle 100 times. For our experiments we have used only the *Short Time Fourier Transform Coefficients* and the *Wavelet Statistical Coefficients* as our features, this means that there are two classifiers working in the hierarchy. The reason we chose to omit the

*Spectral Flux Coefficients* is that through earlier experiments they did not impact the results of our classification.

To create a basis for comparison, we have run our experiments using three different classifiers, *Support Vector Machines* (with hierarchical voting using one SVM for the Wavelet and Short Time Fourier Transform features), *K-Nearest Neighbor* and *Gaussian Naive Bayes Classifier*. Each training cycle consisted of fitting the classifiers using the initial parameters. We then using a grid search to find the optimal parameters, except in the case of the *Gaussian Naive Bayes Classifier* since it is based on probability with no additional parameters. After the optimal parameters for the fold were found, we used the hold-out set of samples from the Stratified K-Fold to verify the performance with regards to the aforementioned criteria.

| **SVM** | Precision | Recall | F1-score |
|---|---|---|---|
| Crackle | 0.90 | 0.73 | 0.81 |
| Normal | 0.85 | 0.95 | 0.90 |
| Avg/Total | **0.87** | **0.87** | **0.86** |

**Table 4.1:** SVM classifiers Precision, Recall and F1-score after running 100 training and cross-validation cycles.

| **KNN** | Precision | Recall | F1-score |
|---|---|---|---|
| Crackle | 0.92 | 0.25 | 0.39 |
| Normal | 0.68 | 0.99 | 0.80 |
| Avg/Total | **0.77** | **0.70** | **0.65** |

**Table 4.2:** K-Nearest Neighbors classifiers Precision, Recall and F1-score after running 100 training and cross-validation cycles.

| **Naive Bayes** | Precision | Recall | F1-score |
|---|---|---|---|
| Crackle | 0.57 | 0.10 | 0.18 |
| Normal | 0.63 | 0.95 | 0.76 |
| Avg/Total | **0.61** | **0.63** | **0.54** |

**Table 4.3:** Gaussian Naive Bayes classifiers Precision, Recall and F1-score after running 100 training and cross-validation cycles.

From the results above we can see that the SVM classifier performs best in comparison with KNN and Bayes classifiers. All of the classifiers are arranged in a hierarchy as stated in the methods, using our two most indicating features (Short Time Fourier Transform and Wavelet Decomposed Statistical Features).

An important observation that we made in our experiments is that the standard deviation in accuracy is fairly low. The accuracy only varies with about 6%

| | Accuracy |
|---|---|
| SVM | 0.87 ±0.06 |
| KNN | 0.70 ±0.07 |
| Bayes | 0.63 ±0.06 |

**Table 4.4:** Average accuracy and standard deviation of the different classifiers.

throughout the 100 training cycles. This is a good indication that there is low variation in results with each cycle, we believe that this is a positive indication that there are common distinctions between crackle signals and normal signals.

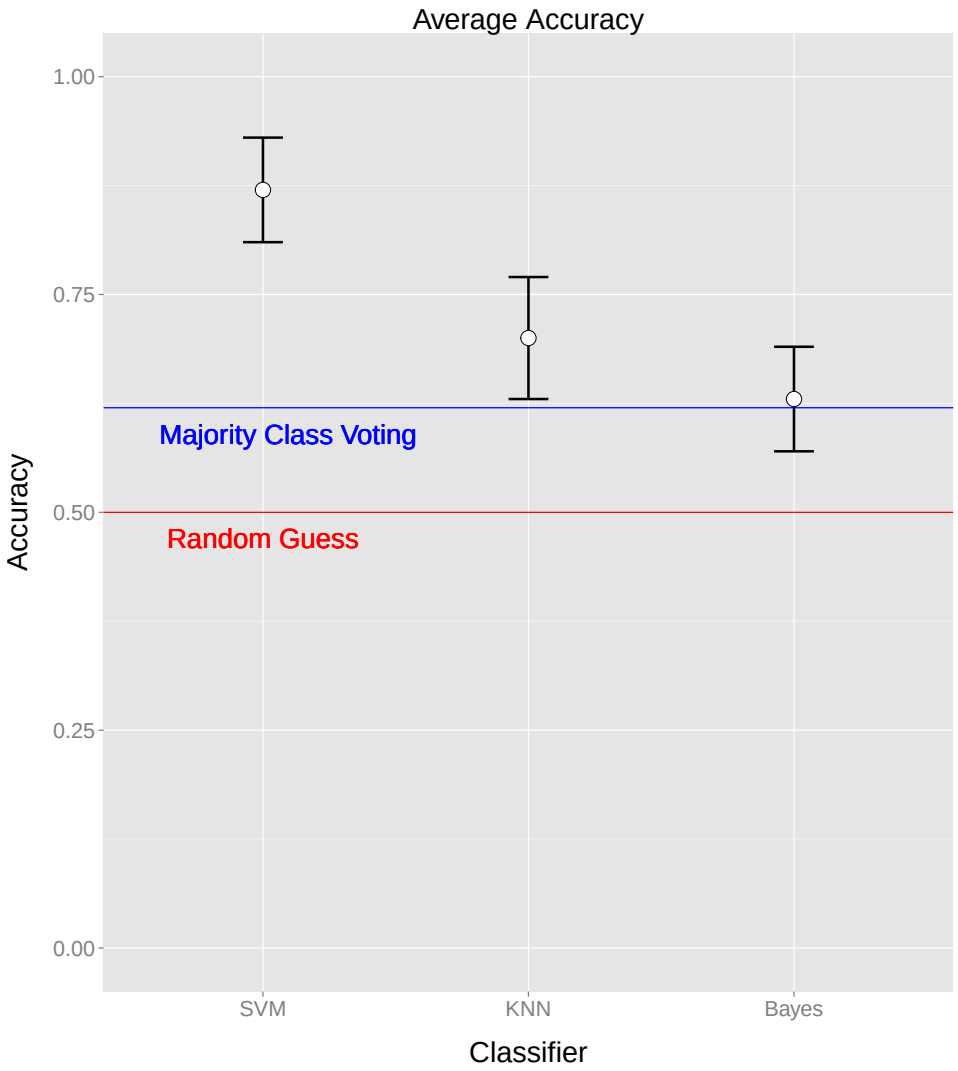



**Figure 4.1:** A plot of the average accuracy of the different classifiers.

However in our experiments, the optimal parameters (found in our grid search) for our classifiers changed quite frequently for each of the training cycles. We believe that this is due to a low amount of training data, only 98 samples in total (37 crackle and 61 normal). This means that even though the accuracy is good, there are too few datapoints to find a common best set of parameters for all training samples. Our SVM parameters changed between using a linear kernel with a high cost parameter, C, and using a RBF[1] kernel with a low cost parameter. We believe that increasing the amount of training data might also

1. Radial Basis Function

help to eliminate some of the false positives that occur in our normal audio files. As we discussed there are additive noise in a number of these normal audio files that are very similar to the crackle signals. These are a source of false positives in our experiments, but it is something that we cannot discard as isolated events, since ambient noise is a reality of clinical settings.

## 4.2 Experiments on Real Data

In this section we will go through the common cases that we have seen in our experiments when applying our pipeline to real data. All of these audio files have already been classified by the investigators from the *Inter-observer study*.

In Figure 4.2 we see a typical crackle audio file, with the crackles annotated in red. For most of these types of samples, the pipeline analysis are able to find almost all crackles.

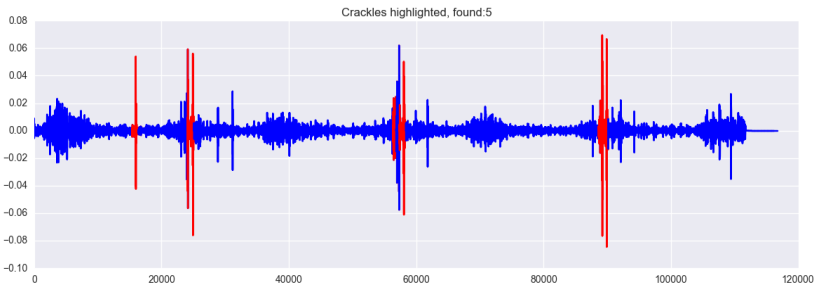



**Figure 4.2:** A typical crackle audio file.

In some more extreme cases we see examples of what is referred to as *Fine Crackles*. These are typically shorter and have less power than the standard *Coarse Crackles*, our classifier are able to find these crackles as well, but it does not distinguish between *Fine* and *Coarse* types of crackles.

We consider these findings to be promising, since these are audio files recorded in what would be a typical clinical setting. But we have also found that there are many problematic samples as well. There are many cases of ambient noises that imitates crackles and produces false positives (Figure 4.4). When listening to these files we can hear that they are in fact close to crackles as well. This is of course also a problem for Medical Staff when it comes to listening to the lungs, so if we can improve our solution to discriminate between these ambient noises and true crackles, it can prove very useful in both practice and training of said Staff.

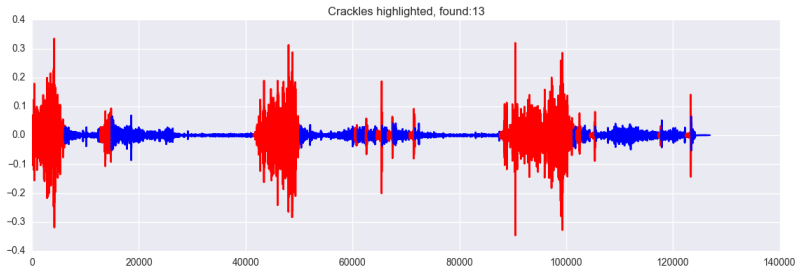



**Figure 4.3:** A rarer case of crackle audio files. This particular one is recorded from a patient diagnosed with pulmonary fibrosis.

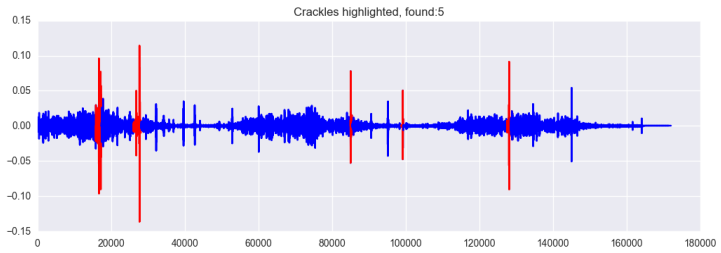



**Figure 4.4:** A problematic audio file which contains ambient noise. These spikes are labelled falsely as crackles.

This shows of course one of the more problematic ones, but there are some normal audio files which have significantly fewer false positives, often between 1 to 6 false crackles. We have come up with an idea for a post-processing step to eliminate these looking at how often a crackle occurs in the audio file, and also if there is some sort of cyclic pattern to their occurrence if they do appear more than once. A single crackle, even if it is a true positive, does not necessarily have any significance (Figure 4.5).

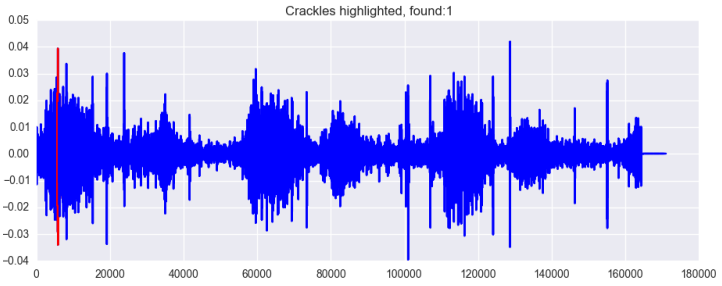



**Figure 4.5:** A normal audio file containing a single false positive, this might be eliminated with a post-processing step.

We experimented with real data (full sized audio files) that have been classified by the experts at the *Inter-observer study*. The main problem seems to be varying results, and that each experiment is dependent on the parameters found with

each SVM. Due to this we found that our rate of false positives for all normal audio files are up to 1,3 false positive crackles per audio file, while the rate of true positives are about 5,3 crackles per audio file. We believe this is due to the small amount of training data, which will be a focus for our future work where we need to expand our training set. As we discussed earlier, these false positives can also be eliminated in post-processing by having a certain number of crackles that must appear in an audio file before reporting it as a positive finding.

## 4.3 Comparison to Related Work

In the paper *Automatic Crackle Detection Algorithm Based on Fractal Dimension and Box Filtering*[Pinho et al., 2015] a team of researchers from the University of Aveiro, Portugal suggest a technique of automatic crackle detection. The methods they use can be summerized in three steps. 1) Extract the window of interest that may contain a crackle, they do this by using fractal dimension and box filtering teqniques. 2) Verify the signal if it adheres to the CORSA[2] established criteria. and 3) Characterization of the Crackle's parameters. Their methods of detecting crackles are based on previous work done in the field of examining the nature of Crackles. One of the criteria they use to determine if a window contains a Crackle is the initial deflection width (IDW) and the largest deflection width (LDW).

Although these Crackle parameters seems to have worked out well in their tests, it does not exclude the possibility that there are signals which mimic these parameters in normal breathing. Which they do not mention in their work, and might have an impact on the accuracy of their developed software. Their tests were performed in 24 audio files and comparing to a multi-annotator gold standard, and seems to perform very good, on these 24 cases. The advantage that our project has over this study is that the number of sample audio files vastly outnumber their test cases (at the end of the study up towards 18 000 audio samples). Which means problems that can appear in a normal clinical setting, is more likely to be discovered at a testing stage.

They also highlight an important problem with most developed techniques today:
"*Despite the high values of sensitivity and specificity associated with these techniques, limited testing have been performed with respiratory sound files recorded in clinical settings and validated against a multi-annotator gold standard*"

2. need citation: Basic techniques for respiratory sound analysis. Eur. Respir. Rev. 2000;10:625–35.

Most of the experiments performed in automatic crackle detection, either have a low amount of audio samples, or audio samples collected from a few patients. These sample sets might not be representative of the general populus.

The paper *Pulmonary crackle detection using time–frequency and time–scale analysis*[Serbes et al., 2013] have taken a very similar approach to our own project, and also inspired our approach. The study is based on comparing different classifiers to each other and using different techniques for feature extraction, comparing the results to find the optimal method. The two different feature extraction methods are *Time Scale* and *Time Frequency*, where the *Time Scale* analysis consists of a 64 scale wavelet transform using three different wavelet families such as Morlet, Paul and Mexican Hat. For the *Time Frequency* analysis they use a 64 point Fourier transform using a number of different window types such as Gaussian, Blackman, Hanning, Hamming etc. These two types of analysis is then integrated over time and frequency to obtain the different behaviours of the signal with regards to different domains. The features extracted together with the original signal is then fed to each their classifier which outputs a probability estimate for each feature and using simple voting to determine the class of a given signal.

They have a sample set of 6000 audio signals, 3000 containing normal breathing and 3000 containing a crackle signal. Each sample consist of a 512-point window at a sampling frequency of 9,6 kHz, and have been classified by physicians. Now this may be the biggest difference from our project as they do not have full audio files which they divide up, but rather a pre-divided set of signals that have been classified by physicians.

## 4.4 Discussion

For future work we plan to evaluate additional types of classifiers. SVMs is often a good approach to two class problems (both linear and non-linear) but historically, Hidden Markov Models are used more frequently in speech recognition. Since speech recognition is closely related to the current project, it would be natural to also test the solution using this classifier.[Gales and Young, 2007] One of the advantages of our implementation is that the classifier type is completely interchangeable, which will make it easy to test with different classifiers in the future.

We also want to expand our training set, trying to utilize as many of the audio files in the *Inter-observer study* as possible. We also need to focus on

As we have seen in our test results on real data, we intend to improve the ability of our classifier to distinguish false positives that occurs in many of our normal samples. We need to develop a method to distinguish the crackle-like noises from the actual crackles that we have mentioned earlier. One solution we have seen is the power in the spectrum, crackles usually keep their power all the way up to 1000.0 Hz, while the crackle-like noises loses some of this power up towards the same frequencies. It is a marginal difference, so we need design experiments to determine whether or not this characteristic can be used. Increasing our training set might also solve this problem, as the classifiers have a good cross validated accuracy, but have conflicting results when testing on actual data. Our classifiers are able to find crackles in the crackle samples, but they also find crackles in the normal samples.

Another feature that is often used in speech recognition is the *Mel Frequency Cepstral Coefficients* (MFCC). The MFCC is a transformation that maps frequencies to the Mel-scale, which is a non-linear representation of frequency, that closely approximates how humans perceive frequency or pitch.

# 5

# Conclusion

In this report we have presented a problem of classification between two classes of audio, recorded from lungs by means of a stethoscope. We found that a signal processing and machine learning approach to this classification is feasible, and we have developed a pipeline that is capable of this analysis. Overall the pipeline we have developed seems to solve at least the simplest classification problems, being able to distinguish the most obvious crackles from normal audio. However, there are some problems which naturally occurs in a clinical setting, that we will need to take into consideration. We believe that since our pipeline has been developed using data recorded in a clinical setting, we had to handle some of the problems that may occur in such a setting. We have found a challenge with false positives, but we believe that we can eliminate these in future work. Our approach provides the basis for our future work, as it is an efficient way of finding crackle signals. We believe this tool will be useful for general practitioners and the experts at the *Inter-observer study* a tool in their analysis as well. In addition, we believe our approach is not limited to pulmnoray crackles, but other types of bodily sounds as well such as jaw clicking, and abnormal clicking sounds from knee joints.

# Bibliography

[Bishop, 2006] Bishop, C. M. (2006). Pattern recognition and machine learning.

[Butterworth, 1930] Butterworth, S. (1930). On the theory of filter amplifiers. http://www.researchgate.net/file.PostFileLoader.html?id=560fd0695f7f71adfa8b4588&assetKey=AS%3A280454457511950%401443876961677. [Online, Last Acessed: 07.12.15].

[Crochiere and Rabiner, 1981] Crochiere, R. E. and Rabiner, L. R. (1981). Interpolation and decimation of digital signals - a tutorial review. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1456237. [Online, Last Accessed: 07.12.15].

[Deng and Li, 2013] Deng, L. and Li, X. (2013). Machine learning paradigms for speech recognition: An overview. http://research.microsoft.com/pubs/189008/tasl-deng-2244083-x_2.pdf. [Online, Last Accessed: 18.11.15].

[Domingos, 2012] Domingos, P. (2012). A few useful things to know about machine learning. http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf. [Online, Last Accessed: 04.12.15].

[Gales and Young, 2007] Gales, M. and Young, S. (2007). The application of hidden markov models in speech recognition. http://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf. [Online, Last Accessed: 25.11.15].

[Göğüş et al., 2015] Göğüş, F. Z., Karlık, B., and Harman, G. (2015). Classification of asthmatic breath sounds by using wavelet transforms and neural networks. http://www.ijsps.com/index.php?m=content&c=index&a=show&catid=36&id=125. [Online, Last Accessed: 15.10.15].

[Pinho et al., 2015] Pinho, C. M. R., Oliveira, A. L., Jácome, C., Rodrigues, J. M., and Marques, A. (2015). Automatic crackle detection algorithm based on fractal dimension and box filtering. http://www.researchgate.net/publication/281857717_Automatic_Crackle_

Detection_Algorithm_Based_on_Fractal_Dimension_and_Box_Filtering. [Online, Last Accessed: 01.10.15].

[Platt, 1999] Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. http://www.researchgate.net/publication/2594015_Probabilistic_Outputs_for_Support_Vector_Machines_and_Comparisons_to_Regularized_Likelihood_Methods. [Online, Last Accessed: 04.11.15].

[Serbes et al., 2013] Serbes, G., Sakar, C. O., Kahya, Y. P., and Aydin, N. (2013). Pulmonary crackle detection using time–frequency and time–scale analysis. http://www.sciencedirect.com/science/article/pii/S105120042003089. [Online, Last Accessed: 26.08.15].

[UiT, ] UiT. Tromsøundersøkelsen. https://uit.no/forskning/forskningsgrupper/gruppe?p_document_id=367276. [Online, Last Accessed: 24.11.15].

[van den Oord et al., 2013] van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc.

[Zakeri et al., 2015] Zakeri, V., Arzanpour, S., and Chehroudi, B. (2015). Discrimination of tooth layers and dental restorative materials using cutting sounds.