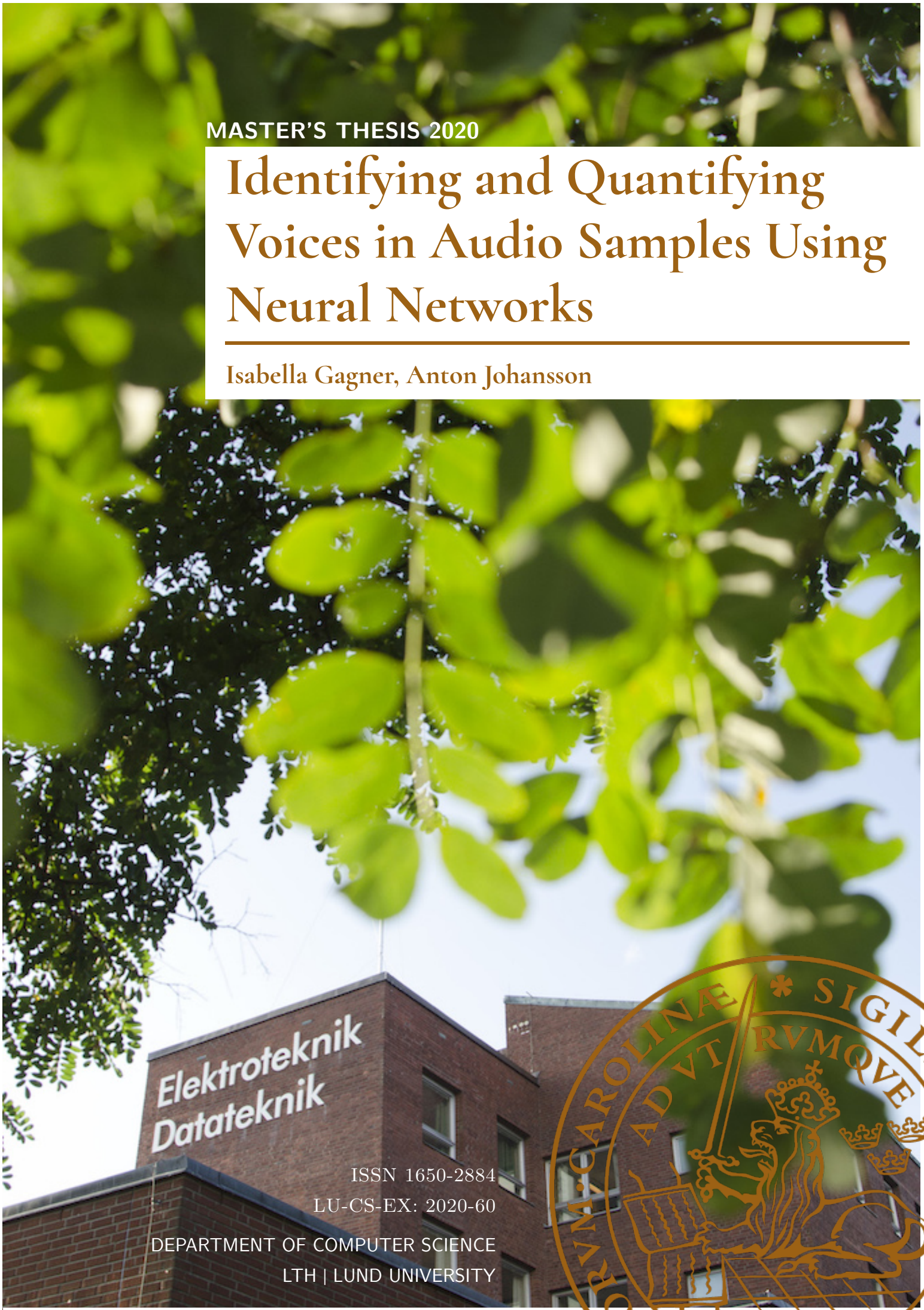
MASTER'S THESIS 2020

Identifying and Quantifying Voices in Audio Samples Using Neural Networks

Isabella Gagner, Anton Johansson



ISSN 1650-2884
LU-CS-EX: 2020-60

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-60

Identifying and Quantifying Voices in Audio Samples Using Neural Networks

Isabella Gagner, Anton Johansson

Identifying and Quantifying Voices in Audio Samples Using Neural Networks

Isabella Gagner
tfy15iga@student.lu.se

Anton Johansson
tfy15ajo@student.lu.se

September 28, 2020

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Colin Nordin Persson, colin@minut.com

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Abstract

By analysing audio samples of an environment, important information regarding the environment could be gained. In this thesis, we examined and compared mel-frequency cepstrum coefficients (MFCC) used in a convolutional neural network (CNN) and speaker diarization methods, to evaluate if it was possible to approximate the number of speakers in a recording.

Results showed that the speaker diarization method performed well for 1 to 5 speakers, but worse for 6 to 10 speakers, as overlapping speech increased, with an overall *mean absolute error* (MAE) of 2. The CNNs showed promising results for all classes, with an overall MAE of 1. On a gold standard dataset, the CNN model was the only one to produce results, with an overall MAE of 2.

The CNN model could be feasible for the problem at hand, but a better dataset must be annotated to determine how well it would work on real conversations.

Keywords: convolutional neural network, speaker diarization, machine learning, audio analysis, neural network

Acknowledgements

To write our master's thesis during a world wide pandemic was unexpected, and challenging at times. We would like to give a big thank you to Pierre Nugues, who always made sure that we were coping with the circumstances, and always asked an extra time if we were okay.

We would also like to thank Colin Nordin Persson, our supervisor at Minut, for his support and insights regarding machine learning and audio analysis.

Contents

Chapter 1

Introduction

During the last couple of years, as the technology industry is moving towards *internet of things* (IoT) designs, smart homes have become a new trend. This means connecting home devices (such as washing machine, fridge, speakers etc.) to the home WiFi, and then, for example, receiving continuous updates from the machines into applications on your smartphone. Of course, home alarms are also included in this trend – however, in a time where personal integrity and privacy on the internet is discussed more and more, many security systems still rely on cameras (amongst other things) for home surveillance, thus essentially jeopardising privacy. In addition to this, home alarms are often expensive and require professional installation services (larmkollen.se, 2020).

The founders of the Minut company realised that there was an unexplored market for a small, modular, easy-to-install home alarm device, which did not use cameras. So, they created they Minut device, with the motivation being home awareness without cameras. This requires an analysis of motion, sound levels, temperature, and humidity, amongst other things (minut.com, 2020). By analysing that data, the Minut device gives the home owner an insight to the state of the property, thus giving the owner the possibility of making sure that everything is alright.

As the Minut device often is used within short-time rental properties, where there usually are limitations considering the number of people allowed to stay in the property, it would be of interest to implement a feature in the Minut device that gave the owner an approximation of the number of people present in their property. Thus, in this thesis, our main focus was the sound, where we aimed at examining if it is possible to quantify the number of unique voices, and thus persons, in a room, given an audio sample of the conversation. We approached this problem with two architectures: *convolutional neural networks* and *speaker diarization* which we evaluated on two speech corpora.

1.1 Background

The lack of cameras makes the Minut device ideal for short-term rental hosts, as the device gives the home owner an insight into the current state of their property. Moreover, the data that is collected and analysed never leaves the device nor is saved on - it is collected, analysed and deleted. This means that the rental hosts are not risking the privacy of the tenant.

This particular use case gives the start of this thesis problem formulation. Analysing the sound can give much insight into the state of the environment, such as if accidents happen (identifying and notifying events such as glass shatter or fire alarm) or how many persons are present in the environment. The latter is of interest as most short-term rental hosts have a maximum number of allowed tenants. By analysing the sound, the Minut device could possibly be able to get an estimation of how many persons have been heard during a time interval, thus being able to notify the home owner if something seems out of the ordinary.

1.1.1 Problem Formulation

The questions to be answered in this thesis are thus:

1. Can we identify *unique* speakers, given a finite length audio clip of people talking, and from that give an accurate estimation of the number of people in the room?

2. Moreover, is it possible to run the model on the Minut device, despite its limitations such as memory capacity?

1.2 Related Work

As the first method we tried in this thesis was using Mel frequency cepstrum coefficients (MFCC) in a convolutional neural network (CNN), the first part of related work will present articles related to that.

The second method we used was speaker diarization, thus the second part of the related works presents articles which are state of the art within the speaker diarization task.

1.2.1 Speaker Recognition

In 2010, Muda et al. examined the possibility of using MFCC for voice recognition tasks, specifically using *dynamic time warping* (DTW) techniques. DTW is a methodology to find similarities between time series that may differ in time or speed. Muda et al. used the MFCC as time series of voices and the DTW method to compare utterances to each other. They created a reference MFCC time series of a voice and then used DTW methods to compare other utterances to the reference, to determine if an utterance was made by the same person or not. These findings were positive and indicated that MFCC indeed could be used for speaker identification tasks.

While Muda et al. achieved positive results with MFCC and DTW techniques, Lukic et al. (2016) argue that the MFCC transform does not contain enough information about the voice attributes (such as pitch) to be able to be used directly for speaker recognition tasks. It

would rather need an additional algorithmic step such as clustering with Gaussian mixture models to work well.

The use of convolutional neural networks for this task has been tested more during the 2010s. Lukic et al. (2016) mention that a common algorithm is using MFCC transforms and convolutional neural networks (CNNs) with clustering algorithms to identify unique speakers. They, in turn, propose a method using spectrograms with CNNs. Their results represent the state of the art at the time. As with most articles within this subject, no overlapping speech was present, and only one person spoke in each sample.

More recently, Ashar et al. (2020) compared a CNN, deep neural network (DNN), and CNN-DNN hybrid approach to identify speakers in a noisy environment. In the first approach, they used spectrograms, in the second one, the MFCC transform, and in the third experiment, they used both. Ashar et al. performed some pre-processing on their signals such as noise reduction and silence removal. In the CNN approach with the spectrograms as input, they used one CNN both for feature extraction and classification and obtained an accuracy of 73%. Using a DNN and the MFCC transform, they reached an accuracy of 80%. They trained a third network, a DNN, on features combined from the first two approaches, which gave them an accuracy of 87.5%. Yet again, this method did not test MFCC with CNN on its own as voice recognition, and the dataset did not contain any overlapping speech. That means that the approach that we were interested in trying - being able to use overlapping speech together with the MFCC transform in a CNN - were not previously tested.

1.2.2 Speaker Diarization

Another more specific task is speaker diarization which tries to answer the question: *Who spoke when?*

Cyrta et al. (2017) tackled the speaker embedding part of this task by training a *recurrent convolutional neural network* (R-CNN) on four transforms of the magnitude spectrogram. The *constant Q transform* (CQT) performed best with a significant improvement over their baseline.

Zhang et al. (2019) proposed to replace the clustering module used by most diarization tasks by a trainable *unbounded interleaved-state recurrent neural network* (UIS-RNN). It uses speaker-discriminative embeddings to model different speakers in a parameter-sharing RNN. It integrates this model with a distance-dependent Chinese restaurant process (ddCRP) (Blei and Frazier, 2011) to be able to model an unknown number of speakers.

Fini and Brutti (2020) improved Zhang et al.'s approach by introducing a new loss function called *sample mean loss*, which better models speakers. They also introduced a way of estimating the parameter in the ddCRP which models the probability for a new speaker to join a conversation directly from the training data.

Moreover, Fujita et al. (2020) proposed a single *end-to-end diarization* model (EEND) instead of the traditional clustering of speaker embeddings. This model uses a neural network based on bidirectional long short-term memory (BLSTM) blocks to directly output speaker diarization results from a multi-speaker audio recording. This addresses some problems clustering-based diarization models suffer from. Firstly, traditional clustering-based diarization models uses unsupervised clustering models which can not be directly optimised to minimise diarization errors. Secondly, they have problems handling speaker overlap since the clustering algorithms assume that there only are one speaker per segment. Lastly speaker

embedding models have trouble adapting to real audio recordings with speaker overlap since they often are optimised on segments with one speaker and without overlap.

Fujita et al. (2019) improved Fujita et al.'s approach by exchanging the BLSTM blocks for self-attention blocks. The self-attention blocks are directly conditioned on all the input frames which enables them to capture both global speaker characteristics and local speech activity in a segment. This makes them better suited for speaker diarization compared to BLSTM blocks because a BLSTM block only captures local speech activity. It can not capture global speaker characteristics since it is only conditioned on its previous and next frame.

Although all these previous works show promising results, they all have quite large and complex models which would be unfeasible to deploy on a small IoT device with limited computing and memory capacity such as the Minut device. Therefore, we investigate if a small CNN is able to estimate the number of unique speakers in a audio clip and compare the result against a speaker diarization method.

1.3 Contributions

This thesis examined whether or not it was possible for a convolutional neural network (CNN) to identify that an audio sample contains several different unique speakers, and quantify the number of speakers. We compared a CNN based approach with MFCC transforms of the audio samples with speaker diarization techniques. To better mirror reality, two data sets were created algorithmically, one of which contains overlapping speech.

The main differences from earlier works were to investigate whether or not these methods were feasible for approximating the number of persons in a room, given that all persons were speaking, and while allowing overlapping speech – which most often is not the case for speaker identification or speaker diarization methods.

Chapter 2
Audio processing

The fundamental form of an audio signal is an acoustic waveform. To be able to analyse a waveform, it needs to be recorded into a format that a computer can handle. This means that the recorded signal has to be discrete. To get the discrete digital audio signal, the analogue signal is sampled at discrete points in time. The resulting wave representation is called a waveform and consists of a time series of discrete values of the wave amplitude at the sampling points. An example of a waveform is shown in Figure 2.1.

The quality of a digital signal is determined by the sampling frequency f_s, which is how often the signal is sampled, and the sampling precision, which is how close the sampling values are to the real wave amplitudes. This suggests that a higher sampling frequency gives a higher quality signal, but as the sampling frequency increases, so does the space required to store the waveform. Therefore, some upper limit must be set on the sampling frequency, so that the memory required for storage is feasible. Luckily the Nyquist–Shannon sampling theorem states that an audio signal can be reconstructed exactly from samples taken at twice the highest frequency in the input signal (Rabiner and Schafer, 2007, p. 98). This means that the sampling frequency only needs to be twice as high as the highest frequency of interest.

These raw waveform signals could be the input to a very deep neural network and give good classification results (Dai et al., 2017), but the scale and size of these networks makes them unfeasible for the Minut device. Therefore, this chapter describes the steps we took to pre-process the data to reduce its dimensionality.

2.1 Interpreting Speech

The human ear is specialised in interpreting human speech, and by studying how the ear works and how the human ear perceives sound, different representations of speech signals can be found. Since interpreting human speech is the main source of interest in this thesis, the pre-processing tries to mimic how the human ear perceives sound.

The human ear is most sensitive to frequencies in the interval 100 Hz to 6 kHz, where

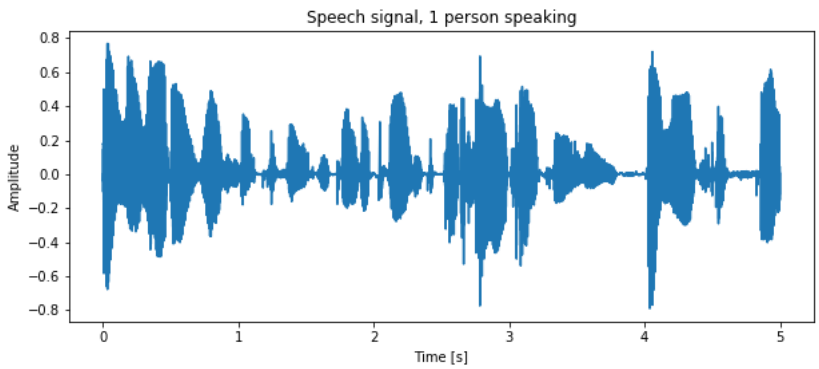


Figure 2.1: Example of the waveform of one person speaking.

most of the frequency range for speech is between 3 and 4 kHz (Rabiner and Schafer, 2007, p. 28). Consequently, frequencies far higher than 4 kHz are not of interest in this thesis and therefore a lower sampling frequency can be used. To be entirely sure that all frequencies of interest are captured, we used a sampling rate of 16 kHz, thus capturing frequencies as high as 8 kHz. Furthermore, speech signals can be considered stationary over time intervals of tens of milliseconds (Rabiner and Schafer, 2007, p. 34), which will prove useful when transforming signals into the time-frequency domain.

2.2 Fourier Transform

To be able to analyse the different frequencies in a continuous time signal $x(t)$, a frequency representation of the signal is needed. We carry this out with a *Fourier transform* which transforms the continuous function $x(t)$ from the time domain to a continuous function in the frequency domain. This transform is defined as:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt, \quad \omega \in (-\infty, \infty), \tag{2.1}$$

where $\omega = 2\pi f$ is the angular frequency (Proakis and Manolakis, 1996, p.242).

2.2.1 Discrete-Time Fourier Transform

For discrete time signals $x[n]$, there is the corresponding *discrete-time Fourier transform* (DTFT), which is defined as:

$$X(\omega) = \sum_{-\infty}^{\infty} x[n]e^{-i\omega n}, \quad \omega \in (-\infty, \infty), \tag{2.2}$$

which transforms the discrete time function $x[n]$ from the discrete time domain to the discrete frequency domain. Furthermore, the DTFT is periodic with a period of 2π since the discrete time signal is limited to frequencies within $(0, 2\pi)$. Hence, $X(\omega) = X(\omega + 2\pi k)$ is true for $k = 0, 1, 2, \ldots, N$ (Proakis and Manolakis, 1996, p. 253). This means that if the

DTFT can be solved inside the interval, it is solved for all frequencies, but since the signal $x[n]$ is of infinite length, generally it still can not be computed.

2.2.2 Discrete Fourier Transform

To be able to compute the transform for a discrete time signal, two things have to be changed from the DTFT. The first one is that it needs to be sampled on a finite set of angular frequencies $\omega_k = \frac{2\pi k}{N}$, $k = 0, 1, 2, \ldots, N-1$ which is possible since it is periodic. The second is that the signal has to be limited, i.e. of finite length. So instead consider the following finite discrete signal:

$$\hat{x}[n] = \begin{cases} x[n], & \text{for } 0 \leq n < N \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Plugging these two changes into the DTFT (equation 2.2) gives:

$$X[k] = \sum_{n=0}^{N-1} \hat{x}[n] e^{-i2\pi kn/N}, \quad k = 0, 1, 2, \ldots, N-1, \tag{2.4}$$

where X is a function of the frequency bin k. This equation is called the *discrete Fourier transform* (DFT) and is generally computable (Proakis and Manolakis, 1996, p. 401).

2.2.3 Short-Time Fourier Transform

When transforming the signal into the frequency domain using the DFT, all time-localization is lost. To instead get a time-frequency representation of the signal, the *short-time Fourier transform* (STFT) is used. First it splits the signal into short, possibly overlapping, frames, then the DFT is computed for each frame. The time-localization of the frame is then used to get the time-frequency representation of the signal. The STFT is defined as:

$$X(\omega, n) = \sum_{m=-\infty}^{\infty} x[n+m] w[m] e^{-i\omega m}, \quad n = 0, 1, 2, \ldots, N-1, \tag{2.5}$$

where $w[m]$ is a window sequence, i.e. a window through which the signal slides and $x[n]$ is the discrete time domain signal. Any choice of window function can be used but usual choices include the Hamming and Hann window functions. In the same way as for the DFT, frequency sampling is done to get the discrete version of the STFT which is defined as:

$$X[k, n] = \sum_{m=0}^{L-1} x[n+m] w[m] e^{-i2\pi km/N}, \quad k, n = 0, 1, 2, \ldots, N-1, \quad N \geq L, \tag{2.6}$$

where N is the number of frequency bins, L is the length of the window sequence, $w[m] \neq 0$ for $0 \leq m < L$ and $w[m] = 0$ otherwise (Oppenheim and Schafer, 2014, Sect. 3.4).

2.3 Mel Scale

Voiced speech and musical sounds have a structure that over short time intervals is fairly periodic, which is a quality known as pitch. Pitch is nonlinear and an attribute of sound that is subjective to humans but relates to the fundamental frequency of the sound. The *melody scale* or *mel scale* was constructed to follow how humans perceive pitch and is defined as:

$$M(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \quad \text{or} \quad M(f) = 1127 \log_e\left(1 + \frac{f}{700}\right), \tag{2.7}$$

where f is the fundamental frequency (Rabiner and Schafer, 2007, pp. 29-31). As humans can discern pitch changes better at low frequencies than higher, the Mel scale will more accurately mirror what the human ear can perceive.

To transform frequency to the Mel scale, filter banks are used. An example of 20 filter banks is shown in Figure 2.2, where the filter banks are linear on the Mel scale and thus logarithmic on the frequency scale.

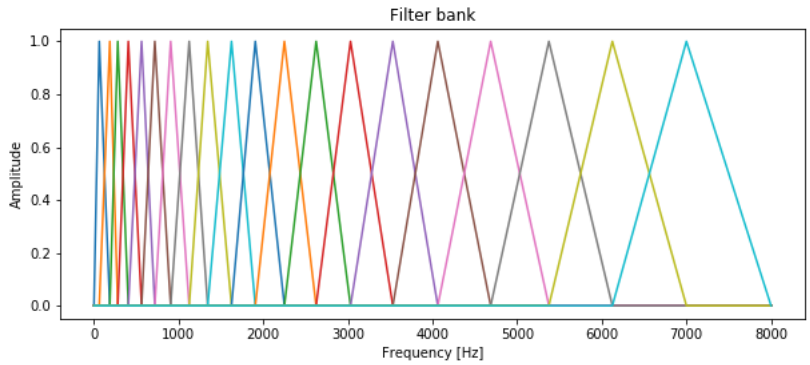


Figure 2.2: Mel filter bank, 20 filters linearly spaced on the Mel scale, thus logarithmic on the frequency scale.

The filter bank will be applied to the STFT of a signal frame, thus resulting in an indication of how much energy was in each filter range. As there are more filters at the lower frequencies, the differences at those frequencies will have higher resolution than for higher frequencies.

2.4 Cepstrum

The *cepstrum* was defined to be the inverse Fourier transform of the log magnitude spectrum of a signal. It is used when exploring the general concept of homomorphic filtering of signals combined by convolution. For a discrete-time signal, the cepstrum is defined as:

$$c_{\hat{n}}[m] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |X_{\hat{n}}(e^{i\hat{\omega}})| e^{i\hat{\omega}m} d\hat{\omega}, \tag{2.8}$$

where $X_{\hat{n}}(e^{i\hat{\omega}})$ is the STFT of the signal. The connection between cepstra and homomorphic filtering of convolved signals is that the cepstrum operator transforms a convolution into

an addition. Furthermore, these properties of cepstra are also what makes them useful for speech analysis since the model for speech builds on convolutions of the excitation, and the vocal tract impulse response. By using a cepstrum then the two convolved signals can be separated (Rabiner and Schafer, 2007, pp. 55-58). A short-time cepstrum can also be used in speech processing to detect periodicity in a signal which can be used to determine if there is someone speaking (Rabiner and Schafer, 2007, p. 65).

2.5 Mel-Frequency Cepstrum Coefficients

Taking advantage of the properties of cepstra and the filter banks previously described, the Mel-frequency cepstrum coefficients (MFCC) were created. They build on the knowledge of how the human ear works and weighs the short-time signal with a filter bank, emulating the human ears' properties with constant bandwidth for lower frequency and larger bandwidth for higher frequencies. The Mel-frequency spectrum is calculated following:

$$MF_{\hat{n}}[r] = \frac{1}{A_r} \sum_{k=L_r}^{U_r} |V_r[k] X_{\hat{n}}[k]|^2, \tag{2.9}$$

where $V_r[k]$ is the triangular weighing function, $X_{\hat{n}}[k]$ is the STFT of the signal, and A_r is a normalisation factor that ensures that a perfectly flat input spectrum gives a perfectly flat Mel spectrum.

Then a discrete cosine transform is applied for each frame according to:

$$mfcc_{\hat{n}}[m] = \frac{1}{R} \sum_{r=1}^{R} \log\left(MF_{\hat{n}}[r]\right) \cos\left[\frac{2\pi}{R}\left(r + \frac{1}{2}\right) m\right], \tag{2.10}$$

where R is the number of filters, Which gives the MFCC for the discrete-time signal. Often the $mfcc_{\hat{n}}[m]$ is evaluated for less coefficients than the number of filters (Rabiner and Schafer, 2007, pp. 70-72). Figure 2.3 shows a summary of the steps to achieve MFCC. Figure 2.4 shows an example signal and its corresponding MFCC representation.

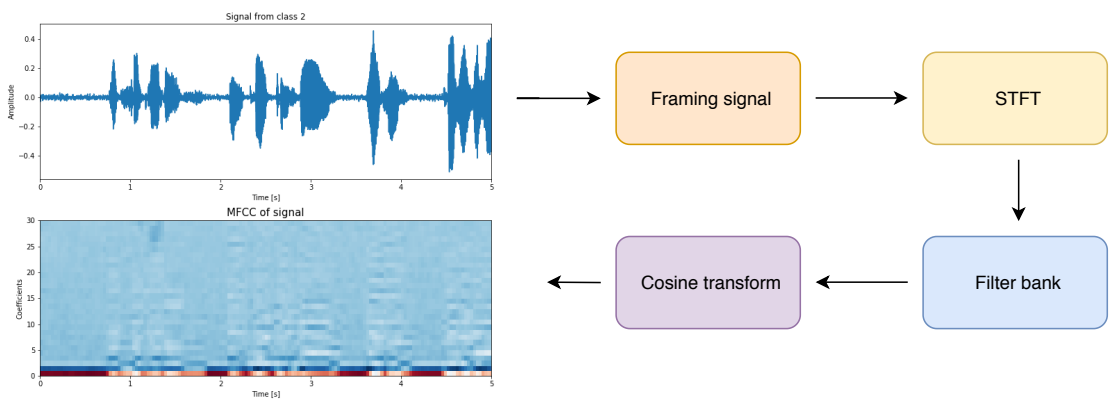


Figure 2.3: The steps from signal to MFCC of signal.

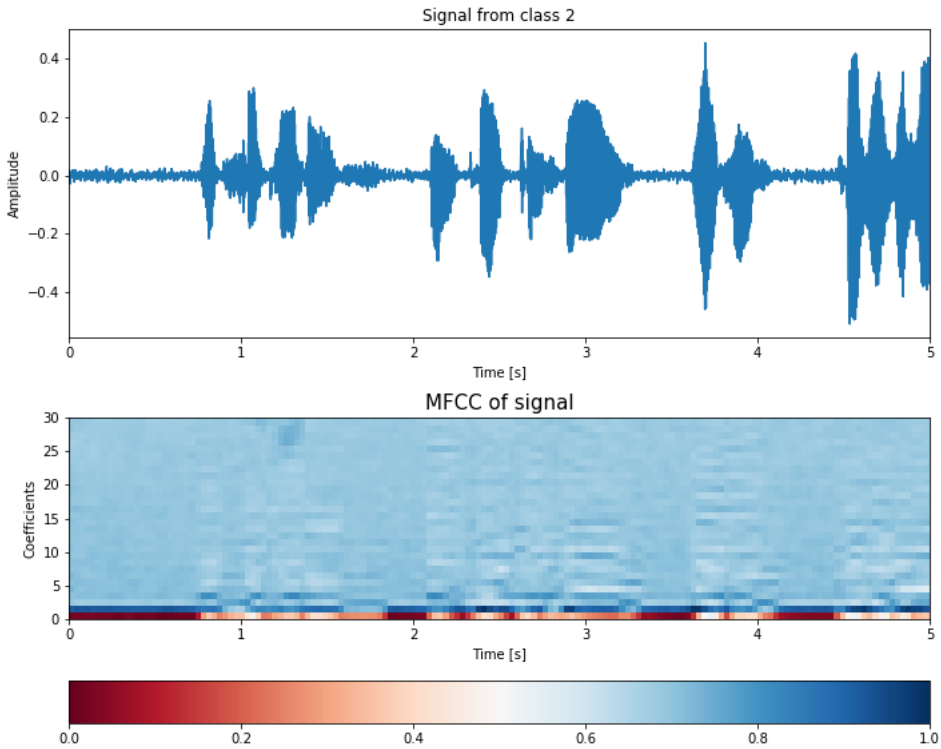


Figure 2.4: A signal and its corresponding MFCC representation. Class 2 means that there are 2 unique voices heard in the clip. It is possible to distinguish some patterns in the MFCC which correspond to the persons talking in the audio signal above.

Chapter 3
Neural Networks

The term artificial intelligence (AI) has, during the last couple of years, become a buzzword known not only to data scientists but also the general public. In popular science, it is a term often surrounded by mystery and utopian or dystopian futures. Machine learning (ML) is a type of AI, and this chapter will explain how deep learning works, which is the machine learning method used in this thesis.

Machine learning operates the opposite way of conventional programming. In conventional programming, a computer is supplied with data and rules to follow, and the computer takes the data, applies the rules, and gives the user an answer (like a calculator, where the data is the numbers and the rules are the arithmetics). With ML, the computer is supplied with data and the answers to each data point, and the computer will, after training, give the user the rules that it found (Chollet, 2017, p. 5). These rules can then be applied to give answers to new data.

3.1 Machine Learning Overview

What does machine learning mean? As defined by Mitchell (1997, p. 2):

> A computer program is said to learn from experience $\boldsymbol{E}$ with respect to some class of tasks $\boldsymbol{T}$ and performance measure $\boldsymbol{P}$, if its performance at tasks in $\boldsymbol{T}$, as measured by $\boldsymbol{P}$, improves with experience $\boldsymbol{E}$.

The task ($\boldsymbol{T}$), performance measure ($\boldsymbol{P}$) and data to gain experience from ($\boldsymbol{E}$) must be well-defined for the machine learning algorithm to work in practice. As an example, given a set of images of cats and dogs, a task defined as:

> Does the image contain a cat?

is a binary classification problem. Define also a performance measure, *percentage of correctly classified cat images*, and a machine learning model can be built.

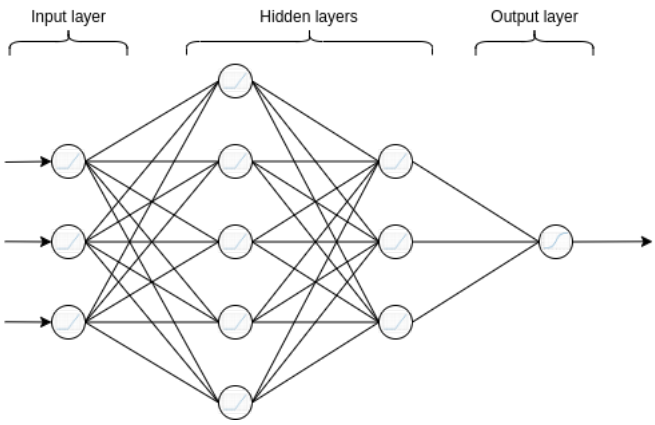


Figure 3.1: Visualisation of a small neural network. Each neuron in the input and hidden layers is depicted with a ReLu activation function, and the last neuron with a Sigmoid function.

When defining the task and the data set, it is important that the data set accurately represents future problems (Mitchell, 1997, p. 6). If the task at hand is to identify cars, training a model on images of cats and dogs will not make a good car classifying model. As such, the data set must contain as many possible variations of data that can be seen in the future as well: If a model is trained exclusively on images of cats with white backgrounds, it may not perform well when shown an image of a cat with a green background.

3.2 Neural Network Structure

The purpose of this section is to introduce the reader to a simple neural network before moving onto convolutional neural networks, as the principles are very much the same. We will focus on describing a linear regression algorithm.

Figure 3.1 shows a standard feed-forward neural network. It consists of an input layer, followed by hidden layers (i.e. all layers that are not input nor output layers), and finally ends with an output layer. Each layer consists of neurons and the whole constitutes a neural network (NN), as explained in Nielsen (2018, chap. 1).

Each layer consists of nodes representing the inputs, where each node in the layer is connected to all the nodes of the next layer. Each connection between two nodes has a weight.

The nodes are given at the construction of the network. Let us denote $\mathbf{x}_N = [x_1^N, x_2^N ... x_n^N]$, the n nodes of layer N and $\mathbf{x}_{N+1} = [x_1^{N+1}, x_2^{N+1} ... x_p^{N+1}]$, the p nodes of layer $N+1$. The weights are trainable parameters. Let us denote w_{ij}^N the weight of the edge connecting node i of layer N to node j of layer $N+1$. We represent the connection weights between a layer with a matrix $\mathbf{W}_N$. Each neuron also contains a bias, and we denote the bias of a layer N as $\mathbf{b}_N = [b_1^N, b_2^N .. b_n^N]$.

The inputs, the weights, and the bias together produce one output through the neuron's *activation function*. This output can then be forwarded to several other neurons. In the case of a sigmoid neuron, the activation function is the sigmoid function, defined as $\sigma(z) = 1/(1 + exp(-z))$, as found in Nielsen (2018, chap. 1), which means that the output o_1^{N+1} from one

neuron x_1^{N+1} is

$$o_1^{N+1} = \sigma\Big(\sum_i w_{i1}^N x_i^N + b_1^{N+1}\Big) = \frac{1}{1 + e^{\sum_i w_{i1}^N x_i^N + b_1^{N+1}}} \tag{3.1}$$

When an ML algorithm is learning, the weights and bias of each neuron are being modified such that the cost function $C(w, b)$ is optimised, as explained in Nielsen (2018, chap. 1). An example of such a cost function, specifically in the context of linear regression, is the mean squared error (MSE), as found in Goodfellow et al. (2016, p. 106). The MSE is defined as equation 3.2, where w are all the weights in the NN, b all biases, n the number of training inputs, and $y(x)$ represents predicted and true values for the model.

$$MSE(w, b) = \frac{1}{2n} \sum_x \| y^{pred}(x) - y^{true}(x) \|^2 \ . \tag{3.2}$$

3.3 Convolutional Neural Network

The convolutional neural network (CNN) performs exceptionally well in image classification tasks, and are thus often used within computer vision. One of the reasons that CNNs are so successful in image classifications is because they are translation invariant, meaning that if they have learned one feature located in one part of an image, they will recognise that same feature even if it occurs at another location in the next image (Chollet, 2017, p. 123). This is not how other NNs work, as they would have to learn each location that the feature could appear in, thus requiring larger and more complex data sets (Chollet, 2017, p. 123).

The building blocks of a CNN are, much like the NN presented in the previous section, neurons, weights and biases, used in an activation function, as explained in Nielsen (2018, chap. 6). Depending on the type of task, the cost function will be different, and as the inputs to CNNs are tensors, the representation will also be different to the simple network described above. The weights and biases are combined into a kernel, and the hidden layers in the NN are called feature maps.

In Figure 3.2, a simplified visualisation of how a kernel slides over an image (input feature map) to produce a feature map is shown. The kernel, spanning over several neurons, will use the values of the neurons, and according to its weights, biases and by using an activation function, will produce an output that is represented in a corresponding position in the resulting feature map, as explained by Nielsen (2018, chap. 6).

3.3.1 Definition of Convolution

To be able to understand how the convolution operation works in a CNN, we must first define convolution. Mathematically, the continuous convolutional operation is defined according to the equation below, as found in Wikström (2014, p. 212).

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t - \tau) g(\tau) d\tau \tag{3.3}$$

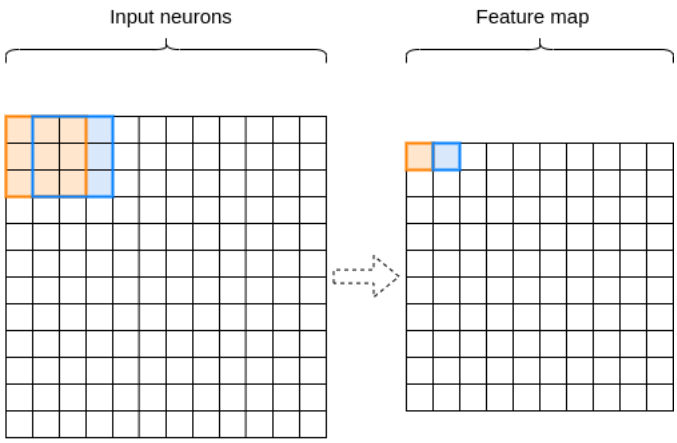


Figure 3.2: Example of how a kernel slides over an input image (where the input neuron values are the pixels) to produce one feature map.

The discrete convolutional operation is defined in equation 3.4 below, where a and b are sequences, as found in Wikström (2014, p.138).

$$(a * b)_k = \sum_{m=-\infty}^{\infty} a_{k-m} b_m \tag{3.4}$$

Generalizing this to a two-dimensional convolution, where convolution is to be computed between two matrices, gives the following equation;

$$C(i,j) = \sum_{m=0}^{Ma-1} \sum_{n=0}^{Na-1} A(m,n) * B(i-m, j-n) \tag{3.5}$$
$$\text{where } 0 \leq i \leq Ma + Mb - 1, \quad 0 \leq j \leq Na + Nb - 1.$$

In equation 3.5, the matrix A has the dimension (Ma, Na), and matrix B has the dimension (Mb, Nb) (MATLAB Computer Vision Toolbox, 2019).

3.3.2 The Convolutional Operation in CNNs

The kernel will, as briefly explained above, slide over the input image (input neurons) and produce a smaller output. The kernel sizes vary, but standard sizes are 3x3 or 5x5. When sliding the kernel over the input neurons, the step size (called *stride length*) can be altered as well. In this thesis we use a stride length of 1, as depicted in Figure 3.2.

As the kernel consists of weights and biases, the output from the activation function (say, a sigmoid function) will be according to equation 3.6, given a kernel of height and width n.

$$\text{output} = \sigma\left(b + \sum_{l=0}^{n-1} \sum_{m=0}^{n-1} w_{l,m} a_{j+l,k+m}\right) \tag{3.6}$$

In equation 3.6, b is the kernel bias, $w_{l,m}$ are the weights of the kernel and $a_{x,y}$ is the input activation situated in x, y (Nielsen, 2018, chap. 6). The signs are different in equation 3.6

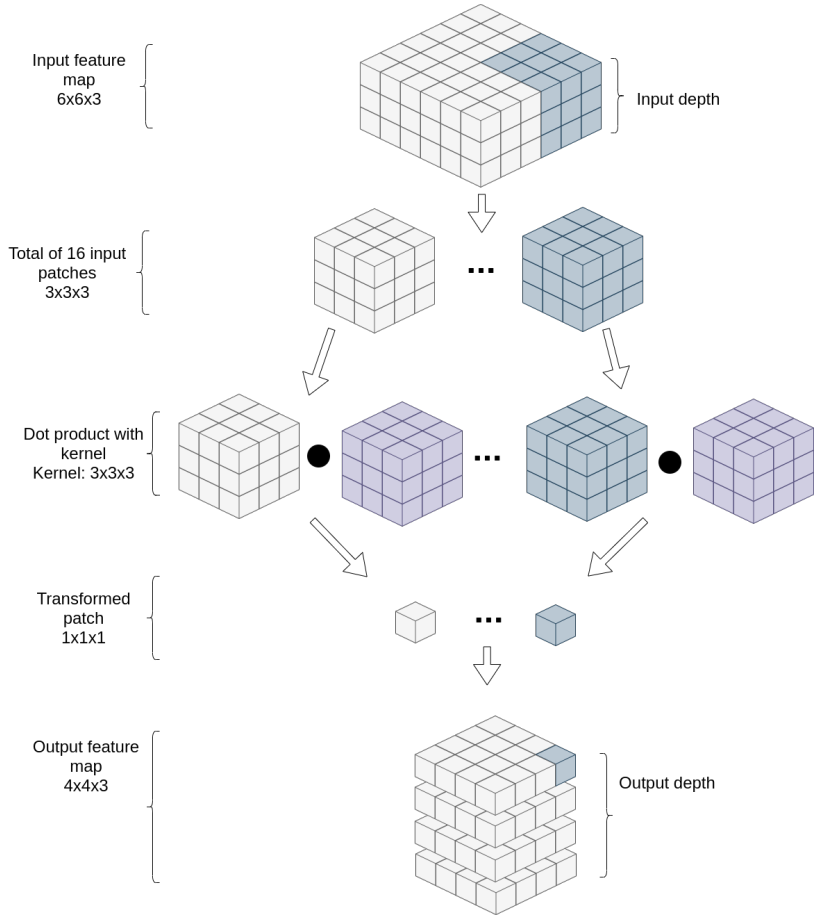


Figure 3.3: A visualisation of how one step of convolution works in a CNN. This is given an input map of depth 3, which could correspond to an RGB image (one map each for the colours red, blue and green). The depth of the output feature map is normally not the same as the input feature map.

compared to equation 3.5, only due to different annotations. This is where the convolutional operation happens in the CNN – the sum in the activation function in equation 3.6 is a convolution operation and corresponds to when the kernel is convoluted with input patches. The output is the value of the hidden neuron (see Figure 3.2 again, the kernel operating on the input neurons corresponds to the orange square in the matrix to the left and the following hidden neuron is the orange neuron in the matrix to the right).

Normally, several kernel operations are applied to create one output feature map. The number of kernels applied will correspond to the final output depth, where each kernel is different, thus producing layers of different information found in the same place of the image. The kernel is purple and visualised as a tensor in Figure 3.3, and the kernel must have the same depth as the input feature map.

In Figure 3.3, the lower right corner of the input feature map and the steps it goes through are highlighted in blue, to show how the spatial information is contained throughout the steps. The input feature map is divided into its patches, then each patch is multiplied via a tensor dot product with the current kernel to produce one output. This output will be applied to the activation function and the output from the activation function will result in

a value that will be represented in one layer of the output feature map. For each new kernel that the input patches are multiplied with, a new layer in the output feature map is created.

3.3.3 The Pooling Operation

In between each convolutional operation, a pooling layer can be added (though it is not always necessary). As an example of the pooling operation, the *max pooling* operation will halve the height and width of the output feature map from the convolutional operation, by letting a (normally 2x2) window slide over each layer with a stride of 2, selecting the max value to represent the window (Chollet, 2017, p. 127).

After the last pooling layer in the model, the feature map must be flattened into a vector. This is done sequentially; each value in the last pooling layer will become a value in the flat vector. See Figure 3.4 for a visualisation of max pooling on one layer, followed by flattening.

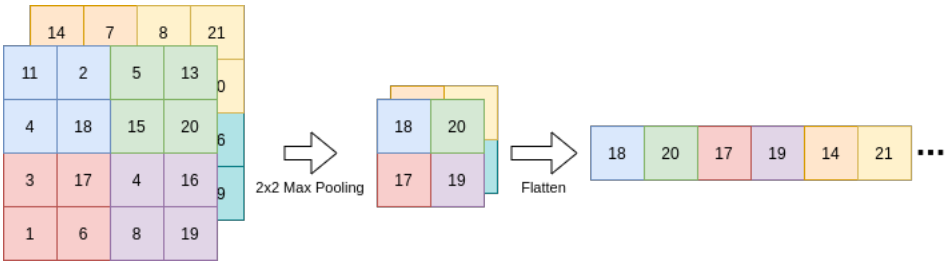


Figure 3.4: Visualisation of the max pooling operation and flattening.

After a flattening operation, a fully connected layer normally follows. A fully connected layer is a layer where each neuron is connected to each other neuron, as depicted in Figure 3.1.

3.3.4 Dropout Layer

Dropout layers can be added to a model in the case of overfitting the model when training. The standard approach to dropout layers is to set a fraction of the output parameters of a layer to zero, thus essentially forcing the model to forget parts of the training (Chollet, 2017, p. 109). The reason behind this is that the model can be very well trained on the training data, but learn it too well (a phenomenon called overfitting), leading it to not generalise well to new samples. By forcing it to forget parts of what it has learned, it can not rely on just a small part of the data which forces it to try to generalise (Chollet, 2017, p. 104).

3.3.5 Classification and Regression

Depending on the problem that a NN is used for, the output from the model can be different. The two main approaches are classification and regression, the former being used for when the problem has pre-defined solutions (such as *how many cats are there in this image?*, where the answer is already known to be some integer in the range *0-5*), whereas regression normally

is used for predicting more continuous values (such as the future price of a house) (Chollet, 2017, p. 85).

With regression, there will only be one output value from the model, and a regression model can be trained on the same data as for a classification problem. With classification, the outputs from the model will instead be as many as there are classes, for example 5 outputs for a classification problem with 5 possible classes. Each output is a probability for that class, and thus the output with the highest probability will correspond to the classified class.

For this thesis, as the problem is *How many persons are speaking during this time?*, both classification and regression are applicable. To create a more fluent model, however, regression has been the go-to. By not forcing the model to be restricted to a certain range of classes, it is easier to see how well it performs in its predictions – how far away from the correct class does it in fact predict? – and thus find inconsistencies in data or model design.

3.3.6 Metrics

When evaluating a NN model, different metrics can be used. In this thesis, as we are handling number of speakers, the metric used for developed models will be the mean absolute error (MAE). MAE will correspond to how far off the model predicts; as an example, if the model predicts 3.4 persons speaking for a sample where there were 3 persons speaking, the error would be 0.4. MAE takes all predictions and averages the absolute value of the errors for each class, and it follows that the closer to 0 that MAE is for each class, the better the model (Chollet, 2017, p. 87).

The most used metric in ML models is accuracy, which is simply "Out of all test samples, how many were correctly classified?".

3.4 Model Bias

When building a neural network with a specific task, it is important to spend time analysing the data set. This is because models that are trained on bad data sets will contain a lot of bias. A machine learning model is only as good as the data it is trained on, meaning that if the data contains bias, the final model will also contain bias (Chollet, 2017, p. 265).

There are countless examples of facial recognition systems performing worse on non-white faces compared to white faces, or voice recognition systems that do not understand accents or dialects of the English language that differ from the average white American male voice (Metz, 2020; Palmiter Bajorek, 2019). There is even an example of an Irish woman failing an automated English proficiency test due to her accent (Palmiter Bajorek, 2019).

3.4.1 Minimising Bias in a Model

There are several different types of bias to be observant of when building a data set. While some bias must be present as they mirror a correct reality (such as more quiet time in a room with few people in, compared to a room full of people), other bias must be examined, taken into consideration and maybe completely removed.

One type of bias are examples such as the above mentioned ones – i.e. bias that stem from socially constructed norms. As the audio analysis in this thesis focuses on separate

voices and not the actual words, it is *possibly* not as important to represent different accents and dialects of a spoken language (English) as it is in, for example, speech recognition tasks (Palmiter Bajorek, 2019). However, the fact that the only spoken language in the data sets is English may or may not affect the final performance of the model when it is generalised to more languages – further testing will deduce that. As there are voice differences between sexes, it would most probably be important to keep the gender distribution evenly balanced in the data set.

Another type of bias to be observant of is more specific to how each sample is built. When constructing a machine learning model, the programmer assumes that the model will look at a certain feature (or several features) $\boldsymbol{A}$ in a sample to classify it. As machine learning is nothing more than statistics, it is crucial that the data set does not display patterns that teach the model to look at other, hidden and possibly unwanted, features $\boldsymbol{B}$ or $\boldsymbol{C}$ when classifying. If these features $\boldsymbol{B}$ and $\boldsymbol{C}$ are derived from a wrongly built data set, the model will not have learned what the programmer thought and will thus generalise poorly.

By analysing the data set and its classes in all ways, types and forms, the programmer can minimise unwanted bias in the model. Convolutional neural networks also have the advantage of being easy to visualise, as it is classification of images, and methods such as grad-CAM (Chollet, 2017, p. 160) are available to see what the model looks at when running.

Chapter 4
Speaker Diarization

The *speaker diarization* (SD) problem, which is closely related to the first research question of this thesis, tries to answer the question:

> Who spoke when?

and is often solved in separate stages. In short, most speaker diarization methods split the process into the following steps:

1. Separate the input into a number of utterances where speech is detected using voice activity detection methods;
2. Extract speaker-discriminative embeddings ($\boldsymbol{i}$- or $\boldsymbol{d}$-vectors) from each utterance; and finally,
3. Cluster the embeddings (Zhang et al., 2019).

This chapter introduces the theory behind these steps, and in Section 5.4.2, we propose how these methods can be used to approximate the number of persons in a given room.

4.1 Voice Activity Detection

Given an audio sample containing a mixture of speech and noise, the purpose of a *voice activity detection* (VAD) algorithm is to detect where in the sample speech is present. Generally, the performance of a VAD algorithm is highly affected by the signal-to-noise ratio (SNR) (Moattar and Homayoonpoor, 2010), meaning that if there is less noise, the algorithm performs better. According to Ko et al. (2018) and Moattar and Homayoonpoor (2010), conventional VAD algorithms are based on heavy assumptions regarding the statistical characteristics of the signal (for example assuming that the noise is a stationary process, at least for some time). This leads to the algorithms being very sensitive to changes in the SNR.

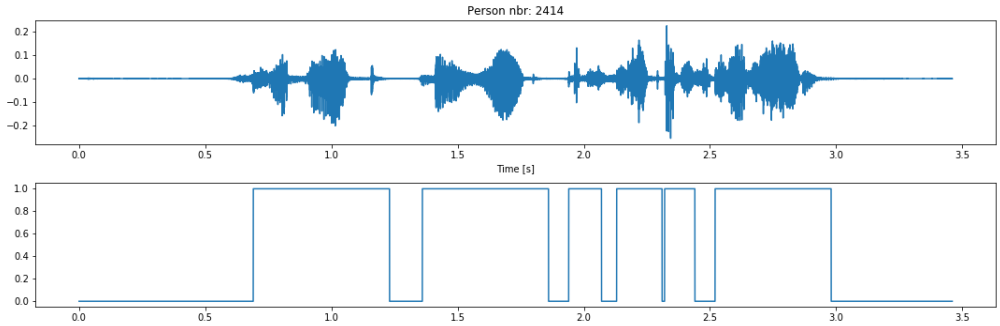


Figure 4.1: A speech signal as well as where speech is found (=1) and no speech is found (=0).

Table 4.1: A simple example of what embeddings can be.

	Height	Flower diameter	Colour	Nbr of flowers
Daffodil	60	12	0.12	1
Forgetmenot	30	1	0.6	7
Dandelion	10	4	0.18	1

In this thesis, we used the open source projects `py-webrtcvad` and a model based on a VAD found on Github by Usoltsev (2015), where `py-webrtcvad` is a wrapper of the Google VAD algorithm called WebRTCVAD.

The WebRTCVAD algorithm is based on the Gaussian mixture model for VAD, see Ko et al. (2018) for an explanation, and Figure 4.1 shows how `py-webrtcvad` works with a low sensitivity setting. We first tried to apply the `py-webrtcvad` algorithm to our dataset, but it could not handle the high SNR. We decided to use the other VAD for the diarization model instead. We chose it because it was easy to understand and to adapt to our needs. It works by comparing the total energy in the signal to the energy in the speech frequency band and uses a simple percentage to determine if there is speech in a signal or not.

4.2 Embedding

A speaker diarization system needs an embedding for each voice snippet to identify unique voices. The embedding acts as a multi-dimensional representation of voices and enables the model to differentiate between different speakers. It also enables clustering in a later stage (Variani et al., 2014).

There are two major types of embeddings in regards to speaker diarization methods – i-vectors and d-vectors. The former is the conventional way of creating embeddings and is normally based on Gaussian mixture models, and the latter uses deep neural networks instead (which gives the name d-vector) (Variani et al., 2014).

One way to think of embeddings is as vectors, where each entry corresponds to an attribute describing the object. As an example, imagine vectors of four parameters each, where each vector describes a flower. The four parameters could be height, flower diameter, colour (as a number between 1 and 0), and number of flowers per plant. Table 4.1 shows these embeddings.

Table 4.2: A similarity matrix on the flowers. Daffodil and Dandelion are deemed as the most similar flowers

	Daffodil	Forgetmenot	Dandelion
Daffodil	1	0.96	**0.98**
Forgetmenot	0.96	1	0.93
Dandelion	**0.98**	0.93	1

Part of the idea behind using embeddings is being able to compare otherwise complex entities to each other. This is usually done using the cosine similarity, defined in Equation 4.1, where **A** and **B** are vectors. Using the cosine similarity on the embedding vectors in Table 4.1, the most similar flowers would be Daffodil and Dandelion, as presented in Table 4.2.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4.1}$$

4.2.1 D-Vectors

A d-vector is, as mentioned above, created by a deep neural network. The type of network used for this could be *long short-term memory* networks (LSTMs), which are recurrent networks (as opposed to the simple feed-forward network in Figure 3.1). Recurrent networks have a *memory*, as opposed to feed-forward or convolutional networks (Chollet, 2017, p. 196), as they keep an internal state containing information on what they have previously seen. Wan et al. (2017) showed that LSTM networks performed better than other networks in this task. The main advantage of using d-vectors instead of i-vectors is that the model can be trained on a large amount of data, thus making the model more resistant to a variety of acoustic environments (Zhang et al., 2019).

In the model from Wan et al. (2017) the d-vector embedding for an utterance is the normalised vector from the output of their trained LSTM network when the utterance is used as the input. For other embedding models, the d-vector embedding for an utterance may be the vector extracted from the last hidden layer of the network after using the utterance as input, as described in Variani et al. (2014).

The d-vectors are supposed to develop features during training that discriminate between different voices. That means that there should be similar vectors for one person saying different things, but different vectors for two different people saying either the same or different things. To accomplish this efficiently, special loss functions for the networks have been developed and Wan et al. (2017) describe one of them.

4.3 Clustering

We can apply a clustering algorithm to cluster embeddings, where the algorithm looks at similarities and differences in the embeddings. The goal of this is to find clusters, where different speakers are distinct from each other, thus creating as many clusters as there are

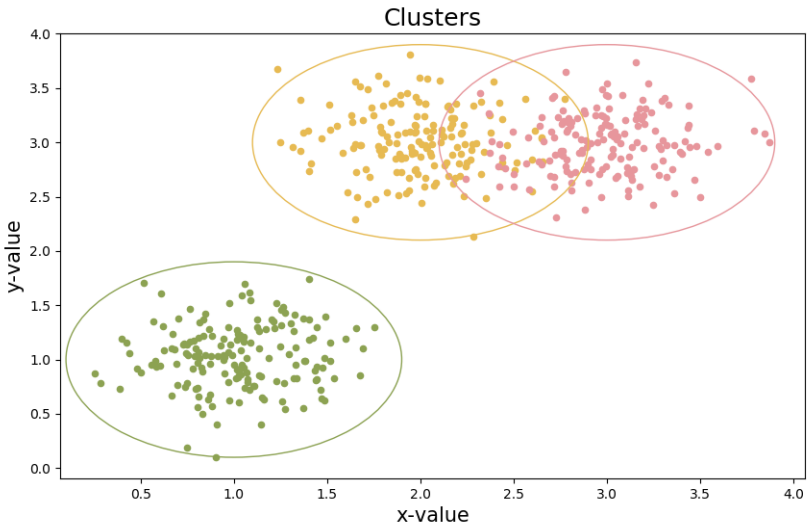


Figure 4.2: Cluster plot.

speakers. Normally, the clustering algorithms are unsupervised, however there are examples of supervised clustering modules, as explained in Zhang et al. (2019).

In this thesis, we used a supervised clustering module called *unbounded interleaved-state recurrent neural networks* (UIS-RNN); see Zhang et al. (2019) for a description of UIS-RNN. When the clusters are formed, new embeddings can be assigned a cluster and with that provide information on which speaker the embedding corresponds to.

Normally in speaker diarization methods, overlapping speech is completely removed when doing embeddings and clustering. As overlapping speech is prominent in the data sets for this thesis, we included it.

Figure 4.2 shows an example of how three clusters could look like. In the figure, some clusters are overlapping and are quite spread out. For speaker diarization, ideally, more delineated and separated clusters would be desirable. This would enable the algorithm to identify different separate speakers more easily.

Chapter 5
Method

This chapter will present CNN and SD methods we developed, together with tools and datasets. We will first present the baseline, then an analysis of the different datasets, and finally an explanation of the CNN and SD models.

5.1 Baseline

The baseline was inspired by Stoter et al. (2018), in which classes 0 to 10 were built using the LibriSpeech clean dataset (Panayotov et al., 2015), transformed via STFT, and then classified using an LSTM network. In this work, we built the baseline using the same dataset, but with an MFCC transform and a regressional CNN instead. The classes 0 to 10 consisted of 0 to 10 voices in overlapping speech, where each sample was 5 seconds long. Creating a baseline for the thesis enables us to estimate and compare final results of other models.

5.1.1 Tools

To create the MFCCs, we used the library `librosa` (McFee et al., 2020). Each signal, being 5 seconds long and with a sample rate of 16,000 Hz, was thus 80,000 samples long. The MFCCs of each signal were computed with 30 coefficients using the `librosa` function `mfcc`. We used the default framing and hop length, which meant a frame length of 128 ms and a hop length of 32 ms. The resulting size of each sample tensor was thus `(30,157,1)`.

When working with the CNN model, we used `Keras` (Chollet et al., 2015), which is an API based on `Tensorflow` (Abadi et al., 2015). For analysis of the model results, we used functions from `sklearn` (Pedregosa et al., 2011).

Table 5.1: Baseline model summary.

Layer name (Type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 155, 64)	640
max_pooling2d (MaxPooling2D)	(None, 14, 77, 64)	0
conv2d_1 (Conv2D)	(None, 12, 75, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 6, 37, 128)	0
conv2d_2 (Conv2D)	(None, 4, 35, 256)	295168
max_pooling2d_2 (MaxPooling2D)	(None, 2, 17, 256)	0
dropout (Dropout)	(None, 2, 17, 256)	0
flatten (Flatten)	(None, 8704)	0
dense (Dense)	(None, 256)	2228480
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257

Total params: 2,598,401
Trainable params: 2,598,401
Non-trainable params: 0

Table 5.2: Overview over the different created datasets.

Dataset	Classes	Specifications
1	0-5	No overlap, minimised bias
2	0-10	Overlap, biased, mirroring gold standard
Gold standard	1-5	Recording from coffee session

5.1.2 Model

We kept the model for the baseline simple, consisting of only three convolutional layers of size 3×3, with stride 1. The two dropout layers were of 25% each. Table 5.1 shows the complete model. Running the model 150 times gave the results presented in Figure 6.1 in Chapter 6.

5.2 Creating Datasets

During the development of the baseline, we realized that the dataset from Stoter et al. (2018) was not representative of a realistic sound scene. Every person was speaking at the same time, which is not how a dialogue looks normally. In addition to that, we wanted to be able to test our models on longer samples. As such, we decided to create new datasets algorithmically.

The sections below present and analyze each dataset. Table 5.2 shows a summary of the datasets.

5.2.1 Gold Standard

For us to be able to test the developed models on real life data, we recorded a coffee session between us and friends. The recording lasted 9 minutes, and was done by using a smartphone

on the table between speakers. We then annotated the recording by hand, and the classes 1-6 could be extracted from 194 15-second clips, using a hop length of 3 seconds. Because the hand-annotation was very time consuming, we only used one recording from one coffee session, thus only generating classes 1 to 6.

From the annotation, we extracted an analysis that possibly could represent the way our small friend circle's dialogue culture looks like. The analysis of this *gold standard* (GS) dataset is presented in Section 5.2.4 below, together with dataset 2.

5.2.2 Tools

When creating the datasets, we used tools like `webrtcvad` (Wiseman, 2019) and `librosa` (McFee et al., 2020). We used `webrtcvad` for finding speech, and `librosa` to read and save the samples.

5.2.3 Dataset 1

The first dataset consisted of classes 0 to 5. We minimised all bias that we could find to try to force the CNN to identify the number of speakers only based on properties found in each unique voice during each clip. This led to the dataset containing no overlapping speech at all. Other bias that we found included: average quiet time per class, average number of clips that were present in each class, unique person distribution over the dataset and the classes, as well as gender distribution over the dataset.

Each sample was 7 seconds long, with class 0 being only background noises and class 5 being 5 unique voices speaking at different times in the sample. The voices used were from the LibriSpeech clean speech dataset (Panayotov et al., 2015).

The making of a sample of a class for this dataset consisted of two steps:

1. The first step was to go through all speech files from the LibriSpeech dataset. We divided each speech file into smaller speech chunks, and saved each chunk with its corresponding speaker. To divide the speech files into chunks, we used the python open source package `webrtcvad` (Wiseman, 2019)

2. The next step was mixing together random speech chunks of random speakers, thus creating samples of classes 1-5.

Figure 5.1 shows an example of class 5, with no overlapping speech and 5 unique voices. Again, when using the `librosa` function `mfcc` on the samples, we set the frame length as 128 ms and the hop length as 32 ms. This resulted in the sample tensors being of size `(30,219,1)`.

Analysis of Dataset 1

Table 5.3 shows how many samples we created for each class, as well as the average number of speech clips and quiet time, where quiet time is defined as when no voice is heard (which means that only background noise may be present). Class 0 was limited by the amount of background clips found in the open source projects `scaper` (Salamon et al., 2017) and `MS-SNSD` (Reddy et al., 2019).

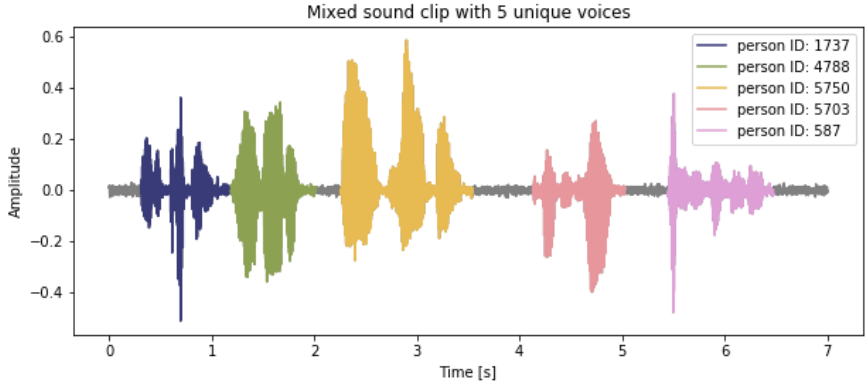
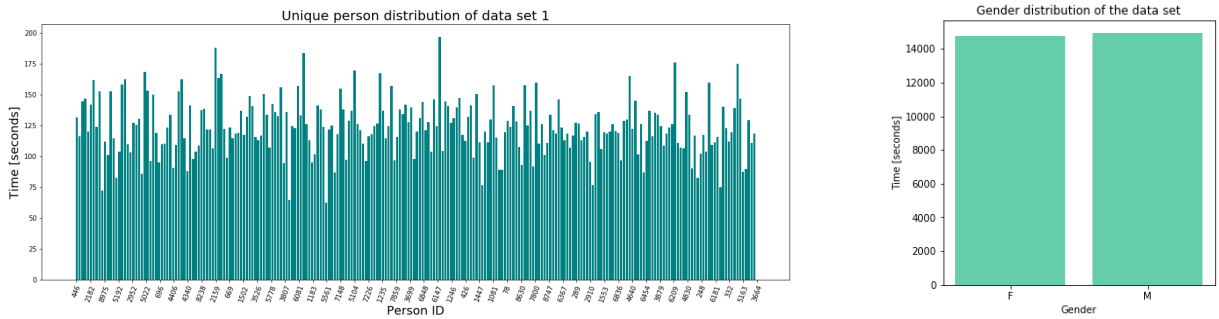


Figure 5.1: Example signal of class 5, from the non-overlapping and minimised bias dataset.

Figures 5.2(b) and 5.2(a) show gender distribution as well as unique person distribution over the dataset. These figures show that no bias towards any gender or any unique speaker(s) should be prominent in the final model.

Table 5.3: Table over general and quiet time statistics of the dataset with respect to each class.

	General		Quiet time	
Class	Samples	Avg. amount of clips	Mean [s]	Variance [s^2]
0	976	0.0	7.0	0.0
1	1200	5.0	2.0504	$791 \cdot 10^{-6}$
2	1200	5.0	2.0515	$798 \cdot 10^{-6}$
3	1200	5.0	2.0512	$840 \cdot 10^{-6}$
4	1200	5.0	2.0504	$809 \cdot 10^{-6}$
5	1200	5.0	2.0517	$853 \cdot 10^{-6}$



(a) Unique person distribution. Not all unique ID:s are shown as labels on the x-axis.

(b) Overall gender distribution.

Figure 5.2: These images show the total speaking time over the dataset with regards to unique speakers and gender.

5.2.4 Dataset 2

For the second dataset, we allowed some bias, such as how much overlapping speech each sample contained. Since we had no data to base the sample statistics on, we had to assume two things:

1. that the quiet time would decrease with classes, i.e. that if there is only one person speaking, there is generally more quiet time compared to 10 persons speaking,

2. that the amount of overlapping speech should increase with the the number of people speaking in the clip, to better mimic reality.

The algorithm for this dataset was slightly different than for dataset 1, and below we explain the algorithm for creating a sample of a class;

1. We decide the class of the sample, for example 3, then 3 unique person ID:s from the LibriSpeech dataset are picked at random.

2. Each unique person that should be present in the sample is assigned a random amount of utterances in the range $(1, 4)$; each of a random length and placed at random places in the clip (however one person cannot overlap himself/herself).

3. Each utterance is extracted randomly from the persons LibriSpeech signals, and `webrtcvad` is *not* used – instead, a random part of the waveform is extracted, to fit the earlier defined length of utterances.

4. We save only the samples that fulfil the conditions on the amount of overlap and quiet time per class.

A schematic image of how one of these samples could look is shown in Figure 5.3. Class 0 consisted of the same samples as for dataset 1. However since the samples were 15 seconds instead of 7, they were significantly fewer. Again, when using the `librosa` function `mfcc` to create the MFCC transformation of each sample, we set the frame length as 128 ms and the hop length as 32 ms. This resulted in the sample tensors being of size `(30,469,1)`.

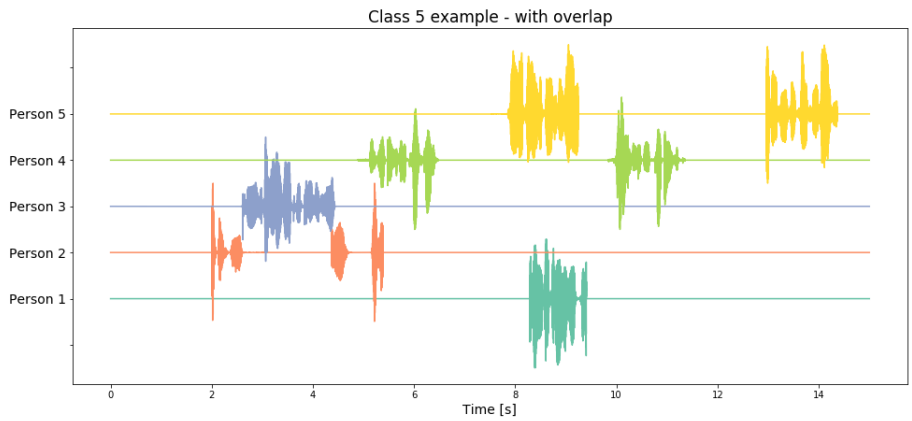
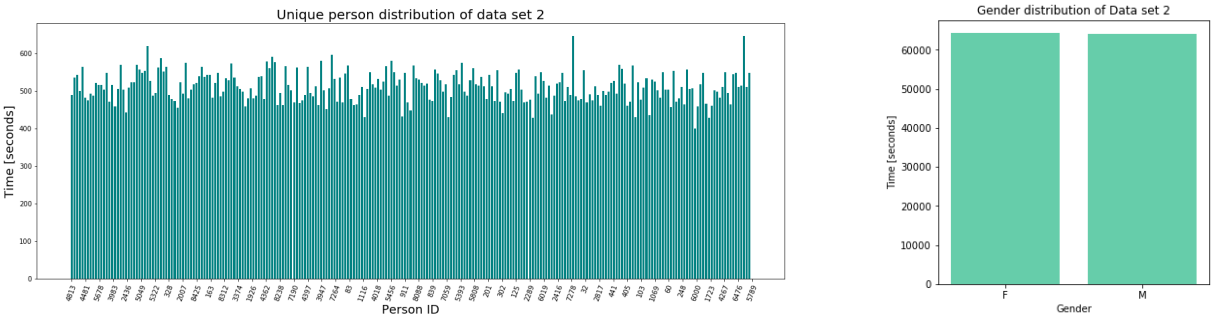


Figure 5.3: An example of class 5 with overlapping speech.

Analysis of Dataset 2

The purpose of this dataset was to have something that could compare to the GS dataset, essentially to answer the question if this method could be feasible for the problem at hand. As such, the analysis of this dataset is compared to the analysis of the GS dataset.

Again, we controlled the gender distribution and total speaking time for each person, as we still did not desire gender bias or bias towards one speaker. These results are shown in Figures 5.4(a) and 5.4(b).



(a) Unique person distribution. Not all unique ID:s are shown as labels on the x-axis.

(b) Overall gender distribution.

Figure 5.4: These images show the total speaking time over dataset 2 with regards to unique speakers and gender.

Table 5.4 shows some comparison between the two datasets regarding the overlapping speech statistics. For classes 7-10, the mean overlap values were more or less linearly extrapolated from classes 1-6, since we had no real-life data for these classes to start from. We only controlled the mean overlap value when creating the dataset, thus explaining why the variances between GS and dataset 2 are different. Some differences in the statistics were allowed when creating dataset 2, especially for the quiet time metric, to not overfit the final model to statistics from only one conversation (the gold standard), thus hoping for a more generalised model. In Table 5.5, the quiet time metric is compared between the datasets.

5.3 Convolutional Neural Networks

We tested two different approaches. The first one consisted of different regressional convolutional networks, and they are presented in this section.

5.3.1 CNN Models

When evaluating the CNN approach, we tested different models on the different datasets. In Table 5.6, a summary of which models were tested on which datasets is shown. After the baseline, we tested three different modifications to the baseline model, here called Models 1 to 3. These models are presented below in Tables 5.7 to 5.9. In each training, 50 samples from the gold standard dataset was appended to the validation set.

Table 5.4: Table of overlapping speech statistics in Golden Standard and dataset 2.

	Dataset 2 overlap			Gold Standard overlap		
	Mean	Variance	Samples	Mean	Variance	Samples
Class 0	0.0	0.0	444	–	–	–
Class 1	0.0	0.0	1073	0.0	0.0	3
Class 2	0.9383	0.1428	1000	1.3166	0.6872	13
Class 3	1.2432	0.1653	1000	1.8260	2.6170	63
Class 4	2.4260	0.5773	1000	3.3205	5.6752	53
Class 5	2.9080	0.6245	1000	4.1756	8.6475	43
Class 6	3.8650	0.7536	1000	5.41702	16.0435	15
Class 7	5.4576	1.3051	1000	–	–	–
Class 8	6.4331	1.5031	1000	–	–	–
Class 9	7.3925	1.6203	1000	–	–	–
Class 10	8.4700	1.6189	948	–	–	–

Table 5.5: Table of quiet time statistics in Golden Standard and dataset 2.

	Dataset 2 quiet time			Gold Standard quiet time		
	Mean	Variance	Samples	Mean	Variance	Samples
Class 0	15	0.0	444	-	-	-
Class 1	9.2342	4.4469	1073	1.0045	1.4046	3
Class 2	7.4571	3.3693	1000	2.5191	5.5437	13
Class 3	6.6683	3.0208	1000	3.8207	5.5488	63
Class 4	5.8322	2.4541	1000	2.8071	5.2280	53
Class 5	5.3831	1.9808	1000	2.5619	3.6688	43
Class 6	4.8104	1.6337	1000	2.1913	3.9581	15
Class 7	4.4295	1.6331	1000	-	-	-
Class 8	4.0624	1.4183	1000	-	-	-
Class 9	3.7960	.3790	1000	-	-	-
Class 10	3.4730	1.2021	948	-	-	-

Table 5.6: A summary of which models were tested on which datasets.

	Audiolabs	Dataset 1	Dataset 2	Gold Standard
Baseline model	x	x	x	x
Model 1			x	x
Model 2			x	x
Model 3			x	x

Table 5.7: Model 1 summary.

Layer name (Type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 467, 128)	1280
max_pooling2d (MaxPooling2D)	(None, 14, 233, 128)	0
conv2d_1 (Conv2D)	(None, 12, 231, 256)	295168
max_pooling2d_1 (MaxPooling2D)	(None, 6, 115, 256)	0
conv2d_2 (Conv2D)	(None, 4, 113, 512)	1180160
max_pooling2d_2 (MaxPooling2D)	(None, 2, 56, 512)	0
flatten (Flatten)	(None, 8704)	0
dense (Dense)	(None, 256)	29360640
dense_1 (Dense)	(None, 1)	513

Total params: 30,837,761
Trainable params: 30,837,761
Non-trainable params: 0

Table 5.8: Model 2 summary.

Layer name (Type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 467, 128)	1280
max_pooling2d (MaxPooling2D)	(None, 14, 233, 128)	0
conv2d_1 (Conv2D)	(None, 12, 231, 256)	295168
max_pooling2d_1 (MaxPooling2D)	(None, 6, 115, 256)	0
conv2d_2 (Conv2D)	(None, 4, 113, 512)	1180160
max_pooling2d_2 (MaxPooling2D)	(None, 2, 56, 512)	0
dropout (Dropout)	(None, 2, 56, 512)	0
global_average_pooling2d (Gobal Average Pooling)	(None, 512)	0
dense (Dense)	(None, 256)	262656
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	513

Total params: 1,739,777
Trainable params: 1,739,777
Non-trainable params: 0

Table 5.9: Model 3 summary.

Layer name (Type)	Output Shape	Param #
conv2d (Conv2D)	(None, 29, 468, 2)	10
conv2d_1 (Conv2D)	(None, 28, 467, 4)	36
conv2d_2 (Conv2D)	(None, 27, 466, 8)	136
conv2d_3 (Conv2D)	(None, 26, 465, 16)	528
max_pooling2d (MaxPooling2D)	(None, 13, 232, 16)	0
conv2d_4 (Conv2D)	(None, 12, 231, 32)	2080
max_pooling2d_1 (MaxPooling2D)	(None, 6, 115, 32)	0
conv2d_5 (Conv2D)	(None, 5, 114, 64)	8256
max_pooling2d_2 (MaxPooling2D)	(None, 2, 57, 64)	0
global_average_pooling2d (Global Average Pooling)	(None, 64)	0
dense (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 1)	65

Total params: 15,271
Trainable params: 15,271
Non-trainable params: 0

For Model 1, we used the baseline model as a base. We added more filters, and the drop out layers were removed. The fact that there were larger samples and more filters resulted in the dramatic increase of trainable parameters compared to the baseline model.

In Model 2, we once again used the baseline model as a base, now testing the impact of a global average pooling layer instead of a flattening layer. In this model, we increased the number of filters in each convolutional layer.

Model 3 we optimised for memory, meaning more convolutional layers and as many max pooling layers as was possible for the sample inputs. As can be seen, the total amount of trainable parameters is much smaller than for the other models, resulting in the trained model being smaller as well.

For the training of the models, a learning rate of `lr = 1e-5` was used together with a `ReduceLROnPlateau` (with `factor=0.5` and `patience=5`). Because of the size of the samples, a batch size of 4 was used, and `EarlyStopping` with `patience = 10` was applied as well.

5.4 Speaker Diarization

In this section, we describe the speaker diarization method. The speaker diarization method is the second approach tested on dataset 2.

5.4.1 Resemblyzer on Overlapping Data

To investigate how well the pre-trained embedding model from `Resemblyzer` handled overlapping speech, we chose 10 clips from 10 different speakers from LibriSpeech. First the model was fitted with the 100 non-overlapping clips. Then each of the clips from on person were overlapped on each clip from every other speaker which resulted in 4500 more clips.

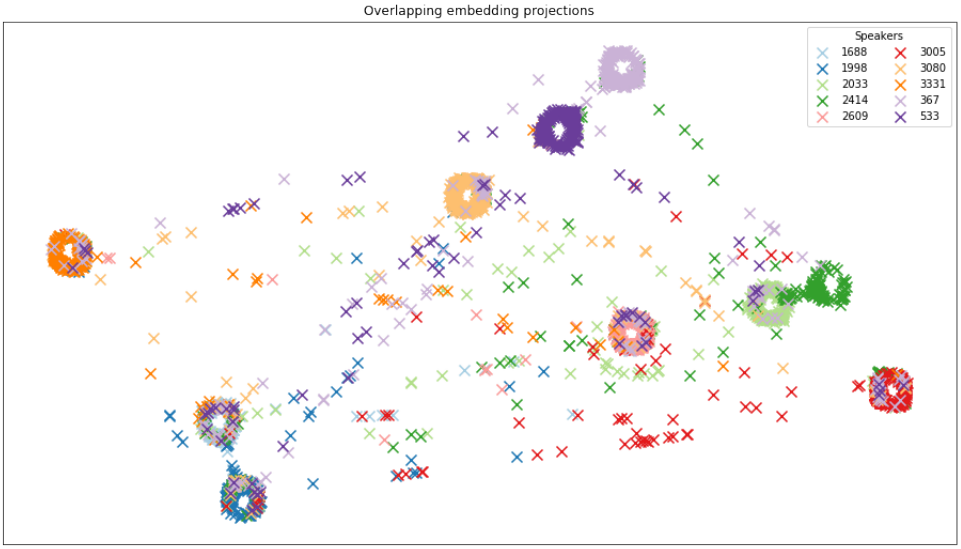


Figure 5.5: The projected embeddings of the overlapped speech from each speaker, where the label for each point is the supposed dominant speaker.

Each of them were then embedded by the model and projected. The resulting projections is shown in Figures 5.5 and 5.6.

The figures shows us that `Resemblyzer` handles overlapping speech surprisingly well since most of the overlapping clips are clustered around the correct cluster. Therefore, it should be well suited for our problem where overlapping speech is abundant but still needs to be embedded to the correct speaker.

5.4.2 Speaker Diarization Model

The speaker diarization model is built as shown in Figure 5.7. We chose to use the pre-trained embedding model which is about 16.3 MB large from the `Resemblyzer` package (Wan et al., 2017). It handles both the pre-processing and embedding of signals. Therefore, the speaker diarization model had two parts where we could experiment with different parameters. Those parts were the VAD and the clustering model.

After we extensively tested different parameters to use for the VAD we settled on the following:

1. We set the parameter `sample_window` to 20 ms.
2. We set the `sample_overlap` to 10 ms.
3. We changed the `speech_energy_threshold` to 0.6 which means that 60% of the energy needs to be within the speech band for the sample to be considered speech.
4. We decreased the lower boundary of the speech band `speech_start_band` to 200 Hz, and finally
5. We increased the higher boundary `speech_end_band` to 8000 Hz.

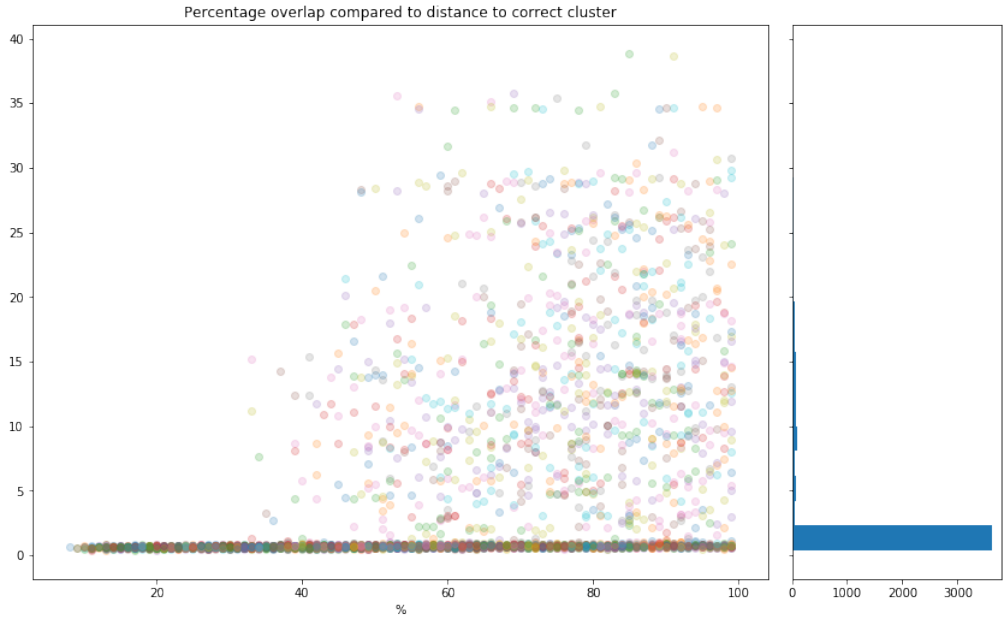


Figure 5.6: To the left is a scatter plot where each point is an overlapping clip. The y-axis shows how far away the sound clip is from the correct cluster and the x-axis shows the percentage of overlap it contained. To the right is a bar graph showing how many clips ended up a specific length from the correct cluster. It was included since it is hard to see the quantities in the scatter plot.

We chose these settings because they work well for our dataset. Moreover, we discarded the smoothing in the VAD and created our own instead. It iterates over all the sample windows, checks a slice from the beginning of the current window and 1200 ms forward to see if 20% of the sample windows inside the slice are labelled as speech. If there are more than 20% labelled as speech, it changes all the labels within the slice to be labelled as speech. Then the iteration continues at the first sample window after the slice. On the other hand, if there are less than 20% labelled as speech the iteration continues to the next sample window and does the same test again.

After our smoothing we have a number of 1200 ms long slices which should be the parts with speech from the signal. To label these slices with the correct speaker we check in the annotation for the dataset who speaks the most during the slice and assigns that label. If two or more people spoke the same amount, it prioritises people that do not have a previous slice from the signal labelled as them.

For the clustering and diarization model, we used `UIS-RNN`; see Section 4.3. We changed a few parameters:

1. We set `enforce_cluster_id_uniqueness` to false since the ids for the speakers persisted between different clips.

2. We changed `grad_max_norm` to 10 instead of 5.

3. We lowered `learning_rate` to 0.0005, and finally

4. We lowered `train_iteration` from 20,000 to 500.

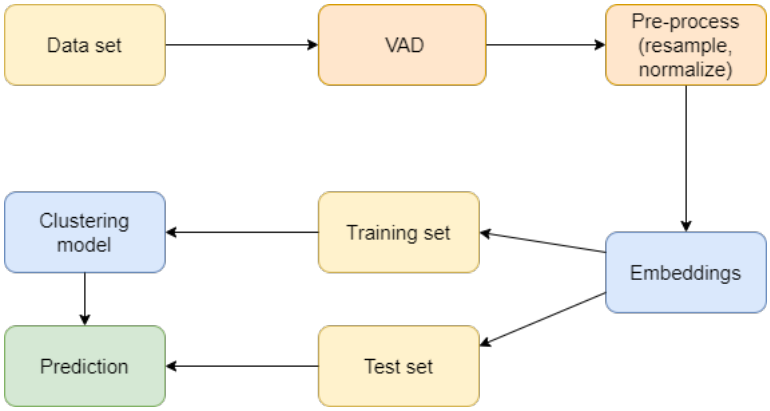


Figure 5.7: Flow chart of how the speaker diarization method works

Then we trained and tested the model 6 times shuffling the dataset between each run. We used 80% of the dataset as training data and 20% as testing data. To get the final result we took the average MAE of each class over all six runs. Finally we used the result to generate a plot over the different MAEs for each class.

We only got results for the diarization method on dataset 2 and not on the gold standard. The reason is that the signal-to-noise ratio (SNR) was inconsistent in our gold standard dataset, and the rudimentary VAD we used was not good enough to be able to separate speech and noise.

Chapter 6
Results

In this section, we present the results from the baseline, convolutional neural networks and speaker diarization methods. The baseline consisted of only one model, whereas we tested three additional CNN models and one speaker diarization method on the built datasets.

6.1 Baseline

As we trained the baseline model 150 separate times, the results are presented as histograms of the mean absolute error (MAE) of each class in Figure 6.1. As can be seen, every class had an MAE below 2, and that even for the higher classes such as 9 or 10 the overall error was only approximately 1.50. As this baseline was used as a guideline for the coming experiments, we decided that an overall MAE below 2 for classes up to 10 would be an acceptable result.

6.2 CNN on Dataset 1

For dataset 1, where bias was minimised, we only tested the baseline model. We ran the model 6 times on dataset 1, however, as none of the results were feasible (see the confusion matrix in Figure 6.2), no other models were tested. As the confusion matrix of these runs shows, the model predicts all samples as either class 2 or class 3, depending on the run. In Figure 6.3, the MAE of each class is shown, as well as the overall computed mean MAE.

These results indicate that the model could not in fact find any voice-specific attributes to base its prediction on. If the model can not find any pattern in the data it still tries to minimise the error. It does this by always predicting one of the classes in the middle since they have the smallest combined error to all classes. As is obvious from the confusion matrix in Figure 6.2, the model did at least recognise a difference between class 0 and the other classes.

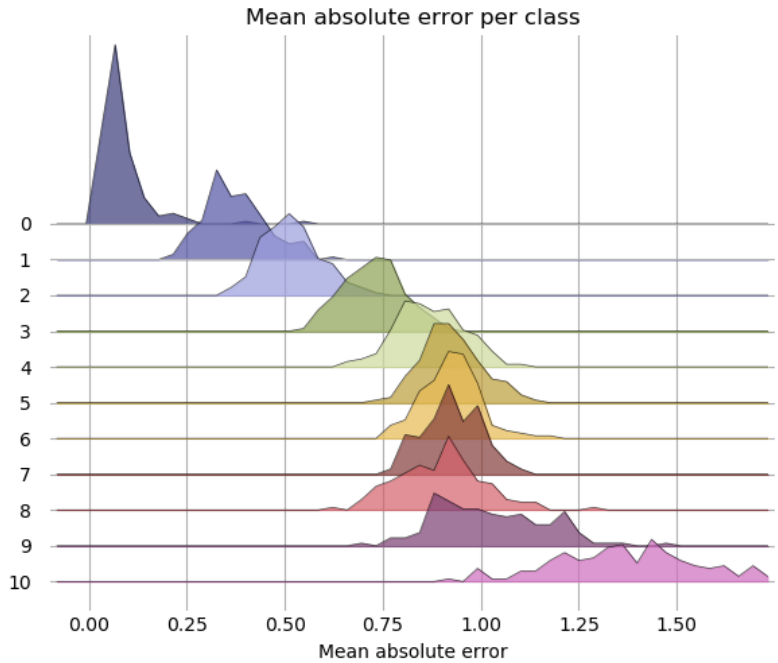


Figure 6.1: Baseline mean absolute errors per class, after running the model 150 times. The *y*-axes correspond to the classes 0-10.

Despite the fact that this model could not discern different classes based on only voice attributes, it is still possible that another, deeper model, or one with an entirely different architecture, would be able to find such differences.

6.2.1 Grad-CAM

Despite the model inaccuracy, we created Grad-CAMs for one example of each class. Figure 6.4 shows an example heat map of class 3. Even though the Grad-CAM shows activation in some parts more than others, it is not only adhering to the patterns where voices are audible, but also during quiet time. Again, this could explain why the model does not converge. As the samples were so similar over the classes, the model could only guess at which attributes to look, resulting in activating both where voice patterns were prominent and where they were not.

6.3 CNN on Dataset 2

For dataset 2, we tested the baseline model, along with three additional models as well. Two of the three additional models were slight modifications of the baseline model, whereas the fourth one was optimised for memory. The results from these models are presented in this section.

Figures 6.5 and 6.6, show the four models and their results on dataset 2 and gold standard. Each model was trained six separate times, and the interpolation is shown between the mean MAE of each class. The overall mean is computed as the mean of the class mean MAE:s.

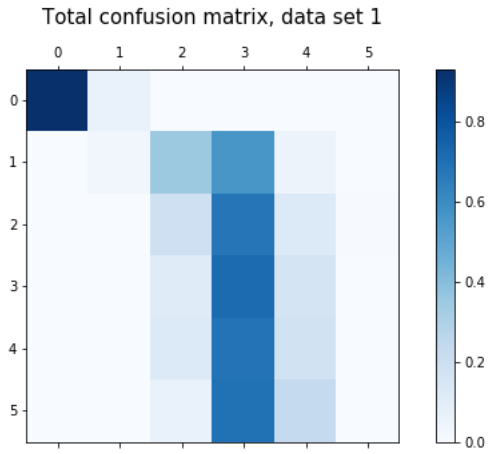


Figure 6.2: Total confusion matrix. This was computed by taking the mean of all confusion matrices.

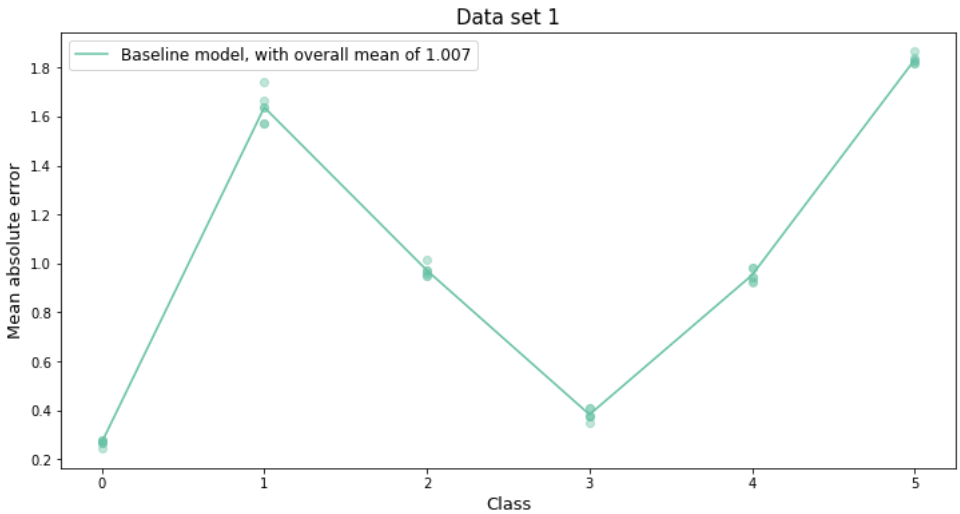


Figure 6.3: Mean absolute errors of baseline model on dataset 1.

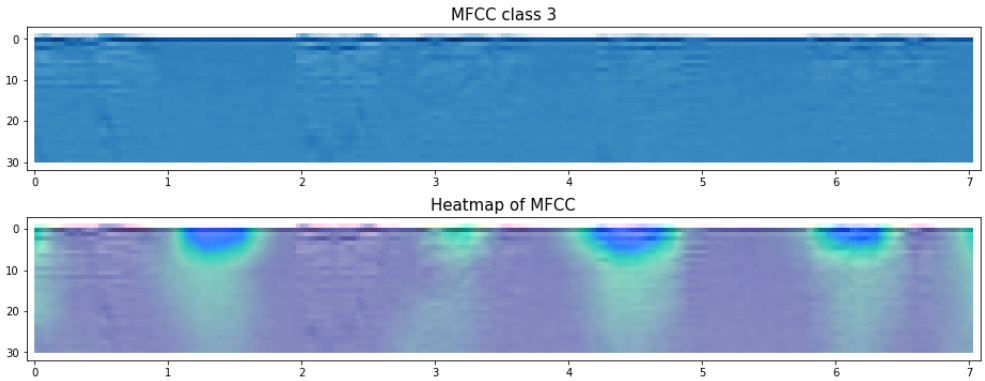


Figure 6.4: Grad-CAM of class 3 of the baseline mode of dataset 1.

Table 6.1: Average size of each fully trained model.

	Model size
Model 1	62.16 MB
Model 2	248.30 MB
Model 3	193.86 kB

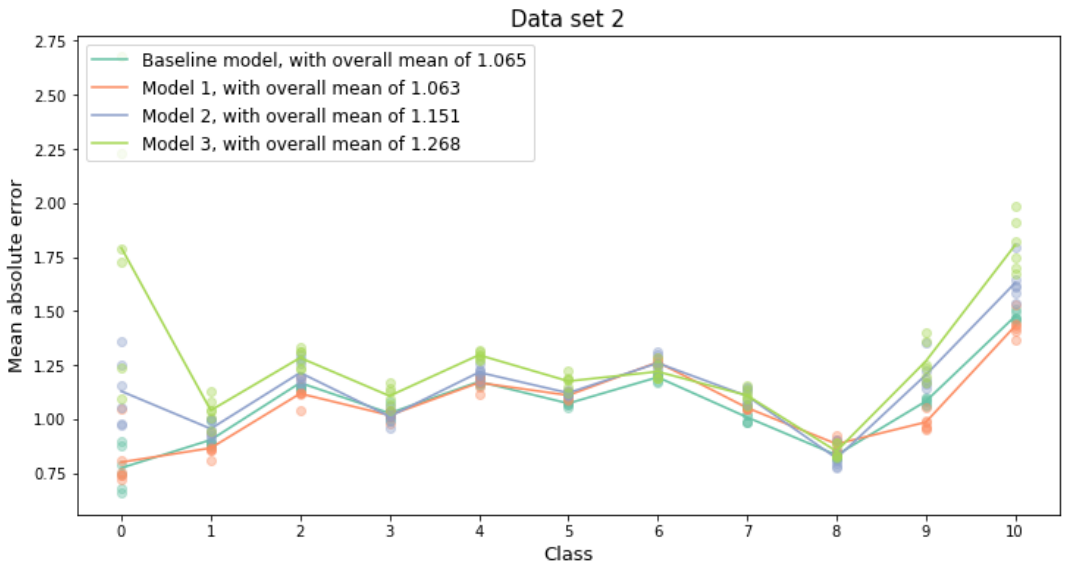


Figure 6.5: Model mean absolute errors on dataset 2. Each dot represents the MAE of one run, and the interpolation is between the mean MAE of each class and model.

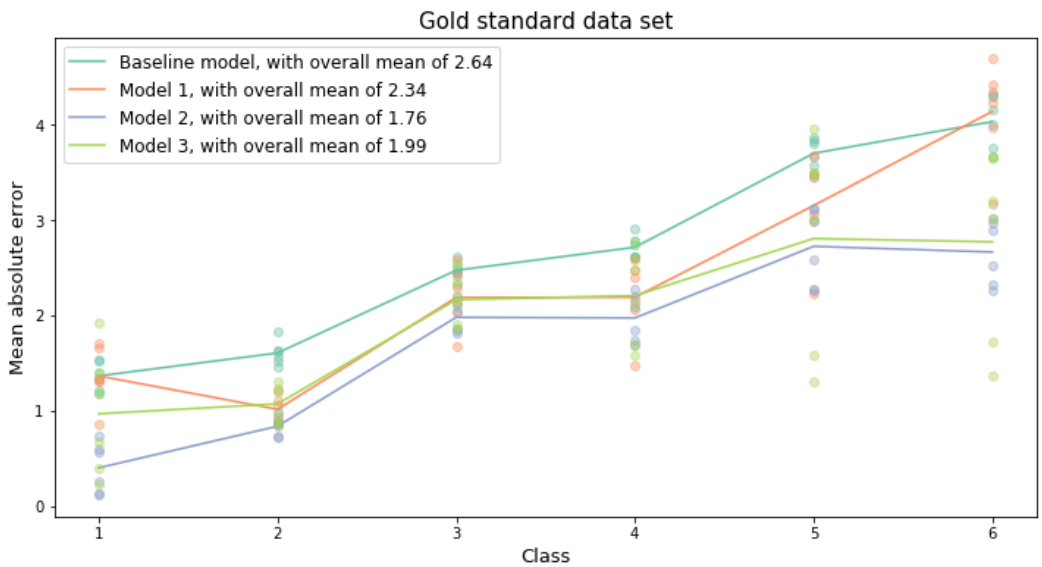


Figure 6.6: Model mean absolute errors on Gold standard dataset. Each dot represents the MAE of one run, and the interpolation is between the mean MAE of each class and model.

As is clear from the figures, the models all show promise for dataset 2 as all MAE and overall mean MAE fall below 2. Looking more specifically at the class differences, it is noticeable that the MAE for class 0 was around 1, up to 1.4. This means that in some cases, especially for Model 3, the model did not discern any prominent differences between a voice being present and not.

Class 0 should be the class with the absolute least MAE, and the fact that there is no obvious difference between the MAE of class 0 and the other classes up to 8, is interesting. Class 0 did have the least amount of samples (around 400), which most likely had some impact on the result. Looking at how well the baseline model performs on class 0 in dataset 1, which is the same data but in shorter sequences, the MAE is clearly below 0.4. It is therefore somewhat safe to assume that this result would improve with more samples for class 0, both as the training data would increase but also the predictions, resulting in an incorrect prediction not affecting the final MAE as much.

When looking at the classes 1-8, the MAE is quite similar, where a slight zig-zag pattern can be distinguished. A possible explanation for this pattern is that the model might be able to discern some classes better. There is more difference between classes 1 and 3, than between 1 and 2, therefore the model might guess class 1 or 3 for class 2 more often than it would guess class 2, resulting in a slightly higher MAE for class 2. The same reasoning follows for more classes.

Analysing the behaviour of classes 8-10, there is reason to believe that some statistics in the making of the dataset might not have been free of error. This is partly because of the rapid increase in MAE: from around 1 in class 8, to almost 2 in class 10, but also because the MAE is lowest for class 8, which is not probable. However, the results are still below 2 and thus comparable to the baseline model.

Despite these results on dataset 2, the models do not perform nearly as well for the gold standard dataset. There are probably a number of reasons behind these results, where the signal-to-noise ratio probably is the most prominent one. For Model 1, the MAE is as high as 4.5 for class 6, thus indicating that it did not always recognise any essential difference between class 6 and 1 or 2. Model 2 got the lowest overall mean of 1.76 on the GS dataset, where the MAE for the higher classes were between 2 and 3. This is a better result, however it is still slightly higher than desirable for class 6, which is already a relatively low class. A result of MAE strictly below 2 for class 6 would have been acceptable.

Looking at how much memory each model consumes, where as little memory as possible is positive, it is obvious that Model 3 outperforms the other models, see Table 6.1. Model 3 performs worse on dataset 2 than it does on the gold standard, where it comes in as second best. However, it does perform better than anticipated given its size as the results are indeed comparable to the other models. Yet again, that Model 3 performs worst on dataset 2 but second best on the gold standard could imply that it is flexible enough to generalise well over datasets.

6.3.1 Grad-CAM

We extracted Grad-CAMs from the model that performed best on the gold standard: Model 2. Comparing these Grad-CAMs with the one from the baseline model on dataset 1 shows a clear difference: these models indicate that there are certain areas of each sample that the model focuses on when predicting. The Grad-CAM is extracted from the last convolutional

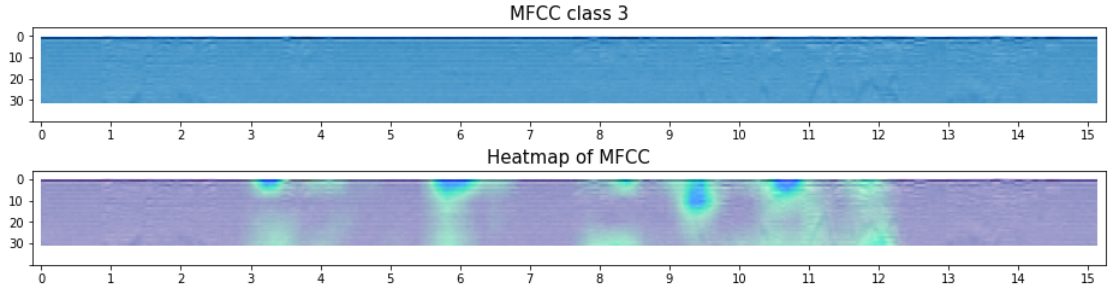


Figure 6.7: Grad-CAM of Model 2 on a sample of class 3 from dataset 2.

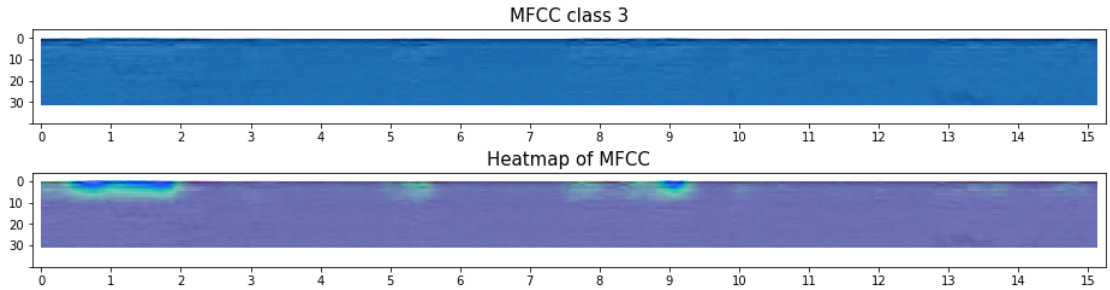


Figure 6.8: Grad-CAM of Model 2 on a sample of class 3 from the gold standard dataset.

layer in the model. Figure 6.7 shows the Grad-CAM of a sample of class 3 of dataset 2, and Figure 6.8 the Grad-CAM of a sample of the same class but from the gold standard dataset.

Analysing the differences in the Grad-CAM of the samples, it is obvious that the model is activated much less in the gold standard sample than in the dataset 2 sample. Again, the most probable reason for this is the signal-to-noise ratio that probably differs quite a bit between some samples of the gold standard and dataset 2. One thing to take note of is that the model activates where there are voices heard in both samples, meaning that it actually bases its decision on the metric we had assumed: the voices.

6.4 Diarization on Dataset 2

Figure 6.9 shows The results from the runs of the diarization model on dataset 2. In the same way as for the CNN, the model was trained six separate times, and the line is the mean MAE of each class.

Since the embedding algorithm has to have something to embed, this model does not classify class 0. The VAD is responsible for distinguishing between the signals with and without speech, but it has a quite low sensitivity. So, in dataset 2, the VAD did not find any speech in any sample from class 0, but also failed to find speech in a few samples where there were speech.

Because of the low sensitivity of the VAD, signals that should be of higher classes are instead labelled as one or two classes lower, since the VAD did not find all instances of speech. This in turn means that instead of having 1000 signals of each class when training the clus-

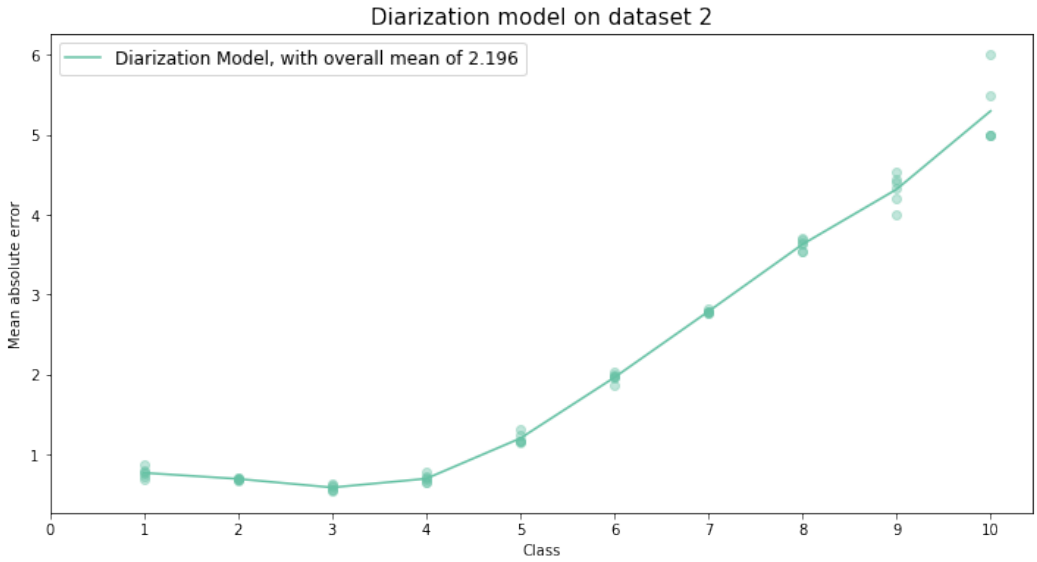


Figure 6.9: Mean absolute errors for the diarization model on dataset 2 for 6 different runs.

tering, there were approximately 1500 of class 1 which decreased to around 800 of class 7 and of class 10 there were only 5 signals.

However, this error did not seem to impact the result, because when testing with the same number of signals for all classes up until class 7 the results looked the same as they do in Figure 6.9. If it would have an impact, we expected the classes up to 7 to have approximately the same MAE but they did not. Instead the MAE started to rise at class 5 just as the model that is trained on an unbalanced number of signals in each class.

Figure 6.9 shows that the diarization method works well up to class 5. But, it seems to be unable to discern between more than about 5 or 6 speakers since the MAE increases almost linearly above class 5. This is supported by the fact that almost all signals of the higher classes were classified as 5 or 6.

This result is what we expected since most of the literature around speaker diarization and the packages we used described how signals with overlapping speech was discarded. Dataset 2 is built assuming that the overlapping speech increases with more speakers, which leads to the increasing MAE for the higher classes. So the model is not good enough to be able to discern so many different speakers with overlapping speech.

For the larger classes overlapping speech increases which also means that non-overlapping speech decreases. Moreover, if non-overlapping speech is what the diarization model needs to work well. A way to mitigate the decreasing amount of non-overlapping speech might be to increase the length of the signal. This would increase the likelihood of a speaker to speak uninterrupted since there is more time for everyone to speak.

6.5 Model Comparisons

Comparing the results for the CNN models and the diarization model on dataset 2, all the CNN models perform better overall, see Table 6.2 for a summary. When looking closer,

Table 6.2: Table summarising the MAE of each class and model. The lowest MAE per class is marked with bold, as well as the lowest overall MAE for both data sets.

Dataset 2

	0	1	2	3	4	5	6	7	8	9	10	Overall
Baseline	**0.77**	0.9	1.17	1.03	1.17	**1.07**	**1.19**	**1.01**	0.83	1.08	1.48	1.065
Model 1	0.8	0.87	1.12	1.02	1.17	1.11	1.26	1.05	0.89	**0.99**	**1.43**	**1.063**
Model 2	1.13	0.96	1.21	1.01	1.22	1.12	1.26	1.11	**0.82**	1.20	1.63	1.15
Model 3	1.79	1.04	1.28	1.11	1.30	1.18	1.22	1.11	0.85	1.27	1.81	1.27
SD model		**0.77**	**0.70**	**0.59**	**0.70**	1.20	1.96	2.79	3.63	4.32	5.30	2.20

Gold standard

	0	1	2	3	4	5	6	7	8	9	10	Overall
Baseline		1.36	1.60	2.47	2.71	3.70	4.03					2.64
Model 1		1.36	1.01	2.19	2.19	3.15	4.14					2.34
Model 2		**0.40**	**0.83**	**1.98**	**1.97**	**2.72**	**2.66**					**1.76**
Model 3		0.97	1.07	2.16	2.21	2.80	2.77					1.99

the diarization model performs slightly better for classes 1-4, however as the MAE increases rapidly for classes 6-10, the diarization model loses its lead.

The CNN models are much simpler compared to the diarization model. If they instead would be larger and more sophisticated, we suspect that they would perform better for the lower classes, maybe even as low as the diarization method, with an MAE strictly below 1.

Considering also the memory usage, one CNN model reaches slightly below 200 kB in size, where the models used on the Minut device as of today are around 70 kB. The embedding part of the diarization method is 16.3MB large, and that is not even counting the clustering of the embeddings. The fact that the CNN model is both simpler and smaller, and produces almost as good results for all classes, means that it is probably the more feasible method for this problem.

Chapter 7

Conclusion

This chapter presents the conclusions drawn from the results of the different experiments. First, we discuss the CNN models and data sets, followed by speaker diarization methods, and finally we outline some proposals for future work.

7.1 CNN Models

When looking at the results for the different CNN models on the different data sets, and Model 2 in particular, it is clear that there were still too much difference between the gold standard data set and dataset 2. Perhaps the models had performed even better, had the signal-to-noise ratio been more consistent in the gold standard, or more randomised in dataset 2. Apart from that, the metrics quiet time and overlapping speech must also be taken with a pinch of salt – this was based on a 9 minute conversation of one group of friends. This is by no means enough to build a model from, as a conversational culture might look entirely different in other friend groups, also perhaps depending on language and general culture.

To fully capture this, either dataset 2 must incorporate all possible types of bias, where these types of metrics must be fully examined before building a data set of this type algorithmically. However, the best way to move forward could be to record and annotate several conversations from several countries and cultures, and simply train the model on that data.

When it comes to comparing dataset 2 with the gold standard, some assumptions may have been slightly wrong, such as the quiet time metric. It is possible that the quiet time does indeed decrease as the classes increase. However it is possible that the total decrease is lower than initially estimated. The same is applicable for the overlapping speech metric – maybe the overlapping speech does not increase as much as estimated.

7.2 Speaker Diarization Model

Looking at the results from the diarization model, it seems reasonable to conclude that it could work well for estimating speakers in small groups, approximately up to 5 separate speakers. However, 15 seconds is not a very long time and longer clips might give the diarization model a better chance to separate more speakers.

7.3 Conclusions

The problem formulations to be answered in this thesis were:

> Can we identify *unique* speakers, given a finite length audio clip of people talking, and from that give an accurate estimation of the number of people in the room?

and

> Is it possible to run the model on the Minut device, despite its limitations such as memory capacity?

As for the first question, the experiments from dataset 1 show that it was not possible to identify unique voices, however the results from the CNN on dataset 2 show that it is still possible to estimate the number of speakers given more bias than only unique voices.

The results show that the diarization model works better than the CNN models for class 1-4 and the CNN models are better for class 6 and above. However, since the CNNs are so simple compared to the diarization, it is probably possible to build a CNN that outperforms the diarization model for all classes for this specific problem.

As for the second question, there is one CNN model that shows promise for being used for the Minut device. However it is still around 120 kB too large memory-wise. It could be used as a start for the company, however we have realised that the real problem lies not with the CNN model, but with the data for training.

We have shown that it is possible to give a good estimate of the unique number of speakers in a finite length audio clip using a CNN. Moreover, for a small number of speakers a diarization model can also achieve this. The possibility for these models to operate on the Minut device however is still unanswered. It is something that they will have to look further into.

7.4 Future Work

A large part of this thesis work has gone into creating datasets algorithmically, and we deem the second dataset to be the biggest weakness of the work. This is because we cannot be sure that dataset 2 does reflect reality enough to solve the problem at hand. While the CNN model does show some promise, it is nearly impossible to know if it would actually be feasible in a real-life application, as there is still too much difference between the GS data set and dataset 2.

Thus, the first step in the possible future work would be to build and annotate a real-life data set. There are a number of ways to go about solving that, and we propose a solution of gathering a number of people in a room, placing one microphone at the centre of the room, and one microphone on each speaker. The microphone in the centre will be recording the data used for the model, while the microphones attached to each speaker will be used to aid annotation of which speaker spoke at what time. Another thing to take into consideration is whether or not to annotate non-verbal sounds, such as laughter.

Besides developing the data set, more emphasis could be laid on pre-processing the signals before transforming them. There are a number of transforms that could be added to for example improve the signal-to-noise ratio.

As some of the models in this thesis have been constrained by memory and thus been kept quite simplistic, it would be interesting to try deeper CNN networks, or recurrent NNs combined with a CNN, as there is a possibility that deeper networks can capture patterns that our models could not.

As for the speaker diarization part of the work, it would be interesting to use a more sophisticated VAD. We could maybe even include something like speaker change detection to be able to increase the quality of the cut up signals that the embedding model receives from the VAD.

Other interesting things would, of course, be to test more embedding and clustering methods since we ended up just using two already proven and easy to use Python packages: `Resemblyzer` for the embedding and `UIS-RNN` clustering.

7.5 Ethical Aspects

The recording of conversations between people, with the intention of estimating how many people there were that took part in the conversation, can come with some ethical problems. Especially if they are not directly aware of being recorded. Questions about how the data is used, whether or not a product like this can comply with GDPR as voices are biodata, or if anyone can replay ones conversations afterwards naturally arise. These questions are natural and understandable, and need to be addressed.

Claiming that a product is 100% safe regarding privacy and personal data is a stretch, as no one can guarantee such a thing, however the company Minut has gone to quite some lengths to protect their customer's data. As an example, the sensitive data never leaves the device, and are discarded as soon as the computations on the data is done. The data that leaves the device is anonymous, such as noise levels in dB (i.e. not the actual recording), or number of movements under the device. The product is completely compliant with the GDPR laws.

As such, we deem that it is up to the customer to decide if it is a risk worth taking, and if they think that the company is trustworthy enough to buy and use their product.

7.6 Work Distribution

As we were two persons writing this thesis, we divided the work between us. Isabella made the baseline, datasets and the first method that consisted of the convolutional neural network models. Anton spent time trying to duplicate Audiolabs results, and then moved on to

researching, examining and testing the speaker diarization method. As we both had previous experience with CNNs, the baseline and the first method was not as time consuming as creating datasets and understanding and implementing speaker diarization methods.

Regarding the writing of the report, Anton wrote Chapter 2, 4.1 and 4.3. Isabella wrote Chapter 3 and 4.2. The division of Chapters 5 and 6 was according to above - Isabella wrote the parts about the baseline and CNN method, Anton wrote the part about the speaker diarization method. We both collaborated on Chapter 1 and 7.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ashar, A., Bhatti, M. S., and Mushtaq, U. (2020). Speaker identification using a hybrid cnn-mfcc approach. In *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, pages 1–4.

Blei, D. M. and Frazier, P. I. (2011). Distance dependent chinese restaurant processes. *Journal of Machine Learning Research*, 12(8).

Chollet, F. (2017). *Deep Learning with Python*. Manning.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Cyrta, P., Trzciński, T., and Stokowiec, W. (2017). Speaker diarization using deep recurrent convolutional neural networks for speaker embeddings. *Advances in Intelligent Systems and Computing*, page 107–117.

Dai, W., Dai, C., Qu, S., Li, J., and Das, S. (2017). Very deep convolutional neural networks for raw waveforms. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 421–425. IEEE.

Fini, E. and Brutti, A. (2020). Supervised online diarization with sample mean loss for multi-domain data. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7134–7138. IEEE.

Fujita, Y., Kanda, N., Horiguchi, S., Xue, Y., Nagamatsu, K., and Watanabe, S. (2019). End-to-end neural speaker diarization with self-attention. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 296–303. IEEE.

Fujita, Y., Watanabe, S., Horiguchi, S., Xue, Y., and Nagamatsu, K. (2020). End-to-end neural diarization: Reformulating speaker diarization as simple multi-label classification. *arXiv preprint arXiv:2003.02966*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Ko, J. H., Fromm, J., Philipose, M., Tashev, I., and Zarar, S. (2018). Limiting numerical precision of neural networks to achieve real-time voice activity detection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2236–2240.

larmkollen.se (2020). Hemlarm med larmkollen. `https://www.larmkollen.se/`.

Lukic, Y., Vogt, C., Dürr, O., and Stadelmann, T. (2016). Speaker identification and clustering using convolutional neural networks. In *2016 IEEE 26th international workshop on machine learning for signal processing (MLSP)*, pages 1–6. IEEE.

MATLAB Computer Vision Toolbox (2019). Matlab computer vision toolbox r2019b. `https://se.mathworks.com/help/pdf_doc/vision/vision_ref.pdf`.

McFee, B., Lostanlen, V., McVicar, M., Metsai, A., Balke, S., Thomé, C., Raffel, C., Malek, A., Lee, D., Zalkow, F., Lee, K., Nieto, O., Mason, J., Ellis, D., Yamamoto, R., Seyfarth, S., Battenberg, E., Morozov, V., Bittner, R., Choi, K., Moore, J., Wei, Z., Hidaka, S., nullmightybofo, Friesch, P., Stöter, F.-R., Hereñú, D., Kim, T., Vollrath, M., and Weiss, A. (2020). librosa/librosa: 0.7.2.

Metz, C. (2020). There is a racial divide in speech-recognition systems, researchers say. *The New York Times*.

minut.com (2020). Minut smart home alarm. `https://www.minut.com/`.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Moattar, M. and Homayoonpoor, M. (2010). A simple but efficient real-time voice activity detection algorithm. *European Signal Processing Conference*.

Muda, L., Begam, M., and Elamvazuthi, I. (2010). Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *CoRR*, abs/1003.4083.

Nielsen, M. A. (2018). Neural networks and deep learning. `http://neuralnetworksanddeeplearning.com/`.

Oppenheim, A. V. and Schafer, R. W. (2014). *Discrete-Time Signal Processing*. Pearson Education Limited, Edinburgh Gate Harlow Essex CM20 2JE.

Palmiter Bajorek, J. (2019). Voice recognition still has significant race and gender biases. *Harvard Business Review*.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

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.

Proakis, J. G. and Manolakis, D. K. (1996). *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ.

Rabiner, L. and Schafer, R. (2007). Introduction to digital speech processing. *Foundations and Trends in Signal Processing*, 1.

Reddy, C. K., Beyrami, E., Pool, J., Cutler, R., Srinivasan, S., and Gehrke, J. (2019). A scalable noisy speech dataset and online subjective test framework. *Proc. Interspeech 2019*, pages 1816–1820.

Salamon, J., MacConnell, D., Cartwright, M., Li, P., and Bello, J. (2017). Scaper: A library for soundscape synthesis and augmentation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2017*, pages 344–348. Institute of Electrical and Electronics Engineers Inc.

Stoter, F.-R., Chakrabarty, S., Edler, B., and Habets, E. A. P. (2018). Classification vs. regression in supervised learning for single channel speaker count estimation. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Usoltsev, A. (2015). Vad-python. `https://github.com/marsbroshok/VAD-python`.

Variani, E., Lei, X., McDermott, E., Moreno, I. L., and Gonzalez-Dominguez, J. (2014). Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4052–4056.

Wan, L., Wang, Q., Papir, A., and Moreno, I. L. (2017). Generalized end-to-end loss for speaker verification. `https://github.com/resemble-ai/Resemblyzer`.

Wikström, F. (2014). *Funktionsteori*. Studentlitteratur, Lund.

Wiseman (2019). py-webrtcvad. `https://github.com/wiseman/py-webrtcvad`.

Zhang, A., Wang, Q., Zhu, Z., Paisley, J. W., and Wang, C. (2019). Fully supervised speaker diarization. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6301–6305.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-08-27

EXAMENSARBETE Identifying and quantifying voices in audio samples using neural networks
STUDENT Anton Johansson, Isabella Gagner
HANDLEDARE Pierre Nugues (LTH), Colin Nordin (Minut)
EXAMINATOR Flavius Gruian (LTH)

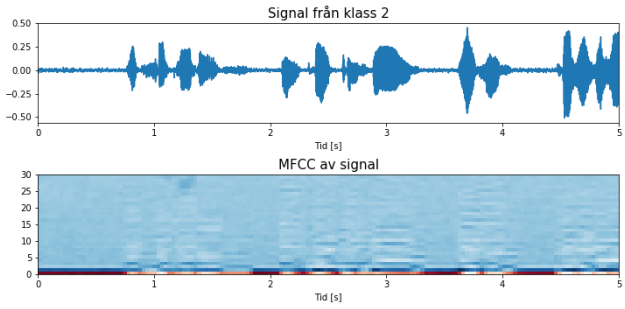
Uppskatta antal pratande personer i ett rum med maskininlärning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Anton Johansson, Isabella Gagner**

Genom att analysera bland annat ljud, rörelser och temperatur, har företaget Minut skapat en *internet of things* (IoT) enhet som låter en ha uppsikt över ett hem utan att installera kameror. Vi kan få mycket information om en miljö genom att enbart analysera ljuden - kan vi även uppskatta hur många personer som hörs i ett klipp?

Varje Minut-enhet analyserar fortgående rörelser, temperatur och ljud för att kunna uppmärksamma ägaren vid ovanliga händelser. Med hjälp av minneseffektiva modeller som körs direkt på enheten kan det ske utan att känslig information lämnar den. En enhet kan känna igen ljud så som glassplitter och brandlarm, och kan skicka notifikationer till ägaren för att uppmärksamma denne på olika saker.

Men var går gränsen för vad vi kan lyckas analysera och känna igen när det gäller ljud från en inspelning? Kan vi lyckas identifiera unika röster och på så vis uppskatta hur många personer som finns inom området som enheten kan höra?



I detta examensarbete har vi undersökt och jämfört två olika maskininlärningsmetoder för att uppskatta antalet pratande personer i ett ljudklipp. För att lyckas med detta skapade vi två egna dataset algoritmiskt. I det första datasetet minimerade vi skillnader mellan ljudklippen (vilket gjorde dem mindre verklighetstrogna) och i det andra försökte vi spegla verkliga konversationer.

De två metoder vi testade var *faltningsneuronnätverk* (CNN) och *speaker diarization* (SD). Ett CNN är ett nätverk som kan lära sig att känna igen mönster i bilder, där vi transformerade ljudsignalerna till bilder kallade *mel-frequency cepstrum coefficients* (MFCC) och tränade vårt CNN med dessa (se figur). Målet med SD är att dela upp ljudklipp i delar, och sedan markera varje del med vilken person som pratat i den. Det första steget är att identifiera när personer talar i ett ljudklipp. Det andra är att omvandla dessa delar till vektorrepresentationer av ljudet som hörts. Slutligen paras dessa representationer ihop med varandra för att bestämma vem som talade när.

Våra resultat visade på att metoden med CNN var mest användbar för detta ändamål, då de modellerna tog minst minne samt presterade bra även för klipp där många röster hördes. Samtidigt måste metoden utvärderas mer med ett äkta, inspelat dataset.