Estimating the risk of insurance fraud based on tonal analysis

Henrik Steneld

Academic supervisor: Maria Sandsten
Supervisor at Trygg-Hansa: Fredrik Thuring

June 2022

Faculty of Science

Abstract

Insurance companies utilize various methods for identifying claims that are of potential fraudulent nature. With the ever progressing field of artificial intelligence and machine learning models, great interest can be found within the industry to evaluate the use of new methods that may arise as a result of new advanced models in combination with the rich data that is being gathered. For this end, we decided to evaluate a Long Short-Term Memory (LSTM) - as well as a residual (ResNet) type of neural network, with the purpose of estimating the risk of insurance fraud based on acoustic properties of conversations between customers and company representatives.

Furthermore, we drew a connection between identifying conversations that regard fraudulent claims and detecting deceptive speech. With this connection in mind, we simulated data representing deceptive speech by artificially altering the pitch and used it to evaluate four types of acoustic features: Filter bank energies, cepstral coefficients, mel-frequency filter bank energies, and mel-frequency cepstral coefficients (MFCC).

We found that a LSTM model could be viable with either feature tried. Additionally, we found that the filter bank energies yielded the best performance and it did so on the grounds of having been computed over a multitaper spectrogram.

We did not find any combination of model and feature that could generalize results from training data onto data used for validation with respect to real conversations between customers and company representatives.

Popular science description

Trying to identify insurance claims that are invalid or even fraudulent is of great importance for insurance companies. Actually succeeding in doing so may not always be straight forward. Insurance companies use different methods for investigating claims that might be of fraudulent nature, some of which includes speaking directly to the customer. Some may argue that it would be considered valuable for the insurance companies to be assisted in such conversations by artificial intelligence that may be able to hint at whether or not what is being said is typical for fraudulent claims. Such an artificial intelligence could of course be considered beneficial within related areas as well, such as in criminal investigations.

The idea with this project is to look into the possibility of constructing a model that can be used for the aforementioned purpose. We look at fields that we consider related to the task, namely speaker recognition and lie detection. The reason for considering speaker recognition is that we are taking the approach of modelling acoustic properties of the customers speech, as opposed to the content of what is being said. From science on lie detection we know that when someone is being deceptive, the pitch in their voice is increasing. This knowledge is used in the sense that we therefore simulate pitch alterations in ordinary speech and evaluate various types of acoustic features as input to models trying to identify the simulated alterations.

Acknowledgement

I would like to thank my supervisor at Lund University, Maria Sandsten, for all the help given throughout this project, it has been incredibly valuable. Also, I would like to thank Trygg-Hansa and especially Fredrik Thuring who in addition to having originally come up with the idea of this project has been supportive and providing all necessary means for completing it.

Contents

Introduction

The aim of this project was to assess the validity of information given by an insurance customer throughout his or her telephone conversation with a representative of that company. It was proposed that tonal analysis of the conversation can determine this validity. We had access to data corresponding to 1324 unique telephone conversations between customers and company representatives where the matter of an insurance claim had been discussed. Furthermore, the data was labeled as being determined to be related to either legit, or fraudulent claims. In order to achieve success in assessing the validity based on tonal analysis, it was proposed to utilize science from the field of speaker recognition, from where candidate acoustic features were deduced. Furthermore, it was proposed that a Long Short-Term Memory (LSTM) type of neural network would be a viable model. In addition to a LSTM model, we also implemented and tested a state of the art residual network (ResNet).

1.1 Problem formulation

We defined the waveform of a relevant audio stream consisting of a telephone conversation between a customer and an insurance company as a stochastic process denoted X, and the correct label for a corresponding audio stream X was defined as Y,

$$Y = \begin{cases} 1 & \text{fraudulent claim} \\ 0 & \text{legit claim} \end{cases} \tag{1.1}$$

The goal was formulated as estimating p,

$$p = E(Y|X) \tag{1.2}$$

utilizing a LSTM- or ResNet type of neural network.

1.2 Approach

To make inference on p, we sought to utilize the data at hand, which was 1324 examples of pairs (X, Y) provided by an insurance company. We recognized that it was likely not feasible to consider X in its entirety. Such data as input to either LSTM- or ResNet model would be to large with respect to the number of time steps. Heavy down-sampling was not considered either as, motivated by the Nyquist-Shannon sampling theorem, we would loose possibly important information laying in high frequencies. Instead we utilized results from within the field of speaker recognition.

Speaker recognition is the field of modelling acoustic properties of the speaker as opposed to the content of what is being said, i.e modelling vocal tract dynamics. In typical practice, features are extracted from audio and models are built based on them. There is no shortage in proposed type of features to extract, and new types are typically being found as tweaks to established ones.

We wanted to select a reasonable type of feature for the task of making inference on p. In order to facilitate the selection, we decided to first work with simulated data. By generating data and manipulating it in a way deemed typical for conversations regarding fraudulent claims, we could evaluate a set of feature candidates before modelling the data set provided by the company.

In order to determine how to simulate reasonable data, we looked at a problem that was proposedly related. Namely that of identifying when someone is being deceptive. We looked at previous work that had been concerned with tonal properties of people being deceptive and we found indications that when someone is being deceptive, the pitch in their voice increase (Levitan et al., 2018; Taylor and Hick, 2007; Villar et al., 2013; Vrij and Semin, 1996). With this in mind, we generated data upon which we artificially altered the pitch upwards in irregular sized time intervals. Then, we evaluated different types of features as input to a LSTM type of neural network for the purpose of identifying data points that had been altered in pitch. By first working with simulated data we also had the option to verify that a LSTM model was a viable selection. Had the LSTM model failed compared to a naïve model on a data set that would clearly be easier to model, it would hardly be worth going forward.

Given that we had information about how the data from the company was composed, namely as telephone conversations, it was speculated that we could improve an estimate of p by considering certain grounds upon which features were computed. We looked at different type of spectrograms and band-frequencies.

Before trying out either the LSTM- or the ResNet model on data provided by the company, we performed pre-processing as means of data simplification. Suppose that the audio stream X could be separated into three tracks such that

$$X = \left(X^{customer} \cup X^{agent} \cup X^{silence}\right) \tag{1.3}$$

Then, it was proposed that only considering features extracted from $X^{customer}$ would be preferable in relation to considering features extracted from X. The argument for removing silent parts could be strengthened by results in e.g.

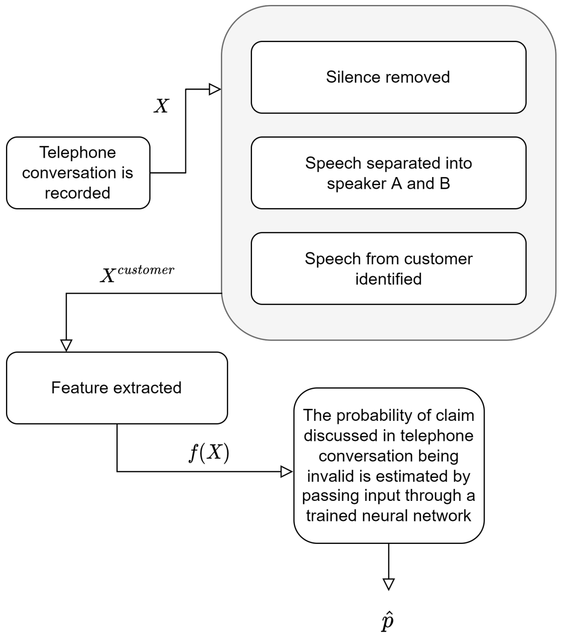
Telephone conversation is recorded
X
Silence removed
Speech separated into speaker A and B
Speech from customer identified
$X^{customer}$
Feature extracted
$f(X)$
The probability of claim discussed in telephone conversation being invalid is estimated by passing input through a trained neural network
$\hat{p}$

Figure 1.1: An overview of the complete process, given a recording X of a telephone conversation between a customer and a company representative, we would like to estimate $p = E(Y|X)$, the probability that the claim being discussed is fraudulent.

(Franklin Nissy & Renisha, 2020).

Finally, the complete process, from audio being generated to p being estimated is given in Figure 1.1 .

1.3 Structure

It was decided to divide this report into three parts. In the first part, we detail the work on audio with simulated pitch alterations. Such pitch alterations are described in Section 3.2. In the second and third part, we look at the data provided by the company consisting of real telephone conversations. The second part is dedicated to pre-processing the data, in particular with speaker diarizaion (identifying who spoke at what time). The third part regards modelling of p given data prepared in the second part and conclusions drawn in the first part. There were especially two conclusions from the simulated data that we hoped would transfer over to the modelling of the data from the company. First, the conclusion of whether or not a LSTM type of model (as described in Section 2.2.1) was adequate at all for identifying pitch alterations. Second, the conclusion of which type of feature that could be extracted from audio that best would encapsulate such alterations, in addition to the basis on which it could be computed on. The essence of feature extraction is detailed in Section 2.1.

1.4 Related work

This thesis is based on research within several areas. In essence, we have focused on the task of identifying deceptiveness (i.e lie detection) as artificially simulated, with the aid of neural networks. As such, we highlight some interesting research that has gone in to the aforementioned fields.

Trying to identify deceptiveness with the use of machines has been a topic of research for a long time. Historically, polygraphs are probably the most well known type of machine for this purpose, having been used frequently by law enforcement for decades. However, it is fairly well known that the efficiency of polygraphs has been heavily questioned by the scientific community. The National Academy of Sciences released a report, National Research Counsil (2003) where they conclude that "Overall, the evidence is scanty and scientifically weak" (p. 212). Nowadays, focus has found its track to more non-invasive methods, such as modelling visual, and verbal cues. Visual cues are typically modelled by means of analysing video content, often accompanied by verbal cues. With respect to verbal cues, there exists work which has proposed certain vocal markers that are linked to deceptive speech. However, we see that it can be hard to conclude the existence of reliable such markers. For example, Sporer and Schwandt (2006) looked into nine markers by analysing 41 previous studies. Out of the nine, they found that pitch and response latency essentially were the only markers that could be reliably associated with deception. With respect to pitch, which plays a rather central role in this thesis, further evidence for it as a reliable marker is found in for example (Levitan et al., 2018) and (Villar et al., 2013). However, there is also evidence for the contrary, i.e that increased pitch is not associated with deceptiveness (Zhang et al., 2022). As such, it is hard to draw viable conclusions other than that the relationship between verbal cues and deceptiveness is a field of interest for many researchers. After all, it is hard to generate or even validate large amount of data where deceptiveness has occurred in a natural fashion.

Plenty of research has gone into neural networks, especially the type of networks that we consider: LSTM and ResNet. In fact, the most cited paper in the 20th century that concerns the architecture of a neural network was the original paper written by Hochreiter and Schmidhuber (1997) detailing the LSTM model (Schmidhuber, 2021). In the 21:st century, the most cited paper that regards a neural network architecture is the paper by He et al. (2016), detailing the ResNet used for winning the ILSVRC classification task (Schmidhuber, 2021). Commonly, when neural networks are implemented for the purpose of detecting fraudulent behavior, it is by means of anomaly detection. For example, a LSTM model has been found to display state of the art performance for fraud detection within healthcare (Snorovikhina & Zaytsev, 2020). Other areas where LSTM networks have been successful in detecting anomalies are for example in ECG time signals (Chauhan and Vig, 2015), and by means of intrusion detection within computer networks (Bontemps et al., 2017).

Method

2.1 Acoustic features

Historically, many different types of acoustic features for speaker recognition has been proposed. Some are found by computations directly on the waveform in the time domain (e.g. Linear predictive codes), while others are computed from the spectrogram in the time-frequency domain (e.g. Mel-frequency cepstral coefficients). The most commonly used feature is likely to be the Mel-frequency cepstral coefficients, which is why we decided to use this as a starting point.

We decided to consider only features that were to be deduced from the spectrogram. Since we considered features deduced from the spectrogram, we sought to evaluate not only features per se, but also how well they performed as input to a model with regards to on what estimate of spectrogram they had been computed. It was proposed that perhaps the result could be improved by estimating the spectrogram either using multitaper techniques or by a parametric approach.

In the following sections we describe the features that we sought to evaluate (Section 2.1.1), the various types of spectrograms considered (Section 2.1.2) and audiobands considered due to the knowledge we had of the data being generated as telephone conversations (Section 2.1.3).

2.1.1 Features

We decided to evaluate four different types of acoustic features as input to the model for the simulated data. Two of which were computed on the ordinary frequency scale, and two of which were computed on the mel-frequency scale. The features that we sought to evaluate on the simulated data was filter bank energies, mel-frequency filter bank energies, cepstral coefficients, and mel-frequency cepstral coefficients (MFCC). The motivation for evaluating MFCC was its popularity in related fields. The reasoning for also evaluating mel-frequency filter bank energies was that they are commonly used as input to neural networks. Mel-frequency filter bank energies are in essence something that has to be calculated in order to retrieve MFCC, where MFCC are formed as a transform. As such, it was speculated that the reason for the superior result in neural networks is that a neural network can learn a suitable transform by itself. This type of speculation was also the motivation for evaluating filter bank energies that are not deduced from the mel-frequency scale. We sought to allow a neural network to learn the scaling in of itself by distributing weights accordingly. Finally, cepstral coefficients are deduced from filter bank energies the same way that MFCC are deduced from mel-frequency filter bank energies, which was the motivation for including them. We detail each type of feature in subsequent sections.

Filter bank energies

Filter bank energies were, out of the four features examined, the simplest. In order to retrieve this feature, we first had to decide the number of filters in the filter bank that were to be computed, as this would be the same as the dimension of the feature vector. A large number of filters would yield a feature vector that is representative of a high resolution on the spectrum. To compute a K_{FBE}-dimensional feature vector at a given instance in time, we applied a filter bank consisting of K_{FBE} evenly spaced band-pass filters of a predetermined shape onto the spectrum. As such, by applying a filter bank to the spectrum, we separated the spectrum into parts, where each part carried information consisting of frequencies at corresponding sub-bands. We decided to use triangular filters due to the similarity with mel-frequency filter bank energies as detailed in the subsequent section. A filter bank with triangular filters is depicted in Figure 2.1. In order to move from an applied filter bank to filter bank energies, we computed the sum of the powers at the frequencies in each band after the filter bank had been applied to the original spectrum. By considering a spectrogram as opposed to a single spectrum, we could by applying a filter bank at each time instance generate a time series consisting of filter bank energies, which was in fact what we sought to retrieve. When we computed filter bank energies, we set $K_{FBE} = 30$. As a final step, we did a log-transform on the complete time series.

Mel-frequency filter bank energies

The mel-frequency scale is a scale that is used in order to correct for the perception of increasing pitch levels. Our hearing is not linear in the sense that an increase in pitch of a fixed amount in the ordinary frequency scale (Hz) will typically not correspond to a perceived increase in pitch that is equal regardless of the initial frequency. The mel-scale tries to fix this. Altering sound by a fixed amount of mels, should ideally imply an altering of a fixed amount in perceived pitch, regardless of initial mels. O'Shaughnessy (1987) described the mel-scale as being somewhat linear between $0 - 1000$ Hz, while being logarithmic above 1000 Hz. There is however no fixed definition of the mel-scale in terms of transformation from the ordinary frequency scale. Historically, there have been proposed mathematical formulas (O'Shaughnessy, 1987; Cassidy & Harrington, 1999; Lindsay & Norman, 1977) as well as there have been deduced

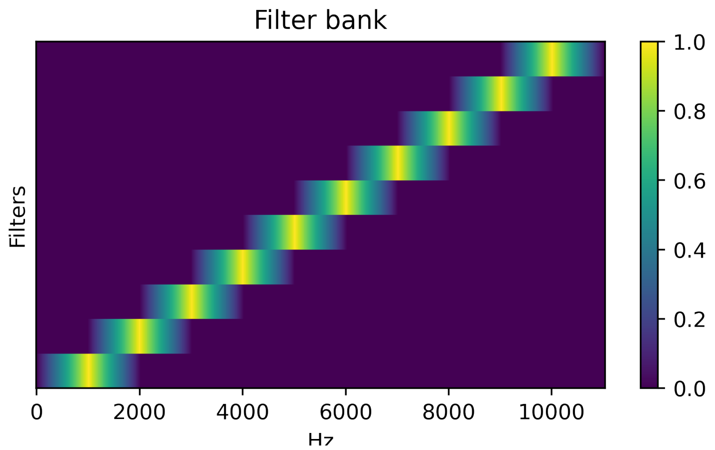

Figure 2.1: Filter bank with triangular filters illustrated. Note that each filter spans an equally large proportion of the frequency axis as does its neighbouring filters. This is in contrast to how a mel-frequency filter bank would look.

tables from acoustic measurements (Baranek, 1949). In order to transform the ordinary frequency scale into the mel-scale, we utilized the package Librosa (version 0.9.1, 2022). Mel-frequency filter bank energies are deduced exactly how the ordinary filter bank energies are deduced with one exception: The K_{MEL} band-pass filters are evenly spaced in the mel scale, but considering the ordinary frequency scale (Hz) they are not. A filter bank of evenly spaced band-pass filters in mel scale is illustrated in Figure 2.2. Traditionally, one uses triangular filters for computing the mel-frequency cepstral coefficients detailed in subsequent section and for that reason, we used band-pass filters of triangular shape when computing mel-frequency filter bank energies. When we computed mel-frequency filter bank energies, we set $K_{MEL} = 30$. Just as in the case for the ordinary filter bank energies, we did a log-transform as a final step.

Cepstral coefficients

Childers et al. (1977) explains that a rather simple model for vocalized speech can be formulated in the time domain as the convolution of signals. First, pulses of air affects the vocal cords which generates glottal pulses that finally reacts on the vocal tract, producing speech as a result. Beigi (2011) defines this type of system in the spectral domain as the product of three components U, G_c and G_v corresponding to: Input from the brain regarding what speech to produce, the nervous system and vocal tract motor control (regarded jointly), and the vocal tract dynamics. As the product of U, G_c and G_v in the spectral domain generate speech H in the spectral domain

$$H = G_v G_c U, \tag{2.1}$$

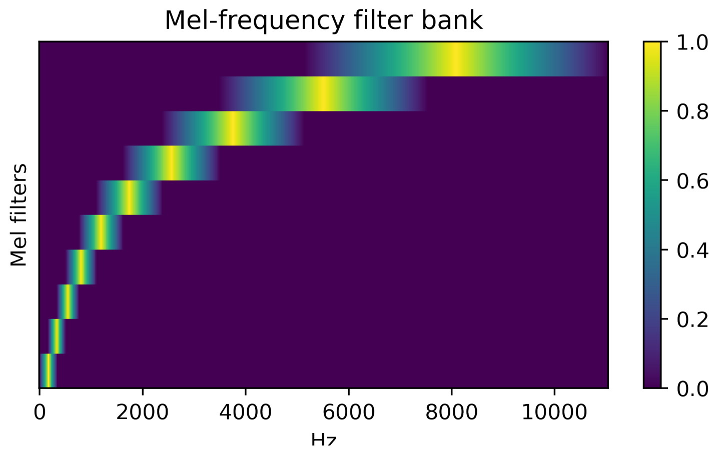

Figure 2.2: Mel-frequency filter bank with triangular filters illustrated. The K_{MEL} band-pass filters are evenly spaced in mel scale, which, as seen in the figure, this is not the case when viewing the ordinary frequency scale (Hz).

we can naturally consider speech in the time domain as well:

$$h(t) = u(t) * g_c(t) * g_v(t) \tag{2.2}$$

In speaker recognition, as separated from speech recognition where interest instead lies in the content of the speech, one is usually interested in modelling the vocal tract dynamics g_v (Beigi, 2011). As such, it is naturally considered beneficial to deconvolve the speech h. This is the main reason for considering the cepstrum as opposed to the spectrum.

A power cepstrum, as first described by Bogert et al (1963) can be computed as the squared inverse Fourier transform of the log-transformed power spectrum (Childers et al., 1977). In the cepstrum, regardless if we consider the squared inverse Fourier transform or just the inverse Fourier transform, the components becomes additive. As we take the log-transform of the power spectrum for the speech we obtain

$$\begin{gathered} \log(H) = \log(G_v G_c U) \\ \Longleftrightarrow \\ \log(H) = \log(G_v) + \log(G_c) + \log(U). \end{gathered} \tag{2.3}$$

Due to the (inverse) Fourier transform being a linear transform, we obtain the power cepstrum denoted $h\hat{(t)}$ as

$$\begin{gathered} \hat{h}(t) = \left(\mathcal{F}^{-1}\left(\log(G_v) + \log(G_c) + \log(U)\right)\right)^2 \\ \Longleftrightarrow \\ \hat{h}(t) = \left(\mathcal{F}^{-1}\log(G_v) + \mathcal{F}^{-1}\log(G_c) + \mathcal{F}^{-1}\log(U)\right)^2 \end{gathered} \tag{2.4}$$

or expressed differently,

$$\hat{h}(t) = (\hat{g}_v(t) + \hat{g}_c(t) + \hat{u}(t))^2 \tag{2.5}$$

Finally, dropping the squared value used in the original definition by Bogert et al. (1963) and denoting such a cepstrum of h as $\tilde{h}$, we arrive at the relation

$$\begin{gathered} h(t) = u(t) * g_c(t) * g_v(t) \\ \Longrightarrow \\ \tilde{h}(t) = \tilde{g_v}(t) + \tilde{g_c}(t) + \tilde{u}(t) \end{gathered} \tag{2.6}$$

By focusing on the vocal tract dynamics and treating everything else as noise, the cepstrum allows for noise to be assumed additive.

Cepstral coefficients are what we call the frames that arise by taking an inverse Fourier transform of the log-transformed filter bank energies. Note, would the filter bank energies have been computed on non-overlapping rectangular (boxcar) windows, they would effectively be a resolution reduction of the power spectrum, meaning that cepstral coefficients as we defined them would effectively be a low-resolution cepstrum. When computing coefficients from filter bank energies, one typically keep only the first few resulting coefficients. We decided to keep the first 13 coefficients, meaning that computing cepstral coefficients doubled as a dimensionality reduction of filter bank energies.

Mel-frequency cepstral coefficients (MFCC)

Mel-frequency cepstral coefficients are to mel-frequency filter bank energies what cepstral coefficients are to filter bank energies. They are the inverse Fourier transform of the mel-frequency filter bank energies. Typically, as is how we decided to implement MFCC, one can use an inverse discrete cosine transform instead of an inverse Fourier transform (Beigi, 2011). Just as for the cepstral coefficients, when computing MFCC from mel-frequency filter bank energies, we discarded some coefficients, keeping only the first 13.

2.1.2 Spectrogram

Recall that we sought to evaluate different estimates of spectrograms for extracting features in relation to identifying pitch altered audio. We define three spectrograms estimated in different ways as S_B, S_{MT} and S_{AR}. They correspond to a baseline, a multitaper spectrogram and a spectrogram estimated under parametric assumptions. To begin detailing each spectrograms considered, we first define an ordinary spectrogram $S(t, f)$ with a general window that is to be used for deducing S_B, S_{MT} and S_{AR}. We do this by first considering a discrete-time short-time Fourier transform (STFT) as defined by

$$W_h(t, f) = \sum_{t_i=0}^{t_N-1} w_{t_i} h^*(t_i - t + M/2) e^{-i2\pi t_i f} \tag{2.7}$$

where w is the input signal (in our case the audio in its waveform) sampled at discrete instances in time with sampling distance T such that $t_i - t_{i-1} = T$. $f = 0, \frac{F_s}{L}, \frac{2F_s}{L}, \ldots$ and L is the number of frequencies considered. $F_s = 1/T$ is the sample frequency and h is the window function of length M centered at t. It should be noted that in order to utilize the efficient algorithm that is the fast Fourier transform (FFT) appropriately, L should be chosen so that $L = 2^I$, where I is any integer (Sandsten, 2020). From the discrete-time short-time Fourier transform we complete the definition of the ordinary spectrogram as

$$S_h(t, f) = |W_h(t, f)|^2 \tag{2.8}$$

Baseline

We defined S_B to be the ordinary spectrogram S_h with h being a Hamming window, a window function first proposed by Richard W. Hamming which is the most popular window in the field of speech processing (Beigi, 2011). The window was of a length corresponding to approximately $20ms$. The reason for considering STFT's of $20ms$ was that the waveform generated by speech is generally considered to be approximately stationary in such time frames. Furthermore, we computed S_B at a time resolution of approximately $10ms$, meaning that we had a 50% overlap of the windows. The discrete-time STFT and the Hamming window was computed in Python using the package SciPy.

Multitaper

Next we considered instead S_{MT} as an alternative estimate of the spectrogram utilizing multitapering techniques. A multitaper spectrogram is computed by taking an (possibly weighted) average over K_{MT} ordinary spectrograms, where each of the K_{MT} ordinary spectrograms are computed using different window functions h_k (known as tapers in multitaper context),

$$S_{MT}(t, f) = \sum_{k=1}^{K} \lambda_k S_{h_k}(t, f) \tag{2.9}$$

Common choices for K_{MT} lies in the range $2 - 10$ (Sandsten, 2020). For S_{MT}, we chose to consider $K_{MT} = 7$. We used Slepian functions as window functions. Slepian functions are also known as discrete prolate spheroidal sequences (DPSS) or Thomson multitapers (Sandsten, 2020). The windows h_k, $k = 1, \ldots, K_{MT}$ was retrieved again using the package SciPy in Python. We did not consider weights other than $\lambda_k = 1/K_{MT}$. Just as in the baseline case, we used windows of length corresponding to approximately $20ms$ and computed the multitaper spectrogram S_{MT} over a time grid with resolution approximately $10ms$. The use of multitaper techniques could be motivated by the result found in (Kinnunen et al., 2012) or (Alam et al., 2013).

Parametric

We also looked at a parametric estimate of the spectrogam, which we defined as $S_{AR}(t, f)$. In the parametric set up, we were assuming that the signal (being audio in its waveform) could be estimated over intervals of $20ms$ as an Auto-regressive (AR) model of order 10. This parametric setup was motivated by the fact that telephone communication is performed via linear predictive coding and synthesizing where it is common that a linear code of 10 dimensions is computed. We used the package Librosa (version 0.9.1, 2022) to estimate the AR-coefficients. They were estimated over $20ms$ worth of audio but re-estimated every $10ms$ so that we had a similar time resolution as we had with the baseline and multitaper spectrograms. In essence, we had constructed a 10 dimensional time series over a time grid with resolution $10ms$ where at every time instance we had an estimated AR model. From any estimated AR model we could compute a spectral density and by doing so, at every time point, we could generate the spectrogram S_{AR}. A spectral density R can be computed from an AR(K_{AR}) model with innovation variance σ^2 as

$$R(f) = \frac{\sigma^2}{| \sum_{k=0}^{K_{AR}} a_k e^{-i2\pi f k} |^2} \tag{2.10}$$

Finally, we define $S_{AR}(t, f)$ to be the spectral density $R(f)$ computed from an AR(K_{AR}) model that describes a time series centered at time t.

By means of illustrating the different spectrograms that we had decided to look into, we simulated a time series consisting of 500 time steps originating from an AR(4) model, as defined by

$$x_t = 0.8x_{t-1} - 0.7x_{t-2} + 0.6x_{t-3} - 0.5x_{t-} + \epsilon_t \tag{2.11}$$

with Gaussian innovations

$$\epsilon_t \in N(0, 1) \tag{2.12}$$

and estimated the power spectral density (i.e the spectrogram at a fixed time point). The reason for illustrating the estimated spectral densities over data simulated from a true auto-regressive model is to also illustrate how well a parametric estimate works if the parametric assumption is in fact true. We did not expect audio generated by telephone conversations to take the shape of pure auto-regressive models, although it was proposed that there would be a rather strong resemblance. The result is depicted in Figure 2.3.

2.1.3 Audio bands

In addition to evaluating different features deduced from different spectrograms, we looked into whether or not the full frequency range available should be utilized or not. Most of the data we considered was generated by telephone conversations. In telephone communication sampling occurs at varying intensities dependent on available device and protocol. Furthermore, not all frequencies are

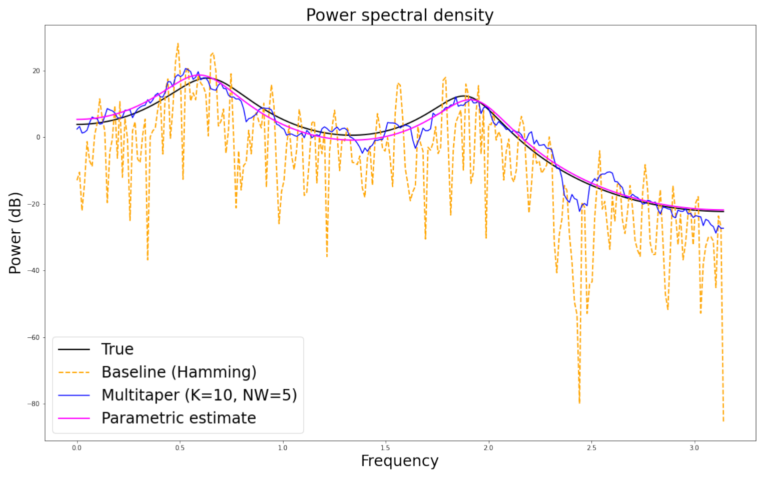

Figure 2.3: Illustration of the differences between the approaches that we considered for estimating the spectrogram. Here, the true power spectral density (spectrogram at fixed instance in time) is depicted along with estimates of different kinds. For the parametric estimate, the AR coefficients were estimated via Yule-Walker equations in Python with statsmodels (version 0.14.0, 2022)

Table 2.1: Different voice bands at which telephone communication tends to take place (Cox et al., 2009). We evaluated features extracted from the audio, considering frequencies limited to the different bands detailed in the table. The idea was that perhaps most information is retained in the lower frequencies and as such it could be beneficial to consider only eg. narrowband range, even though data points are possibly generated from wider bands. If performance loss (in regards to identifying a pitch alteration) was negligible or even negative at narrower bands, it would possibly be beneficial to consider only narrower bands in the data set consisting of real data, which contained telephone conversations generated at unknown voice bands. Performance loss in the widest bands would indicate that either information outside of what is generated through telephone communication is acting as distortion, or that the decrease in frequency resolution is detrimental.

Voice band	**range (Hz)**
Narrowband	$200 - 3400$
Wideband	$50 - 7000$
Superwideband	$50 - 14000$
Fullband	$20 - 20000$

transmitted. For example, wideband ($50 - 7000$ Hz) is recommended[1] for Voice over LTE (VoLTE), while it is demanded by 3GPP to use at least narrowband ($200 - 3400$ Hz). One could consider four bands typically used in telephone communication, all listed in Table 2.1. The idea with evaluating features extracted from different bands was to try and decide if it would be feasible to consider the full range of frequencies, or only parts. For example, given that it is somehow determined that the feature dimension cannot exceed V (due to e.g. complexity or computational cost), using V bands-pass filter energies from the range $200 - 3400$ Hz would mean a higher resolution in the features than by considering V bands-pass filter energies over $20 - 20000$ Hz. As audio in the data set provided by the company was likely to be generated by communication at varying band widths, it was proposed that it furthermore could be beneficial to consider a narrower band for consistency among data points.

2.2 Neural Network

Recall that we looked into estimating p being the probability that an insurance claim is fradulent conditioned on a recording of a telephone conversation. In order to acquire such an estimate we had decided to utilize neural networks. It was proposed that a neural network could learn to detect differences in the input conditioned on the state of Y. We decided to primarily work with a LSTM type of network, but to also implement a version a ResNet model for use on data

[1]AMR-WB is recommended, which is either in 50-6400Hz or in 50-7000Hz

provided by the company.

2.2.1 LSTM Model

Long Short-Term Memory (LSTM) networks are a branch of recurrent neural networks (RNN). An ordinary RNN is rather simple in its idea, and we present the idea in its simplest form with a single hidden unit. Input x_t, $t = 1, \dots, n$, is fed to a hidden unit h_t which is also fed by h_{t-1}. In turn, h_t feeds y_t and h_{t+1}. For x being an input vector of length n, we say that U, V, W are weights that are scalar for $(1, n)$ dimensional input, but d-dimensional vectors for (d, n) dimensional input. Furthermore, y is an output of length n, meaning that for every sequence x, a sequence y of the same length will be yielded as output. We denote an activation function, such as for example the Sigmoid function as ϕ and observe that the complete model can be defined as

$$\begin{cases} y_t = \phi_y(Wh_t) \\ h_t = \phi_h(Ux_t + Vh_{t-1}) \end{cases} \tag{2.13}$$

The simple model with one hidden unit is depicted in Figure 2.4.

With a RNN, It is possible to stack layers of hidden units. When doing so, instead of feeding $h_t^{(0)}$ to $h_{t+1}^{(0)}$ and y_t, one directs the feed to $h_{t+1}^{(0)}$ and $h_t^{(1)}$. To stack multiple hidden units in one layer, proceed just as for an ordinary neural network, i.e by feeding the input directly to all units of the same layer.

If one is interested in a single output from an input sequence (sequence to vector), such as we were when estimating the risk of fraud based on a complete time series, one allows the hidden unit to produce output only to itself and for the final value of y, i.e for y_n.

When training a RNN, it is common to use back propagation through time, which essentially is an algorithm that unravel the RNN and then performs back propagation the same way as is done for an ordinary plain artificial neural network (ANN). When the input consists of a long sequence, i.e for large n, the unraveling of a RNN leads to a very deep model, which in turns means that the gradient is of risk for either ”vanishing” (being suppressed to values close to 0), or to ”explode”. Development of the LSTM model stems primarily from work on suppressing the issue with vanishing or exploding gradient. Hochreiter and Schmidhuber (1997) described in a well-cited paper how a model involving multiplicative gates could partly solve the issues with the gradient. The full model of a LSTM network with memory cells, as Hochreiter and Schmidhuber (1997) called them, is a rather complex model.

The models that we implemented of the LSTM type all had varying number of layers of hidden LSTM units, each connected to the next, conceptually likewise to how a simple RNN has its hidden units connected. At the final LSTM

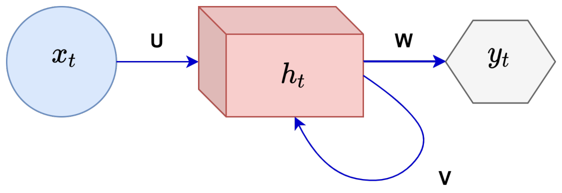

Figure 2.4: Illustration of a basic recurrent neural network with a single hidden unit. For a given instance in time t, the hidden unit h_t produces one output that is fed to both y_t and h_{t+1}. The hidden unit h_t takes both x_t and h_{t-1} as input.

layer, the output was fed to an ordinary ANN, which in turn produced the output that we sought to evaluate. We implemented the model in Python using the package Keras (version 2.7, 2022). Loss function was set to be binary cross entropy. We used Adam as learning rate algorithm, as it is the most commonly used (Amidi & Amidi, n.d).

2.2.2 ResNet model

A residual network (ResNet) is a type of neural network which incorporates residual connections between layers in order to deal with the problem of vanishing gradient. The residual connections can be seen as shortcuts between layers of the network, such that output from one layer is passed on not only directly to the next but also to a layer connected via a residual.

Residual networks gained attention when He et al. (2016) constructed such a network with a depth of no less than 152 layers that won the ILSVRC classification task on the ImageNet test set in 2015. However, residual networks may be used for tasks other than image classification. In fact, one co-author of the original paper that detailed a LSTM network describes the residual network as a feedforward version of a LSTM network (Schmidhuber, 2021), which has its strength primarily on sequential data. Fawaz et al. (2019) demonstrated that a residual network outperforms many other types of neural networks that can be used for classifying time series. Their result shows that out of eight different architectures tried out on both uni-variate and multi-variate time series (for a total of 98 data sets), the residual type of network performs best on average.

We implemented the same architecture that Fawaz et al. (2019) used for their ResNet, with the means of trying it on our data set. The model consisted of three blocks, where each block along with the input had a residual connection to adjacent blocks. Furthermore, each block consisted of three convolution layers and batch normalization. The three convolution layers in the first block had 64 filters, while the layers in the second and third block had 128 filters. In each block, the kernel size was 8, 5 and 3 for the three layers respectively. After the final block there was a layer of global average pooling (GAP) and a single node with Sigmoid as activation function.

Part I: Simulated data

3.1 Introduction

In this part, we looked at the evaluation of a LSTM model in conjunction with various features. We wanted to evaluate the model on the basis of simulated pitch alterations in a fashion that had been reasoned to be typical in conversations regarding insurance claims. As previously explained, such pitch alterations were speculated to take the form of upward alterations at various intensities in irregular time intervals. This was motivated by the proposal that when someone is talking about a fraudulent claim, they are essentially being deceptive in intervals. Focus lay on evaluation of the various features that had been proposed in Section 2.1.1. With respect to the LSTM model per se, we wanted to verify that it performed considerably better than a simple naïve model. Furthermore, the features that were of interest could all be deduced from spectrograms. It had been proposed that prior knowledge of the data, being that it consisted of noisy telephone audio, could be a reason for considering either S_{MT} or S_{AR} as defined in Section 2.1.2 and therefore we also evaluated features computed over these.

3.2 Altering pitch

In order to simulate relevant data we had to perform pitch alterations in a controlled fashion. There exists many different approaches for altering pitch in an audio stream. Many of which are detailed by Royer (2019). In general, common methods are by operations either in time domain, or in frequency domain. One approach that operates in the time domain is to stretch the time interval, and then re-sample so that the original length is preserved. Time stretching could be performed by the use of a phase vocoder, originally described by Flanagan and Golden (1966). By means of simplicity, we implemented an alternative approach also described by Royer (2019), where one instead operates in the frequency domain. The operation is rather simple: Scale the spectrum with a factor determined by the number of semitones to shift so that the shifted

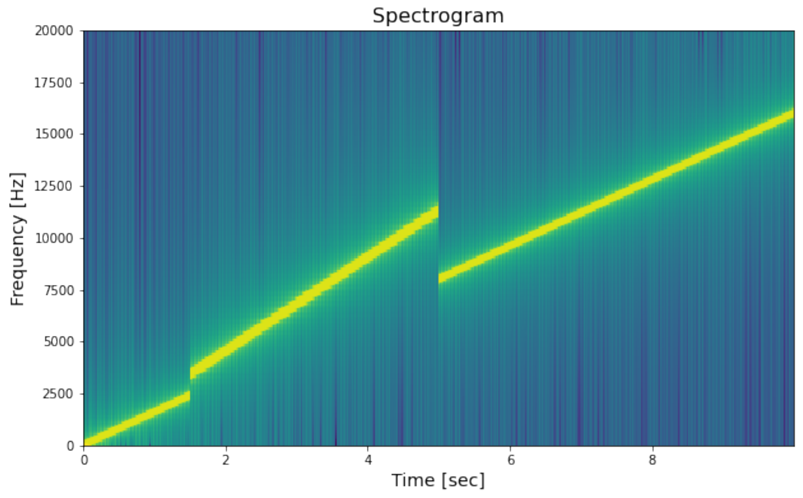

Figure 3.1: The spectrogram of a linear chirp with pitch altered upwards by six semi-tones in the time interval 1.5 - 5 s. Note how the pitch is altered more at higher frequencies. The method for altering pitch was to scale the spectrogram according to $S_{\text{shifted}}(t, f) = S(t, f/\beta), \quad \beta = 2^{6/12}$.

spectrum R_{shifted} is given by

$$R_{\text{shifted}}(f) = R(f/\beta), \quad \beta = 2^{k/12}$$

where k is the number of semitones to shift by. A pitch alteration using this method is illustrated in Figure 3.1.

3.3 Data generation

Pitch alteration was to be performed on some audio that would lay as ground for the simulated data set. For this purpose, 15 minutes of speech was recorded by one person. The recording was done with an iPhone 13 Pro and the speech consisted of reading from a well-known book. The sampling rate was 48 kHz with 16 bits per sample. The audio stream was loaded in Python using the package Librosa (version 0.9.1, 2022). Thereafter the audio stream was cut into 450 clips, two seconds each. Half of the clips, chosen at random, were altered in pitch. For those clips altered, only part of the clip was altered in pitch. The starting point (between $0 - 1$ seconds in) was chosen at random for each clip. So was the end point (minimum 0.25 seconds after start, maximum at full length). The amount of shift that was performed was chosen at random for each clip and it was set to be either one or two semitones up, always shifting up.

Table 3.1: Data sets used in the set up. Original data set constructed by extracting two second clips from 15 minutes of book reading. Half of the resulting data points were at random time instances altered by a random amount in pitch. Data set D_A and D_B are generated by processing the original data set. Feature extraction and evaluation was performed only on processed data sets.

Data set	**Description**
D_0	Original data set constructed by random pitch alterations. Sampling rate at $48kHz$.
D_A	Down sampling of original data. Target sampling rate set to $8kHz$.
D_B	Data generated by recording of telephone conversation where original data set was played back.

From the generated clips, two new data sets were formed. The first data set was formed by having all clips down sampled from 48 kHz to 8 kHz. The second data set was formed by replaying the pitch shifted audio (at 48 kHz) through actual mobile telephones. Note, when replaying audio through telephones we could not be certain at what audioband the audio was transmitted and recieved. However, given that both the network operator and mobile phones used supported VoLTE it was assumed that the band was at least wideband, possibly wider. With regards to the first data set, the reason for down sampling was two folded. First, there are many different methods for shifting pitch, some which preserves acoustic characteristics of the speaker better than other. By heavy down sampling it was hoped that the impact from the choice of pitch shifting method would be attenuated. Second, it seemed reasonable to mimic the sample rate one commonly would find in telephone audio.

The complete data sets were separated into three sets each: Model (50%), Validation (25%) and Test (25%). We denote the data sets as detailed in Table 3.1.

3.4 Evaluation

In order to obtain results that could be interpreted as the tendency of the acoustic features to catch the artificially altered pitch, we looked at the test data set and counted how many correct classifications that were made using respective feature as sole input. The main metric of interest was determined to be the rate of correct classification, known in the context as accuracy. Furthermore, we took note of the loss from the loss function as an additional metric of certainty. Loss would give an insight into the ability of correctly estimating the probability of pitch being altered.

The method for classifying the clips was the same for all features in order to

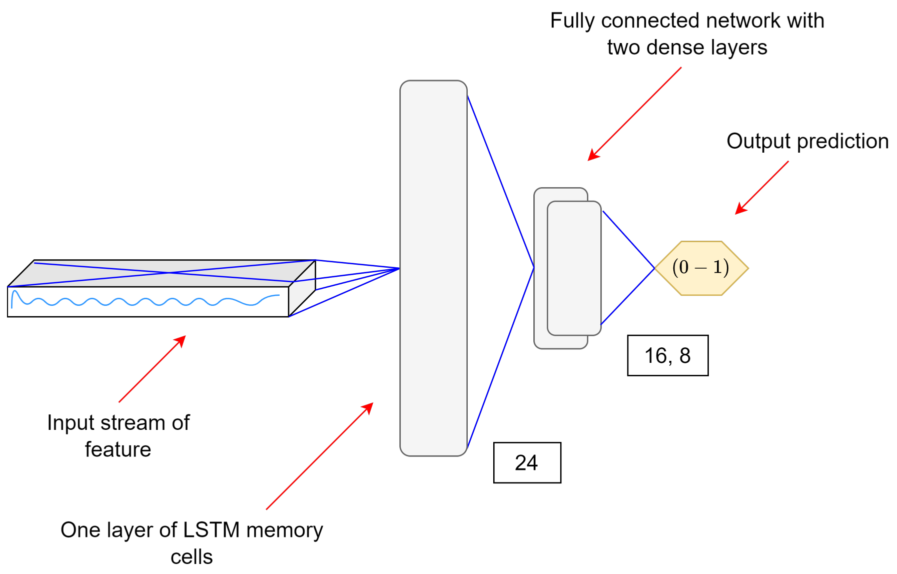
Fully connected network with two dense layers
Output prediction
(0 – 1)
16, 8
24
Input stream of feature
One layer of LSTM memory cells

Figure 3.2: Architecture of neural network. Input is an audio stream of shape $(124, d_X)$, followed by a layer of 24 LSTM memory cells and two dense layers with 16 and 8 nodes respectively. A single dense with Sigmoid activation function as output.

minimize additional influence over the result apart from that yielded by the features themselves. For classification, a LSTM network as described in Section 2.2.1 was designed. The network was kept rather simple in its architecture. It accepted matrices of dimension $(124, d_X)$ as input, where the number of columns d_X was considered dimensions of the acoustic features and each row as the features represented at different instances through time (124 in total for $2s$ audio). First, a layer of 24 LSTM memory cells. Thereafter, two layers with 16 and 8 dense nodes respectively were added for increased complexity. Drop-out was implemented in all layers in order to reduce the tendency for over-training. Drop-out rate was set to 0.2. Early stopping was implemented such that training would not continue if more than 20 epochs consecutively had not yielded any improvement in loss on validation set. The architecture is depicted in Figure 3.2.

3.5 Results

All in all, we evaluated a total of 58 combinations of features, data sets, spectral estimates and frequency bands. The results for D_A is presented in Table 3.2. Results for D_B is presented in Table 3.3.

We note that on data set D_A, we obtained the best result by considering features

Table 3.2: Result of evaluation on data set denoted as D_A. Presented is the accuracy of which the LSTM model managed to correctly classify data points as being either pitch shifted or not. Loss as defined by the binary cross entropy is presented in parenthesis. Features denoted, F_0: Filter band energies, F_1: Mel-frequency filter band energies, F_2: Cepstral coefficients, F_3: Mel-frequency cepstral coefficients (MFCC). Note that all featuers are detailed in section 2.1.1.

Baseline (S_B)		**Multitapering (S_{MT})**	
Feature	**Accuracy (loss)**	**Feature**	**Accuracy (loss)**
F_0	0.96(0.15)	F_0	0.97(0.09)
F_1	0.94(0.19)	F_1	0.96(0.19)
F_2	0.96(0.16)	F_2	0.96(0.12)
F_3	0.96(0.17)	F_3	0.97(0.12)

computed on the multitaper spectrogram. This was prominent for all features where either accuracy, loss, or both was improved from baseline while none of the metrics were worsened. Furthermore, we saw that the best result was found when considering filter bank energies (detailed in Section 2.1.1).

When considering data set D_B, it was not clear which method for estimation of spectrogram performed best. On average, the parametric approach had the worst accuracy and loss out of the three. Considering the baseline and multitaper approach over all four features at all bands, then for the 16 combinations, we found the highest accuracy with baseline eight times and with multitaper spectrogram eight times yielding a tie in such a sense. The accuracy was on average best when considering the narrowest of the bands (narrowband), and worst when considering the widest band (fullband). Furthermore, the variability of the metrics was least for the narrowband.

3.6 Conclusions

First, we concluded that utilizing an LSTM for identifying pitch altered audio is considerably better than leaving it to chance. A naive model, defined as a coin flip (recall that half of the data was pitch altered, the other half not) would by design yield an expected result of 50% correctly classified clips. Even after passing the audio through cellular phones (as in D_B), a LSTM network managed to correctly classify more than 90% of the clips.

We concluded that it was beneficial to consider filter bank energies as acoustic feature. This conclusion was made primarily on the basis of results found for data set D_B.

Results on D_A suggested that it could be beneficial to utilize multitapering techniques, as for the features tried they all showed an improvement in either

Table 3.3: Result of evaluation on data set denoted as D_B. Presented is the accuracy of which the LSTM model managed to correctly classify data points as being either pitch shifted or not. Loss as defined by the binary cross entropy is presented in parenthesis. Features denoted, F_0: Filter band energies, F_1: Mel-frequency filter band energies, F_2: Cepstral coefficients, F_3: Mel-frequency cepstral coefficients (MFCC). Note that all featuers are detailed in section 2.1.1. Result marked as x means that no result within 20 epochs was found.

Baseline (S_B)				
Feature	**Narrowband**	**Wideband**	**Superwideband**	**Fullband**
F_0	0.91(0.28)	0.94(0.24)	0.93(0.27)	0.83(0.40)
F_1	0.93(0.27)	0.86(0.48)	0.90(0.41)	0.89(0.35)
F_2	0.93(0.25)	0.90(0.27)	0.87(0.38)	0.86(0.44)
F_3	0.90(0.25)	0.89(0.60)	0.86(0.43)	0.86(0.35)

Multitapering (S_{MT})				
Feature	**Narrowband**	**Wideband**	**Superwideband**	**Fullband**
F_0	0.91(0.24)	0.92(0.28)	0.94(0.21)	0.83(0.41)
F_1	0.91(0.32)	0.90(0.27)	0.90(0.36)	0.90(0.37)
F_2	0.94(0.19)	0.91(0.48)	0.81(0.55)	0.76(0.48)
F_3	0.89(0.26)	0.84(0.49)	0.87(0.34)	0.81(0.40)

Parametric (S_{AR})				
Feature	**Narrowband**	**Wideband**	**Superwideband**	**Fullband**
F_0	0.79(0.61)	0.81(0.37)	0.81(0.46)	0.76(0.64)
F_1	x	0.90(0.33)	0.89(0.38)	0.89(0.39)
F_2	x	0.94(0.38)	0.81(0.42)	0.77(0.50)
F_3	0.83(0.41)	0.90(0.45)	0.87(0.36)	0.80(0.71)

accuracy, loss, or both, while also not showing any worsening. We found this to be interesting, as typically features are not computed over multitaper spectrogram but over spectrograms formed using Hamming windows.

Data set D_B showed that with audio generated by telephone communication, it should be viable, given the present setup with feature dimensions, to consider a narrower band as opposed to anything wider (even though possibly available).

As for translation to the data set provided by the company, we concluded to primarily look at filter bank energies at the narrowband frequencies $200 - 3400$. The reason for going forward with features generated at such a narrow band was primarily that we saw a relatively large worsening in accuracy of pitch shift identification considering bands too wide. This in combination with the fact that we could not know over which audio band a given telephone conversation had taken place, making the decision a bit conservative as well. We also concluded that the filter banks were to be computed on spectral estimates obtained utilizing multitapering techniques. After viewing the results, we could not see any reason for considering a parametric spectral estimate.

Part II: Data pre-processing

4.1 Introduction

This part regards pre-processing of the data set that was provided by the company. In particular, we wanted to extract only parts of the audio streams, ideally being generated by speech from the customer as opposed to silence or speech from the company representative. Recall that we had defined the three parts constituting any data point X as $X^{customer}$, X^{agent} and $X^{silence}$. In most cases, an audio stream X in the data set was generated by a telephone conversation. As such, there was a lot of different environmental background noises, as well as varying audio quality, meaning that we had to use adaptive techniques for preparing $X^{customer}$, preferably with few hyper-parameters to be tuned. The act of identifying parts in audio generated by speech is commonly referred to as voice activity detection. Identifying who spoke when can be referred to as speaker diarization.

4.2 Voice activity detection (VAD)

Voice activity detection was implemented as means of removing silent parts from the audio. Beigi (2011) describes that an efficient method for removing silence is to identify when the signal energy drops below a certain threshold. Such a threshold can be determined in various ways, Beigi (2011) proposes for example to set the threshold based on some initial time (e.g. first 0.2 seconds) where no speech would be assumed present. However, as the data consisted mostly of telephone conversations, we noted that setting such a threshold to be accurate for the entire data set would be a challenge. Some conversations had for example initial loud tones generated by telephone noise. As such, it was proposed to construct an adaptive threshold based on a long-term moving average. Voice activity detection was ultimately performed by identifying when a short-term moving average of the energy in the signal dropped below a threshold determined by a weighted long-term moving average.

For energy level E calculated as the sum of powers over all frequencies and re-calculated at every $1/T$ seconds, define time intervals as,

$$t_s = \textit{time interval (length) of short-term moving average}$$
$$t_l = \textit{time interval (length) of long-term moving average}$$

and moving averages as,

$$\textit{short term: } M_s(t) = \sum_{s=0}^{\lfloor t_s T \rfloor} E(t-s)$$
$$\textit{long term: } M_l(t) = \alpha \sum_{s=0}^{\lfloor t_l T \rfloor} E(t-s)$$

Then we can further define an indicator for voice activity as

$$V(t) = 1\{M_s(t) < M_l(t)\} \tag{4.1}$$

To remove silent parts from the audio stream we then discarded parts of the audio stream where $V(t) = 0$. The idea is illustrated in Figure 4.1. Before calculating the short term and long term moving averages however, weighting of the frequency spectrum was performed. The reason for weighting the spectrum was that without doing so the result was dis-satisfactory, which was speculated to be due to noise at frequencies not typical for human perception. It was speculated that the noise interfered with the energy level generated by the actual voice, and as such it was proposed that it would be beneficial to boost the amplitude at the frequencies typical for human perception while also attenuating the frequencies more typical for noise. For this, a binary high-pass filter was introduced with cut-off at 80 Hz. In addition, the A-weighting function, being a spectral weighting function commonly used for environmental noise measurements, was applied. This led to more satisfactory result. Illustration of the moving averages and indicator function can be seen in Figure 4.2 which is from an actual telephone conversation. In the end, the hyper-parameters was set by trial and error to be $t_s = 0.1,\ t_l = 60,\ \alpha = 0.1$.

4.3 Speaker diarization

Speaker diarization was the act of separating the audio made up of telephone conversations into parts, where each part belonged to different speakers. As we made the assumption that all conversations consisted of speech from two distinct persons, we could adopt a technique suggested by Beigi (2011). First, we would utilize an algorithm for detecting turn points, which would be the time points where there had possibly occurred a change in speaker. This would result in many small disjoint segments. Second, as the number of turn points

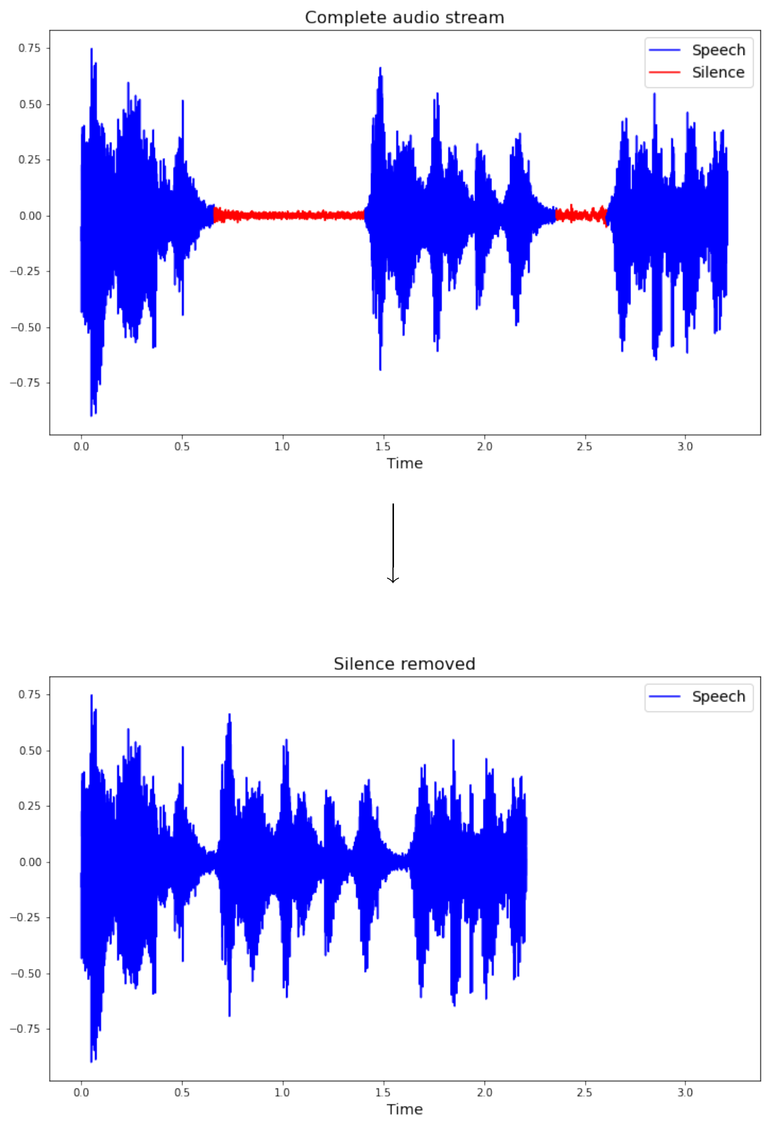

Figure 4.1: The method for removing silent parts of audio visualized. Segments consisting of silence is first identified, thereafter removed from the audio stream, generating a truncated version where only speech is present.

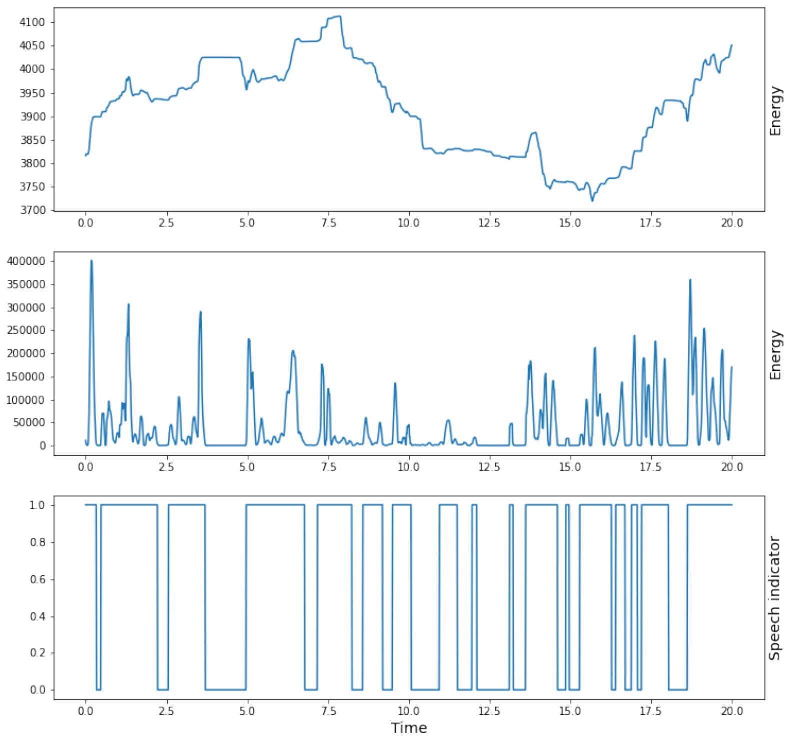

Figure 4.2: Illustration of voice activity detection during 20 seconds of an actual telephone conversation. From top to bottom: weighted long term moving average of energy (60 s), short term moving average of energy (1/10 s) and resulting indicator function V as defined in (4.1). Note that the indicator match up with activity in short term energy.

preferably would be an overestimation (Bonastre et al., 2000), where each represent a candidate[1] time point for a change of speaker, we would try to merge smaller segments into larger[2], labelling each the speech of either speaker A or B. The technique is illustrated in Figure 4.3.

In order to find turn points, we implemented a version of a method suggested by Delacourt and Wellekens (2000). Given an audio stream, we calculated a spectrogram with frames of $20ms$, overlapping by 50% utilizing a Hamming window. From the spectrogram we extracted MFCC as detailed in section 2.1.1. From here on, the procedure of finding turn points was separated into two phases. First we tried to find many turn points, over-segmenting the audio by quite a lot. Thereafter we would try to reduce the number of turn points by merging adjacent segments. The two phases and the merging of segments are detailed in subsequent sections.

4.3.1 First phase: Locating turn point candidates

We formed pairs of adjacent windows from the feature vectors, each stretching l seconds. With the feature vector time series denoted $\{X(t)\}_{0\le t\le T}$, define the pair of l-seconds large windows as

$$W(t) = (W_0^{(t)}, W_1^{(t)}) \quad : \quad \begin{cases} W_0^{(t)} & = \{X(u)\}_{t-l\le u<t} \\ W_1^{(t)} & = \{X(u)\}_{t\le u<t+l} \end{cases}$$

The pair $W(t)$ is illustrated in Figure 4.4. From $W(t)$ we could set up hypothesises, form a test statistic $C(W(t))$ and yield a comparable measure with any $W(s)$, $s \neq t$. The null-hypothesis set up for any W was that the two windows W_0 and W_1 was generated by the same multi-dimensional Gaussian distribution, i.e

$$H_0 : W_0 \in N_d(\mu, \Sigma), \;\; W_1 \in N_d(\mu, \Sigma)$$
$$H_1 : W_0 \in N_d(\mu_0, \Sigma_0), \;\; W_1 \in N_d(\mu_1, \Sigma_1)$$

and the test statistic was the log generalized likelihood ratio [3],

$$C(W) = -2\ln\left(\frac{\sup_{\mu,\Sigma} \mathcal{L}(W_0 \cup W_1,\ \mu; \Sigma)}{\sup_{\mu_0,\Sigma_0} \mathcal{L}(W_0,\ \mu_0; \Sigma_0)\sup_{\mu_1,\Sigma_1} \mathcal{L}(W_1,\ \mu_1; \Sigma_1)}\right)$$

By constructing a grid of evenly spaced time points

$$K = t_k \ : \ \begin{cases} k = 0, 1, \ldots, n \\ t_0 = l \\ t_n = T - l \end{cases}$$

[1]Rather than a true time point for a change of speaker

[2]Instead of simply assuming a change of speaker at each turn point

[3]As recommended for spontaneous speech by Delacourt and Wellekens (2000)

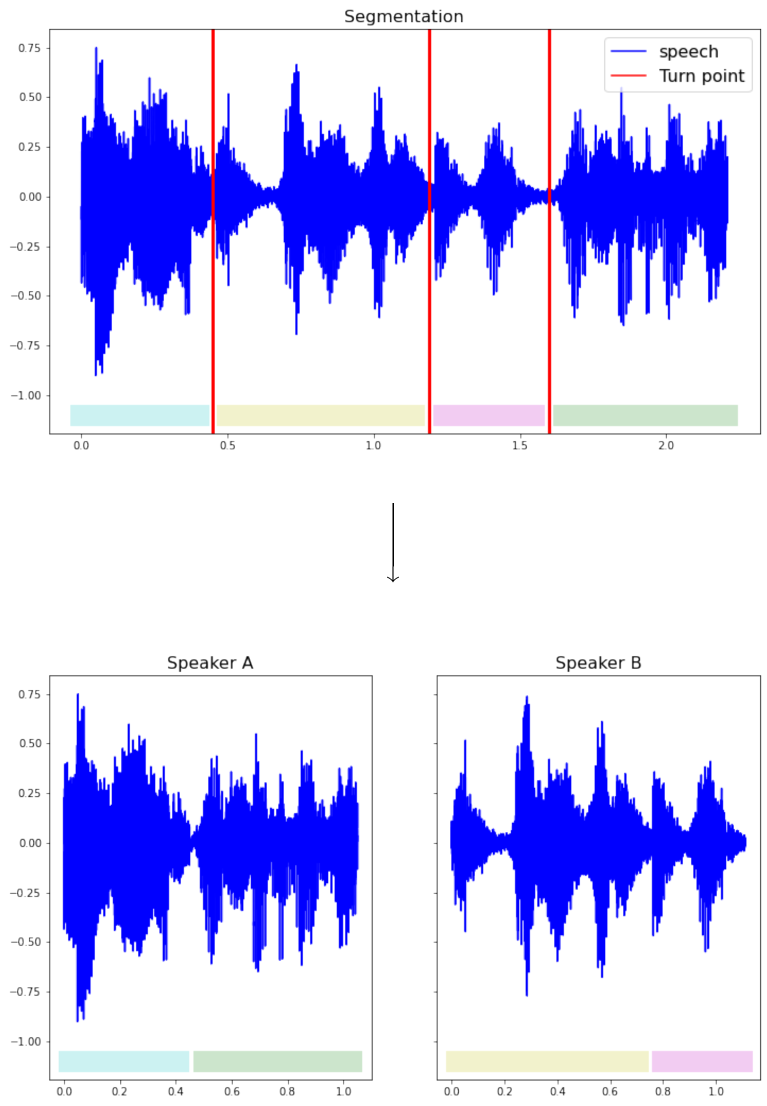

Figure 4.3: The method for separating speakers illustrated. Note that silent parts are removed. First, turn point candidates are identified (red vertical lines). Thereafter, the segments generated in-between turn point candidates are clustered using agglomerate hierarchical clustering until there remains two clusters, representing speaker A and B.

we could calculate the realization C_K for the entire grid, where naturally

$$C_K := \{C(W(t))\}_{t \in K}$$

Now, if a change of speaker actually occurred at time t, then one can expect that in general, $C(W(t))$ would be larger than $C(W(s))$ for any s not being an instance in time where a change of speaker occurred. As neighbouring time points around a change of speaker arguably does not constitute a change of speaker, local maximum in C_K would indicate a possible turn point. As such, the idea as described by Delacourt and Wellekens (2000) was that time points where local maximum were found were in the first phase considered a turn point as long as the height relative to adjacent local minimums where larger than some fraction η of $SD(C_k)$. In addition, for local maximums closer to each other than some threshold, the (relatively) smallest one would be discarded. In essence, the idea for a time point t_k being considered a turn point is illustrated in Figure 4.5.

In practice we noted that the generated plot did not really resemble Figure 4.5 as there was a lot more noise, taking the shape of many small peaks. This posed a problem, as large peaks would be considered smaller than expected due to a local minima (caused by C_K being noisy) located close to the peak maxima. To counter the issue with C_K being too noisy, we looked at the prominence of the peaks. The prominence of all peaks was calculated. Peaks with prominence less than the fraction η of $SD(C_k)$ were discarded. Thereafter, the peaks were sorted from smallest to largest (based on prominence) and in an iterative manner they were discarded if they were within the threshold of being to close to another (larger) peak.

When preparing the data, we used parameters lining up with what Delacourt and Wellekens (2000) found suitable for telephone conversations. We set window length l to 2 seconds. η was set to be 1/2. The resolution of the grid, i.e $[t_k - t_{k-1}]$ was set to 1/10 seconds. The threshold for two local maximums being considered too close in time was set to one second.

4.3.2 Second phase: Reducing turn point candidates

From the first phase, there had typically been extracted numerous turn point candidates. Before clustering segments, we tried to merge adjacent segments by identifying turn points that could be dropped. One option for achieving this was to: Given two adjacent segments separated by one turn point candidate, set up the hypothesises

H_0 : *The two segments are generated from the same multivariate Gaussian distribution.*

H_1 : *The two segments are generated from different multivariate Gaussian distributions.*

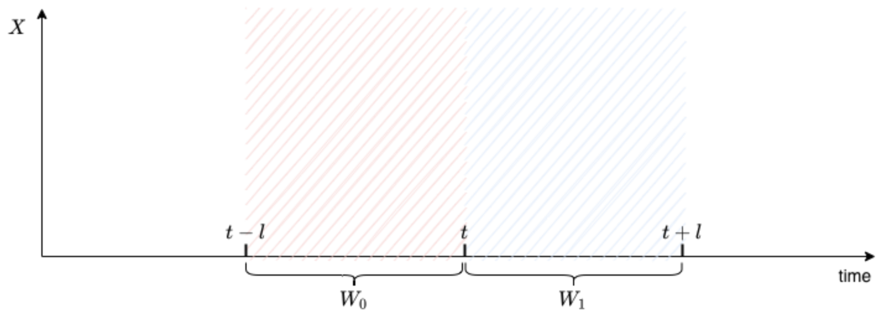

Figure 4.4: Illustration of a pair $W(t)$ of windows W_0 and W_1. Time t separates the two windows that contains all the points $\{X(s) \; : \; s \in W_0\}$ and $\{X(s) \; : \; s \in W_1\}$ respectively.

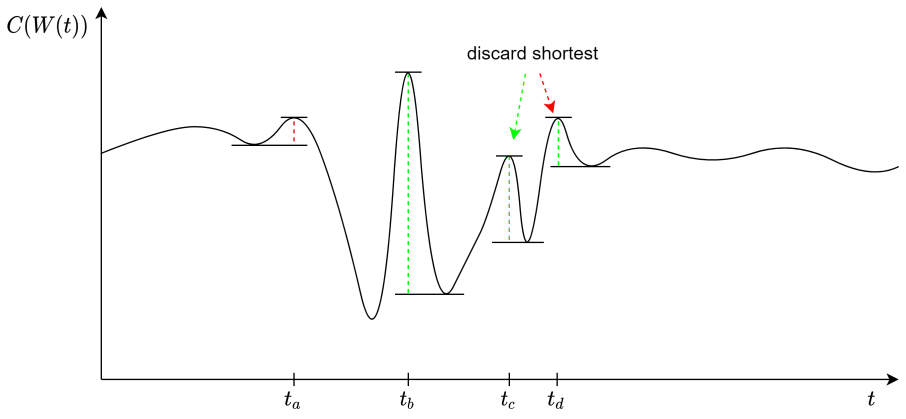

Figure 4.5: Illustration of condition for considering a time point to be a possible location for a change of speaker (turn point candidate). Here, time t_b and t_c are regarded as turn point candidates. t_a is discarded as the height of the peak is less than a pre-determined fraction η of $\mathrm{SD}(C(W_K))$. t_d is discarded not due to the size of the peak being to small in itself, but due to t_c and t_d being to close: so we discard the smallest of the peaks.

and test with some pre-determined significance level. If the test statistic would fall in the critical region: keep the turn point candidate. If not: discard the turn point candidate.

The use of BIC as metric for clustering speaker segments was first proposed by Chen and Gopalakrishnan (1998). Now, instead of identifying a suitable test statistic and significance level, we adopted BIC as metric. BIC as metric is motivated by Delacourt and Wellekens (2000) who describes that for the second phase, the segments tend to be longer than the window size used in the first phase and therefore BIC as metric is appropriate. Importantly, the aim was still to identify if a turn point candidate could be discarded or not based on the hypothesises set up, and the means was still to compare two adjacent segments with their union. For two segments Z_0 and Z_1 with union denoted Z, the BIC distance is defined as

$$\Delta\text{BIC} = -R + \lambda P,$$

where R is given as

$$R = \frac{N_Z}{2} \log \mid \Sigma_Z \mid - \frac{N_{Z_0}}{2} \log \mid \Sigma_{Z_0} \mid - \frac{N_{Z_1}}{2} \log \mid \Sigma_{Z_1} \mid,$$

and P given by

$$P = \log(N_Z)\frac{1}{2}\left(p + \frac{1}{2}p(p+1)\right)$$

for p being the dimensions in one feature vector. Note the hyper-parameter λ which is set in conjunction with a threshold for ΔBIC in order to conclude when two segments should be represented as one combined segment. Through trial and error, we set λ to equal 1.5 and a threshold for ΔBIC to 0. An example of C_K generated from an audio stream with corresponding turn points is depicted Figure 4.6 where both turn points from the first phase is displayed as well as which could be removed as a result of the second phase.

4.3.3 Merging segments

The final step in separating the speakers was to merge segments into clusters, one representing each speaker. One approach often employed for this means is Agglomerative Hierachial Clustering based on some similarity measure (Park et al., 2021). We decided to implement this kind of "bottom-up" clustering (further described in subsequent paragraph). The same acoustic feature (MFCC with 13 coefficients) that was used when identifying turn points was used now. Furthermore, we used the same measure of similarity (ΔBIC) and hyper-parameters for clustering as we did when reducing turn points candidates (λ = 1.5 and threshold equal 0). The stopping criterion was naturally set to be when we had reached two clusters.

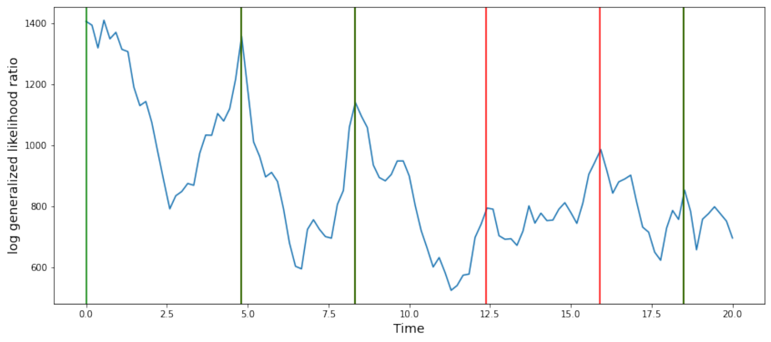

Figure 4.6: Turn points from both phases displayed along with $C(w(t))$ from a real audio stream. All vertical lines indicate a turn point candidate found in the first phase. Only turn points marked as green vertical lines are kept after the second phase, whereas the segments separated by red lines are considered too alike.

Agglomerative hierarchical clustering

Agglomerative hierichical clustering is a clustering technique where one works from the bottom and up. Starting with a set of data elements, one merge the two elements that are closest as defined by some measure of distance into a new element. Thereafter one repeats the procedure. This is done until there exists only as many element as is the target number of clusters. The method is visualized with the dendrogram depicted in Figure 4.7.

4.4 Speaker identification

After having merged segments into two clusters as described in the previous section, we had arrived at the possibility to split up the audio file into two distinct tracks (recall Figure 4.3). The issue from here on was to identify the person of interest, i.e to identify a track representing $X^{customer}$.

From additional data provided we knew that there were 14 individual company representatives responsible for all telephone conversations in the data set. Furthermore, each company representative had been assigned a unique identifier which was matched against each data points. As such, we could group data points by the company representatives. This was key to identify which track generated from X that was going to represent $X^{customer}$. For each such group consisting of a number of audio streams

$$X_0, \ldots, X_j, \ldots, X_G$$

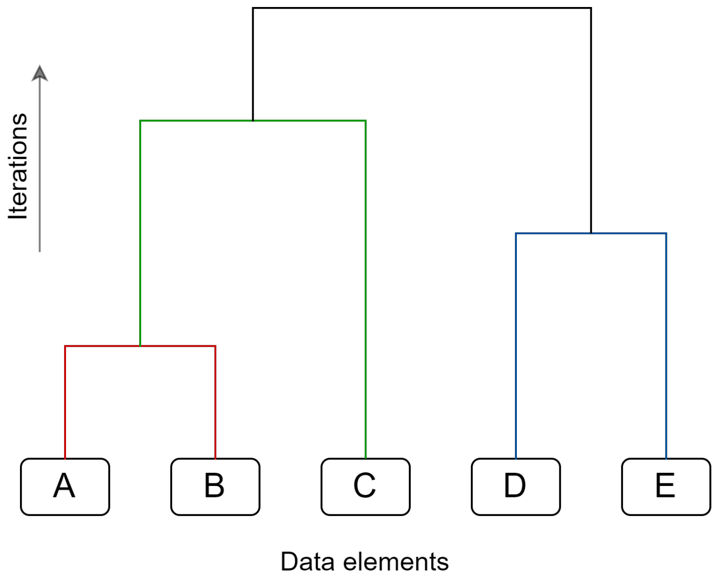

Figure 4.7: Agglomerative hierarchical clustering visualized. One merge data elements until only the number of desired clusters remains. In this figure, if one seek to find for example three clusters, one would stop after merger depicted in blue, yielding clusters: {A, B}, {C}, and {D, E}.

one could label the two tracks as A_j and B_j, where one ideally should represent $X_j^{customer}$ and the other X_j^{agent}. The problem was to identify if A_j or B_j should represent $X_j^{customer}$ and therefore be kept (while the rest would be discarded).

By re-using the acoustic features (MFCC with 13 coefficients) that was used for identifying turn points, we could form a sort of fingerprint for each track A_j and B_j as being the estimated expected values and covariance matrix of the acoustic features. Recall that we had assumed a multi-variate normal distribution for the acoustic features. The hope was that for two audio streams X_k and X_l, we could match the two tracks most similar (not both being from one stream), and assume that those two tracks would be generated by X_j^{agent}. This would work essentially since for two audio streams we had four tracks but only three distinct speakers. By iterating through each group of company representatives, we would for each audio stream X_j in a group let the other audio streams in the same group cast a vote on either A_j or B_j being represented by X_j^{agent}. The track with the highest number of votes would be discarded as it would be assumed to be consisting of audio generated not by the customer but by the company representative. To implement this we needed to have a measure of similarity between fingerprints. A measure that has been used for example in the context of image segmentation but under similar assumption on the statistic is the Bhattacharyya measure of similarity (Coleman & Andrews, 1979). The Bhattacharyya measure can be calculated for two multi-variate Gaussian distributions p and q as

$$D(p,q) = \frac{1}{4}\ln\left(\frac{1}{4}\left(\frac{\sigma_p^2}{\sigma_q^2} + \frac{\sigma_q^2}{\sigma_p^2} + 2\right)\right) + \frac{1}{4}\left(\frac{(\mu_p - \mu_q)^2}{\sigma_p^2 + \sigma_q^2}\right)$$

4.5 Results

The method for extracting $X^{customer}$ from X could not be evaluated thoroughly as it would mean a lot of manual work in labeling audio segments. However, by drawing a sample $\hat{X}_1^{customer}, \ldots, \hat{X}_n^{customer}$, and manually determining for each audio stream if it is clear that the majority of the speech is uttered by the customer and not the company representative, we could at least provide a measure of the success in speaker identification conditional that the speaker diarization was satisfactory. This measure would however then be of interest not only for evaluating the speaker identification, but considering that speaker identification would fail if speaker diarization had failed, this measure would contribute in the understanding of the level of success for the complete data preparation.

Speaker identification

Denote $S_1, \dots, S_n$ to be such that for any $1 \leq k \leq n$,

$$S_k = \begin{cases} 1, & \text{if } \hat{X}_k^{customer} \text{ is identified correctly}^4 \\ 0, & \text{otherwise} \end{cases}$$

then, assuming that S_i and S_j are independent for $i \neq j$ and that

$$E(S_i) = E(S_j) = p$$

we can construct a confidence interval for p, being the probability of $X^{customer}$ "successfully" being extracted from X with the method in place. We consider the sum

$$B_n = \sum_{k=1}^{n} S_k$$

and note that if we consider S_k, $k = 1, \dots, n$ as i.i.d (disregarding any possible systematical error), we get

$$B_n \in \text{Bin}(p, n)$$

which allows for an exact confidence interval for p as

$$I = \Big(\inf\{p \ : \ P(B_n \leq x) > \frac{\alpha}{2}\}, \ \sup\{p \ : \ P(B_n \geq x) > \frac{\alpha}{2}\}\Big)$$

with x being the realization of B_n, i.e the number of successes. The confidence interval I is also known as a Clopper-Pearson interval, which was computed using the equivalent expression involving the quantiles of a Beta distribution (Thulin, 2013).

$$I = \left(B\left(\frac{\alpha}{2}, x, n - x + 1\right), \ B\left(1 - \frac{\alpha}{2}, x + 1, n - x\right)\right)$$

The resulting 95% confidence interval ($\alpha = 0.05$) after drawing a random sample of size $n = 20$ was computed as

$$I = (0.68, \ 0.99)$$

where x had been retrieved as $x = 18$.

[4] subjectively, based on manually determining if a clear majority of the speech is uttered by the customer.

Part III: Real data

5.1 Introduction

After having first evaluated features and the concept of utilizing a LSTM model on simulated data, we had the set up in place for modelling data provided by the insurance company. We recall that the conclusions that had been drawn from the simulated data was that we should seek to evaluate filter bank energies as input, and to compute such energies on the basis of a multitaper spectrogram stretching a narrow frequency band of 200-3400 Hz. We modelled the real data in two attempts, first utilizing a LSTM model and second by the implementation of a ResNet model.

5.2 Data description

We had access to 1324 recorded conversations between customers and company representatives. Most of the conversations were via telephones. The majority of the audio files that was generated had been compressed into mp3 files, thus most commonly being at a sample rate of 44.1 kHz. We utilized Python and the package Librosa (version 0.9.1, 2022) to load the audio files into their respective waveforms. Audio files which was not sampled at 44.1 kHz was still loaded into a waveform at 44.1 kHz. This was done by utilizing a functionality in Librosa for down and up sampling.

5.3 Models and data handling

As described in Section 2.2.1, LSTM networks are a branch of recurrent neural networks that does not suffer from vanishing gradient in the same extent as predecessors. Likewise, for data sets consisting of long time series (2000+ steps) tried out by Fawas et al (2019), the ResNet model performed best on average out of the models tried (Fawas et al., 2019), indicating that the model would not suffer from vanishing gradient given rather lengthy input sequences. However, it was still to our understanding that it would not be viable to consider either network for time series of more than a few thousand time steps. As some of the data points X in the data set yielded features just short of half a million time steps, we concluded that we had to cut up data points in order to be able to successfully train the networks. We detail the method for dealing with the lengthy time series in subsequent sections.

5.3.1 First attempt: LSTM

We implemented a LSTM model as described in Section 2.2.1. The least complex model we tried consisted of a layer of 24 LSTM memory cells, followed by two hidden dense layers with 32 and 16 nodes respectively. From this model, we increased the complexity by first adding nodes to existing layers, subsequently adding on new layers. We kept doing this until we found a model capable of learning from the training data. We set a limit to the number of epochs at 2000, if no improvement above baseline determined by random prediction had been made before the end of the last epoch, we considered the model design incapable. We tried three different input lengths: 450, 900 and 1800, corresponding to approximately 10, 20 and 40 seconds worth of audio respectively. All data was kept, meaning that long data points was cut into many shorter data points.

5.3.2 Second attempt: ResNet

We implemented the ResNet model detailed in Section 2.2.2. On the grounds of data points possibly being too long with regards to number of time steps, we decided on two approaches. First, we tried to cut data points into new shorter data points of length: 450, 900, 1800 and 5400, corresponding to approximately 10, 20, 40 and 120 seconds worth of audio respectively. Second, we tried to segment each data point into segments of 13500 time steps (corresponding to approximately five minutes worth of audio). From each segment, a new data point was constructed by merging time steps from each segment in chronological order such that the size of the new data point was 1800. For example, with a data point Z consisting of 30000 time steps, we created two segments S_1 and S_2, where S_1 consisted of the first 13500 time steps from Z and S_2 the following 13500 time steps. To construct new data points of length 1800, we had to pick 900 time steps from each segments. Therefore, each segment was divided into $13500/900 = 15$ sub-segments:

$$S_1^{(1)}, S_1^{(2)}, \ldots, S_1^{(15)}$$

with corresponding notation for S_2. New data points Z_i for $i = 1, \ldots, 15$ was then formed by merging $S_1^{(i)}$ with $S_2^{(i)}$. As such, the resulting length for Z_i would be 1800 time steps. Note, the last 3000 time steps in Z was discarded ($13500 + 13500 = 27000$). Just like with the LSTM model, we set the maximum number of epochs to 2000.

5.4 Results

We failed to find any generalization onto the validation data. In both attempts, we observed both under-fitting and over-fitting, meaning that a model either did not fit the training data or that a model fit the noise in the training data as opposed to the general pattern sought to be found. We aimed to see an improvement in the loss on the validation set before over-fitting occured, but did not manage to do so.

5.4.1 First attempt: LSTM

We evaluated models for time series that were of varying length (450, 900 and 1800 time steps). The least complex model that we found which could learn from data was a model with nine hidden layers, three of which were LSTM layers with remaining five being ordinary dense. The model is detailed in Figure 5.1. However, it was only for data being of 450 time steps that we saw any learning taking place. Furthermore, it was not until after around 1000 epochs that the model started to learn. We also saw that loss in the validation set started to worsen at the same time, meaning that the model learnt to identify noise in the training set (over-training) as opposed to any general patterns. See figure 5.2 which shows how loss in the validation set starts to increase at the same time as loss in the training set starts to decrease. In fact, we could not find any model utilizing LSTM memory cells that managed to generalize any training onto the validation set (within 2000 epochs).

5.4.2 Second attempt: ResNet

Contrary to the LSTM model where we had a hard time finding a model capable enough to learn from the data, we saw that the ResNet model managed to fit the training set within the first few epochs. However, we did not see any generalization onto the validation set. This was the case in both approaches considered for handling the number of time steps in the data. We experimented with label smoothing (factor 0.1) and with increasing the kernel size, as well as halving the number of filters. This led to the model having to go through many more epochs before learning. Nevertheless, the model eventually did fit the training set, but still no generalization could be seen onto the validation set.

5.5 Discussion and future research

We did not find any type of model, neither as a version of LSTM nor ResNet able to generalize any key structures from training data over to validation data. This was of course something that we had sought to achieve.

First and foremost, there are many ways to deal with lengthy time series data points for the different type of models and we considered only rather simple

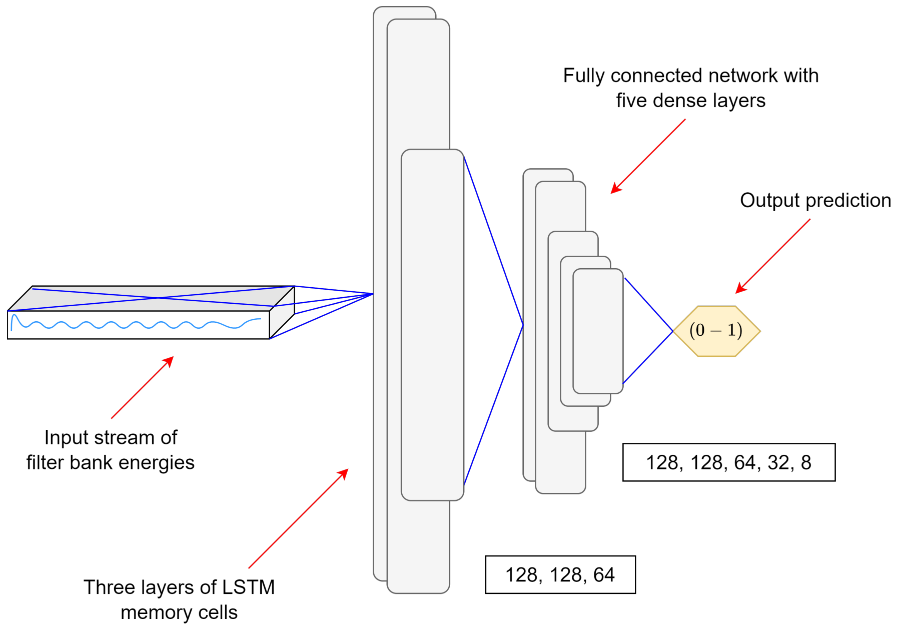

Figure 5.1: Design of the LSTM type of network that managed to learn from data, although no generalization onto validation data occurred. Input was the feature that had been extracted and was varying in length but remained 30 dimensions wide. Three layers of LSTM memory cells followed, where the first two layers operated as sequence to sequence with the output fed into the next layer. The final layer of LSTM memory cells had a single output value for each cell (64 in total), meaning that the fully connected network that was attached was fed by an input of 64 dimensions. The input to the fully connected network of five hidden dense layers all with ReLU activation function yielded an output mapped by a Sigmoid function in the range 0-1. The final output was to be interpreted as the probability of fraud.

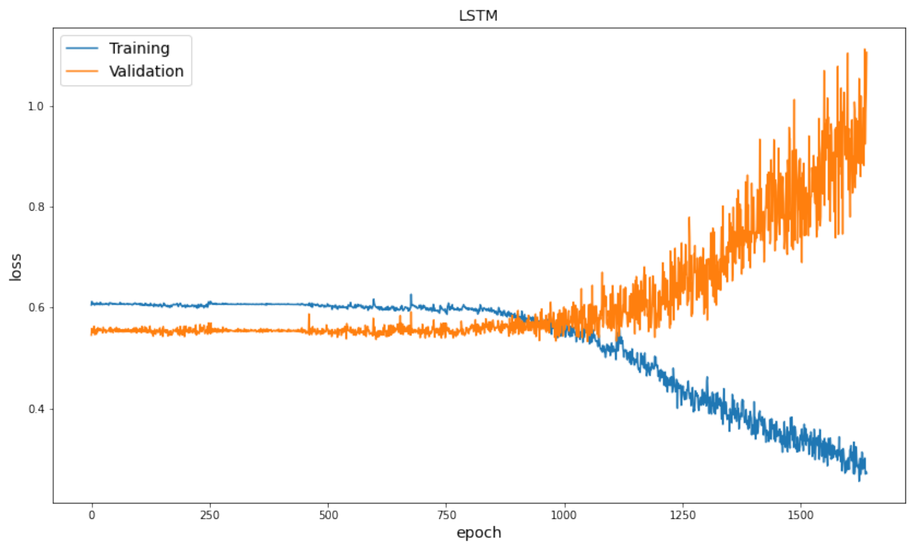

Figure 5.2: Loss from training a LSTM network on data points that are cut down in length to 450 time steps each. It is clear that overtraining occurs (at around 1000 epochs) before any generalization has taken place.

approaches. Even though LSTM can handle varying length input, we decided to use input of fixed length for the reason of time efficiency. We trained the model with mini-batches, typically of size 64, and as such each input to every individual mini-batch had to be of the same length. In order to fully take advantage of the way that LSTM can train on data with varying length we would have to drop the concept of mini-batches, increasing computational time. The ResNet on the other hand, being a convolutional type of network with residual connections, mandates input of fixed length. This means that all input to a ResNet model will have to be of the same length regardless of initial length. By cutting up data into segments that are reasonable in size, i.e somewhere in the range of a few hundred time steps up to possibly a few thousand, we loose long term dynamics from the conversations. We tried to handle this by applying an adaptive type of approach for lengthy inputs, however we believe that possibly important full-conversation type of dynamics is lost regardless. One possibility that we considered to evaluate but found to be outside the scope of this project was to implement a compression mechanism. Especially, we considered that it might be interesting to look at the use of an auto-encoder in order to project sequences consisting of N time steps into compressed sequences of M time steps, $M < N$.

Ignoring the issues with lengthy inputs, we recall that we made the assumption that whatever is changing when someone is discussing an invalid claim is (amongst others) the pitch, which we considered due to its proposed link to properties of deceptive voice. This assumption could perhaps be far-fetched, or even wrong. However, the impact of the assumption was likely not very large, as both the features and models that we considered were rather standard within the fields of speaker recognition and time series modelling anyway.

Bibliography

Beigi, H. (2011). *Fundamentals of speaker recognition.* Springer, Boston, MA.

Royer, T. (2019). Pitch-shifting algorithm design and applications in music (Dissertation).

Flanagan, J. L., Golden R. M. (1966). Phase vocoder. *The Bell System Technical Journal, 45*(9), 1493-1509. DOI:10.1002/j.1538-7305.1966.tb01706.x.

Franklin Nissy, M. F., Renisha, G. (2020). Telephone Voice Speaker Recognition Using Mel Frequency Cepstral Coefficients With Cascaded Feed Forward Neural Network *Iconic Research And Engineering Journals, 3*(8), 164-170

Bonastre, J. -F., Delacourt, P., Fredouille, C., Merlin, T. and Wellekens, C. (2000). A speaker tracking system based on speaker turn detection for NIST evaluation. *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100), 2*, II1177-II1180. DOI: 10.1109/ICASSP.2000.859175.

Delacourt, P., Wellekens, C. J. (2000). DISTBIC: a speaker-based segmentation for audio data indexing. *Speech Commun. 32*, 111–126. DOI:10.1016/S0167-6393(00)00027-3

Shaobing Chen, S., Gopalakrishnan, P. S. (1998). Clustering via the Bayesian information criterion with applications in speech recognition. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181), 2*, 645-648. DOI:10.1109/ICASSP.1998.675347.

Park, T. J., Kanda, N., Dimitriadis, D., Han, K. J., Watanabe, S., Narayanan, S. (2021). A Review of Speaker Diarization: Recent Advances with Deep Learning. DOI:10.48550/arXiv.2101.09624

Coleman, G. B., Andrews, H. C. (1979). Image Segmentation by Clustering. *Proceedings of the IEEE, 67*(5), 773-785. DOI:10.1109/PROC.1979.11327

Villar, G., Arciuli, J., Paterson, H. (2013). Vocal pitch production during lying: Beliefs about deception matter. *Psychiatry, Psychology and Law, 20*(1), 123–132. DOI:10.1080/13218719.2011.633320

Levitan, S., Maredia, A., Hirschberg, J. (2018). Acoustic-Prosodic Indicators of Deception and Trust in Interview Dialogues. DOI:10.21437/Interspeech.2018-2443.

Cox, R., Campos, S., Lamblin, C., Sherif, M. (2009). ITU-T coders for wideband, superwideband, and fullband speech communication [Series Editorial]. *Communications Magazine, IEEE, 47*, 106-109. DOI:10.1109/MCOM.2009.5273816.

Charbonneau, K., Colin, N., Gaspar, R., Ule, H. (2012). A-weighting the equal loudness contours. *The Journal of the Acoustical Society of America 131*(4):3502. DOI:10.1121/1.4709236

McMinn, T. (2013). "A-weighting": Is it the metric you think it is?. *Annual Conference of the Australian Acoustical Society 2013, Acoustics 2013: Science, Technology and Amenity*, 165-168.

Amidi, A., Amidi, S. (n.d). *Deep Learning Tips and Tricks cheatsheet.* https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks

Thulin, M. (2013). The cost of using exact confidence intervals for a binomial proportion, *Electronic Journal of Statistics 8*, 817-840. DOI:10.1214/14-EJS909

Fawaz, I. H., Forestier, G., Weber, J., Idoumghar, L., Muller, P-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery 33*, 917–963. DOI:10.1007/s10618-019-00619-1

McFee, B., Metsai, A., McVicar, M., Balke, S., Thomé, C., Raffel, C., Zalkow, F., Malek, A., Kyungyun Lee, D., Nieto, O., Ellis, D., Mason, J., Battenberg, E., Seyfarth, S., Yamamoto, R., viktorandreevichmorozov, Choi, K., Moore, J., . . . Thassilo. (2022). librosa/librosa: 0.9.1 (0.9.1). Zenodo. [computer software]. DOI:10.5281/zenodo.6097378

Sandsten, M. (2020). *Time-Frequency Analysis of Time-Varying Signals and Non-Stationary Processes. An Introduction.*

Vrij, A., Semin, G. R. (1996). Lie experts' beliefs about nonverbal indicators of deception. *Journal of Nonverbal Behavior, 20*(1), 65-80. DOI:10.1007/BF02248715

Taylor, R., Hick, R. F. (2007). Believed cues to deception: Judgments in self-generated trivial and serious situations. *Legal & Criminological Psychology, 12*(2), 321-331. DOI: 10.1348/135532506X116101

Zhang Z., McGettigan C., Belyk M. (2022). Speech timing cues reveal deceptive speech in social deduction board games. *PLoS ONE 17*(2):e0263852. DOI:10.1371/journal

National Research Council (2003). *The Polygraph and Lie Detection.* Washington, DC: The National Academies Press. DOI:10.17226/10420

Beranek, L. L. (1949). *Acoustic measurements.* New York: McGraw-Hill

O'Shaughnessy, D. (1987). *Speech Communication: Human and Machine.* Addison-Wesley Publishing Company, pp. 150

Cassidy, S., Harrington, J. (1999). *Techniques in speech acoustics.* Springer, pp. 18

Lindsay, P. H., Norman, D. A. (1977). *Human information processing: An introduction to psychology* (2nd ed.). New York: Academic Press.

Kinnunen, T., Saeidi, R., Sedlak, F., Lee, K. A., Sandberg, J., Hansson-Sandsten, M., Li, H. (2012). Low-Variance Multitaper MFCC Features: A Case Study in Robust Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing, 20*(7), 1990-2001. DOI:10.1109/TASL.2012.2191960

Alam, J., Kinnunen, T., Kenny, P., Ouellet, P., O'Shaughnessy, D. (2013). Multitaper MFCC and PLP features for speaker verification using i-vectors. *Speech Communication, 55*(2), 237-251. DOI:10.1016/j.specom.2012.08.007

Snorovikhina, V., Zaytsev, A. (2020). Unsupervised anomaly detection for discrete sequence healthcare data. *AIST*

Chauhan, S., Vig, L. (2015). Anomaly detection in ECG time signals via deep long short-term memory networks. *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 1–7. DOI:10.1109/DSAA.2015.7344872

Bontemps, L., Cao, V. L., McDermott, J., Le-Khac, N-A. (2017). Collective Anomaly Detection based on Long Short Term Memory Recurrent Neural Network. *arXiv.* DOI:10.48550/ARXIV.1703.09752

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. DOI:10.48550/ARXIV.1512.03385

Schmidhuber. (2021). *The most cited neural networks all build on work done in my labs.* https://people.idsia.ch/ juergen/most-cited-neural-nets.html

Sporer, S.L., Schwandt, B. (2006). Paraverbal indicators of deception: a meta-analytic synthesis. *Appl. Cognit. Psychol., 20*, 421-446. DOI:10.1002/acp.1190

Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. *Neural Comput. Nov 15;9*(8). 1735-80. DOI:10.1162/neco.1997.9.8.1735

Childers, D. G., Skinner, D. P., Kemerait, R. C. (1977). The cepstrum: A guide to processing. *IEEE 65*(10). 1428-1443. DOI:10.1109/PROC.1977.10747

Bogert, B.P., Healy, M.J., Tukey, J.W. (1963). The quefrency analysis of time series for echoes : cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking.