UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

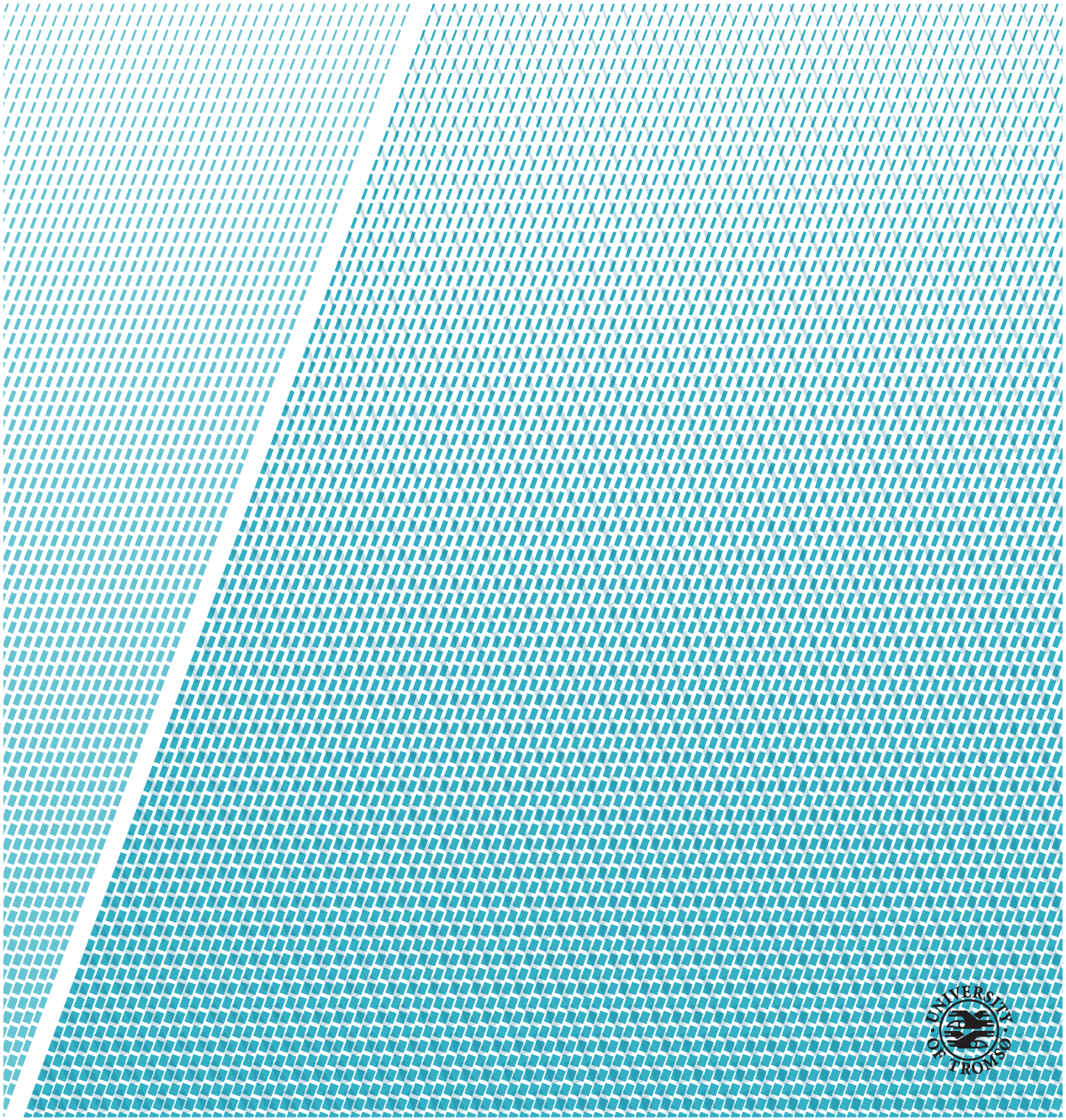
Faculty of Science and Technology
Department of Computer Science

Automated Lung Sound Analysis

—

Morten Grønnesby

INF-3981, Master Thesis in Computer Science, Spring 2016



Abstract

Lungs sounds has been used as a diagnostic tool for centuries. The usefulness of listening to lung sounds, or pulmonary auscultation, as a definite diagnostic method has been diminished by advances in medical imaging such as chest X-Ray, but these advanced methods also bring a higher monetary and time cost. In addition, when the severity of pulmonary conditions changes, audible symptoms change immediately, while x-ray imaging does not show the same immediate change. The stethoscope is still used as a screening method and has great potential for use in continuous monitoring, as it is a simple, non-invasive, and low cost method. Therefore, lung sounds are still important today, and there is a need for better training tools, and automatic analysis methods that can be integrated with stethoscopes to advance the technology of one of the most common medical tools today.

As a part of Tromsøundersøkelsen, researchers are now recording lung sounds to create a gold standard database of lung sounds and categorizing them based on occurrences of abnormal sounds. They are investigating the validity of pulmonary auscultation as a diagnostic method.

Earlier approaches have achieved good results in this field, but have lacked a large dataset and gold standard to validate performance in a general setting of clinical data. We present our approach to automatically analyze lung sounds, and classify abnormal sounds found in audio files of recorded breathing. We employed signal processing and machine learning techniques and implemented an analysis pipeline to perform the classification. We achieved a cross-validated F1-score of 83.5% using a Support Vector Machine performing classification on window excerpts containing *Crackles* from recordings of breathing. We also did preliminary evaluation the classification for *Wheezes*, and found a F1-score of 64.6%.

With our pipeline we have also implemented a GUI for a web application that we can deploy as a working prototype. We believe that with this approach we have created a basis for a core technology, that can be integrated with mobile platforms to serve as a home monitoring device, training tool or medical equipment.

Acknowledgements

I would like to thank my supervisor *Lars Ailo Bongo*, I am left with great respect and awe for the amount of dedication and expertise you have in your field. Thanks to my co-supervisors *Hasse Melbye*, *Einar Holsbø* and *Robert Jenssen* for helping me with suggestions and advice on my thesis. Thanks to *Michael Kampfmeyer* for helping me with machine learning problems in your spare time, and *Bjørn Fjukstad* (and *Einar Holsbø*) for welcoming me in their office space and lab.

Also thanks to my classmates *Eirik Mortensen*, *Jarl Fagerli* and *Ruben Mæland* for enduring this endeavour with me, and for the community we have had throughout the years.

A special thanks goes to my family at home in Trondheim, *Annlaug Grønnesby*, *Ole Anders Grønnesby* and *Lise Grønnesby* for their encouragement and support in my studies, as well as my best friend *Espen Schmitz*.

Contents

List of Figures

List of Tables

1

Introduction

Physicians routinely listen to lung sounds through stethoscopes during general examinations or when patients indicate respiratory distress. Such lung auscultations are an important method for physicians in decisions on treatment and referral for X-ray. However, auscultation is a subjective method and improper treatment and referrals accumulate an increased time and monetary cost. Training physicians is a challenging task because of varying perception of sound and lack of common terminology, though the latter have come more into focus for pulmonary experts. As a consequence of these challenges, better tools for training are required and a gold standard of abnormal lung sounds is greatly needed.

Training physicians using such tools, would help them to more accurately diagnose and decide a course of treatment and referral. A better set of tools for detecting abnormal lung sounds could also be used for self-monitoring by patients during treatment.

In this thesis we present our approach to *Automatically analyzing lung sounds* to detect abnormal sounds, and our tools that could aid physicians in training themselves to recognize these abnormal lung sounds.

1.1 Background

Here we describe the data extraction methods, and data classification methods that the doctors use in order to achieve a gold standard of categorized lung sounds. Even though there are typically additional patient information, our domain in this thesis will be audio.

1.1.1 The Tromsø Study

The Tromsø Study, or Tromsøundersøkelsen in Norwegian, is a repeated epidemiological study that has been conducted periodically since 1974, and has seen in total 40 051 different people participating. [Jacobsen et al., 2012] The original aim of the study was to discover reasons behind and combat high mortality due to cardiovascular diseases in Norway. Later the study has extended its scope to include for example respiratory diseases, from which the new gold standard dataset of lung sounds is being built. The Tromsø Study takes place each 5-6 years, currently the 7th round.

In Tromsøundersøkelsen, prof. Hasse Melbye and MD. Juan Carlos Aviles Solis are collecting lung sounds from over 3000 participants. From each of the participants, the investigators collect a total of 6 recordings, sequentially recorded at 6 different places on the torso. When recording lung sounds, the researchers use a stethoscope with a fitted microphone in the tubing. The 6 recordings are taken from two locations on the upper front of the torso, two on the upper back, and two on the lower back of the torso. All files are captured in Wave (.wav) format at 44,100 Hz sampling rate.

This accumulates to a repository of 18 000 individual recordings of lungs, all taken within a clinical setting and with the possibility to link with other data such as health records. Data is recorded in a setting that is close to what General Practitioners encounters when performing routine examinations, which is possibly a noisy environment.

1.1.2 Abnormal Lung Sounds

The goal of the lung sound study is to investigate how accurate lung sounds are as biomarkers for diagnosing, monitoring and treating lung diseases. Lung auscultation is one of the simplest non-invasive screening for lung disease, or other diseases that afflict lungs as part of the symptoms such as congestive heart failure. Therefore, it is a quick and cheap way of screening patients. However, audible symptoms are prone to subjectivity of the investigator. Therefore, investigating the validity of the stethoscope (and lung auscultation) as a

diagnostic method for lung disease is important, but also the creation of tools that can aid in both training and screening using auscultation. The study focus on the two most common abnormal lung sounds, *Crackles* and *Wheezes*, and creating a gold standard with expert classified abnormal sounds.

Crackles (also referred to as *Rales* or *Crepitations* in earlier research) are short explosive *clicking* or *crackling* sounds that occur due to opening of small airways, with a short duration often ranging between 5-40 ms. They can occur in most places of the lung, and can be present in one (unilateral crackles) or both lungs (bilateral crackles) simultaneously. Basal crackles are used to describe crackles that originate from the bottom of the lungs.

Crackles can be divided into several main types depending on the characteristic of the sound; coarse, medium and fine, as well as wet or dry. Fine crackles are often soft, high pitched and short, while coarse are louder, lower pitched, and last longer. Most commonly crackles can be heard during an inspiratory phase, and depending on the type, either late or early. Coarse crackles tend to occur in early inspiratory phase, while fine crackles occur in late inspiratory phase.

Crackles can also occur in healthy lungs, but a persistent presence of crackles indicate opening of small airways and small cavities (alveoli) in the lungs being collapsed by fluid, exudate, or lack of aeration during expiration. These symptoms often occur in patients with pneumonia, pulmonary fibrosis, acute bronchitis and other conditions. [Forgacs, 1978] Crackles are also very subtle sounds, so a microphone that is rubbing over cloth and chest hair, might actually produce similar sounds.

Wheezes are continuous musical sounds, which can last up to a whole inspiration or expiration cycle. They are usually caused by air being forced through small paths due to obstructions in airways, creating a *whistling* sound. Wheezes can be heard detected over the whole chest as well as the trachea, which have proven to be a good method of detecting wheezes in asthma patients. [Sanchez et al., 1993, Pasterkamp et al., 1997]

As with the crackles, wheezes can vary a lot from person to person, and the sound depends on cause, severity and auscultation method and location. Wheezes can be indications of respiratory conditions such as asthma attacks and different types of allergies that causes narrowing or obstruction of airways. Wheezing can also occur in healthy lungs when the airflow velocity increases during physical exercise.

Lung sounds are difficult to define in a general sense, because of its inherent link to anatomy and condition, which also makes training challenging. As Murphy

put it in his paper *Auscultation of the lung: past lessons, future possibilities* [Murphy, 1981] when talking about the lungs sonic signal:

> This signal can be seen to vary with recording site, flow rate, lung volume, body position, and various breathing manoeuvres. It is likely that the sound changes with growth, development, and age, as well as with minimal environmental insults. The signal is so complex and varies so much that it appears at times to be random or unpredictable. It is more likely, however, that the sonic signal reflects the underlying anatomy and pathophysiology.

1.1.3 Gold Standard

A gold standard in medical terminology, refers to a diagnostic test, which can be both with or without restrictions, that is the best available (most definite). While in Machine Learning, a gold standard usually refers to a manually annotated training set or test set. Gold standards are useful to evaluate if classification in Machine Learning is general enough in comparison to a base truth.

When classifying lung sounds for the gold standard, a team of three doctors categorize each audio file individually. Then if all agree, the file is categorized as the agreed class. If there is disagreement between the individual classification, the file is saved for an agreement meeting, where all doctors meet and discuss the possible category. If an agreement can be reached, the file is stored on record as the agreed class, otherwise, it is discarded. At the time of writing, the experts have manually classified about 2500 audio files.

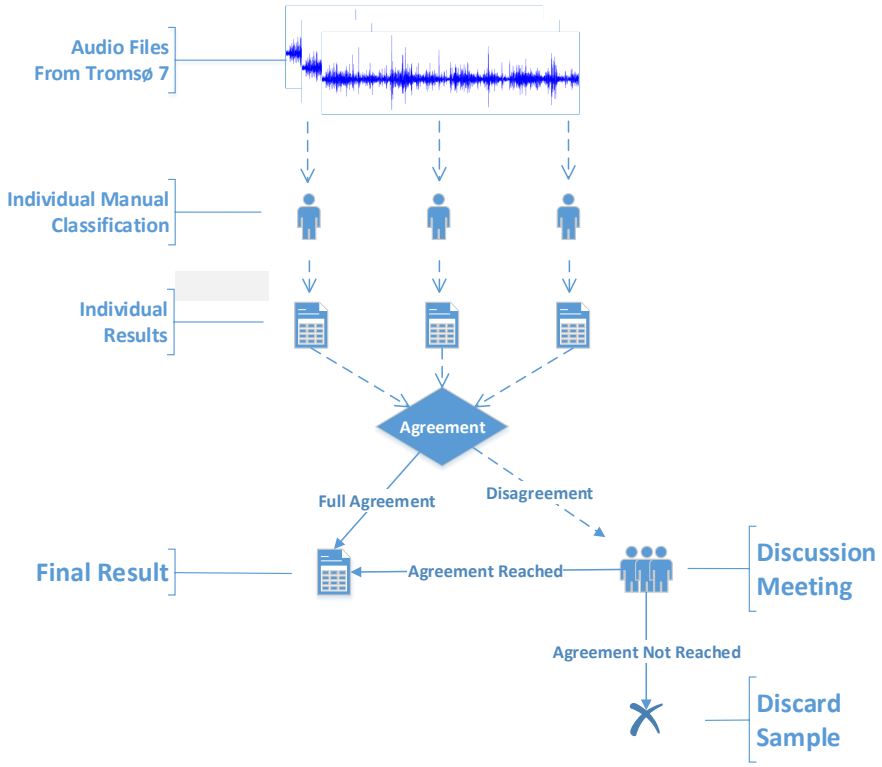


Figure 1.1: Doctors agreement protocol for lung sounds

Our goal is to develop an approach for automatic analysis of these sounds, using the gold standard. An automatic analysis system can summarize and utilize the knowledge this large repository holds. While medical tests often need a high amount of *Specificity* (True Negative Rate) and *Sensitivity* (True Positive Rate). For a test to be called a *Gold Standard Test*, it has to be the best available. [Kanchanaraksa, 2008] While our results are similar to other studies of the same nature, the access to a gold standard of about 18 000, this study is the first of its kind to encompass such a large scope. [Gurung et al., 2011]

1.1.4 Machine Learning

We believe that machine learning is the best choice for an automated analysis. Machine learning have become a standard in data analysis tasks, both in the cases of large amounts of data and non-trivial pattern recognition in complex data. It is also cost efficient, scalable (somewhat) and have the potential to achieve near expert-level precision in classification tasks. [Jordan and Mitchell, 2015]

There are three types of general machine learning approaches; supervised, unsupervised and reinforced. For a task where there are two clear classes (and the goal is to classify data), we use a supervised approach. In order to implement a supervised machine learning algorithm, we first need to generate a labelled training set. Labelled training sets are sets of data were we know which class they belong to. Using this knowledge it is possible to generate a model that represents the statistically significant features of the data.

When we have a set of data with labels for two or more classes, it is possible to *train* a model to either recognize all three classes, or train two models to recognize one positive and negative class (binary classification). We chose to use the latter, due to the difference in duration of the two abnormal sounds. Finding wheezes within a window of 90 ms would be very hard to achieve and finding crackles in 3 second windows would be equally difficult.

1.2 Approach

There are several challenges in creating and training a machine learning model for classification of abnormal lung sounds:

1. **A large amount of representative data** is required to develop and validate machine learning models. More data makes it more likely that the data is representative of the general case. And it reduces the risk of over-fitting models and it becomes more robust against outliers. In cases where data is scarce it is usual to generate synthetic samples. Though the disadvantage of synthetic data is that it may not be correct in regards to the real world, especially for audio, which is statistically non-stationary.

2. **Generate a labelled training set** to use with a supervised learning algorithm. The audio files that are used in this project are already labelled as either containing crackles or not, but we also want to find individual crackles in each file. Therefore, we need to extract individual crackles from audio files at a finer granularity, this reduces the amount of normal data and emphasizing abnormal samples.

3. **Preprocess the files to be classified** in the same way that the training data has been pre-processed. Reduce the size of each classification task and more accurately pinpoint locations of the abnormal lung sounds within a given audio file.

4. **Find and extract features** from the data. We need to find features that represents our data while reducing the number of dimensions that

our classifier has to consider. This is due to the curse of dimensionality, which entails that generalizing something correctly becomes exponentially harder as the dimensionality of the data (or feature set) increases. [Domingos, 2012]

5. **Select, tune and train a classifier using our training set** to create a model that is able to perform accurate and precise predictions on new data.

6. **Validate our model** using cross validation to measure specificity and sensitivity, and the expert classified gold standard from the Tromsø study, to evaluate classification accuracy on clinical data.

7. **Create a usable interface** to infer classes for unseen data, and provide a visual representation of the results to an end user.

To the extent of our knowledge, we have not seen any related work that is able to fulfil all these requirements. In related work we often see that lung sounds are a niche field, which is mostly of interest to medical researchers. Doctors today have also questioned the usefulness of the stethoscope in diagnosis, as Gupta explains in his article *The stethoscope: The iconic medical tool* [Gupta et al., 2016] In machine learning very few engineers and machine learning specialists are exposed to these kind of problems, and tend to favor the more popular problems such as image classification, speech recognition and recommender systems. Another reason that lung sounds have not been directly adopted in machine learning is that there has never been a good gold standard to train and evaluate classification. Other machine learning tasks draw from popular available databases such as ImageNet (images), MNIST (handwritten digits), NimStim (facial expressions). However, there are interest in the field as seen with the thesis *A framework for automated heart and lung sound analysis using a mobile telemedicine platform*, which are currently undergoing tests in India, [Kuan, 2010] and the successful start-up *EKO Core Stethoscope*. [Wong, 2015]

This work builds on our previous study, *Pulmonary Crackle Detection using Signal Processing and Machine Learning* [Grønnesby, 2015], we developed a method for classifying crackles in audio files. The method we used were based on two main features, *Statistical moments in wavelet decomposition* and *Short Time Fourier Transform*, and using two SVM classifiers, one for each feature type. Though a little unmotivated, the reason we used two classifiers were that our features did not give a clear separation, so each classifier would partake in a voting scheme. Subsequently we assumed that our features were not descriptive enough of our data, and that a separation between the classes was hard to distinguish. A lot of the previous work was heavily dependent on our

classifiers recovering for features with an exaggerated number of dimensions without adding information.

1.3 Proposed Solution

Our approach is outlined in the following list:

1. From the time since our previous study, even more data have been gathered. Furthermore, more of the gathered data have been manually classified, so the amount of data to evaluate our approach have been increased as well. However, we use a subset of the gold standard as an early benchmark in evaluation, and save major evaluation for future work.

2. We have increased our training set, with a new Data Annotation Tool that simplifies the process of generating labelled training samples. Where we previously had 37 crackle samples and 61 normal samples, we have now increased the respective amounts to 178 and 208 each. In addition, we have also annotated 22 wheeze samples to begin preliminary tests for wheeze detection.

3. We have improved upon the crackle detection from the previous work. Exploring new features and simplifying the classification process. To alleviate this problem, we have reduced the dimensions in favor of a higher number of features. From having 3 different high dimensional feature types, resulting in a vector with more than 300 dimensions, we have selected 5 different feature types of only 1 dimension each, resulting in a vector with only 5 dimensions. This requires a less complex classifier such as K-Nearest Neighbor (KNN), subsequently this decreases our training time as well.

4. Training is performed in the same way as our previous study, but required time have decreased due to lower dimensions in our feature vectors. This makes our pipeline more responsive through our GUI.

5. We have done evaluations on classifying Wheezes, and have found some preliminary results. Wheezes, as opposed to Crackles, are lasting, melodic and continuous sounds. Features that work for crackles may not be applicable to wheezes directly due to this difference.

6. In addition, we have improved and further developed our front-end, incorporating a waveform display and audio player embedded in the

page. This allows researchers to review their data quickly, and also correct any errors that might occur in classification.

1.4 Evaluation

We evaluate our implemented pipeline by cross-validation, in addition we have also evaluated our pipeline on a subset of the gold standard lung sound database. We found a *Precision* of 85.5% ±6.1 and a *Recall* of 83.6% ±10.2, giving a F1-score of 83.5% ±3.6. In our tests on full audio files we found that our classifier was able to classify 14 of 23 crackle files as containing crackles, and 186 of 247 as not containing crackles (normal). The results on full audio files are not as accurate as per window, which indicates that we need to correlate the windows that are being classified to the audio file itself.

1.5 Contributions

We make the following contributions:

- Description of a Machine Learning based approach to preprocessing, feature engineering, classifying and representing two different abnormal lung sounds, *Crackles* and *Wheezes*.
- A pipeline for fast analysis of lung sound audio files containing lung sounds and a web based front end GUI for presenting analysis results in a readable manner for doctors and medical students.
- Evaluation using a subset of the, to our knowledge, largest epidemiological dataset of expert annotated lung sounds.

1.6 Conclusion

In conclusion our results show that the approach has acceptable sensitivity and high specificity, for individual windows, and we believe with the completion of our future work that we can achieve close to the same accuracy for full audio files as well. Together with completion of our future work, we believe that our core technology has applications in devices such as mobile devices, medical equipment or as a web api. Various use cases ranging from self-monitoring by patients, training of medical staff and students and automatic monitoring in

medical equipment.

2 Methods

In this chapter we motivate and describe the methods used in our approach. To complete our challenges, we have to complete the following steps: Generating a training set, Preprocessing Audio files, Feature Selection and Selection and Training of a Classifier.

2.1 Training Set Generation

We divide our data into three classes; normal, crackle or wheeze. While the gold standard has been classified by 3 experts, only one expert has classified each individual window of the training set. Our training set contains smaller windows from the classified parts of the gold standard. The reason we have chosen to extract smaller, shorter windows is because of the large amount of information audio file contains. Therefore, we believe searching for events (crackles or wheezes) that last either between 5-30 milliseconds or between 0.5-3 seconds in 15 seconds of audio is counter-intuitive. We believe that this is reminiscent of a *Needles-in-Haystack Problem*. [Moreland and Truemper, 2009]

Therefore, we further manually label smaller windows containing the three classes we are looking for. To aid the manual labelling, we implemented a data exploration tool (Figure 2.1). Each audio file lasts for about 15 seconds, and contains normal breathing as well as crackles. So we had to extract smaller excerpts, or windows, from these files and save them as individual files. For

crackle samples we have used a size of 4096 samples per file, or 92 ms. This will guarantee that the windows we label as crackles contains at least one or more whole crackles, since coarse and fine crackles typically lasts between 5-30 ms [Sovijarvi et al., 2000]. Each of the windows overlap by 50% with the previous window, so if a crackle occurs on the edge of a window, it will be contained in the next window. Figure 2.2 shows this process of manual labelling of windows.

The same procedure is applied to wheezes, but the window size is bigger since wheezes typically last for orders of magnitude longer. We chose to use 131 072 samples per window, which is about 2.97 sec. To have a consistent size when classifying wheezes, we also extracted normal samples with the same amount of samples.

As of now, we have two training sets. The first is the Gold Standard from Tromsø 7, and the second is from the inter-observer pilot study conducted prior to Tromsø 7. [Aviles-Solis et al., 2015] We chose to only use data from the Tromsø 7 study, since the dataset from the pilot study had already been edited to highlight the abnormal lung sounds. As we will discuss later in this chapter, all features are scaled using a standard scaler, so variations within the training set might produce a skewed scale when fitting the scaler.

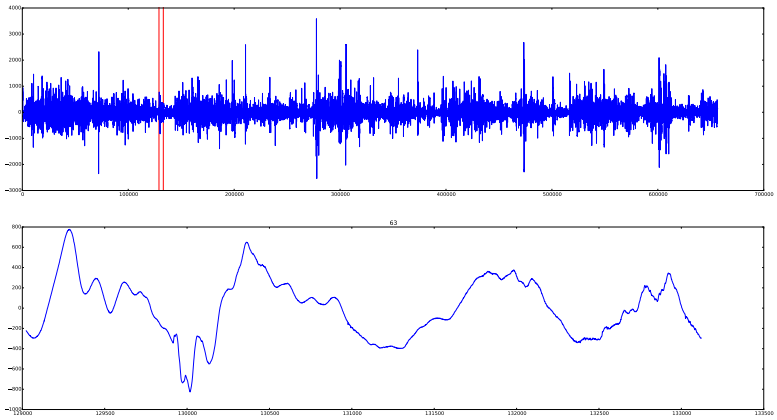


Figure 2.1: Data annotation tool. By sliding a window along the signal, smaller windows can be examined and saved

Generating a lot of training data is a time consuming task, but we chose to do this manually to a better ensure that our training set contains only true positive samples.

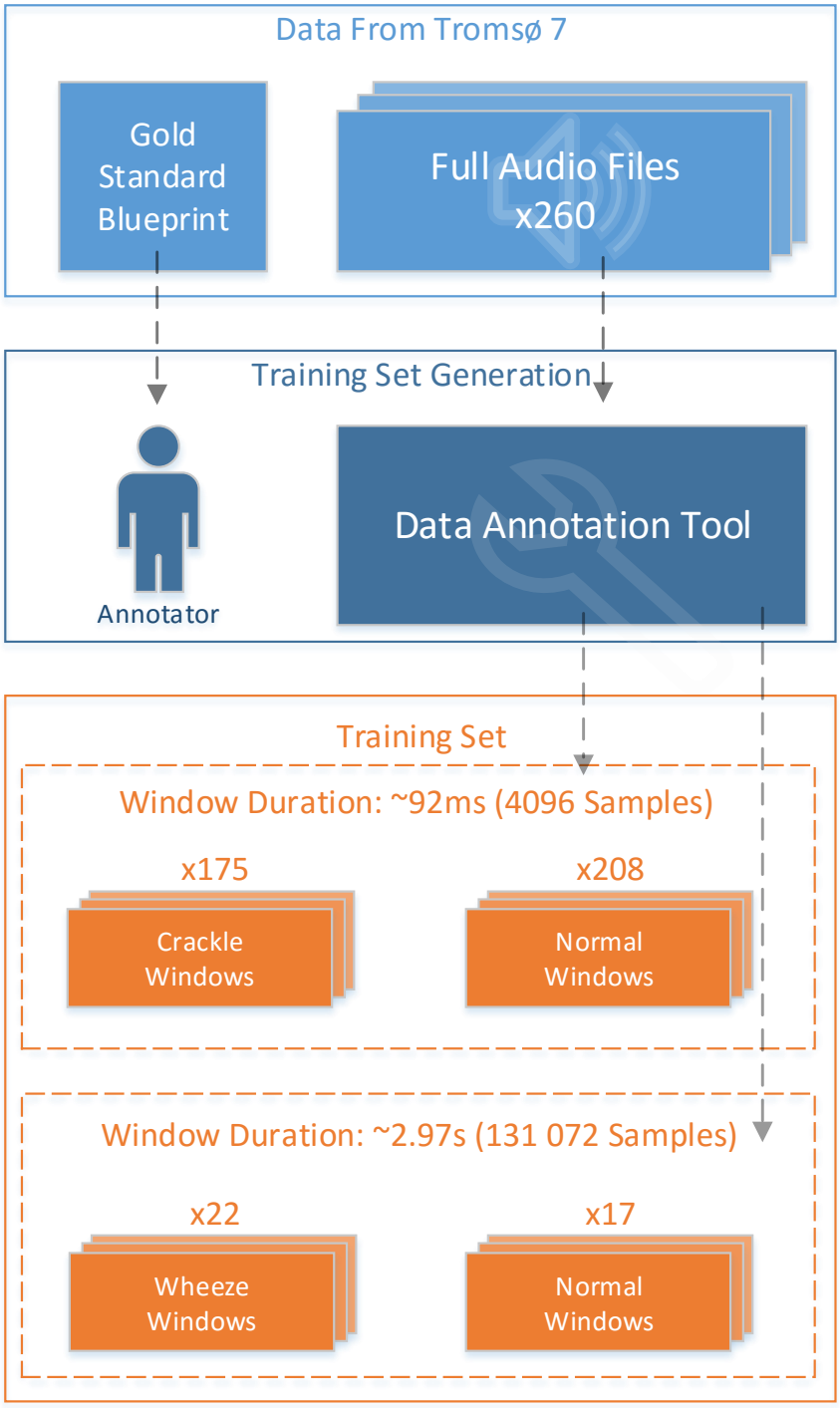


Figure 2.2: Training set generation with a single annotator

2.2 Preprocessing

Every audio file in our repository consists breathing recorded over approximately two respiratory cycles, lasting on average 15 seconds. These audio files are sampled at 44 100 Hz, equivalent to the sampling rate of Compact Discs. When recording digital audio, 44,100 Hz has become the preferred sampling rate due to the Nyquist-Shannon sampling theorem, which is a fundamental bridge between continuous-time signals and discrete-time signals (often called analog and digital signals respectively).

Theorem 2.2.1. *A sampled waveform contains all the information without any distortions, when the sampling rate exceeds twice the highest frequency contained in the sampled waveform.*

Since the abnormal sounds of interest occur between 50 - 2400 Hz, we have the option to down-sample the audio files quite a bit, but have chosen not to. The features we have chosen to use does not require heavy computation, so therefore we want to keep the samples as close to their original format as possible. We consider feature extraction to be part of the preprocessing steps, so the only preprocessing done before feature extraction is windowing of full audio files. The training set is already windowed, and does not require windowing. Figure 2.3 shows windows applied to an audio file.

The different feature types are outlined in the next section.

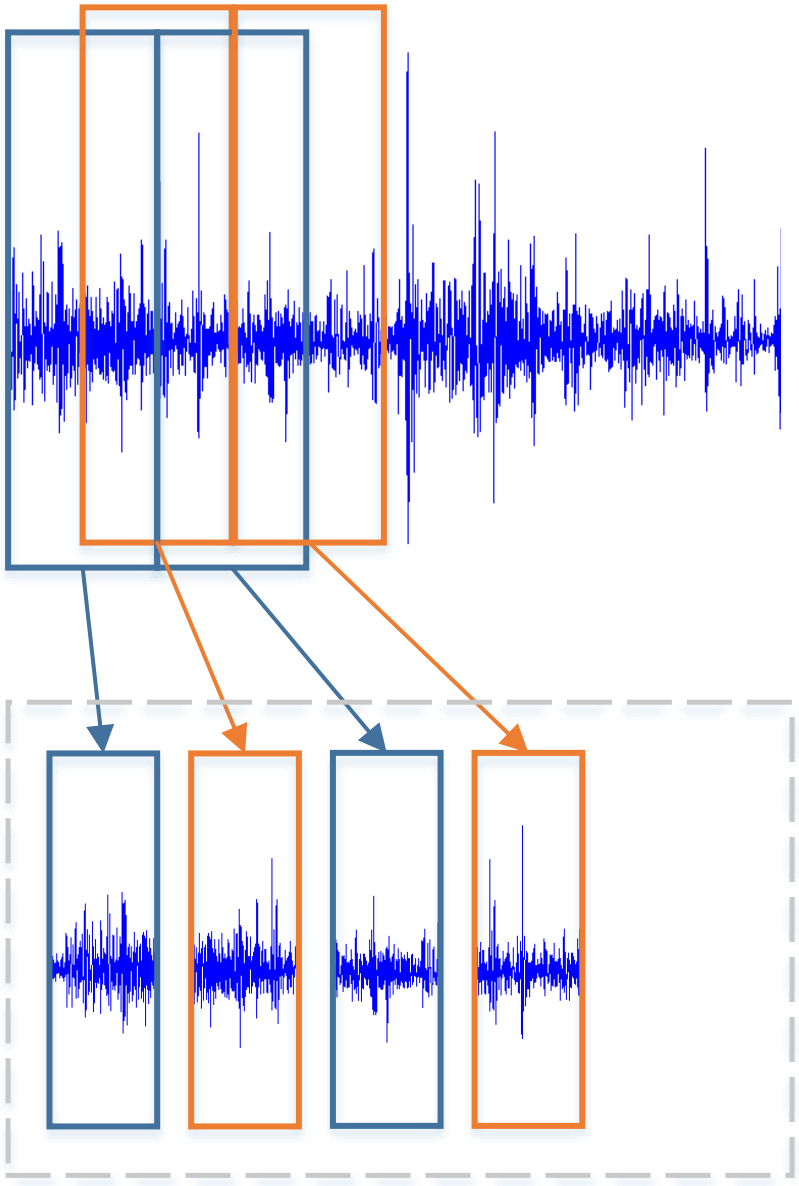


Figure 2.3: Windowing of an audio file with overlapping windows (50%), using a sliding window of 4096 samples (crackles) or 131 072 samples (wheezes)

2.3 Feature Selection

In this section we will present some of the features that we have evaluated in our approach. Throughout our study we have evaluated some features that did not end up in our pipeline due to inconclusive results. These alternate features are summarized in the section Alternate Features. Additional features for other types of learning (such as deep learning) is explained in Chapter 5, Related Work.

We evaluated each of our features individually to get an idea of how well they will perform as single feature vectors. The results are presented in the Results & Evaluation section. We have evaluated the same features for both crackles and wheezes and found that they are not universally applicable (see Chapter 4 Evaluation and Section Future Work in Chapter 5, Conclusion).

2.3.1 Time Domain Features

We chose to find features that are both dependent on the time domain and the frequency domain, since both are important for classification of Wheezes and Crackles. Crackles may be more dependent on the time domain rather than the frequency domain, due to its short-lasting explosive nature. We collected the features described below from the time domain. These features, as with the MFCC are on a per window basis, meaning the features are calculated on 4096 samples or 92ms of audio. Each audio file will be divided into around 300 windows for Crackles. Wheezes have a longer window and therefore, we collect about 8 windows per file at a rate of 131 072 samples per window or 2.97ms. The time domain features are calculated directly on the audio without any prior transformation.

Variance

The variance within a time series, or any vector for that matter, is defined as a measure of the spread of a distribution.

In our case the distribution is audio amplitudes over time. Our evaluation and results have shown that crackle windows have more variance than normal windows due to their explosive nature. Normal windows may vary more in terms of zero crossing rate, but the spread is higher for crackles as they usually contain more power, or have a higher amplitude, than normal breathing. The variance is related to *Shannon's entropy* which have been used to locate heart sounds for the purpose of eliminating these from lung sounds. [Yadollahi and Moussavi, 2006]

Range

The simplest features of our feature set, is the maximum value of the audio file subtracted from the minimum value of the audio file. This gives us a certain range for each audio file, and since crackles have an explosive popping noise, we believe the range of crackle windows will be higher than normal breathing. Note that this feature is highly dependent feature scaling, as it is highly sensitive to noise and other artefacts that may cause sudden high amplitudes in the audio. The formula for the range of a signal S gives us:

$$R(S) = |Max(S) - Min(S)|$$

Sum of Simple Moving Average

While Simple Moving Average is closely related to stock market price fluctuations, it can be applied to a signal as well. The sum will give an indication of how much the signal is changing over the course of the time it lasts. We have used two different granularity levels of this feature. The coarse version looks at all 4096 samples as one signal, while the fine version calculates the value for smaller sub-windows of 800 samples each, sliding 100 samples at a time. The fine version only keeps the window with the highest amount of change. Calculating the Sum of Simple Moving Average for a full sized window is done according to the following formula:

$$SMA_{coarse}(Sig) = \sum_{n=1}^{len(Sig)} |Sig_{n-1} - Sig_n|$$

And to apply the same formula to smaller windows and selecting the window with maximum change, we divide the window into n smaller sub-windows and apply the same formula:

$$SMA_{fine}(Sig) = Max(SMA_{coarse}(win_1), SMA_{coarse}(win_2), ..., SMA_{coarse}(win_n))$$

2.3.2 Frequency Domain Features

While the time domain features are extracted from a window of a signal, which we do not transform prior to extraction, frequency domain features

are calculated on frequency spectrum magnitudes. We obtain the frequency spectrum magnitudes by calculating a Fast Fourier Transform and only keeping the real parts of the coefficients (absolute values) or by using a Short Time Fourier Transform to calculate spectrograms.

Spectrum Mean

The mean value of the spectrum gives us an indication of the central tendency in the frequency domain. Crackles that occur in breathing often carry more power in higher frequencies. The center of the power distribution would naturally have a higher value for any windows containing crackles, though we have observed that this is a tendency rather than a rule.

2.3.3 Feature Scaling

Audio data is non-stationary and fluctuating, so each recording might have a slightly different sound, lower gain, noise etc. To deal with this we need to scale our features, standardizing each feature category across all training samples. Using a standard scaler, we are able to achieve this. Outliers and divergence between training samples are minimized through this process, and brings all features to a standard scale compared with other samples. This is especially important with classifiers such as KNN, as they calculate distance metrics between points to determine class membership. Any outliers as a result of unscaled data will have an impact on classification accuracy. Attaining poor accuracy because we do not scale our features is bad enough, but attaining a good accuracy because features are not scaled properly is a much bigger concern. So scaling all features to a common scale based on the training set is done for all audio files.

2.3.4 Summary

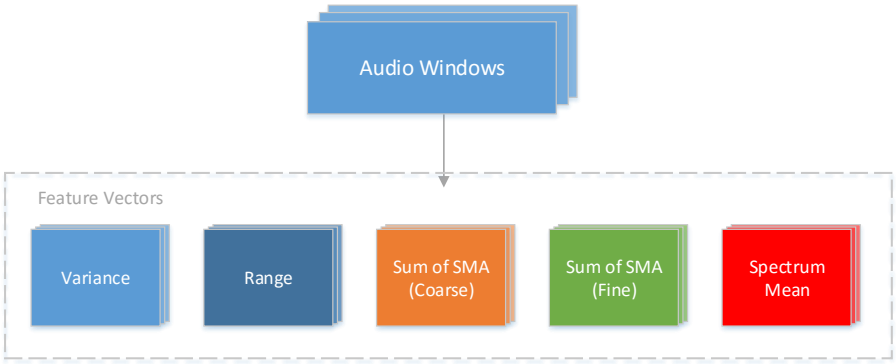


Figure 2.4: Each window is represented by a 5-dimensional feature vector.

We chose to keep the five preceding features (Figure 2.4 shows our final feature vector), which have proven to be equally as good as our old features, with fewer dimensions. [Grønnesby, 2015] We want to use as few dimensions as possible due to the *Curse of Dimensionality*. A term coined by Bellman [Bellman, 1957] and explained in terms of machine learning by Keogh et al. as:

> For machine learning problems, a small increase in dimensionality generally requires a large increase in the numerosity of the data, in order to keep the same level of performance for regression, clustering, etc. [Keogh and Mueen, 2011]

Since we have a training set of 173 and 208 samples, we want to stay below 10 dimensions. With each dimension added, the amount of training data needed

2.4 Alternative Features

We have tested some additional features, and have some preliminary results. But these are at a too early stage to be integrated or we did not find any conclusive results that they would increase classification precision.

MFCC

The Mel Frequency Cepstral Coefficients is a technique for feature extraction that has seen great success in speech and music recognition, and is a part of almost all modern applications in these fields [Logan et al., 2000] [Hasan et al., 2004]. The algorithm for computing the MFCC of an audio signal uses the following steps:

1. Framing the signal
2. Compute the Discrete Fourier Transform (DFT) for each window
3. Apply the Mel-filterbank to convert frequency to the Mel-scale
4. Take the Log amplitude of the Mel-scaled spectrum
5. Compute the Discrete Cosine Transform on the Mel-scaled Log amplitudes

To understand what we gain from using the MFCC transform, we need to explore each step individually. Audio signals are nonstationary, meaning their statistical properties change over time. Therefore, when framing a signal into smaller chunks we are able to analyze a signal that is approximately stationary. Then applying the fourier transform to each of these chunks creates a spectrogram of the signal. This spectrogram denotes spectral content of the signal in the Hertz scale, so applying the Mel-filterbank converts the Hertz values into the Mel-Scale. The Mel-Scale is a perceptual scale of pitch; which models pitch closer to what humans perceive rather than actual Hertz values. Further, taking the Log amplitude of the Mel-scaled spectrum gives a power spectral density estimation, which shows the energy of the different frequency bins. Lastly we compute the Discrete Cosine Transform of the log power spectrum as if the log spectrum were a signal. Normally these types of transformations are used in data compression (audio, imaging and video), and to obtain the original signal we would apply the *Inverse Fourier Transform*. However, when we compute the *Discrete Fourier Transform* in the second step, only absolute values are kept, which means phase information is lost. Therefore, we use the *Discrete Cosine Transform*, rather than the *Inverse Fourier Transform*. The resulting coefficients, the MFCC, is a cepstral representation of the audio clip. A cepstrum contains information about the rate of change at different spectrum bands, which would be the Mel-spaced frequency bins.

MFCCs have been the state-of-the-art standard in speech recognition for a long time, and it has applications in Music Information Retrieval as well. Due to the nature of crackles, being short explosive sounds, lasting less than 100 ms, the

MFCC might not be directly applicable. In addition, the MFCC are susceptible to changes in loudness.

The MFCC have historically been used with sequence classifiers such as Hidden Markov Models, so its usefulness is somewhat dependent on the type of classifier. They are also meant to model a large vocabulary in speech recognition, while our classification problem is binary. Moreover, the MFCC can have quite a few dimensions depending on the signal frame size, so they cannot be used directly in classifiers where careful feature engineering is required.

2.4.1 Spectrogram Image Analysis

In the article *The detection of crackles based on mathematical morphology in spectrogram analysis.* [Zhang et al., 2015] The authors investigated a method of classifying crackles based on a generated spectrogram using image analysis techniques. The authors found in their experiments that a crackle often leaves an elliptical pattern in the spectrogram, from which features can be extracted. We replicated this method by calculating the spectrogram of a signal using the short time fourier transform, and using a histogram equalization to increase the contrast of the spectrogram. Further we used thresholding to normalize each value to either 0 or 1. Though we were able to replicate the spectrogram processing techniques, we were unable to find the elliptical structure of the crackle present in our experiments.

We can see one of the main problems we encountered using this approach in Figure 2.5 is that we can see something that resembles an elliptical pattern, but a very similar pattern also occurs in files that the experts have classified as a normal. Without access to the actual data used in the study it is very hard to compare exactly to what the Zhang et al. have done. Using a wavelet decomposition to compute the spectrogram, as they did in their study, rather than the short time fourier transform might give a better frequency resolution.

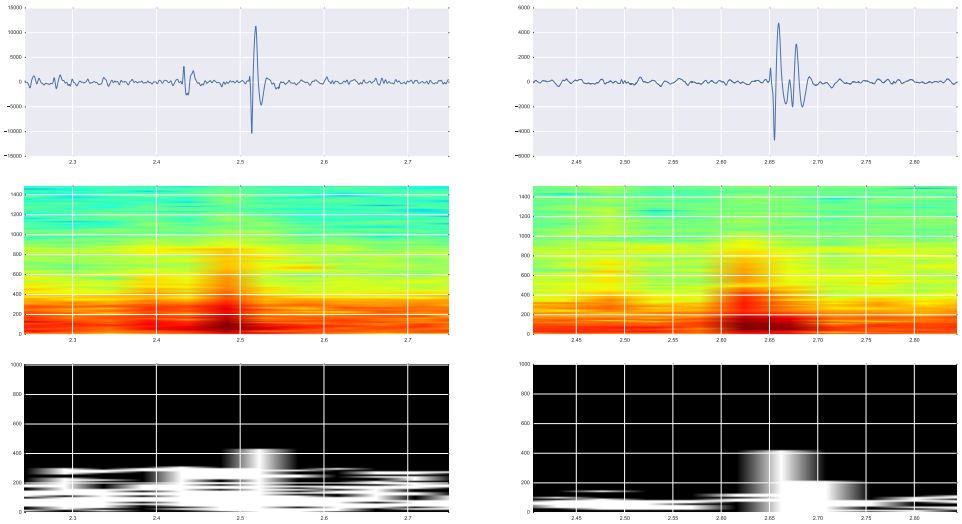


Figure 2.5: Comparison of two signals with accompanying spectrogram plots and histogram normalized spectrogram. Left: Crackle, Right: Normal Breathing (Noisy)

2.5 Breathing Cycle Detection

We have done some experiments with detecting the breathing cycle of an audio file. Since most crackles happen at an inspiratory phase and wheezes at an expiratory phase, determining the breathing cycle could help eliminate false positives. By restricting the area where our pipeline searches for abnormal sounds we would eliminate possible false positives that happen in between breathing cycles. To do this we have implemented a method by looking at the smoothed root mean square of the signal. We smooth the signal by convolving it, and then look for the minima of the resulting RMS curve. This presents an issue though, since we are looking for global minima, it is natural that these are located either at the start or at the end of the signal. As we can see in Figure 2.6, there are one local minima at the start of the signal, and four at the end. So for us to make this method feasible for breathing cycle detection, we would need to restrict the location of the global minima, using local minimas instead. It could be as simple as saying that the minima cannot be located within the X first or last samples of the signal, though it is not a very elegant solution. Another way to do this would be to use a Parzen Window (Kernel Density Estimation), in order to estimate the probability density function of a part of the signal. Then comparing the different estimates to find the parts with the lowest or highest distributions to find breaks between breathing cycles or find the peaks in inspiratory or expiratory cycle.

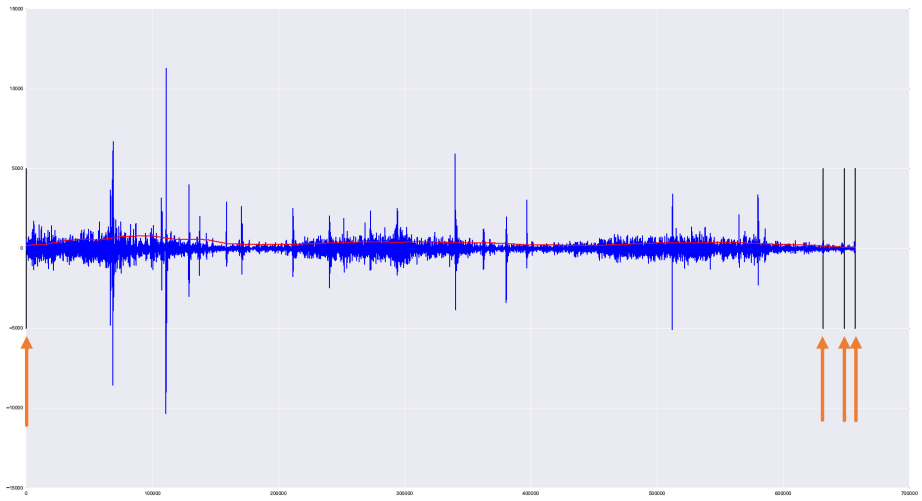


Figure 2.6: Waveform with the RMS curve plotted in red. The black vertical lines are the minimas of the RMS curve.

2.6 Classifier

In our evaluation we have evaluated 3 classifiers. In our experiments we found that the SVM performed best on our features. We used K-Nearest Neighbors and AdaBoost (Decision Trees) for comparison.

2.6.1 K-Nearest Neighbor

The K-nearest Neighbors (KNN) method is a *non-parametric*, *lazy* method used in both regression and classification. It is *non-parametric* in the sense that it does not make any assumptions about the structure of the underlying data, such as Gaussian distribution or linear separability. And it is *lazy* since it does not require any training step. In KNN classification, class membership of an unseen data point is determined by the k closest training samples in the feature space. Using a distance metric such as *Euclidean*, *Manhattan*, *Mahalanobis*, the KNN labels new points based on the majority of the nearest points (see Appendix A for common distance metrics).

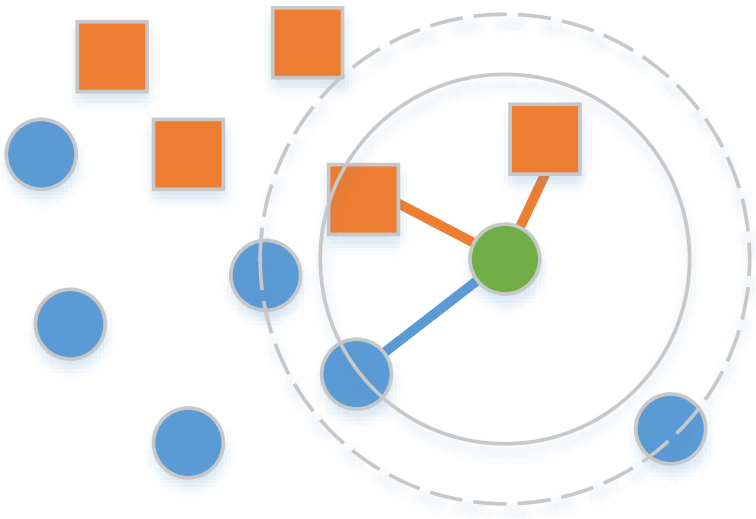


Figure 2.7: KNN with k=3 (solid line), the new green point would be labelled an orange square. For k=5 (dotted line), the point would be labelled as a blue circle.

The selection of k is dependent on the data, while a small value for k can give very distinct boundaries between two classes, and be useful if the data have very small margins (which is our case, see Figure 4.3) but it also affected by a noisy (irrelevant) features. For larger values of k, the classifier becomes more robust against noise, but some of the distinction between classes will be lost. [Peterson, 2009]

Dynamic Time Warping

A type of distance metric called dynamic time warping could be paired with the KNN to work better for signals that differ in time and speed. Which for Crackle detection may be an advantage, and for Wheeze detection (which vary more than Crackles in durations) may have an even bigger impact. The main problem with using dynamic time warping is that the operation has a complexity of $O(m^2)$ for two signals of length m, and since KNN already have a classification complexity of $O(n^2)$ for n training samples, classification becomes time consuming. For simple feature vectors as the ones we use, dynamic time warping does not perform much better than simpler distance metrics such

as Euclidean Distance. But if we used raw audio data in KNN, dynamic time warping could give a much better result, but would take an increased amount of time to compute.

For the dynamic time warping to have a desired effect, the signal should be a time-dependent series, rather than just our statistical summary features. It would be possible apply dynamic time warping to the MFCC vectors, and using the distance itself as a feature. However, the larger the vectors compared are, the longer time it will take to compute the signal similarity. [Müller, 2007, Ding et al., 2008]

2.6.2 Adaptive Boosting and Decision Trees

AdaBoost, or *Adaptive Boosting*, is a method of chaining together a number of weaker classifiers to obtain a new classifier that is the weighted average of the weaker classifiers. With a sufficient number of iterations, the error of the final classifier can become quite low, although, there is a certain danger of over-fitting. [Schapire et al., 1998]

The basic idea of *AdaBoost* is to have an optimally constructed classifier, that satisfies the function:

$$f(x) = sign\{F(x)\}$$

where

$$F(x) = \sum_{k=1}^{K} \alpha_k \Phi(x)$$

where $\Phi(x)$ is the base classifier, which returns a binary class label. K denotes the number of classifiers being boosted, and α is the weight associated with the k^{th} weak classifier. Finding the α is done through iterative, or stepwise, optimization of m steps, where $F_{m-1}(x)$ is the previous, optimized iteration. So to compute the optimal values for step m, we would compute the cost function:

$$\alpha_m = argmin J(\alpha)$$

where

$$J(\alpha) = \sum_{i=1}^{N} exp(-y_i(F_{m-1}(x_i) + \alpha\Phi(x_i)))$$

While stepwise optimization is usually suboptimal compared to direct optimization, direct optimization of trees is a highly complex task, and often impossible to carry out. [Theodoridis and Koutroumbas, 2009, p. 231–232]

Decision Trees is a common classifier to use with *AdaBoost*. Which we tried to employ in our project, but it has a high training time compared to SVMs, and does not increase the overall accuracy of the classification.

2.6.3 Support Vector Machines

Support Vector Machines or SVMs [Boser et al., 1992] are a type of linear classifiers, much in the same fashion as linear perceptrons where two classes, c_1 and c_2 are assumed to be linearly separable by a hyperplane:

$$f(x) = w^T x + w_0 = 0 \qquad f(x) \begin{cases} c_1, & if\ w^T x > 0 \\ c_2, & if\ w^T x < 0 \end{cases}$$

Finding a hyperplane is the training step of the linear perceptron, but the problem is that there is no way to know if the hyperplane separates the two classes by an equally large margin. [Theodoridis and Koutroumbas, 2009, p. 93] This is where the *Support Vectors* of the SVM are important. By minimizing the cost function of the parameters w, w_0 so that:

$$minimize \quad J(w, w_0) \equiv \frac{1}{2}|w|^2$$

$$subject\ to \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1, 2, ..., N$$

We obtain the maximum margin between the two respective classes, with equal length to the hyperplane. [Theodoridis and Koutroumbas, 2009, p. 120–121] However, not all classification problems are linearly separable, there might often be a few samples of classes that are non-separable. To deal with this problem we can introduce *slack variables*, ξ, so that we ignore a certain amount

of samples within the support vector margins, or misclassified samples. The different constraints are given by:

$$y_i[w^T x + w_0] \geq 1 - \xi_i$$

Where $\xi_i = 0$ for correctly classified samples, $0 < \xi_i < 1$ for correctly classified samples within the margins and $\xi_i > 1$ for incorrectly classified samples.

By modifying the cost function so that we include ξ and using a positive constant C to control the influence of the *slack variables*:

$$J(w, w_0, \xi) = \frac{1}{2}|w|^2 + C\sum_{i=1}^{N} I(\xi_i) \qquad I(\xi_i) \begin{cases} 1, & \xi_i > 0 \\ 0, & \xi_i = 0 \end{cases}$$

The C parameter can be determined through *Grid Search* by fitting different values for C to different classifiers and selecting the highest scoring classifier. The C parameter trades of the size of the margin with the amount of incorrectly classified samples or samples within the margins of the support vectors. [Theodoridis and Koutroumbas, 2009, p. 124–125]

There is also the case of SVMs where it is not desirable to find linear separations of data, and that a non-linear separation would be advantageous. The way that SVMs solve this problem is by means of *Kernel Functions*. *Kernel Functions* are functions that can produce a mapping where:

$$x \in \mathcal{R}^l \rightarrow \mathcal{R}^k$$

Where the vectors are mapped into a new k-dimensional space, which allows mappings in infinite dimensional spaces, if it is required. At first glance, this would imply that the complexity increases, since k is a higher dimensional space than the input space l. However, according to:

$$y_i^T y_j = (x_i^T x_j)^2$$

The inner product of pairwise vectors in the new higher dimensional space is expressed as a function of the inner product of the corresponding vectors in the original feature space. [Theodoridis and Koutroumbas, 2009, p. 198–200]

As for different mappings, also referred to as *Kernels*, we have found that the *Radial Basis Function Kernel* have given the highest classification accuracy (For a list of the most common *Kernel Functions* see Appendix B).

3

Design and Implementation

Our implemented pipeline is available through a web interface; we use a Flask back-end script to perform analysis on audio files uploaded to the server. The pipeline itself is implemented in Python 2.7 using, amongst others, the popular machine learning library Scikit Learn [Pedregosa et al., 2011]. We chose to use Python and Sklearn due to its flexibility and ease of use. Using Sklearn we can evaluate different classifiers with minimal changes to implementation because all classifiers have the same call convention, and by using *Polymorphism* it integrates seamlessly with our pipeline. The pipeline is portable across different operating systems, provided that Python 2.7 and the required libraries are available. Our pipeline runs single-threaded on a single computer with basic hardware or a virtual machine. Due to low execution time there is no need for a distributed implementation. But if the computational requirements increase, for example implementing a *Deep Learning* algorithm (see future work) we might look into libraries such as *Theano*, a math library for transparent GPU computations [Bergstra et al., 2010, Bergstra et al., 2011] or *MLlib*, machine learning library running on top of the cluster computing framework *Apache Spark*. [Zaharia et al., 2010, Meng et al., 2015]

Low execution times for our pipeline is vital in order to have a responsive interface, that can actually be used in order to do analysis on the fly.

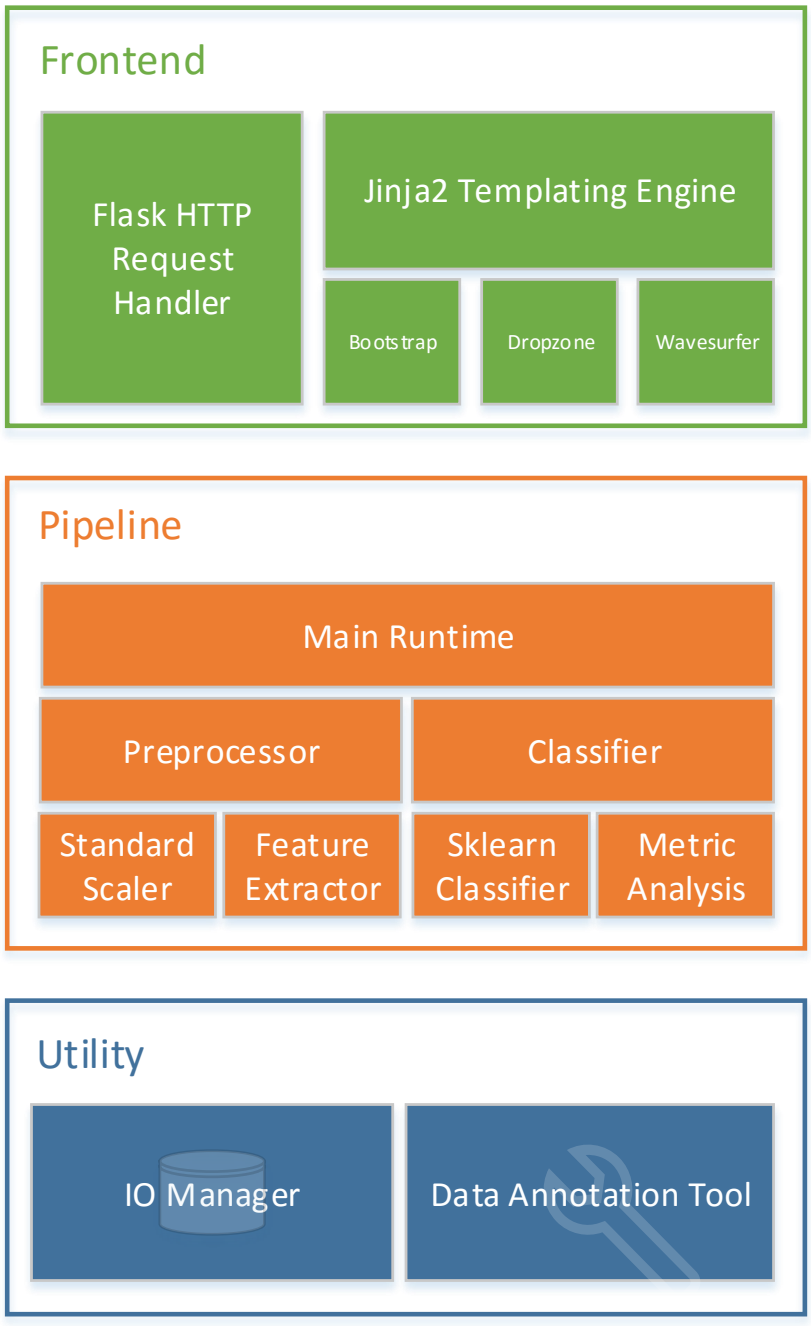


Figure 3.1: Software stack of our system.

3.1 Pipeline

The pipeline consists of four modules: Main, Preprocessor, Classifier and Frontend. In addition there are a few utilities to simplify interaction with the file system, generating a training set etc.

Main Runtime provides an interface to submit audio files for analysis. The main runtime instantiates all other modules and handles all full sized audio files, providing a simple abstraction for other modules.

The Preprocessor is divided into three modules, one base classifier which implements all common functionality between wheezes and crackles, and one specialized classifier for each class. The main difference between these two classes is the size of the analysis windows. Each of the preprocessors needs to be instanced and has to process a training set in order for it to be fully initialized. The reason is that both preprocessors use a standard scaler module which needs to be fitted before it can be used on new data.

The Classifier implements training and inference of the classification algorithm itself. Upon initializing the classifier, it can be run in three different modes: *train*, *metric*, *load*. In *Train* mode the classifier will collect the training set and use the referenced preprocessor (with a fitted standard scaler) to collect all features from the training set. When the training set is obtained, the classifier will train the assigned machine learning algorithm and persist the classifiers data to disk. *Metrics* mode will run 100 cycles of training and validating the classifier of choice. When splitting the training set, 70% is randomly sampled and used for training, while the remaining 30% is used to validate the classifier. The F1-score of the classifier is calculated using the validation part of the training set at each iteration. After the 100 cycles we calculate the average and standard deviation of the F1-score.

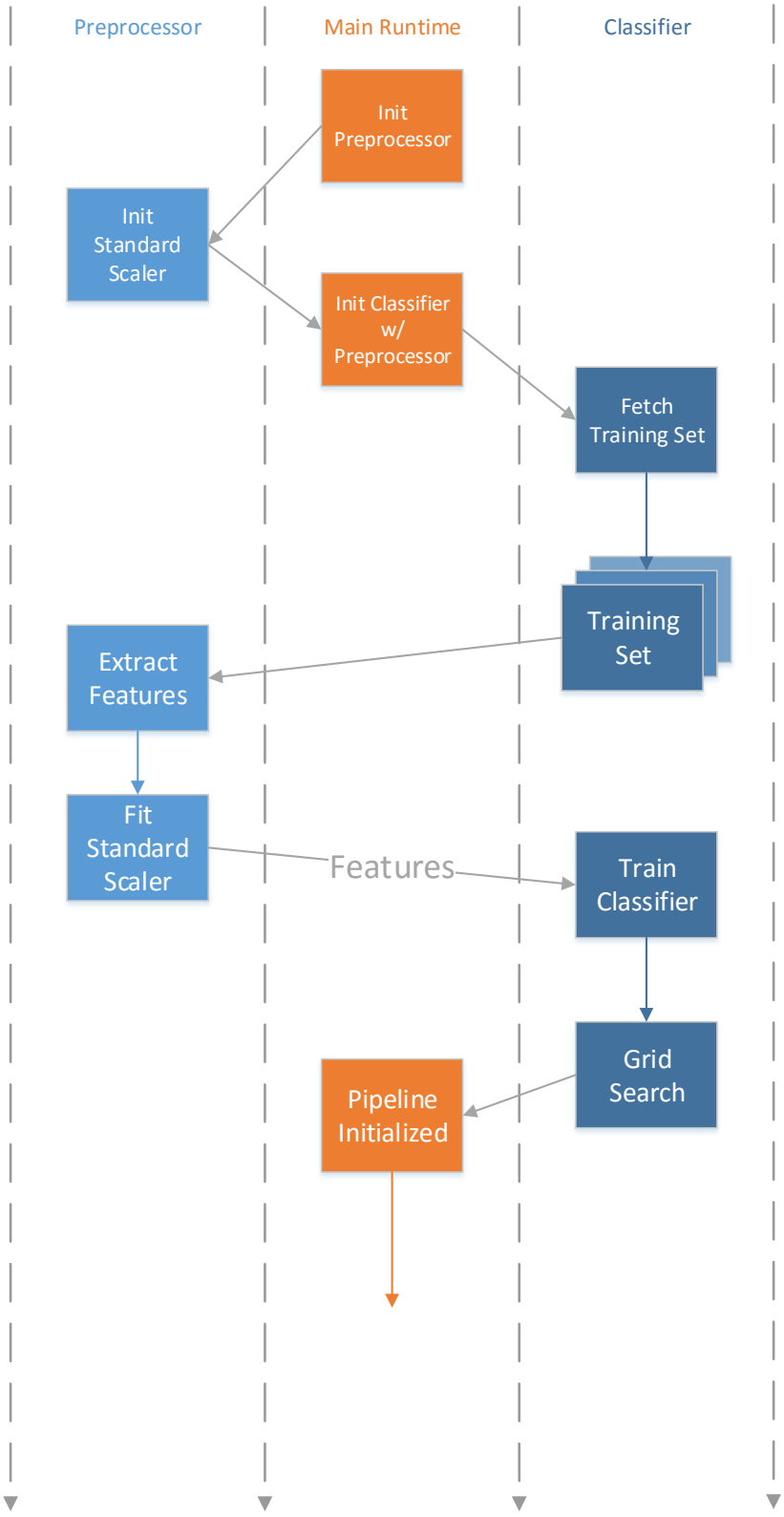


Figure 3.2: Initialization process of the pipeline. The preprocessor must be initialized before the classifier, since the Standard Scaler must be fit on the training set.

3.2 Frontend

Our frontend library built around the flask http library and the jinja2 templating engine. The frontend provides an access-point and interface for general practitioners to upload and analyze audio files of lung sounds. The page is built using the *Bootstrap* framework for the visual representation, *Dropzone.js* for file uploading and *Wavesurfer.js* to generate waveforms and integrating an audio player in the browser. We believe that for our solution to be useful we need to provide a good user interface, that satisfies the needs for all research collected in Tromsø 7. Therefore, the Frontend prototype have been developed with requests from our collaborators at Tromsø 7.

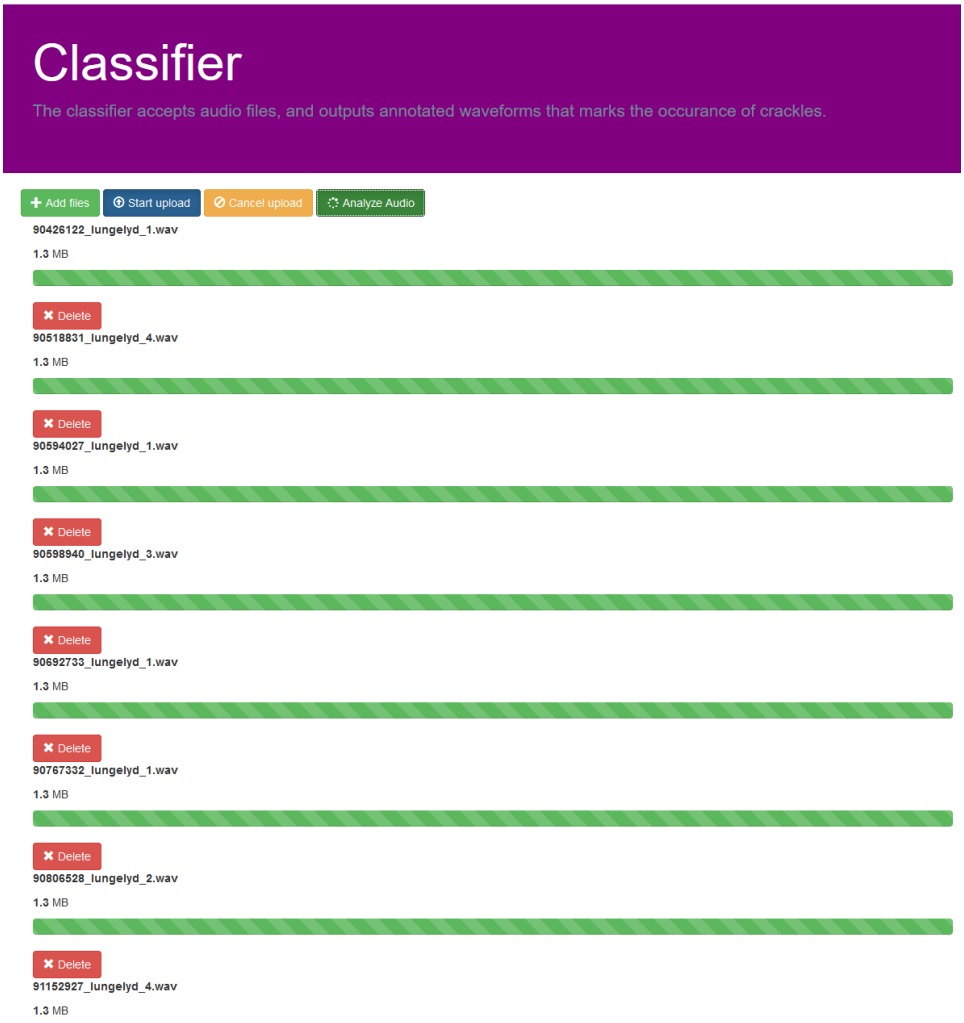


Figure 3.3: Upload page, has the ability to upload a number of files simultaneously.

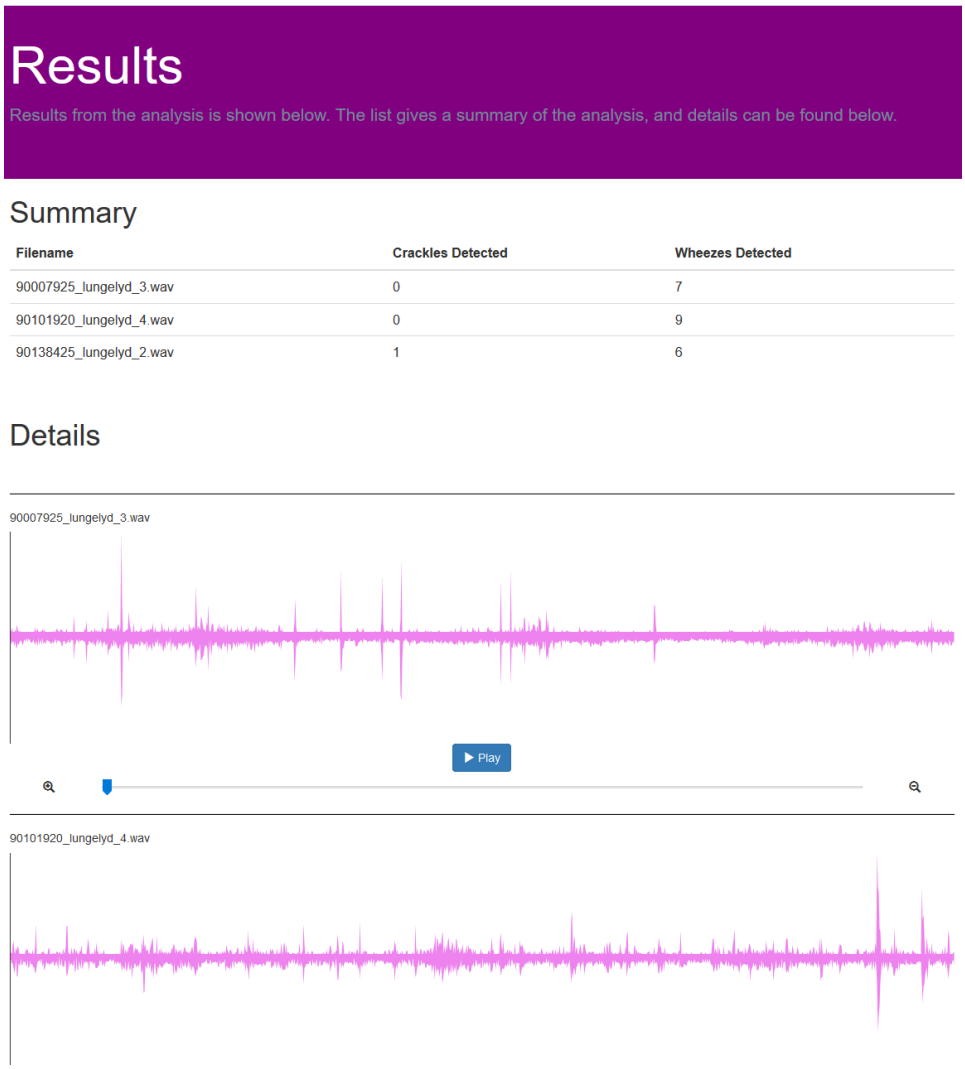


Figure 3.4: Results page, with a summary at the top of the page and an integrated audio player for each file. Numbers are for example purposes.

We can see that the classification and results sections of the front end (Figure 3.3 and 3.4), that users are able to upload files to the server. These files can then be analyzed by clicking the *Analyze Audio* button (Figure 3.3) and the user will be redirected to the results section after a few seconds, depending on how many files are uploaded at the same time. On the results page, the user is presented with a summary of all files analyzed and how many crackles or wheezes were detected. Below the summary, the user can go through each file and listen to the audio file in real time from the browser. We also plan to annotate the waveforms, in another color, marking where the abnormal findings are located.

3.3 Libraries

To implement such a system, we are dependent on a number of libraries to simply our implementation. The main libraries are listed here with a short description:

Scikit-Learn is an extensive machine learning library built with Python. It implements most of the common machine learning algorithms, feature selection and data transformation functions. We use this library mainly for the machine learning algorithms and validation of our classifiers.

Librosa is a python library for audio and music analysis. We use this library for calculating the fourier transform, short time fourier transform and the MFCC. It also supports calculating Spectral Centroids.

Flask has been one of the more popular lightweight http libraries for Python. Although it is not as extensive as Django, it has a much simpler interface and requires much less code in order to work properly. Although most of this is subject to opinion, we chose to use Flask because it is a library we are familiar with.

Numpy is the most complete scientific computation library available for Python. Implementing most common methods from Linear Algebra, Statistics and provides n-dimensional arrays/matrices. Most scientific libraries for Python use Numpy to some extent.

Scipy is another library for scientific computation such as numpy, but also provides signal processing techniques.

3.4 Prototype Deployment

We ran our prototype using cProfiler for Python, and on a machine with the following specs:

Windows 10 Pro 64-bit
Intel Core i5-4570s @ 2.90GHz (4 Physical Cores)
6Gb RAM Python 2.7.9

Our initialization process consumed about 1.44 seconds, including sequential grid search of 64 SVM parameter combinations (192 fits). For deployment this process can be run separately and the resulting classifier and scaler can be saved to disk for later use with a user interface (one of the options of the classifier).

When classifying audio files, our pipeline used 1.08 seconds to preprocess and classify 319 windows of audio, when classifying crackles. So not accounting for network latency, it would take little over 1 second to analyze one audio file and report it back through our Web interface.

Our prototype is ready to deploy to a virtual machine. While it could work with any server that have support for *CGI* (*Common Gateway Interface*) scripts, we think that the best solution would be to use a Linux VM using Apache. Assuming that there is no performance change from Windows to Linux (which there may be), what we gain from using a Linux VM instead of a Windows VM is ease of use. Working with CGI scripts within Windows IIS is significantly harder than with Apache. The VM does not need to have high specs, as a 1-2 cores, 4-6 GB of ram and about 80GB of disk space should be sufficient.

An additional feature that our collaborators at the Tromsø 7 have requested is the possibility to export results from our front-end. Since all the data is available both server and client side, we believe this could be integrated in before deployment of the prototype, either as plain text, or in a CSV format.

4

Evaluation

Our methodology for testing our pipeline is divided in two parts, the first part is cross validation of the classifier, using the training data and splitting it into a train-test data set. We evaluated the 5 dimensional feature vector (See section Summary in Methods), both with and without the MFCC vectors.

4.1 Crackle Results

We first tested each feature type individually to see which has the highest indication of separation between classes. Each of the features was tested by running a train-validate cycle 100 times, and then averaging the F1-score across all cycles. The train-validate split is done on the training set which contains only windows. We have collected 175 crackle windows and 208 normal windows. Each cycle splits the training set into 70% training and 30% validation, then each training cycle runs a grid search to tune parameters. It is important to note that parameter tuning through *Grid Search* requires an internal cross-validation to score classifiers for each of the parameter combinations. The grid search is a 3-fold cross-validation on the 70% used for training and after the best-effort parameter tuning is performed, the classifier is refitted with the whole 70% of the training data. When the classifier is trained, it is then validated by predicting the last 30% of the training set, or the validation part, for which the F1 score, precision and recall is calculated and the whole process is then repeated.

Precision, or Positive Predictive Value, is the measure of how many samples labelled as positive, where true positives. A high precision shows how well the classifier are able to correctly label data. Recall (also referred to as *Sensitivity*) is the amount of positive samples that the classifier recognized, that is the true positive rate. Recall shows how sensitive the classifier is to positive samples, and how many positive samples we can expect the classifier to miss. The F1-score is the *harmonic mean* between the two preceding measurements, and gives an indication of overall classification accuracy.

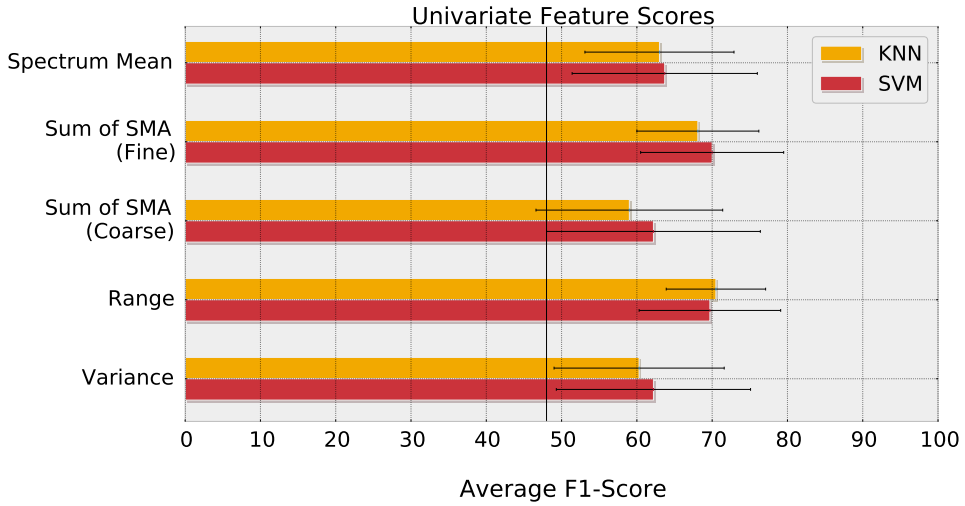


Figure 4.1: Average F1 scores for individual features, vertical line marks the random guess F1-score.

We can see that most of the F1-scores for each feature scores between 60% - 70%, which is not a great score, but it is better than random guess. So theoretically we should be able to combine these features to get a better separation in a higher dimensional space. The Scatter Matrix, Figure 4.3, gives an overview of how much separation there are between the normal and crackle classes. While there is separation between classes, there is also overlap between the two classes. The results from the Linear SVM (Table 4.1) supports this observation, since it shows a high precision, but low recall due to the overlap of samples, and not an obvious linear separation.

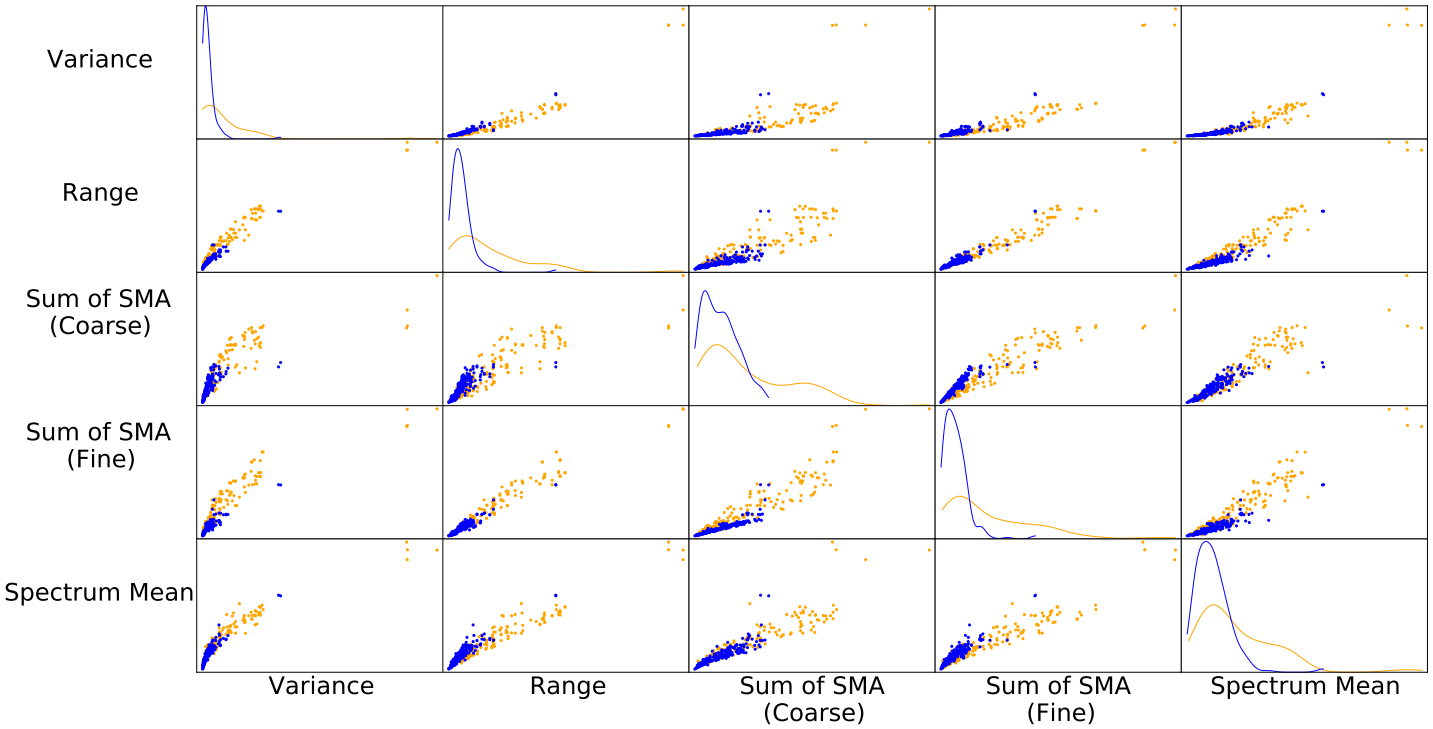


Figure 4.2: A scatter matrix of the 5 feature dimensions of the training data. Blue points are Normal samples and Yellow points are Crackle samples. The Diagonal shows a Gaussian Kernel Density estimation

After testing each individual feature on its own, we also ran a cross-validation cycle using all features. All classifiers are compared to a dummy classifier which votes according to a given strategy. We chose to compare with the *Stratified* voting strategy, where the classifier votes proportional to the number of samples in each class.

Table 4.1: Comparison of classifier performance on crackles

Classifier	Precision	Recall	F1-Score
SVM (RBF)	85.6 ±6.1	83.6 ±10.2	83.5 ±3.6
KNN	84.4 ±6.9	82.3 ±11.3	82.5 ±4.7
AdaBoost (Decision Tree)	82.7 ±5.2	81.8 ±8.3	81.9 ±4.5
Linear SVM	88.7 ±6.7	67.3 ±8.6	76.0 ±5.4
Dummy Classifier (Stratified)	49.7 ±7.7	49.7 ±7.4	49.5 ±6.8

From the table above we can see that on average, the SVM is the best performing classifier. And running grid search for each cycle, the SVM classifier tends to favor a *Radial Basis Function Kernel* and a *C* parameter between 1000 and 2000. So extrapolating from our experiments we believe the SVM is our best choice.

After selecting the SVM as our best performing classifier, we evaluated it on 270 full sized audio files, where a file would be classified as having a significant amount of crackles if the number of positive windows surpasses 30. Out of these audio files, there are 23 containing crackles and 247 containing normal breathing, so a very uneven distribution between classes. As we can see from the confusion matrix (Table 4.2), even with a very high threshold, we have a Recall of 60,8%, a Precision of 18.6% and an F1-score of 6.3%. The results are significantly lower than on individual windows, we discuss the possible causes and solutions to this in the next section (Discussion).

Table 4.2: Confusion Matrix in classifying full audio files

	Crackles	Normal
Crackles	14	9
Normal	61	186

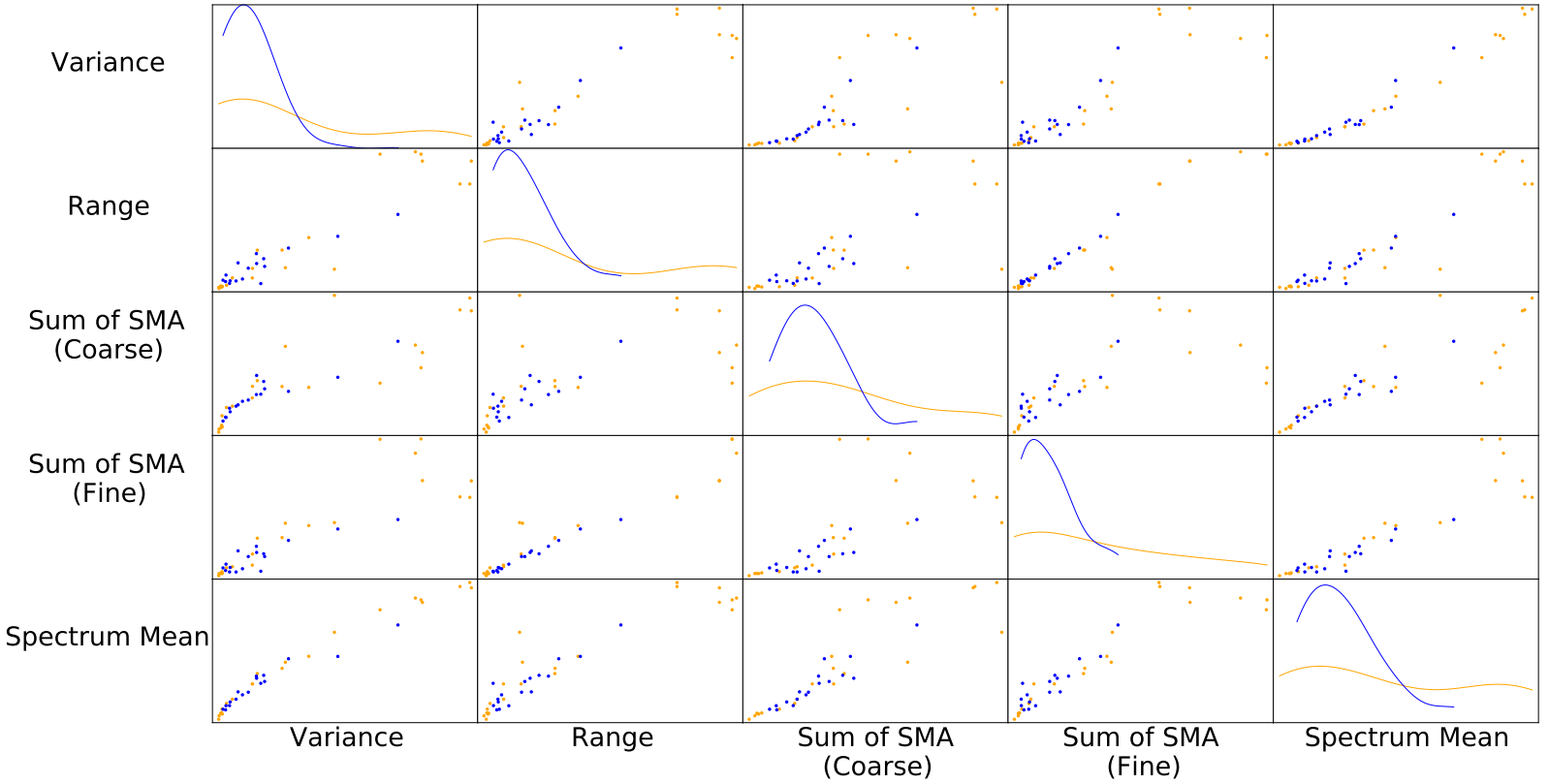


Figure 4.3: A scatter matrix of the 5 feature dimensions of the training data. Blue points are Normal samples and Red points are Wheeze samples. The Diagonal shows a Gaussian Kernel Density estimation

4.2 Wheeze Results

We also tested the features with the preliminary training set for wheezes to get an overview of how well the features perform for wheezes as well.

We ran the evaluations with an SVM, since this was the best performing classifier for crackles. The evaluations were performed in the same manner as with crackles; running a train-validate cycle 100 times with different training set folds. Due to having few wheeze samples in the training set, the results might not be as conclusive as for crackles.

Table 4.3: Results with an SVM using an RBF kernel

Classifier	Precision	Recall	F1-Score
SVM (RBF)	64.8 ±19.0	74.3 ±20.8	63.7 ±24.8

The results vary a lot, and the scores are generally not much better than random guess. We believe that this is due to the nature of wheezes and how they differ from crackles.

4.3 Front End

While we have not conducted a formal user study, the design of the front end is based on inputs from our collaborators at the Tromsø study. The front end works as intended to our knowledge, and the GUI itself is responsive as well. We have not done any load testing for the front end yet.

4.4 Discussion

While our results for crackles are generally good on a per window basis, we have not seen the same results for wheezes, which we believe is due to the fundamental differences in both duration and characteristics between the two abnormal sounds.

4.4.1 Class Imbalance

In practice, only about 5% of all audio data contains abnormal lung sounds. This presents a natural class imbalance between abnormalities and normal breathing. Since we divide input audio files into 300 windows for 15 seconds of audio in crackle analysis and 8 windows in wheeze analysis. From our

results we have achieved a F1-score of 83.5% in cross validation. Which is an acceptable result on its own, but looking at the analysis pipeline, as many as 300 windows may be classified individually for each audio file. This means that after 100 windows, there is a high chance that some windows will be false positives. So to accurately predict the contents of an audio file, without having a high threshold for amount of crackles, the F1-score of window classification would have to be close to 100%.

4.4.2 Correlation

The way our pipeline performs classification, we treat the windows as independent. The pipeline sees the 319 windows from an audio file as individual, independent classification tasks. We believe this is the reason that our experiments on full audio files have not produced the same accuracy that we have seen in cross-validation. This is a weakness of the per-window-classification approach, and we believe that it could be used in conjunction with a meta-analysis either as part of the preprocessing step, or a post classification step.

So as we talked about earlier in the Methods chapter, breathing cycle detection can help to narrow down the area of interest as a preprocessing (or postprocessing) step, and therefore also the number of valid windows to classify. Since we know that crackles and wheezes are related to breathing cycle, correlating the findings of the classifier can be beneficial to classification accuracy.

4.4.3 Wheeze Features

While crackles generally have short duration and a very distinct wave in the time-domain, wheezes does not show the same characteristics. Wheezes contain a lot more information in the spectral domain, with a sound that is reminiscent of whistling, instead of short pops. Wheezes also have a much longer duration than crackles, so each window of for wheeze analysis is 32 times larger than each crackle window. Therefore, we believe that features from the spectral domain will work better for wheezes, compared to the crackle focused features that we have used in our approach. Wheezes have a higher amount of sustained energy in the lower frequency spectrum, and one approach could be to correlate time and frequency to compare energy densities. We believe that Parzen windows or other kernel density estimations is one approach that we could test in future work.

5

Related Work

Machine learning has become a highly popular field, due to the increasing power of modern computer hardware, and the decreasing cost of the aforementioned hardware. With more computing power, both scientific communities and corporations are able to model much more complex decision-support and recommendation systems. While machine learning algorithms such as K-Nearest Neighbor does not require a high amount of time to train, more complex classifiers such as SVMs, ensemble methods and deep networks. With lots of dedicated hardware, the training time can be reduced and each model optimized, and then the best model can be chosen and deployed on smaller hardware such as mobile phones.

5.1 Echonest

Echonest is an example of music recommender systems that use features and machine learning methods to determine and sort different genres of music. This sorting is based on simple features such as energy, loudness, tempo, dancability etc. Genres that have similar features are clustered closer together, enabling recommendations based on distance. In their research they have come up with a way of *fingerprinting* songs. [Schindler and Rauber, 2012]

5.2 Deep Neural Networks & Deep Learning

The first research that modelled Artificial Neural Networks (ANN) was from Warren McCulloch and Walter Pitts in 1943, with their paper *A Logical Calculus of Ideas Immanent in Nervous Activity*. [McCulloch and Pitts, 1943] Though there were some research into ANNs through the 1950s and 1960s, limited computing power prevented experimentation with large ANNs. Nearly 15 years later with the invention of the *Backpropagation* algorithm, research into ANNs became popular again. However, ANNs gave away to simpler classifiers such as SVMs, which outperformed ANNs in both accuracy and training time. The way that ANNs worked was using simple *Perceptron Units* in one or two hidden layers and using weighted connections to an input and an output layer (Figure 5.1). Running networks with more hidden layers was usually infeasible, again due to limited computational power.

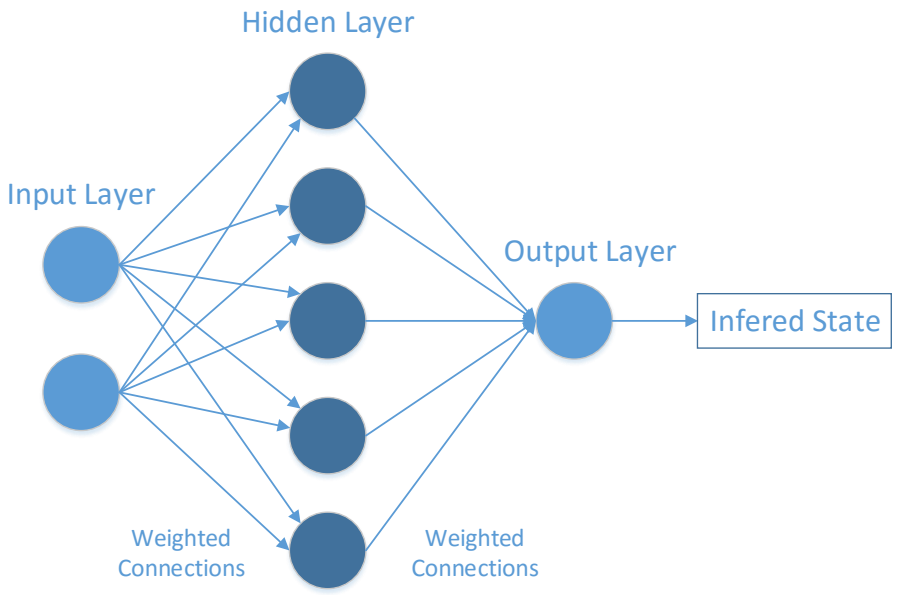


Figure 5.1: Artificial Neural Network

Now in the 21st century, research into ANNs have again become popular, but in the form of *Deep (Neural) Networks* and *Deep Learning*. Deep Learning is a technique of hierarchical machine learning using multiple layers of non-linear processing. One of the successful approaches to Deep Learning have been with Deep Networks, which have become a re-branding or buzzword for Artificial Neural Networks. Deep Neural Networks are basically ANNs with multiple hidden layers, which presents the opportunity of creating more complex models

of non-linear structures, but also increases the time and space complexity of training models in the same way ANNs were limited by in earlier research. The reason optimization problems in Deep Neural Networks have a high time complexity is due to its iterative nature in training.

5.2.1 AlphaGo

Recent popularity in Deep Learning, and Google (Deep Mind) spearheading the research with their AlphaGo program, it seems that many future classification and recommender tasks may be solved by Deep Learning.[Wang et al.,] AlphaGo's application in playing the game of Go, may be reminiscent of IBMs Deep Blue Chess Computer, that beat the reigning champion Gerry Kasparov in 1997.[Campbell et al., 2002] There is a key difference from the Deep Blue breakthrough in 1997 and the win that we saw from AlphaGo this year, and it is the general complexity of the problem. While Deep Blue had the opportunity to calculate a number of moves and choosing an optimal solution, this type of brute force search would be infeasible for a game such as Go. A full sized Go playing board has a game-tree complexity of approximately 10^{360}. So the way that AlphaGo were able to play Go, was to learn from a database of Go games, imitating other players, then playing against earlier generations of itself. It uses a Monte Carlo tree search to find moves based on the knowledge it had learned through its training process. What this means is that the program has a general intuition of the game it is playing.

Deep Mind researchers have stated that the AlphaGo can learn a number of processes other than games as well. This is where the interest from the field of medicine comes into play, Zhang et al. have presented an idea of applying AlphaGo to problems in medicine. [Zhang, 2016] A lot of problems in medicine, such as lung sounds, often requires experience and intuition that is not present in rule-based computer systems.

5.2.2 Distributed Deep Networks

One of the ways that AlphaGo can be trained within a reasonable amount of time, is the architectural and methodical advances that have made in distributing machine learning problems. In his paper *Large Scale Distributed Deep Networks* [Dean et al., 2012], Jeff Dean et al. lays out the ground work for distributing deep networks across a cluster of nodes, and facilitating training using two approaches, *Downpour SGD* and *Sandblaster L-BFGS*. Andrew Ng and Jeff Dean have also applied neural networks in unsupervised learning to recognize higher-level concepts from YouTube videos. [Le, 2013] Geoffrey Hinton, who is often named as the *Godfather of Deep Learning*, have worked on Neural Networks for

decades, and have published many advances in training [Hinton et al., 2006] and applications of Deep Networks. [Krizhevsky et al., 2012]

Using Deep Auto-encoders together with a Deep Network have shown great promise in the field of speech recognition, one major advantage is that it takes away the need for hand crafted feature engineering. [Deng et al., 2010]

5.3 Lung Sound

With all the research going into Deep Learning, and the advances that are being made, it seems that it might become a new standard for learning and modelling non-linear problems, where large amounts of data is available. Our dataset consists of close to 18,000 individual recordings of lung sounds, although we have only worked on a much smaller subset of the data to create our model, a Deep Learning approach might actually be feasible with the full dataset available.

We see that many of the related works on lung sound classification have achieved good results, but do not fulfil all the challenges we believe is important for creating a feasible approach that can be used to support researchers and employees in the medical fields.

5.4 Eko Devices

The company Eko Devices have created a Bluetooth connected stethoscope, their Eko Core device, with an accompanying iOS app, for which they have submitted a pending patent. [Wong, 2015] Their mobile app facilitates recording from the Bluetooth connection and it can push audio to their cloud based storage. Another side of their app is related to telemedicine, in the form of live streaming audio to physicians.

While they have some form of comparison to a lung sound database, which is stated as one of the last claims of their patent application, there is no detailed description of it. However, this claim is one of the last of their patent, and they do not emphasize that this is one of their core claims. Furthermore, with devices such as the Eko Core, there are possibilities to deploy solutions to mobile devices and that our approach can be used in conjunction with such devices.

5.5 Thesis, MiT

In the thesis *A Framework for Automated Heart and Lung Sound Analysis Using a Mobile Telemedicine Platform*, Katherine L. Kuan focuses on a diagnosis system that includes lung sound analysis. They have created an end system that can be deployed with low costs. To our knowledge, they are in a testing phase of their system and that they have an increasing database of lung sounds.

6

Conclusion

In this thesis we presented and motivated our approach for automatic classification of lung sounds. Based on the large dataset and gold standard from the Tromsø 7 project we have implemented a pipeline utilizing signal processing techniques and machine learning to classify abnormal lung sounds in audio recorded from lungs. We have created an interface for our classification pipeline, and a pipeline that is fast enough to perform real time classification of data.

We cross-validated our approach, and we showed that we have viable results in classifying individual windows for crackles, and that we need another set of features for wheezes. However, the results are not accurate enough per audio file, but we know that abnormal sounds are correlated with respiratory phases, and that we plan to use this in future work to increase the accuracy in classification of audio files.

Together with Future Work (next section), we believe that we could develop a core technology for use in mobile devices and medical equipment. Some of the use cases that we have mentioned involves self-monitoring by patients, automatic adjustment in medical equipment such as respirators, training tools for medical professionals and students.

6.1 Future Work

We want to implement a meta-analysis that utilizes our implemented approach to detect crackles and wheezes in correlation with the breathing cycles found in the audio files. We believe that this may further increase the accuracy of our approach. Applying deep learning to the problem could also be an interesting approach, since this have been theorized to be the new standard in Machine Learning. Convolutional Neural Networks have had great success in image classification, and work by applying different transforms to an image, could have some applications in audio as well. Classifying spectrogram images is also an approach which could be explored.

Convolutional Neural Networks would also save a lot of work in feature engineering, as the feature selection process itself takes place within the network in the form of the different kernel convolution. While Deep Networks and Deep Learning techniques can handle a large number of dimensions in input data, and can train accurate models often using only raw data as input, for instance all pixels of an image. They also require a large amount of training samples to achieve this. Up to now, the gold standard has about 2500 audio files, which means applying Deep Learning to lung sounds may become viable in the near future. However, the class imbalance mentioned in discussion means that there are few positive samples compared to negative.

With our pipeline, other types of sounds could be integrated as well. Heart sounds could potentially be integrated in our pipeline, as well as annotation of respiratory phase.

Bibliography

[Aviles-Solis et al., 2015] Aviles-Solis, J., Halvorsen, P., and Melbye, H. (2015). Inter-observer variation in categorizing lung sounds. *St. Petersburg Electrotechnical University "LETI" St. Petersburg, Russia 24th–25th September 2015*, page 11.

[Bellman, 1957] Bellman, R. (1957). Dynamic programming princeton university press. *Princeton, NJ*.

[Bergstra et al., 2011] Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., Desjardins, G., Warde-Farley, D., Goodfellow, I., Bergeron, A., et al. (2011). Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*.

[Bergstra et al., 2010] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

[Campbell et al., 2002] Campbell, M., Hoane, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1):57–83.

[Dean et al., 2012] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. (2012). Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231.

[Deng et al., 2010] Deng, L., Seltzer, M. L., Yu, D., Acero, A., Mohamed, A.-R., and Hinton, G. E. (2010). Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech*, pages 1692–1695. Citeseer.

[Ding et al., 2008] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E. (2008). Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552.

[Domingos, 2012] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87.

[Forgacs, 1978] Forgacs, P. (1978). The functional basis of pulmonary sounds. *CHEST Journal*, 73(3):399–405.

[Grønnesby, 2015] Grønnesby, M. (2015). Pulmonary crackle detection using signal processing and machine learning. *Capstone Project in Computer Science, University of Tromsø*.

[Gupta et al., 2016] Gupta, O. P. et al. (2016). The stethoscope: The iconic medical tool. *Journal of Mahatma Gandhi Institute of Medical Sciences*, 21(1):6.

[Gurung et al., 2011] Gurung, A., Scrafford, C. G., Tielsch, J. M., Levine, O. S., and Checkley, W. (2011). Computerized lung sound analysis as diagnostic aid for the detection of abnormal lung sounds: a systematic review and meta-analysis. *Respiratory medicine*, 105(9):1396–1403.

[Hasan et al., 2004] Hasan, M. R., Jamil, M., and Rahman, M. G. R. M. S. (2004). Speaker identification using mel frequency cepstral coefficients. *variations*, 1:4.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[Jacobsen et al., 2012] Jacobsen, B. K., Eggen, A. E., Mathiesen, E. B., Wilsgaard, T., and Njølstad, I. (2012). Cohort profile: the tromsø study. *International journal of epidemiology*, 41(4):961–967.

[Jordan and Mitchell, 2015] Jordan, M. and Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

[Kanchanaraksa, 2008] Kanchanaraksa, S. (2008). Evaluation of diagnostic and screening tests: validity and reliability.

[Keogh and Mueen, 2011] Keogh, E. and Mueen, A. (2011). Curse of dimensionality. In *Encyclopedia of Machine Learning*, pages 257–258. Springer.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kuan, 2010] Kuan, K. L. (2010). *A framework for automated heart and lung sound analysis using a mobile telemedicine platform*. PhD thesis, Massachusetts Institute of Technology.

[Le, 2013] Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE.

[Logan et al., 2000] Logan, B. et al. (2000). Mel frequency cepstral coefficients for music modeling. In *ISMIR*.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Meng et al., 2015] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2015). Mllib: Machine learning in apache spark. *arXiv preprint arXiv:1505.06807*.

[Moreland and Truemper, 2009] Moreland, K. and Truemper, K. (2009). *Machine Learning and Data Mining in Pattern Recognition: 6th International Conference, MLDM 2009, Leipzig, Germany, July 23-25, 2009. Proceedings*, chapter The Needles-in-Haystack Problem, pages 516–524. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Müller, 2007] Müller, M. (2007). Dynamic time warping. *Information retrieval for music and motion*, pages 69–84.

[Murphy, 1981] Murphy, R. L. (1981). Auscultation of the lung: past lessons, future possibilities. *Thorax*, 36(2):99–107.

[Pasterkamp et al., 1997] Pasterkamp, H., Kraman, S. S., and Wodicka, G. R. (1997). Respiratory sounds: advances beyond the stethoscope. *American journal of respiratory and critical care medicine*, 156(3):974–987.

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

[Peterson, 2009] Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, 4(2):1883.

[Sanchez et al., 1993] Sanchez, I., Avital, A., Wong, I., Tal, A., and Pasterkamp, H. (1993). Acoustic vs. spirometric assessment of bronchial responsiveness to methacholine in children. *Pediatric pulmonology*, 15(1):28–35.

[Schapire et al., 1998] Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of statistics*, pages 1651–1686.

[Schindler and Rauber, 2012] Schindler, A. and Rauber, A. (2012). Capturing the temporal domain in echonest features for improved classification effectiveness. In *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation*, pages 214–227. Springer.

[Sovijarvi et al., 2000] Sovijarvi, A., Dalmasso, F., Vanderschoot, J., Malmberg, L., Righini, G., and Stoneman, S. (2000). Definition of terms for applications of respiratory sounds. *European Respiratory Review*, 10(77):597–610.

[Theodoridis and Koutroumbas, 2009] Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition (4th edition)*. Academic Press.

[Wang et al.,] Wang, F.-Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., Zhang, J., and Yang, L. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120.

[Wong, 2015] Wong, V. (2015). Mobile device-based stethoscope system. US Patent App. 14/152,278.

[Yadollahi and Moussavi, 2006] Yadollahi, A. and Moussavi, Z. M. (2006). A robust method for heart sounds localization using lung sounds entropy. *Biomedical Engineering, IEEE Transactions on*, 53(3):497–502.

[Zaharia et al., 2010] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. *HotCloud*, 10:10–10.

[Zhang et al., 2015] Zhang, K., Wang, X., Han, F., and Zhao, H. (2015). The detection of crackles based on mathematical morphology in spectrogram analysis. *Technology and Health Care*, 23(s2).

[Zhang, 2016] Zhang, Z. (2016). When doctors meet with alphago: potential

application of machine learning to clinical medicine. *Annals of Translational Medicine*, 4(6).

Appendix A: Distance Metrics

A few of the most common distance metrics between two vectors $\vec{x}$ and $\vec{y}$ of length n:

Euclidean

$$E(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

Manhattan

$$M(\vec{x}, \vec{y}) = |x_1 - y_1| + |x_2 - y_2| + \ldots + |x_n - y_n|$$

Chebyshev

$$C(\vec{x}, \vec{y}) = max(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|)$$

Mahalanobis

Mahalanobis requires a covariance matrix S and a vector of means μ.

$$Mh(\vec{x}, \vec{y}, S, \vec{\mu}) = \sqrt{((\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu}))}$$

Appendix B: Kernel Functions

A list of common kernel functions used in SVMs

Linear

$$K(x, z) = x^T y + c$$

Radial Basis Function

$$K(x, z) = exp(-\frac{|x - z|^2}{\sigma^2})$$

Polynomial

$$K(x, z) = (x^T < +1)^q, \quad q > 0$$

Sigmoid

$$K(x, z) = tanh(\alpha x^T y + \gamma)$$

Appendix C: Included Source Code

The source code that is included in this thesis is structured in the following way:

```
/src
|
| - /Classifier
  |
  |- /audio
  |- /classifier
  |- /data <- contains the training sets
  |- /preprocessor
  |- AudioSorting.py
  |- Config.py
  |- IOManager.py
  |- Main.py <- set up in metrics mode
|
| - /Frontend-server
  |
  |- /lib <-- pipeline implementation goes here
  |- /static
  |- /templates
  |- /user_data
  |- app.py
```

The source code itself is not open-source, since the University of Tromsø has a commercial interest in this project, but it will be made open-source at a later time.