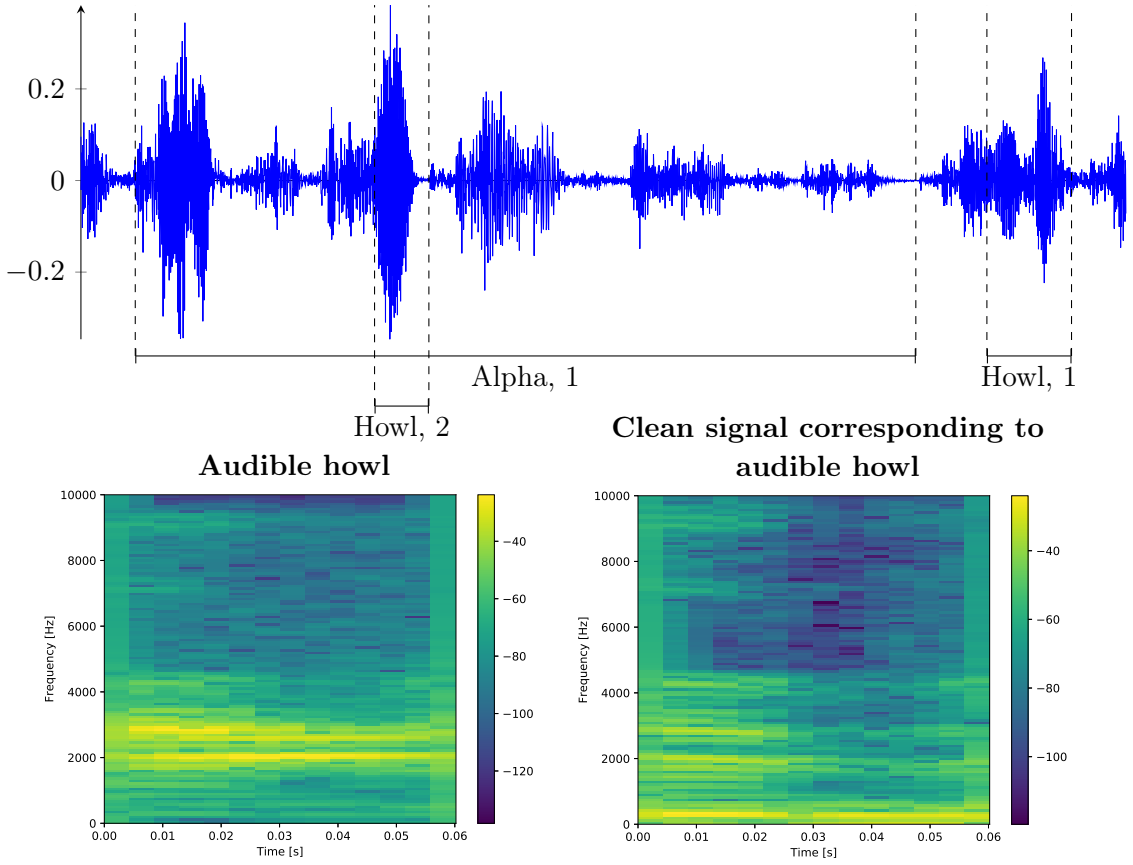
Machine Learning based Evaluation of Feedback Cancellation Systems in Hearing Aids



Jakob Sloth Lauridsen
Signal Processing & Computing
Aalborg University
Date: 4th June 2020

AALBORG UNIVERSITY
DENMARK

Aalborg University
Department of electronic systems
Fredrik Bajers Vej 7
9220 Aalborg Øst
www.es.aau.dk

Title:

Machine Learning based Evaluation of Feedback Cancellation Systems in Hearing Aids

Theme:

Signal Processing & Computing

Project Period:

Spring semester 2020, 10th semester

Participants:

Jakob Sloth Lauridsen

Supervisors:

Jesper Jensen and Zheng-Hua Tan

Pages: 53
Date of Completion: 04-06-2020

Abstract:

State-of-the-art hearing aids are equipped with a feedback cancellation system that prevents howls and other artefacts to be of annoyance to hearing impaired users. Detection of artefacts caused by acoustic feedback is essential for evaluation of the feedback cancellation system.
The detection of artefacts is formulated as a classification problem using spectro-temporal information from the hearing aids and a reference as input.
A single layer feedforward neural network model with 1024 hidden neurons is found to have comparable performance to an existing detection system, developed by Oticon. Different approaches to processing of the spectral data is explored revealing, that the neural network model does not need the reference for it to perform as well as the existing detection system. This gives the neural network more flexibility in testing environments and possibly extra applications such as "online" evaluation of the feedback cancellation system.

Preface

This report is conducted in collaboration with Oticion during the spring of 2020. The project could not have been completed without the equipment and data, made available by Oticon. A huge thanks to the people at Oticon, especially Anders Meng and Mojtaba Farmani who has helped with both practical questions, and inputs to the project.

A special thanks to professor Jesper Jensen for his supervision of the project. His inputs to the project and feedback on the report has been a great help and inspiration.

Reading instructions:
The report is divided into chapters numbered by the order they appear. Sections and subsections are numbered as well the same way.

Bold upper case letters are reserved for matrices, ex. $\mathbf{A}$ where $A_{i,j}$ are entries in matrix $\mathbf{A}$. Bold lower case letters are reserved for vectors, ex. $\mathbf{a}$ where a_i are entries in vector $\mathbf{a}$.

Figures, tables and equations are numbered by the chapter they appear in as well as the order. Figure 5 in chapter 4 will thereby be numbered as 4.5.

Contents

Introduction 1

Adaptive feedback cancellation (AFC) is an essential part of today's hearing aids that ensures that acoustic feedback does not cause annoyance to hearing impaired users. The AFC system used by Oticon is effective, but under certain conditions, unwanted artefacts are present in the delivered sound signal. Part of the evaluation of the AFC system with respect to the sound quality can be done with simulations, but real world testing is also a part of the evaluation method. The current real world testing requires manual annotation of artefacts in the sound signal, which is a tedious and time consuming task.

Oticon proposes that the annotation can be automated using machine learning techniques. This would allow for faster real world testing and also create a framework for which different versions and different models can be compared. The following sections introduce the problem of acoustic feedback in hearing aid systems, how the feedback is dealt with in state-of-the-art hearing aid systems and the side effects of AFC.

1.1 Acoustic feedback in hearing aids

Hearing aids are devices that help hearing impaired people in their everyday life. Today's devices are small enough to be placed behind the ear making them practically invisible. One major drawback of devices being able to fit close to the ear is that it allows for acoustic feedback that can lead to instability of the hearing aid and consequently howling if not dealt with.

Acoustic feedback is created when part of the signal that is played in the ear is recorded by the microphones in the hearing aid. The received signal may lead to positive feedback and consequently howling. A simple model of a hearing aid with a single microphone and an acoustic feedback path is shown in Figure 1.1. The target signal $s(n)$ and feedback signal $v(n)$ is recorded by the microphone and amplified with the linear, time dependent impulse response $g(n)$ in the hearing aid. The amplified signal $u(n)$ is played into the ear of the user. The feedback signal is modelled as the linear, time dependent feedback impulse response $f(n)$ applied to the output signal [1].

The process of amplifying the target signal and playing it to the user is described by a closed

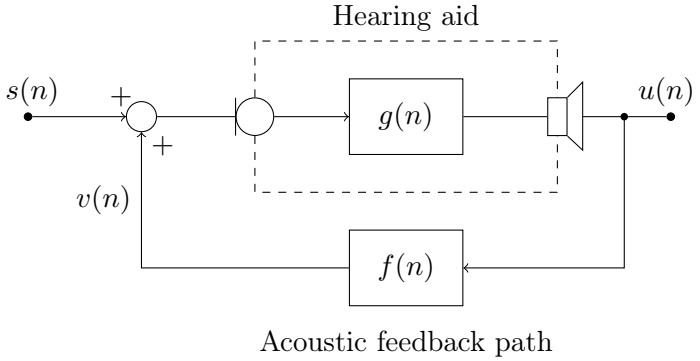


Figure 1.1: Simple feedback model in hearing aids. The gain of the hearing aid and the feedback path constitutes a closed loop that is prone to instability.

feedback loop. A useful relation in closed loop systems is the open loop transfer function which can be expressed in terms of the frequency transformed responses of the direct path through the hearing aid, $G(\omega, n)$, and the feedback path, $F(\omega, n)$,

$$\Theta(\omega, n) = G(\omega, n)F(\omega, n) \tag{1.1}$$

where $\Theta(\omega, n)$ is the time and frequency dependant open loop transfer function. The closed loop transfer function is given by [1]

$$H(\omega, n) = \frac{U(\omega, n)}{S(\omega, n)} = \frac{G(\omega, n)}{1 - \Theta(\omega, n)} \tag{1.2}$$

where $H(\omega, n)$ is the time and frequency dependant closed loop transfer function. Nyquist's stability criterion states that the closed loop system becomes unstable if the two following criteria are met at the same time [2].

$$|\Theta(\omega, n)| \geq 1 \tag{1.3}$$

$$\angle\Theta(\omega, n) = l \cdot 2\pi \quad \forall \quad l \in \mathbb{Z} \tag{1.4}$$

where the criteria can be interpreted as a scenario where the feedback signal is a more than unitary, amplified version of the target signal in phase with the target signal.

The goal of any system that deals with instability issues is to make sure that both criteria are not met at the same time. The criterion in (1.3) requires that either the gain of the hearing aid or the acoustic feedback path is reduced. The gain of the feedback path can be reduced by designing the ear piece such that minimal sound escapes the ear. However closing the ear canal creates an occlusion effect that can be unpleasant for some users both in terms of earpiece comfort and sound perception quality [3]. Reducing the gain of the hearing aid is an alternative to changing the acoustic path. However it is counter productive as the

main purpose of the hearing aid is to amplify sounds. Methods in this category of feedback cancellation detect howls building up in some frequency bands and then apply notch filters to reduce the amplification in those bands [4]. The task of detecting howls is extensivly studied with a great review in [5]. Even though part of the objective of this report is to be able to detect howls in audio signals, there are some critical differences in the requirements to howl detection, that make the existing literature of little use. Howl detection in this report has no real time requirements, and no requirements to numerical complexity of the algorithm. However, the system in the report must simultaneously be able to detect other artefacts than howling and be able to distinguish between them.

The criterion in (1.4) can be addressed by either frequency shifting or phase modulation. Both methods changes the frequency of the feedback signal, breaking the positive feedback loop. Shifting the frequency can be effective in some circumstances, but does not work well with target signals that have harmonic structure such as music [1].

1.2 Adaptive Feedback Cancellation

Another group of approaches to feedback cancellation, add an additional feedback loop within the hearing aid. In literature this group of approaches are called Adaptive Feedback Cancellation (AFC) [1]. Figure 1.2 illustrates the model of the hearing aid with an additional feedback path within the hearing aid. The internal feedback signal $\hat{v}(n)$ is subtracted from the microphone signal.

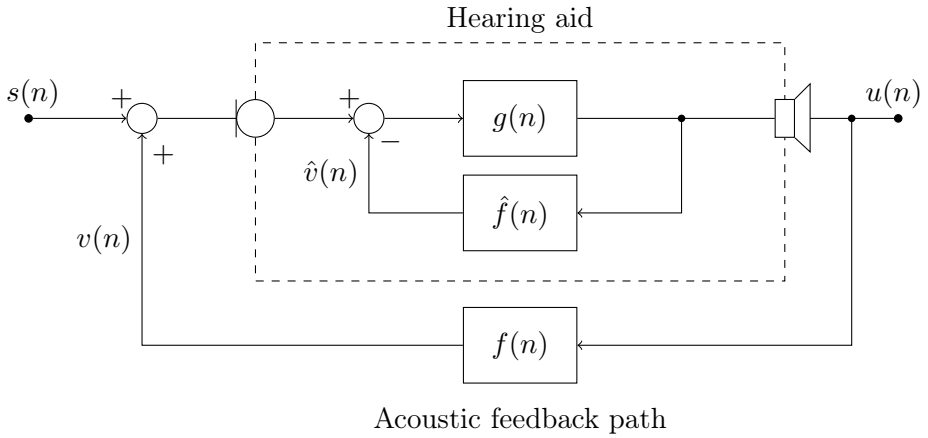


Figure 1.2: Model of the hearing aid with an additional feedback path inside the hearing aid. By approximating the acoustic feedback path, it is possible to negate its effect in closed feedback loop.

The additional feedback loop contributes to the open loop path, which for the new system becomes

$$\Theta_n(\omega, n) = G(\omega, n) \cdot (F(\omega, n) - \hat{F}(\omega, n)) \tag{1.5}$$

where $\Theta_n(\omega, n)$ is the open loop transfer function of the system with AFC and $\hat{F}(\omega, n)$ is the time and frequency dependant feedback component inside the hearing aid.

From (1.5) it is obvious that the magnitude of the open loop gain can be reduced if $\hat{F}(\omega, n)$ is a good approximation of $F(\omega, n)$. In the ideal case, $\hat{F}(\omega, n) = F(\omega, n)$, the open loop gain disappears and the closed loop gain depends only on the direct path of the hearing aid. In practice it is difficult to estimate the acoustic feedback since the signals are correlated [1]. Recent AFC systems attempt to decorrelate signals that are used for estimation of the feedback path by adding inaudible noise to the speaker signal [1].

1.2.1 Assisting AFC with spectro-temporal modulation

The feedback path estimation algorithm can under certain circumstances converge too slow to effectively cancel the acoustic feedback . This can occur when the feedback path changes quickly, such as when a hand is being moved in the vicinity of the ear [1]. When the estimated and the true feedback path are sufficiently different from one another, the hearing aid is prone to instability.

State-of-the-art hearing aid systems use a secondary feedback cancellation system which modulates the speaker signal in frequency bands where instability is detected [6][7]. Instability is detected through estimating the magnitude of the open loop transfer function in both time and frequency. At times and frequencies where the estimated open loop gain exceeds a predefined threshold a spectro-temporal modulation (STM) pattern is applied in the forward path of the hearing aid. The modulation prevents howling immediately, but introduces an artefact in the sound hereafter called alpha. The alpha artefact is less intrusive than howling, but it still degrades the sound quality [6][7].

When the approximation of the feedback path is infeasible, sound quality is degraded whether or not the STM system is active. The degradations can be described as different kinds of artefacts which are an annoyance to the user. The main types of artefacts are summarised in Table 1.1.

1.3 Evaluation of feedback cancellation systems

Evaluation of the adaptive feedback cancellation system can done with simulation, using metrics such as convergence rate and steady state error of the estimation algorithm, or comparsion of signals such as the distance between $f(n)$ and $\hat{f}(n)$ [1]. All simulations rely on

Artefact	Description
Howling	Much like the name suggests, howling is a loud sound typically limited to some band of frequencies sounding similar to a cry or a squeal. Howls can differ in both loudness and frequency, sometimes being inaudible to the hearing aid user but audible to nearby persons.
Alpha	Alpha artefacts sound like bird chirping or a robot effect added to the played signal. It arises from the STM system, when instability is detected. In the spectrogram alpha looks like diagonal stripes, originating from the applied modulation pattern. The artefact can sometimes be visible on the spectrogram but inaudible even to people with normal hearing.

Table 1.1: Summary of the two main types of artefacts.

a known feedback path, that is given before hand. The simulated metrics relate somewhat to sound quality [1].

A critical factor of sound quality is the presence of mentioned artefacts, which significantly degrades the listening experience. The presence of artefacts is tested with a real hearing aid placed in an ear model using different target sound signals and different room acoustics. The sound signals contain speech, music and various sounds from everyday living. Each sound signal is tested using using different variations of hand movement around the ear. For each combination, the speaker signal is recorded. People with normal hearing listen to the recordings while looking at a logarithmic magnitude spectrogram of the signal and marks the presence of the two main types of artefacts in the signal, with a level of severity. Figure 1.3 shows a part of a speaker signal and the corresponding annotations. Each annotated artefact has some type, starting point, duration and severity. Annotated artefacts can overlap.

Severity ranges from visible in the spectrogram but inaudible, to audible, to annoying. The severity is highly subjective, but roughly differentiates the levels of annoyance caused by the artefact. Figure 1.4 shows examples of magnitude spectrograms containing the alpha artefact. In Figure 1.4a the artefact is clearly audible as well as visible in the logarithmic magnitude spectrogram as sinusoidal behaviour in the frequency band 1 kHz to 3 kHz. In Figure 1.4b the artefact is not audible but it is clearly visible in the logarithmic magnitude spectrogram around 7 kHz.

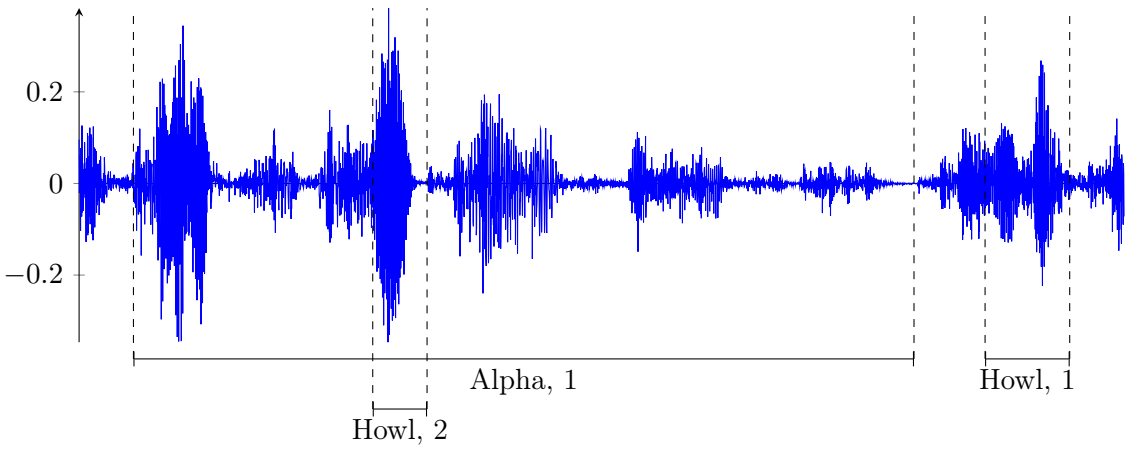


Figure 1.3: Example of an annotation of a part of a signal.

Figure 1.5 shows two examples of a logarithmic magnitude spectrogram where a howl is present. Again, the difference is that Figure 1.5a shows an audible howl and that Figure 1.5b shows a howl that is only visible in the logarithmic magnitude spectrogram. The characteristic of a howl is that the magnitude of some frequency builds up over a short period of time. Both types of artefacts are visible in the logarithmic magnitude spectrogram and can be differentiated from normal sounding signals by humans.

The task of annotating the signals is both time consuming and tedious. Oticon has annotated 114 audio files an average of 2.5 times. In numbers: 44 hours of audio with around 2200 howls and 4500 instances of alpha.

Recently, machine learning techniques have become a popular way of automating manual repetitive tasks. These tasks can be copied by a computer if the problem is well defined in terms of input and output and enough training material is provided. A system that is capable of automating the task specified in this section is of interest to Oticon.

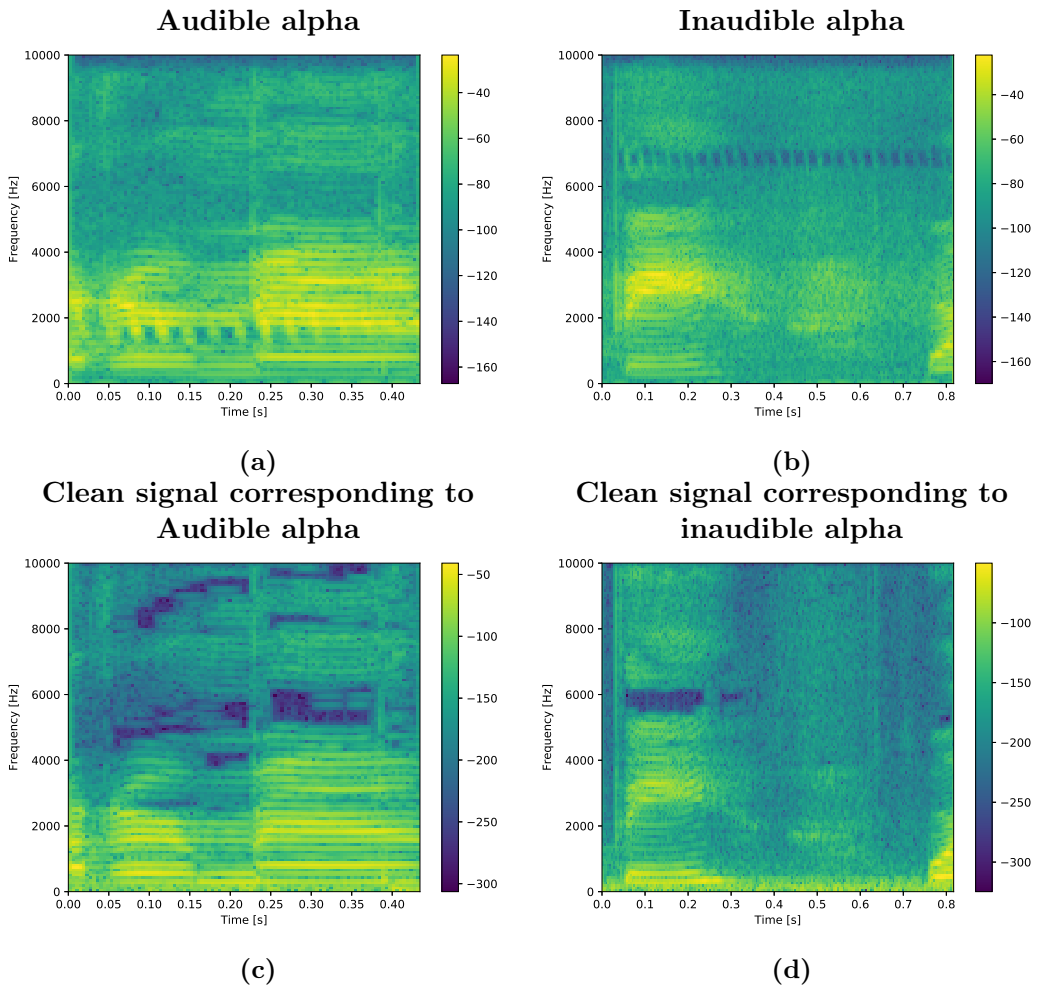


Figure 1.4: Logarithmic magnitude spectrograms of a speaker signal that contain the alpha artefact and the corresponding clean audio signals. (Left column) Audible artefact when pronouncing the word: "Watermen" and the corresponding clean signal. (Right column) Inaudible artefact that is only visible in the spectrogram and the corresponding clean signal.

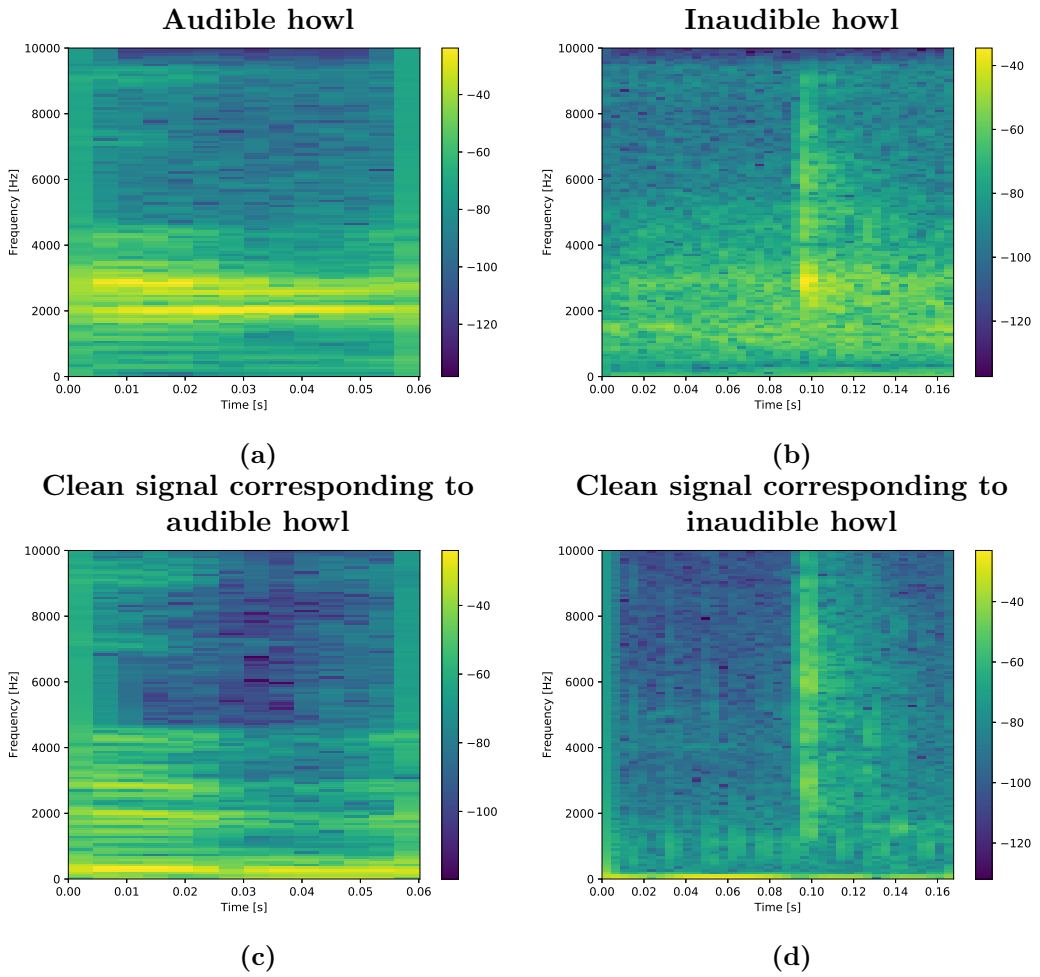


Figure 1.5: Logarithmic magnitude spectrograms of a speaker signal that contain the howl artefact and the corresponding clean signals. (Left column) Audible artefact and corresponding clean signal. (Right column) Inaudible artefact that is only visible in the spectrogram and corresponding clean signal.

Analysis of the annotation task 2

It is critical to model the task of automatically annotating artefacts in a way that fits a relevant machine learning technique. It would be ideal to describe a model where many machine learning tools can be used, but the modelling and choice of machine learning tool is often done simultaneously. The chapter starts with a summary of the annotation task and a discussion of some of the requirements to the desired system. Then the output and the input of the system is defined, making it possible to asses the annotated data at hand. Based on the annotated data, potential problems are presented and discussed.

2.1 The essence of the annotation task

The annotation task and process was explained briefly in the previous chapter. We wish to determine at a given time instant if there is a howling artefact and/or an alpha artefact. Such classification system must make its decision based on audio information related to this given time instant. The system can use any available data and any handcrafted transforms that helps the system make a qualified prediction.

There are no real-time requirements to the system, thus using data that lies in the future and or the past of a given time instant is allowed. In fact, the system can use any amount of previous and future data. This differatiates the system from the systems explored in [5] where detection of howl artefacts has strict, real-time requirements. The developed classification system will not have to run on the hearing aid itself, thus the system is neither limited by severe memory or energy consumption constraints. Execution on a standard laptop PC is satisfactory.

The howl detection algorithms presented in [5] are based on handcrafted theoretical features that indicate the presence of a howl. The classification system presented in this report will be developed based on supervised machine learning techniques, using annotated data made available by employees at Oticon. Each audio file, that correspond to the speaker signal of a test, is annotated by a number of different annotators. Each annotator has the logarithmically transformed magnitude spectrogram of the audio file visually available. The

annotator registers artefacts by listening to the audio, as well as inspecting the spectrogram. Each artefact is described by a type, starting point and duration. Time instants that does not fall within a period of an artefact, are instants of normal operation of the hearing aid. Periods of different artefacts can overlap, ex. a howl and an alpha artefact can be heard and seen in the spectrogram in the same time period.

Two different annotators may disagree about the presence of an artefact or disagree about the starting point and duration of an artefact. Figure 2.1 shows an example of annotations made by two different annotators, where the starting point and duration of the artefacts differs. The spectrogram is what the annotators have available visually and is included to give a context to the annotations, not to show the actual artefact.

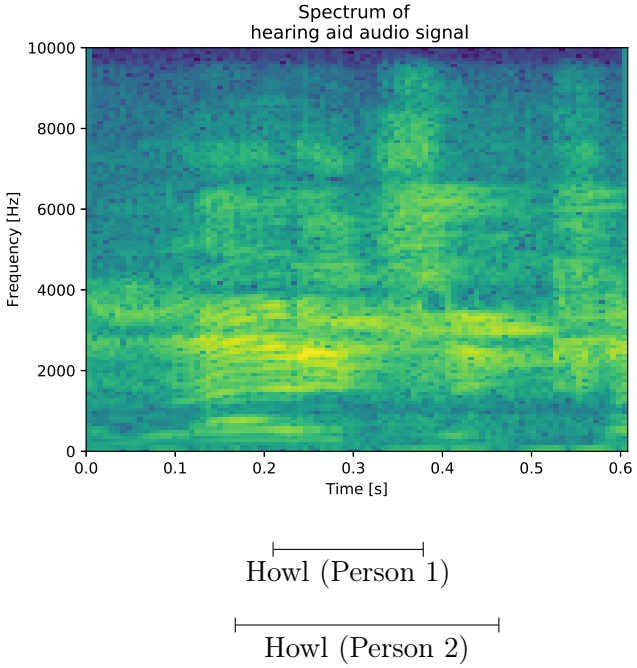


Figure 2.1: Example of disagreement between different annotations of the same signal. Two annotators have annotated a howl with slightly different start time and duration.

Difference between annotations raises the question of who is right and a definition of ground truth. A simple solution would be to select one of the annotations and use that as the ground truth, discarding the other annotation made by different annotators. This method's flaw is that all but one of the annotators experience is lost. Using all available annotations require some set of rules for which set of annotations defines the state of any time instant. Recall that annotated artefacts are done so actively. This means that an annotator that have registered an artefact have either heard or seen that artefact. Contrary, the time instants of normal operation are classified as so only because an artefact has not been registered at that

time instant. It may very well be possible that an annotator 'misses' an artefact, naturally without that person knowing. If it is assumed that all annotated artefact are indeed present, it makes sense to let the presence of artefacts overrule normal operation. Recall that periods of different artefacts can overlap, thus if annotations disagree whether at a time instant there is a howl or an alpha artefact, both of the artefact are there at the same time. In summary the rules for the ground truth are simple. If any annotator has registered an artefact at a given time instant, the ground truth for that time instant will be, that the registered type of artefact is present.

2.2 Model of system output

The output of the classification system must be some qualified prediction of whether some artefact is present at a given time instant and in that case determine which artefact it is. Naturally there are four cases or categories that cover all possibilities.

Empty:	No artefacts are present in the hearings aid's speaker signal.
Howl:	A howling artefact is present in the signal.
Alpha:	An alpha artefact is present in the signal.
Both:	Both a howling artefact and an alpha artefact are present in the signal.

A time instant can only be mapped to a single class as the classes are mutually exclusive.

The definition of output classes, allows for analysis of the proportion of classes in the annotated data set. Figure 2.2 shows a pie chart illustrating the proportion of classes. The data set at hand is dominated by normal, "empty", time instants. The disproportion between the classes is something that must be possible to address or compensate for in the chosen machine learning model. It may also be feasible to preprocess the data in a way that allows for balancing of classes.

2.3 Model of system input

Giving a time instant, as the *only* information to a model, does not seem reasonable if the goal is for the system to predict, whether at that time instant there is an artefact or not. Conveniently, ready available is the "dirty" hearing aid speaker channels and the clean target channel. These audio channels together with the time instant, is the information that the model must be able to predict the presence of artefacts from.

Existing howl detection algorithms [5] use spectral information rather than information from the time domain of the audio signals. The frequency modulation applied in the forward path of hearing aids systems, explained in Section 1.2.1, which is responsible for the alpha artefact, also suggest that the spectral information is relevant to use as input to a classification system.

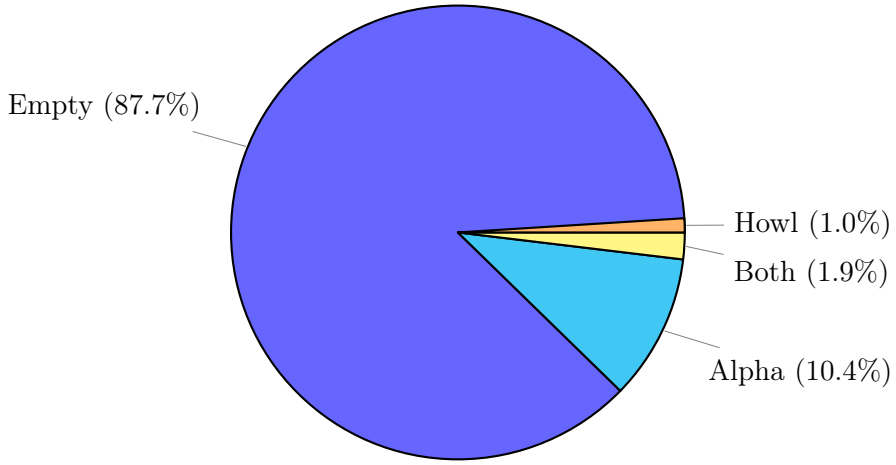


Figure 2.2: Proportion of classes illustrated with a pie chart.

2.3.1 The use of clean and dirty audio

There are three separate, however very correlated, one dimensional signals available. They are: the clean reference channel played to the hearing aid, the left ear hearing aid speaker channel and the right ear hearing aid speaker channel. The two hearing aid channels are perfectly synchronized in time. This is not the case between the clean channel and the two hearing aid channels. There is a lag between the reference and each of the dirty channels.

A meassure that can reveal the lag between two sequences is the cross correlation sequence, if the two sequences are significantly similar [8]. By estimating the cross correlation sequence between the clean channel and the dirty channels separately, a significant peak is found for each channel. Each peak corresponds to the lag between the clean channel and the respective dirty channel.

The lags are slightly different, thus an average is chosen between the two. The clean channel is shifted the appropriate lag, such that the signals are synchronized in time.

2.3.2 Spectro-temporal information with the short-time Fourier transform

The discrete Fourier transform (DFT) is a practical transform used to analyze the frequency contents of finite, discrete signals [9]. An extension to the DFT, and other frequency transforms in generel, which captures the change of frequency content over time is the short-time Fourier transform (STFT) [10]. Ordinary DFT decomposes an evenly sampled time signal using uniformly spaced complex exponentials. Given a discrete time signal $x(n)$ of finite

length, $n = 0, ..., N-1$, the DFT of the signal is given by

$$\text{DFT}\{x\}(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-2\pi j \frac{k}{N} n} \tag{2.1}$$

where $k = 0, ..., N-1$, are frequency bins corresponding to the frequency $\frac{k}{N} \cdot f_s$, where f_s is the sampling frequency of the signal x.

We expect the input for classification to contain time varying frequency information. In order to capture these temporal changes, STFT employs a window moved across the time signal. The time signal can in principle be infinite. Given a discrete time signal $x(n)$ where n denotes a time step, the STFT can be expressed as

$$\text{STFT}\{x(n)\}(m, \omega) = X(m, \omega) = \sum_{n=-\infty}^{\infty} x(n) \cdot w(n-m) \cdot e^{-2\pi j \omega n} \tag{2.2}$$

where m is the time variable, ω is the frequency variable, both in the STFT domain and w is a windowing function. Notice that in this initial definition of the STFT, the frequency variable is no longer fixed to integers, but can be arbitrary frequencies similar to the DTFT [9].

If the window has a finite number of non-zero values, then only those values need to be summed. Let the window length be M. For a fixed choice of slide of the window, m_0, the STFT could be calculated as a DFT with length M and a window.

$$\text{STFT}\{x(n)\}(m_0, k) = X(m_0, k) = \sum_{n=m_0}^{m_0+M} x(n) \cdot w(n-m_0) \cdot e^{-2\pi j \frac{k}{M}(n-m_0)} \tag{2.3}$$

The window length links time and frequency resolution. By increasing the window length through M, the frequency resolution is increased since more frequency bins per time-shift are available. However the longer window decreases the time resolution since each shift of the window requires more time samples. If relevant frequency changes occurs within the window, then these are not captured.

The trade-off between frequency and time resolution is relevant to investigate with the available audio data. Without a classification model, it is difficult to asses which window length result in the better classification. One first way of assessing the trade-off which may not be optimal but indicative is the visual impression of the spectrum for different window sizes. In Section 2.1, examples of artefacts are shown visually using a time varying magnitude spectrum of the hearing aid speaker signal computed using the STFT. Artefacts need to be either audible or visually present in this varying magnitude spectrum to be annotated. Thus if for some window size the artefact is *most* visible, then that window size may be a good choice.

Another relevant parameter of the STFT is the choice of window type. Window types is an extensively studied field [9], with many different window types with varying properties.

An exhaustive numerical study on all possible combinations of windows and window lengths is infeasible considering the scope of this report. A reasonable choice is to use the same parameters as used by humans, though among those, there is not a unified standard. The audio editing software Audacity was used by the annotators. By default the spectrogram option uses a Hanning window with length 1024. The length however, can without loss of visual impression of artefacts be reduced to 256. The choice of shorter window length has the advantage that it reduces the number of dimensions in the transformed signal, which may lead to gains in performance of the classification system.

The specific choice of window length and type leaves a single decision worth presenting. Appropriate stride of the time variable m. A low stride ensures that a lot of spectral information is captured by the STFT, but may also give a lot of redundant information, since the same part of the target signal is used for consecutive valid values of m. If however a too long stride is chosen, some parts of the target signal may be less represented in the transformed signal. If the stride is longer than the window size, some parts of the target signal is not represented in the transformed signal at all. With choice of the Hanning window, an appropriate choice of stride could be half of the window's length, 128 samples. This choice ensures that each sample in the target signal is equally represented in the transformed signal except the ends.

The STFT signal, $X(m, k)$, is complex valued. In Figure 2.1 artefacts are presented visually with the logarithmic magnitude spectrum of the signal. The logarithmic magnitude spectrum of the signal is also used by the human annotators to annotate the audio signals, thus the logarithmic magnitude of the frequency components are indicative for the classification task. The phase, which is not represented in Figure 2.1, could potentially contain useful information, but could also add unnecessary complexity to the input signal. In this report, the phase data will be left out, leaving only the logarithmic magnitude of frequency components to the classification system. The phase data should be included if the the spectrum does not provide indicative information, or could be included in future work.

2.3.3 Using spectral information as input to the clasification system

So far, the audio signals have been transformed with the STFT using a Hanning window with length 256 and 50% overlap. Each complex component of the transformed signal is squared and processed with log10, yielding a time varying logarithmic magnitude spectrum. The signal should be thought of as a matrix with size $C \times F \times T$ where $C = 3$ is the number of audio channels, $F = 128$ is half the number of frequency bins since only half of them are unique due to the nature of the DFT and T is, $T = \frac{2N}{M} - 1$, the number of valid values m can take. Valid choices of m will be referred to as frames. Each frame can be associated to a

time instant, which in this report will be the middle time of the samples from $x(n)$ with are included in the computation of $X(m, k)$.

The classification should be given some part of this matrix and determine which of the four classes that part/time instant belongs to. In this report, time instants that are classfied will coinside with the time instants associated with each frame. The decision of how much information that should be given to the classification system is an interesting choice worth investigating. The amount of data presented to the classification system, from before and after a time instant, will be referred to as context, measured with a number of frames from the logarithmic magnitude spectrum. If the system is given too much context, the input may include information that does not relate to the current time instant. A lot of information also increases the complexity of the input, or increase the dimensionality of the input, which may make the available inputs insufficient for learning due to 'the curse of dimensionality' [11]. On the other hand using too little context, may limit a systems performance since useful information could have been left out.

This trade-off may be the most crucial one presented in this chapter, and will be explored in a later chapter. As a way of controlling how many frames are used for each input, a number of frames, T_f, centred around a frame under investigation, is introduced.

With this, each input, denoted $\mathbf{X}$, given to a classification will be of size $C \times F \times T_f$. A vectorized version if the input is denoted $\mathbf{x} \in \mathbb{R}^{C \cdot F \cdot T_f}$, formed by stacking the two latter dimensions of $\mathbf{X}$ to a single column vector.

2.4 Connecting the system input and output

In this chapter, both a system input and output is defined. For any system to learn the connection between the two, it is necessary to explain how the inputs are related to an output in terms of the ground truth. The related time instant is compared with the ground truth, revealing a class that the input originates from.

Machine learning models for classifying audio inputs 3

Machine learning models used for classification tasks have received an increased amount attention in the recent years. Among many machine learning methodologies, feedforward neural networks are popular and has a broad applicability. Examples include speech recognition [12][13] and in image processing that covers applications from document processing to medical diagnosis [14]. Feedforward neural networks' broad applicability e.g. due to their flexibity, makes them attractive for the application of classfying time instants as well. There is extensive literature on various components of the models, training algorithms and preprocessing methods [11][15].

In this chapter the components of feedforward neural networks are presented. Training of feedforward neural networks is explained including considerations on the loss function and partitioning of data. Then, data preprocessing techniques are presented, which should help the network models perform better. The chapter is concluded with a presentation of the metrics that will be used to evaluate the trained models.

3.1 Components of feedforward neural network models

Feedforward neural network models seek to approximate a function that maps an input $\mathbf{x} \in \mathbb{R}^D$ to an output $\hat{\mathbf{y}} \in \mathbb{R}^K$. The models consist of a network of nodes, organised in layers, feeding intermediate result forward exclusively. The atomic components, nodes, are also referred to as neurons due to their resemblance to neurons in the brain [15]. A single neuron is an affine transform of an input, processed by a non-linear activation function mathematically expresessed as [11]

$$o = f(\mathbf{w}^T\mathbf{z} + b) \tag{3.1}$$

where $\mathbf{z}$ is the input to the neuron, $\mathbf{w}$ is a vector containing weigths applied to each of the entries in the input, b is a bias weight and $f(\cdot)$ is a non-linear activation function. In literature, the bias is often made implicit by appending a constant value to the input $\mathbf{z}$, hiding the bias as the last weight in $\mathbf{w}$ [11]. Moving forward, the later notation is used thus

the output of a single neuron can be expressed as

$$o = f(\mathbf{w}^T \mathbf{z}) \tag{3.2}$$

where the bias is now implicit, hidden in $\mathbf{w}$ by appending $\mathbf{z}$ with a constant. Figure 3.1 graphically illustrates a single neuron.

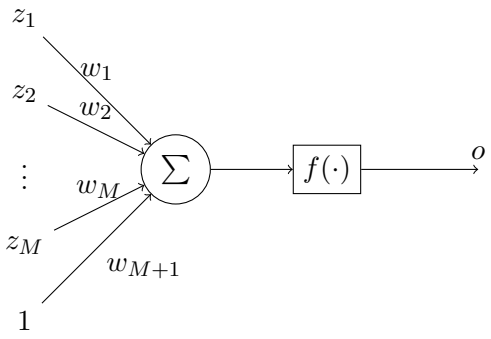


Figure 3.1: Illustration of a node/neuron. The input is scaled by weights, summed and processed by a non-linear activation function. A constant, 1, is appended to the input to include an implicit bias in the neuron.

Neurons are typically arranged in layers, each neuron in the layer connected to the same input but each having their own set of weights. If the layer uses the same activation function, then the mathematical expression of a single neuron in (3.2) can be extended to a layer with an arbitrary number of nodes. By arranging the weights as rows in a matrix, o can be expanded to be a vector. The total activation of the layer can be expressed as [11]

$$\mathbf{o} = f(\mathbf{W}\mathbf{z}) \tag{3.3}$$

where $\mathbf{W}$ is the layer's matrix of weights, each row corresponding to weights of a single neuron and the last column corresponding to the biases. The non-linear activation function f is applied element wise.

Layers of neurons can be placed after one another by using one layer's output as input to the next. Neural networks can consist of an arbitrary number of layers, each with an arbitrary number of neurons. Typically, the input given to the network is referred to the an input layer, but is has no adjustable weights and its size is fixed by the constraints of the problem. The last layer of a network is referred to as the output layer. The width and the activation function of the output layer must fit the purpose of the network. This report's application is multi class classification with four classes, hence an output layer with width four is suitable. Each neuron in the output layer correspond to a class. By using a suitable activation function, the output can be though of as a probability distribution [15]. Between the input layer and the output layer are any number of hidden layers. The output of these layers are hidden within

the network and capture some features of the previous layers, even though these features may be difficult to interpret.

In summary, a model with L layers can be expressed as

$$\hat{\mathbf{y}} = f_L(\mathbf{W}_L f_{L-1}(...f_1(\mathbf{W}_1\mathbf{x})...)) \tag{3.4}$$

where $\mathbf{W}_l$ are the weights in layer l, and f_l is the activation function of layer l. Note that this formulation, makes sense for $L > 1$, but a network where $L = 1$, i.e. a network with no hidden layers also makes sense. A model for binary classification with no hidden layers is referred to as a perceptron in literature [16]. In the case of multi class classification, the network with no hidden layers can be thought of as K parallel perceptrons combined by an activation function.

The optimal choice of layers and activation functions is in practice found experimentally. A neural network with a single hidden layer containing suffiently many neurons, can in theory approximate non-linear functions due to the universal approximation theorem [15]. However, there is no guareentee that the correct weights can be learned. Using multiple layers typically allows the network to achive the same results with fewer weights [15].

3.1.1 Activation functions

Activation functions introduces non linearity to the neural network models and are an essential part that allows classification networks to separate more than just linearly separable classes [11]. A non linear activation function must be applied to all of the networks layers, including the hidden intermediate layers. The activation function of different layers does not need to be the same. Typically, different activation functions serve different purposes in the network. The activation function applied to the output layer has to fit the type of network. In the multi class classification case with K output nodes, the softmax function is a suitable actiavation function [11][15]. The softmax activation function is defined for each entry in the output layer as

$$o_i = \frac{e^{a_i}}{\sum_k e^{a_k}} \tag{3.5}$$

where o_i is the i'th output and a_k is the k'th input. The properties of the softmax function are, that any output is between 0 and 1, and that the total sum of all outputs are equal to 1. These properties are similar to the properties of a probability distribution. Interpreting the output as probability distribution makes the process of using the output to assign inputs to classes easy, since an input intuitively should be assigned to the class that is most likely.

Many of the practical state of the art, neural network models use the rectified linear unit (ReLU) activation function on hidden layers [15]. This function introduces non-linearity, but

uses elements from linear functions, that makes training fast. The ReLU activation function is defined as

$$o_i = \begin{cases} a_i & \text{if} \quad a_i > 0 \\ 0 & \text{if} \quad a_i \geq 0 \end{cases} \tag{3.6}$$

where o_i is the i'th output and a_i is the i'th input. The ReLU activation function makes training fast, since neural network model training typically depends on derivatives. The derivative of the ReLU function is easy and fast to compute.

The network models that are presented in this report uses the ReLU activation function for all hidden layers and the softmax activation function on the output layer.

3.2 Training of feedforward neural networks

The weights in the layers of feedforward neural network models are variables that must be tuned to the application that the model is used in. The fitting of weights is often done using the available pairs of in- and outputs. This concept is called supervised learning. Before training the network i.e. fitting the weights, it is crucial to split data into a training set and test set. It is also important to initialize the network weights strategically as well as select a suitable cost function that the network model should optimize.

3.2.1 Partitioning of data

Besides training a network model, it is just as essential to test and evaluate it. Without dedicated test data, it is impossible to say how a network will handle unseen data. It may be, that the network has "memorized" the training set or over fitted to some specific characteristics that only belong to the training data set. The proportion of the data sets are typically 80% of data for training related use, and 20% for evaluation purposes [15].

Within the data that is put aside for training, there is typically another partition of data. When training a model, parameters that are not part of the network weights also need to be adjusted. These parameters, that are not part of the trainable weights, are referred to as hyper parameters. Hyper parameters include among other things the learning rate and related variables, the number of network layers and the number of neurons in each layer. A validation data set is set a side such that training of the network weights and hyper parameters uses different data. The size of the validation set is typically 20% of the total amount of data [15].

The partitioning of data into three separate parts, training, validation and testing is essential, for training a system that does not over fit to the data that the network is presented to. However deciding which data goes into which part requires some thought. Recall that in the

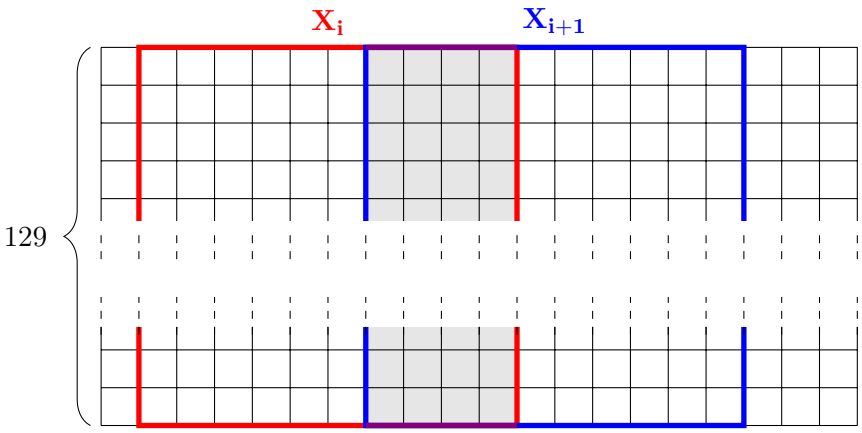


Figure 3.2: Illustration of how two inputs, can share some spectral information. The gray coloured area are the entries that are shared among the inputs $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$.

given application, data belonging to before and after a time instant is presented as input to the network model. If instants are not sufficiently far apart, then inputs will share some spectral information. Figure 3.2 illustrates how two inputs can share some spectral information. It is important that the data in the three sets is not overlapping at all, even if inputs are not exactly the same. To ensure that data is not overlapping between data sets, partition is done at the audio file level. By splitting the data at this level, no concurrent inputs can go into different data sets.

Partitioning of data once into three sets, leaves the question of whether that partitioning was as good as any other partitioning of data. The result may potentially be biased if the designated test set is coincidentally difficult or easy to classify. A method for making sure that the partitioning of data has a negligible influence on the results is K-fold cross validation [11]. K-fold cross validation is a method where the data set is split into K parts. One part is kept for testing, one part is kept for validation and the rest are used for training. K networks are trained and evaluated. For K=5, each part corresponds to 20% of the data from the total data set. We adopt this way of partitioning of data and then show the mean of the performances, together with the minimum and maximum performance.

3.2.2 Initialization of network weights

The initialization of weights has an impact on the final weights of a trained network and can make training converge faster, or even allow a training algorithm to converge, in some cases [15].

The network weights are often initilized using Gaussian distributions. The randomly initilized weights breaks the symmitry of the neurons, whch is important for training [15]. When using

the ReLU activation function in a layer, a heuristic rule is to initialize the biases as small positive constants [15].

The weights in the models presented in this report are initialized by drawing them from a zero mean Gaussian distribution with identity covariance. The biases are initialized as constants of 0.1.

3.2.3 Categorical cross entropy

Before training any network it must be specified what objective the network should optimize. A cost function most often used in multiclass classfication is categorical cross entropy [15][11]. Given the true class of an input in $\mathbf{y}_n$, where $\mathbf{y}_n = \mathbf{e}_k$ and $\mathbf{e}_k$ denotes the vector with 1 in the k'th entry and 0 elsewhere, when the input originates from class k, and an output of a neural network $\hat{\mathbf{y}}_n$, the categorical cross entropy cost function can be expressed as

$$C(\mathbf{y}, \hat{\mathbf{y}}_n) = -\sum_{k=1}^{K} y_k \cdot \log(\hat{y}_k)^* \tag{3.7}$$

where K is the number of classes. Notice that only the term relating to the true class of the input contributes to the error. Since the activation function of the output layer is softmax, entries of the output are limited to values between 0 and 1. If $y_k = \hat{y}_k$, then the error is minimal. For decreasing $\hat{y}_k$, the cost increases exponentially.

When training neural networks, weights are typically not updated for each training pair. Instead, the cost of some set of inputs are summed to form en error, E, and then used for updating the weights. The error can be constructed as a sum of the cost over all of the inputs in the training set. This approach ensures that all data contributes to the updating of weights, but is also a relatively slow approach. Typically, the training data is split into a number of mini-batches [15]. Each mini-batch is used to calculate the error that is in turn used to update the weights. Let $\hat{\mathbf{y}}_b$ be the output of the network when given $\mathbf{x}_b$, the b'th input in a given mini batch. The error of a given mini batch as a function of all of the weights in the network can be expressed as [11]

$$E(\theta) = -\sum_{b=1}^{B}\sum_{k=1}^{K} y_{bk} \cdot \log(\hat{y}_k(\theta, \mathbf{x}_b)) \tag{3.8}$$

where $E(\theta)$ is the error of the mini batch given the trainable weights θ in the network, y_{bk} is the k'th entry in the b'th true label and the equation emphasizes that $\hat{y}_k$ is a function of both an input, $\mathbf{x}_b$ and the adjustable weights, θ, in the network.

The cost function and the error function weights all inputs and all classes equally. Recall from Section 2.2 that the data set available for the problem at hand is very imbalanced,

*The limit of the cost function is $\lim_{x\to 0} x \cdot \log x = 0$

meaning one class is far more represented in the data set than the rest. The nature of the cost and error, means that the majority class will account for most of the error. Thus if no balancing is done to the error function, the network will be trained to minimize the error, mostly accounted for by the majority class.

Balancing of the error and cost function can be done by weighting each of the classes differently [17]. By multiplying each of the terms in the sum by a class dependant weight, the skew in representation can be compensated for. The weighted error function can be expressed as [17]

$$E_w(\theta) = -\sum_{b=1}^{B}\sum_{k=1}^{K} y_{bk} \cdot \log(\hat{y}_k(\theta, \mathbf{x}_b)) \cdot w_k \tag{3.9}$$

where $E_w(\theta)$ is the weighted error function and w_k is the weight associated with class k. Appropiate choices of weights are presented in [17]. In [17] the weight is inverse proportional with the frequency of the classes in the training data set. The weights can be expressed as

$$w_k = \frac{f_{median}}{f_k} \tag{3.10}$$

where f_k is the number of samples of class k in the training data set and f_{median} is the median of the frequency counts among the K classes.

Balancing of the error function is one out of multiple possible ways of addressing the class imbalance. The available time, limits the scope of balancing methods that are investigated in this report. Balancing of classes is discussed again later in a numerical investigation of feedforward neural network models.

3.2.4 Training with stochastic gradient decent

The error function on its own does not tell which weights result in the lowest error. Naturally we would search for a stationary point of weights that minimize the error function, but the nature of the error function and neural networks in general makes it difficult due to the errors non-convex nature. A general technique that works well in many types of optimization is to makes adjustments to the set of weights iteratively [11]

$$\theta^{\tau+1} = \theta^{\tau} + \Delta\theta^{\tau} \tag{3.11}$$

where τ indicate the iteration and Δ is some change. A simple approach that uses the gradient of the error is gradient decent. In the context of errors summed over samples drawn randomly from the entire data set, the algorithm is often referred to as stochastic gradient decent (SGD), where the stochastic adjective refers to the random sampling of mini batches.

The SGD algorithm uses a scaled version of the negative derivative of the error function as the choice of Δ in (3.11). The general iterative approach of updating the weights is [11]

$$\theta^{\tau+1} = \theta^{\tau} - \eta \nabla_{\theta} E_w(\theta) \cdot \theta^{\tau} \tag{3.12}$$

where η is a step length that decides how much the weights are changed for each iteration.

The algorithm intuitively updates the weights in a direction, that reduces the cost as long as the step size is sufficiently small. There exists numerous extensions and variations of SGD, but as a starting point, simple SGD as described in (3.12) should suffice.

The derivate with respect to the network weights are typically computed using the back-propagation algorithm [11]. With backprobagation the gradient is computed in a backwards manner one layer at a time [11].

The weights are updated with the rule in (3.12) after each computation of the derivative of a mini batch. Epochs, which denote the time instant that the network has seen all data in the training set i.e. all data has been processed through a mini batch. Epochs are used as a count for how long a network has been trained for.

3.2.5 Early stopping

When training a feedforward neural network model, there is a significant chance that the model complexity allows the network to over fit to the training data set. To prevent over fitting, training of the network can be stopped at the epoch where the validation loss stops decreasing. Since the validation data set is not used for fitting the network weights, it acts as a test set in that sense. In this report, early stopping is used. The model from the epoch where the validation loss is minimal is used as the final model.

3.3 Preprocessing of data

Conditioning of data before it is presented to the neural network model, can both speed up the training process, and possible increase the networks performance. Different techniques exist for preprocessing of the input. Among relevant technique for this report's application and data, is both whitening and dimensionality reduction. Whitening is the process of decorrelating the variables in the input, and sometimes include mean subtraction. The process is know to reduce training time of neural networks [18]. Dimensionality reduction, is a relevant preprocessing technique, if the data set is of low intrinsic dimension [11]. This means that each high dimensional signal, in this case the spectrograms, can be described by a different but much smaller set of dimensions.

3.3.1 Dimensionality reduction with Principal Component Analysis

A popular machine learning technique, that is useful for finding the most significant dimensions of the data set is Principal Component Analysis (PCA). The technique is used in a large varity of application [11].

As the name suggests, PCA is a deconstruction of the data into the most important signal components. One way of thinking about the signal components is to think about the orthogonal projection, that maximizes the variance when the data is projected onto a one dimensional subspace [11]. Recall that $\mathbf{x}_n \in \mathbb{R}^D$, $n = 1, ..., N_{train}$ are training inputs to the neural network. Let $\mathbf{u} \in \mathbb{R}^D$ be a vector that projects $\mathbf{x}_n$ onto a one dimensional subspace by $\mathbf{u}^T\mathbf{x}_n$. Now, the goal is to maximize the variance of these scalar projections, which can be expressed as

$$\frac{1}{N_{train}} \sum_{n=1}^{N_{train}} (\mathbf{u}^T\mathbf{x}_n - \mathbf{u}^T\boldsymbol{\mu}_\mathbf{x})(\mathbf{u}^T\mathbf{x}_n - \mathbf{u}^T\boldsymbol{\mu}_\mathbf{x}) = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} \mathbf{u}^T(\mathbf{x}_n - \boldsymbol{\mu}_\mathbf{x})(\mathbf{x}_n - \boldsymbol{\mu}_\mathbf{x})^T\mathbf{u} = \mathbf{u}^T\mathbf{S}\mathbf{u} \tag{3.13}$$

where $\boldsymbol{\mu}_\mathbf{x} = 1/N_{train} \sum_{n=1}^{N_{train}} \mathbf{x}_n$ is the sample mean and $\mathbf{S} = 1/N_{train} \sum_{n=1}^{N_{train}} (\mathbf{x}_n - \boldsymbol{\mu}_\mathbf{x})(\mathbf{x}_n - \boldsymbol{\mu}_\mathbf{x})^T$ is the estimated covariance matrix from the the N_{train} data points.

The goal is to maximize the variance $\mathbf{u}^T\mathbf{S}\mathbf{u}$, but this problem is trivial if the magnitude of $\mathbf{u}$ is not constrained. Only the direction of $\mathbf{u}$ is interesting, thus the projection is constrained to have unitary length, $\mathbf{u}^T\mathbf{u} = 1$. With the objective and constraint, we form an optimization problem. The constraint is moved to the objective by a Lagrange multiplier giving the following optimization problem

$$\max_{\mathbf{u}} \quad \mathbf{u}^T\mathbf{S}\mathbf{u} + \lambda(1 - \mathbf{u}^T\mathbf{u}) \tag{3.14}$$

where λ is a Lagrange multiplier. By setting the derivative with respect to $\mathbf{u}$ equal to 0, and rearranging the equation, it reveals that possible solutions to the optimization problem must be eigenvectors of the estimated covariance matrix. This property is shown in the following equation.

$$\begin{aligned} \nabla_\mathbf{u}(\mathbf{u}^T\mathbf{S}\mathbf{u} + \lambda(1 - \mathbf{u}^T\mathbf{u})) &= 0 \\ \nabla_\mathbf{u}\mathbf{u}^T\mathbf{S}\mathbf{u} - \lambda\nabla_\mathbf{u}\mathbf{u}^T\mathbf{u} &= 0 \\ \mathbf{S}\mathbf{u} &= \lambda\mathbf{u} \end{aligned} \tag{3.15}$$

Among the possible solutions, $\mathbf{u}$ should be the eigenvector corresponding to the largest eigenvalue, since this is the one that maximizes the variance [11]. The vector $\mathbf{u}$ is called the most principal component of the data set.

The idea of projecting the data onto a one dimensional subspace can be extended to multiple dimensions, by introduction more vectors, each orthogonal to one another. For some data

set of dimension D we want to project onto a smaller subspace of dimensions M. The most principal components are the M eigenvectors corresponding to the M largest eigenvalues. Notice that the eigenvectors are orthogonal to one another, thus in the transformed space, the data set will be decorrelated i.e. have a diagonal covariance matrix.

PCA, summarized, is to find the eigenvalue decomposition of the sample covariance matrix $\mathbf{S}$ and construct a transformation matrix $\mathbf{U} \in \mathbb{R}^{M \times D}$, where the rows are the desired number of eigenvectors corresponding to the largest eigenvalues. Each input is transformed from a high dimensionality to a lower dimensionality with $\mathbf{Ux}$.

The choice of data used in estimation of eigenvectors from the sample covariance matrix must be done carefully when dealing with different sets of data. The test and validation data sets cannot be used for estimation of principal components. Instead the eigenvectors shall be estimated only from the training data set, and then used to transform all of the data sets. If the distribution of the data sets are similar then the transformed test and validation data sets will have similar properties to the transformed training data set.

3.3.2 Numerical investigation of the eigenvalues of the data set

The intrinsic dimension of data sets can vary widely between applications [11], thus it is necessary to investigate how few eigenvalues can explain the variation in the data set. In this numerical investigation all data from the problem at hand is used, since we only care about finding out how many principal components are relevant and not the transformation matrix $\mathbf{U}$ itself. Figure 3.3 shows a cumulative plot of the eigenvalues in descending order. The plot is normalized with the total sum of the eigenvalues on the y-axis and the total number of dimensions on the x-axis.

Figure 3.3 reveals that as few as 10% of the eigenvalues values makes up 90% of the variance in the inputs, and that each of the following 90% of eigenvalues does not contribute significantly to the variance. If the original inputs are transformed using the most significant 10% of eigenvectors, then the dimensionality will have been decreased by 90%, while retaining 90% of the variance in the original signal. There are generally no correct ways of determining how much of the variance can reasonably be discarded without loss of important information, but from the plot it is obvious that including more than 10% of the data, yield no significant amount of added variance. There is no certainty that the relevant data, that holds the information about an inputs correct class, is not located in the last 10% of variance. One way of assessing if the essential data is lost, is by 'inverting' the PCA on some of the inputs, and see if the characteristics of the artefacts are lost. The inverse of PCA is to scale each eigenvector, with the respective intrinsic dimension which can be expressed as

$$\hat{\mathbf{x}} = \mathbf{U}^T \mathbf{x}_{PCA} = \mathbf{U}^T \mathbf{U} \mathbf{x} \tag{3.16}$$

where $\hat{\mathbf{x}}$ is the reconstructed input and $\mathbf{x}_{PCA}$ is the input, which has been transformed by PCA. Figure 3.4 shows the channels of an input, that falls in the category 'both'. The left column of images, shows the original signal before PCA. The right column shows the reconstructed input, when 10% of principal components are used. It is certainly possible to visually tell that some details of the original input are missing in the reconstructed signal. But the key characteristics are still visually present in the spectrogram, thus this example suggests that the defining characteristics of the input are kept through dimensionality reduction with PCA.

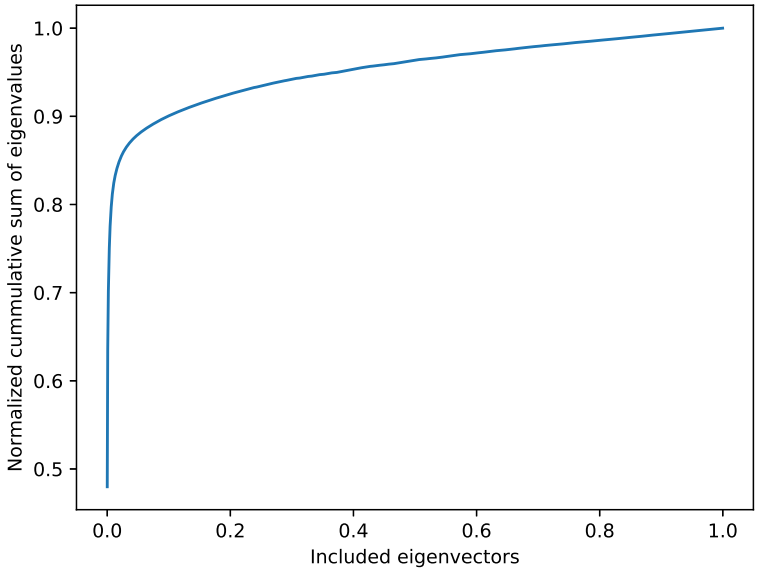


Figure 3.3: Cumulative plot of the eigenvalues in the estimated covariance matrix from the available inputs.

3.3.3 Decorrelating the inputs

Besides reducing the dimensions of the inputs, it is also preferable if the inputs are zero mean and have identity covariance. The process is referred to as *whitening*, and mathematically it is expressed as [11]

$$\mathbf{x}' = \mathbf{H}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}) \tag{3.17}$$

where $\mathbf{x}'$ is the whitened input, $\boldsymbol{\mu}_{\mathbf{x}}$ is mean across each dimension of $\mathbf{x}$ and $\mathbf{H}$ is a transformation matrix with the property that $\mathbf{H}\mathbf{H}^T = \mathbf{S}^{-1}$, $\mathbf{S}$ being the estimated covariance matrix of inputs $\mathbf{x}$. From PCA, the eigenvalues and the eigenvectors of the estimated covariance

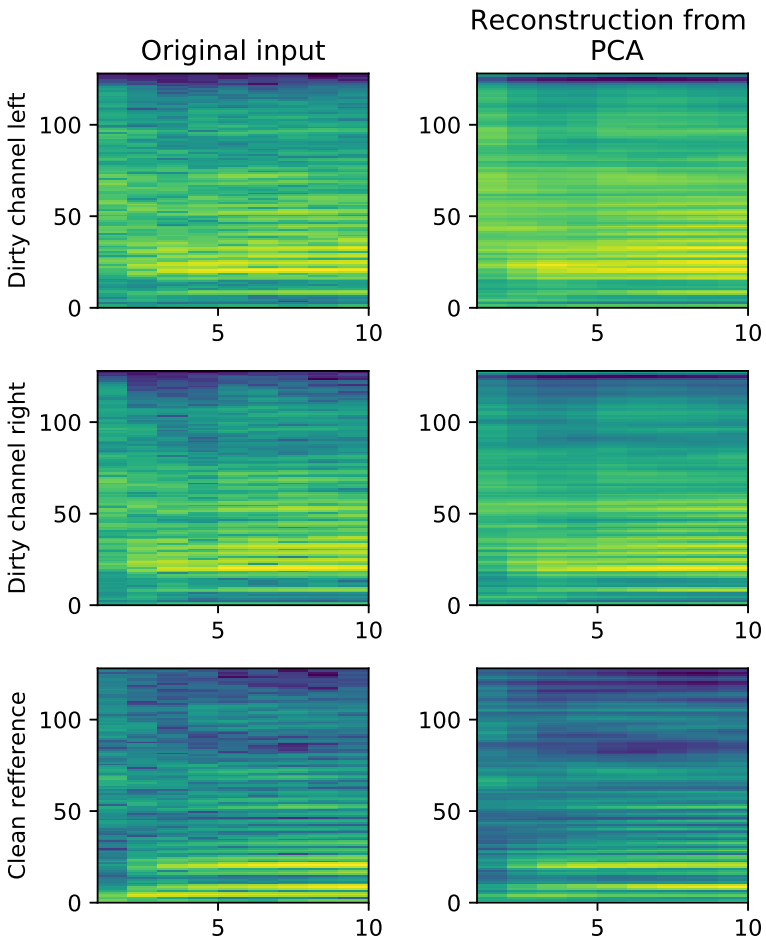


Figure 3.4: Spectrograms of the channels of an inputs, containing the both artefacts. Left column: Original input. Right column: reconstructed signal, using only 10% most principal components. Some details are missing in the reconstructed spectrograms, but the key characteristics remain.

matrix is already given. If $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues in descending order, and $\mathbf{U}$ is a matrix with corresponding eigenvectors, then $\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}$ is a whitening transform

$$\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}(\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}})^T = \mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^T = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T = \mathbf{S}^{-1} \tag{3.18}$$

since $\mathbf{U} = \mathbf{U}^{-1}$. Rows of the diagonal matrix, and colums of the orthogonal eigenvector matrix can be removed without losing the whitening properties. By removing a sufficient amount of rows and columns, both the dimensionality reduction from PCA and the whitening can be accomplished with one transform.

Figure 3.5 shows a block diagram of the preprocessing applied to the the data, before training and evaluation of the neural network model.

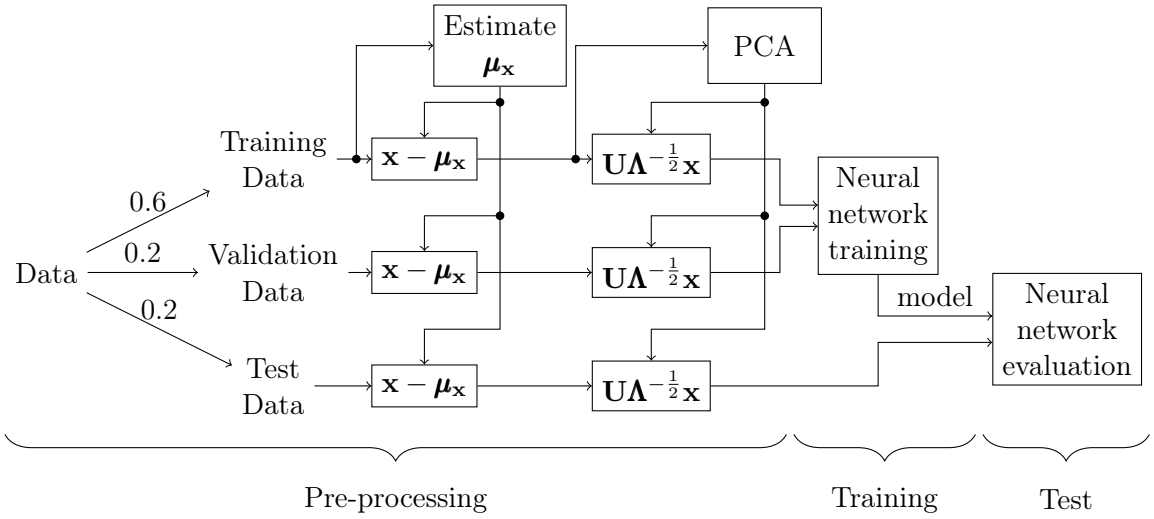


Figure 3.5: Illustration of the preprocessing applied to the data before training and evaluation of the neural network model.

3.4 Evaluation of networks

To assess the performance of trained network models, metrics are used. These allow different models and classification systems to be compared. Recall that for each signal/input, $\mathbf{x}$, that is presented to the classification system, a corresponding output, $\hat{\mathbf{y}} \in \mathbb{R}^K$, is predicted. Each entry, $\hat{y}_i$, indicates the how likely $\mathbf{x}$ is to belong to class i compared to the other classes [15]. Additionally to the predicted label, we also know the true label, $\mathbf{y}$, which has zero entries excluding one entry, y_j, which is 1, indicating the input $\mathbf{x}$'s true class, j. The metrics that are presented should somehow compare $\hat{\mathbf{y}}$ and $\mathbf{y}$ for the inputs in the test data set. The number of samples in the test data set will be referred to as N_{test}.

When comparing the true and predicted labels, it must be clear whether the predicted and true labels are compared directly or by assignments to a specific class. Both approaches are feasible but initially the latter is used in this report. For each predicted label, the corresponding input is assigned to the class that is most likely i.e. the class corresponding to the entry with greatest magnitude. By doing this, closeness of likelihoods is disregarded. The assignment of inputs of classes naturally leads to confusion matrices.

A confusion matrix is a $K \times K$ matrix, denoted $\mathbf{C}$, where $C_{i,j}$ corresponds to the number of inputs that are predicted to be from class i, and that originate from class j. Thus the diagonal of the confusion matrix indicates the correct predictions made by a classification system. The entries off the diagonal, are all combinations of misclassifications. Just by visual inspection of the confusion matrix, it might be possible to get an impression of performance and potential problems. If for example two entries symmetric around the diagonal have a relatively large

value, it indicates that the system struggles with separating the two corresponding classes. Visually comparing two confusion matrices is however not a good metric as it lacks overview and can be subjective.

The confusion matrix can be used to calculate a variety of different metrics. Most metrics originate from binary classification, but can be extended to multiclass classification [19]. The report limits it evaluation to five relevant metrics: Accuracy, sensitivity, precision, F_1 score and Matthews correlation coefficient. Each metric is introduced as a metric for evaluating binary classification systems to show their relation to the reality, and then extended to multi class classification, where two versions must be considered. One version where averaging is done over the whole data, macro averaging, denoted by a subscript M and another version where averaging is done over counts, micro averaging, denoted by a subscript μ.

3.4.1 Accuracy

The metric *accuracy* (ACC) is defined as the trace of the confusion matrix divided by the total sum of the matrix

$$\text{ACC} = \frac{tr(\mathbf{C})}{N_{test}} \tag{3.19}$$

where ACC is the accuracy and $tr(\cdot)$ is the trace operator. A high accuracy generally means a good performance in terms of predicting well, overall. The metric is however skewed by imbalanced data. This is best shown by example: Consider a system that must predict, whether a specific human has been to the moon or not. A system that always predicts that the human has not been to the moon will have a very good accuracy, since most people have not been to the moon, yet. However we would not say that the systems performance is good since from the perspective of detecting people that have actually been to the moon, the system performs terribly.

3.4.2 Precision

Precision or positive prediction value (PPV), in the context of binary classification is the number of *correct* positive predictions, divided by the total number of positive predictions. When extending precision to multi-class classification, we think of precision first in terms of each class individually. Namely, the number of *correct* prediction of a class divided by the total number of predictions of that class [19]

$$\text{PPV}_i = \frac{C_{i,i}}{\sum_j C_{i,j}} \tag{3.20}$$

where PPV_i is the precision of class i. A high precision of class means that most of a system's predictions of that class are correct. However, it does **not** tell if the system has correctly predicted a high proportion of the available inputs originating from that class.

To combine the class specific precisions, PPV_i, an average is used. If an average over counts is used, *micro-averaging*, the metric becomes the same as accuracy, which will provide no new information

$$\text{PPV}_\mu = \frac{\sum_i C_{i,i}}{\sum_i \sum_j C_{i,j}} = \frac{tr(\mathbf{C})}{N_{test}} \tag{3.21}$$

An average over classes, *macro averaging*, denoted with a subscript M provides new information and can be used to asses the precision of a system in general

$$\text{PPV}_M = \frac{\sum_i \text{PPV}_i}{K}$$

where K is the number of classes.

3.4.3 Sensitivity

Sensitivity or true positive rate (TPR) in the context of binary classification is the number of *correct* positive predictions divided by the total number of available positive inputs. When extending sensitivity to multi class classification, sensitivity is thought of first in terms of each class individually. Namely, the number of correct prediction of a class divided by the *total* number of available inputs of that class [19]

$$\text{TPR}_j = \frac{C_{j,j}}{\sum_i C_{i,j}} \tag{3.22}$$

where TPR_j is the sensitivity of class j. A high sensitivity of a class means the system is good at detecting instants of that class specifically. It does **not** tell if the system also makes many false positive predictions in the process, i.e. has a low precision with respect to the class.

If an average over counts is used to combine sensitivities, the metric is the same as accuracy similarily to precision. The macro averaging version of sensitivity can be expressed as

$$\text{TPR}_M = \frac{\sum_j \text{TPR}_j}{K} \tag{3.23}$$

3.4.4 $\mathbf{F_1}$ score

Naturally, a metric combining precision and sensitivity is relevant. The F_1 score is a geometric mean of the precision and sensitivity. This means that if either one of the metrics are low, the F_1 score will also be low. Only by achieving a relatively high precision and sensitivity simultaneously, the F_1 score will be high. The class specific F_1 score is defined as [19]

$$\text{F}_{1_k} = \frac{2 \cdot TPR_k \cdot PPV_k}{TPR_k + PPV_k} \tag{3.24}$$

where F_{1_k} is the F_1 score of class k. The mean version of the the F_1 score can be expressed as

$$\mathrm{F}_{1_M} = \frac{2 \cdot TPR_M \cdot PPV_M}{TPR_M + PPV_M} \tag{3.25}$$

3.4.5 Matthews correlation coefficient

Matthews correlation coefficient (MCC) is a sort of correlation coeficient between the predicted label, and the true label. MCC is often used within the field of binary classification, but the metric is exstended to multiclass classficaition in [20]. In the recent paper [21], MCC is shown to be a more informative metric than accuracy and F_1 score in binary classification since MCC is not subject to bias from imbalanced data sets, and is invariant to class swapping. MCC for multi class classification can be expressed as [20]

$$\mathrm{MCC} = \frac{N_{test} \cdot tr(\mathbf{C}) - \sum_i \sum_j \tilde{\mathbf{C}}_i \hat{\mathbf{C}}_j}{\sqrt{N_{test}^2 - \sum_k \sum_i \tilde{\mathbf{C}}_k (\hat{\mathbf{C}^T})_i} \cdot \sqrt{N_{test}^2 - \sum_k \sum_i (\tilde{\mathbf{C}^T})_k \hat{\mathbf{C}}_i}} \tag{3.26}$$

where $\tilde{\mathbf{C}}_i$ is the i'th row of $\mathbf{C}$ and $\hat{\mathbf{C}}_j$ is the j'th column of $\mathbf{C}$. For a system to achieve a high MCC, it must classify the majority of all classes correctly [21].

3.4.6 Summary of evaluation of classification systems

In summary, five different types of metrics are presented, some with class specific and average versions, and some metrics that measure the systems performance as a whole, namely accuracy and MCC. Each of the metrics measures some different performance aspect of the system, and the perfect system would perform well on all metrics at the same time. Figure 3.6 shows an example of a confusion matrix with all the metrics presented next to it. The key figures that define the overall performance of the system is the average precision, sensitivity and F_1 score and MCC. One should be careful with using accuracy as a metric, without the other measures in mind, since it will mostly depend on the majority empty class.

Even though the results shown in Figure 3.6 are artificial, it is worth familiarizing with interpreting the result and commenting on the corresponding system's performance. As a start, notice that all of available examples of the empty class are correctly classified. Hence the sensitivity with respect to the empty class is 1. However, in the process, a lot of the predictions of the empty class are wrong, hence the low precision. In the given example classes are fairly balanced, and the accuracy can be used to say the system generally predicts 53% of given inputs correctly. The MCC is harder to concretize, but works well if the metric is compared between multiple classification systems.

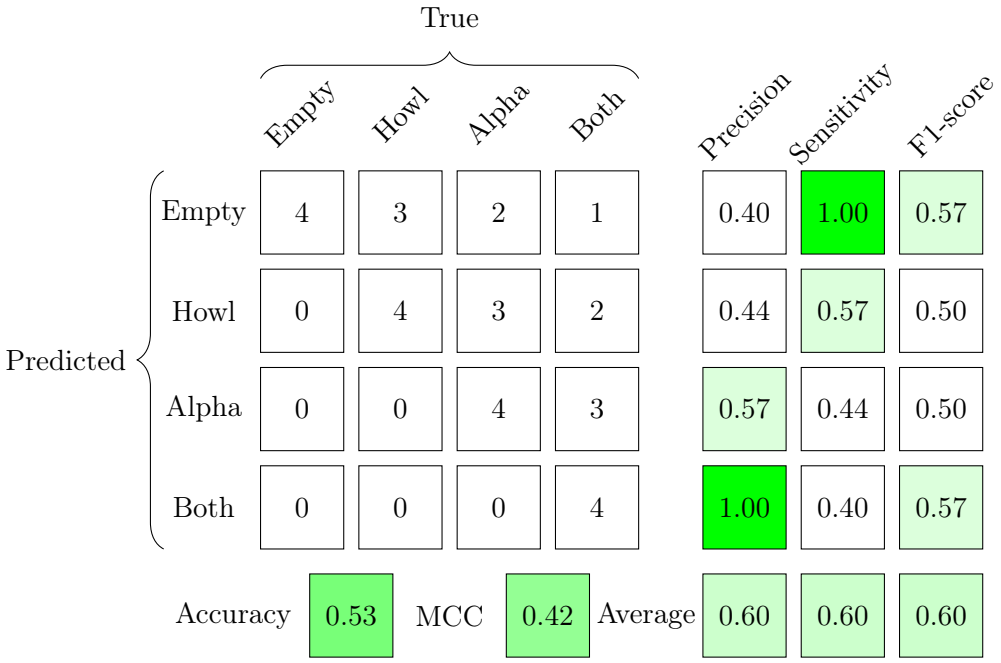


Figure 3.6: Artificial example of a confusion matrix and all the metrics which describe the performance of the system that produced the confusion matrix.

Artefact classification - A numerical investigation 4

In this chapter different feedforward neural network models are trained and evaluated. Before diving into the neural network models, an artefact detection system developed by Oticon in MATLAB is presented and evaluated using the presented metrics. The MATLAB system is used as a reference for all the presented neural network models.

The numerical investigation of neural network models starts with simple models that have a single hidden layer of varying size. The models are trained, compared with each other, with the MATLAB system and a network without hidden layers. The models are evaluated using the metrics presented in Section 3.4.

Before exploring more complex models, the amount of inlcuded context is investigated, reusing the same single hidden layer neural network models. The investigation varies the number of included frames.

The way of including the clean reference channel is investigated, through different ways of processing. Four different ways of handling the reference are proposed and compared in terms of performance using the single hidden layer network models.

By analysing the class specific metrics of the best performing model, classes with the most potential for increased performance are identified. Efforts are done to utilize the potential and directions for further work is given.

The chapter finally compares the best performing neural network model and the MATLAB system in the time domain. The presented metrics disregard the timely relation between concurrent inputs. By converting the classification task back to the annotation task, both interesting results and directions for future work are identified.

4.1 Classifying inputs using a MATLAB detection system

Oticon has developed a system capable of detecting artefacts in hearing aid audio signals, similarly to what is presented in Section 2.1. The MATLAB system requires an audio channel with artefacts and the corresponding clean version of that signal. The system's output is similar to the format of the annotations presented in Section 2.1. Each detected artefact has a starting point and duration. The detected artefacts can be converted to the classification format similarly to how the humanly annotated data is converted. This means that if a time instant falls within a detected artefact, then that time instant is classified as that artefact. If a time instant falls within both types of artefacts, it belongs to the 'both' class. The distance between each time instant can be arbitrarily small. For this report, a time step of 0.05 ms is used corresponding to time difference between samples in the recorded audio signals.

To evaluate the performance of the MATLAB system, the system is first used to annotate all of the audio files in the data set. Recall that each audio signal contains two channels. The annotations of the two channels are merged using the rule of artefacts overruling empty/clean time periods. This is similar to what is used for conflicting human annotations. The single annotation for each file is then converted to the classification format and evaluated the same way as the neural networks, i.e. by comparing the predictions with the ground truth.

Figure 4.1 shows the confusion matrix and metrics of the MATLAB system when evaluated on all of the available audio data. The confusion matrix supports the claim that the classes are highly imbalanced. There are empty inputs in the order of 10^8 and howl inputs in the order of 10^6. When evaluating the system, the accuracy will be dominated by the empty class. The system's performance with respect to the empty class is good both with respect to precision and sensitivity. The performance with respect to the howl class stands in contrast, as the MATLAB system only detects 42% of the time instants where a howl is present, while 80% of its predictions that an input belongs to the howl class are incorrect. The performance with respect to the alpha class and both class are between the other two classes. For those two classes the most significant characteristic is that the precision is much greater than the sensitivity. The system is favoured towards not predicting these classes incorrectly. The averages and especially the MCC, will be referred to later, when the neural networks are evaluated and systems can be compared.

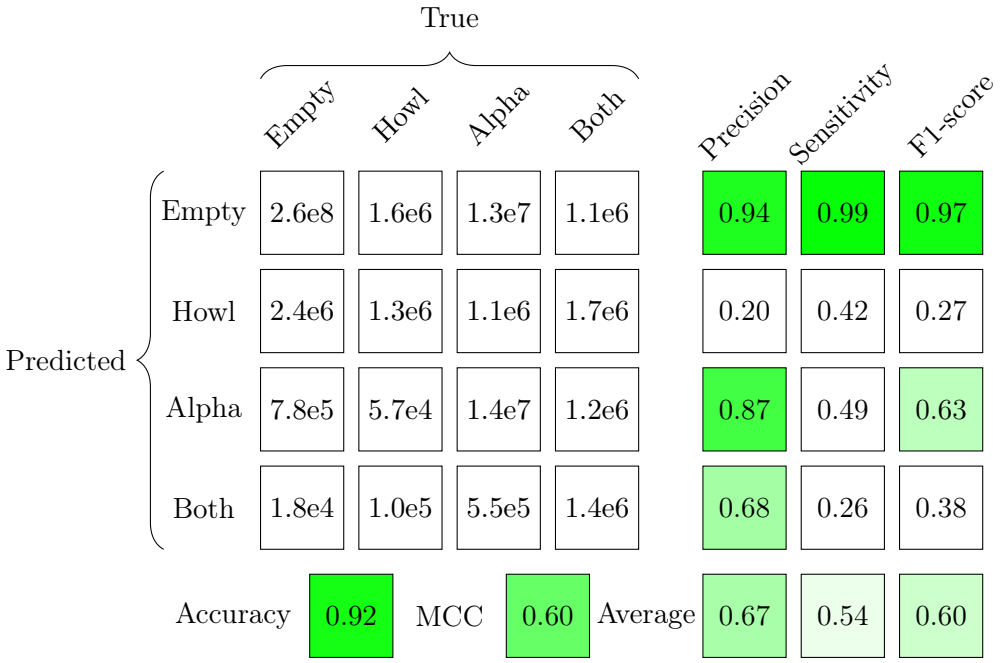


Figure 4.1: Confusion matrix and metrics for the MATLAB system when evaluated on all the audio data at hand.

4.2 Single hidden layer feedforward network models

Single hidden layer feedforward neural networks are an intuitive starting point for investigations of the neural network models. The single hidden layer networks are compared with each other, the MATLAB system and a neural network with no hidden layers, that will be referred to as a baseline network. All networks are trained for 150 epochs, but with early stopping. The weights that are used for evaluation, are from the epoch where the validation loss is minimal. K-fold cross validation with 5 parts is used to test each choice of neurons in the hidden layer. SGD as is used for updating the weights, with a batch size of 32 and the learning rate 0.01 which is found to be reasonable choices. The choice of neurons in the hidden layer starts at 32, and doubles all the way to 1024 neurons.

As for the choice of included context, 10 frames are included in each input. The choice of included frames is investigated further in a later section.

Figure 4.2 shows the average loss of the training, validation and test data set, during training, for the baseline network without any hidden layer on the left, and for the single hidden layer network with 128 nodes on the right. Networks with no or few hidden neurons do not over fit as much as networks with a greater number of hidden neurons. However, the simpler

networks reaches a limit where even the training loss cannot be reduced further. The grey dashed lines in the plots denote the epoch where the validation loss is minimal. It is the weights from this epoch that are used to evaluate the model.

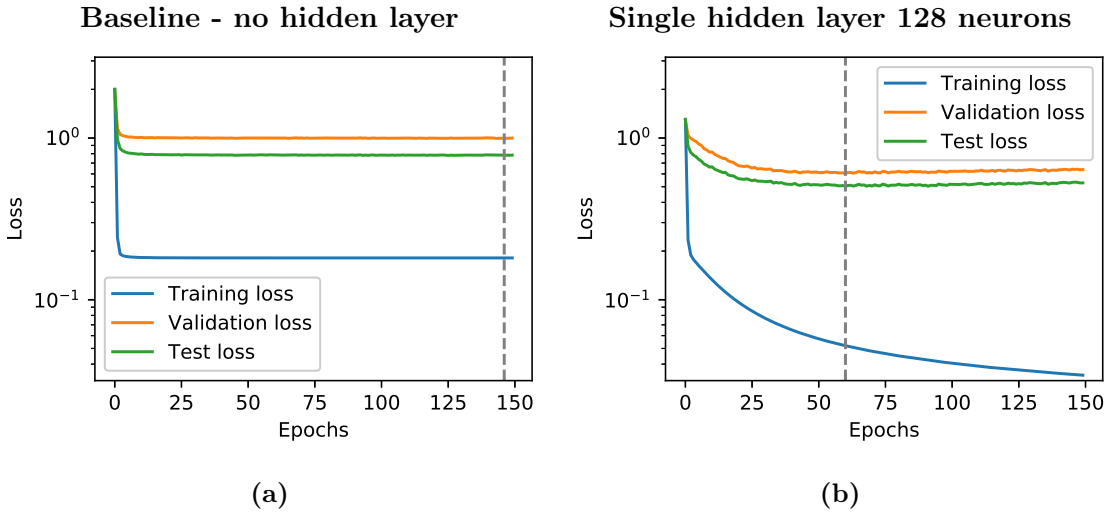


Figure 4.2: Average loss of the training, validation and test data sets during training. **(a)** Loss of the baseline network that does not have a hidden layer. **(b)** Loss of the single hidden layer network with 128 neurons. The vertical grey dashed line denotes the epoch where the validation loss is minimal.

Figure 4.3 shows plots that compare the network models with respect to the five presented metrics. Each bar in the plots is the average metric, for all five trained networks with the same number of neurons. The dark errorbars denote the maximum and minimum value among the five trained models. The MATLAB system is included as a horizontal dashed line in each plot. Comparing the network models with each other, reveals that an increased number of neurons, increases the metrics in general. The gain shrinks with each doubling of neurons. Notice that sensitivity decreases for networks with more than 128 nodes. However, the loss in sensitivity is made up for in precision, since the F_1 score shows an increasing trend.

None of the network models perform better than the MATLAB system overall. The networks only score higher on the sensitivity metric, but it comes at the cost of a much lower precision. The F_1 score reveals that when combining the two, the MATLAB system is better overall. The MCC metric also suggests that the MATLAB system performs better.

4.3 Investigation of included context

Parallel to the investigation of different network models, it is equally interesting to study how much context should be included in each input for the network to achieve the best performance. The free parameter that decides how much context is included in each input is the number of included frames T_f presented in Section 2.3. In this report, the search is

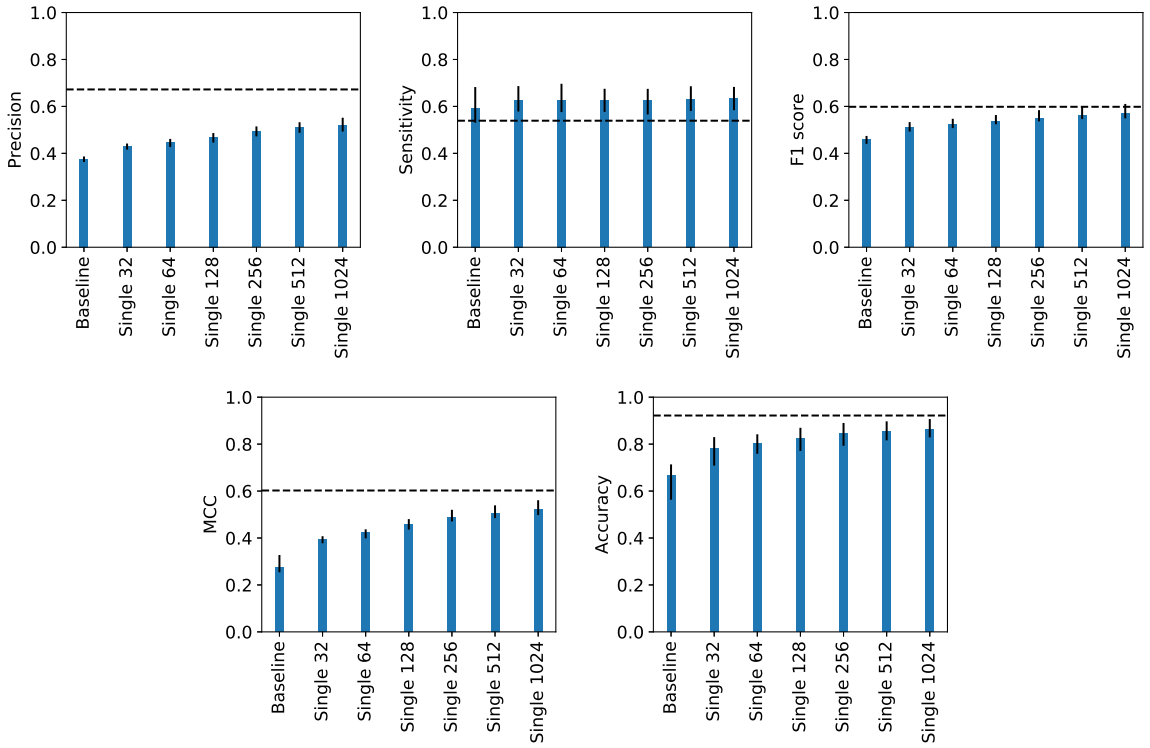


Figure 4.3: Key metrics for network models, with a single hidden layer of varying width. The baseline network is a model with no hidden layer. The trend is that performance increases with the growing width of the hidden layer, but that the performance reaches a plateau.

limited to 5, 10 and 15 frames. All three choices of included frames are investigated using the single hidden layer networks.

Figure 4.4 shows a plot of the metrics for the single layer networks for each of the three choices of included context. The results show that number of included frames and performance are proportional. Among the limited choices of included frames the best performance is achieved when using 15 frames. The 15 frames corresponds to approximately 100 ms. Future work should investigate including even more frames, and determine if there is a limit to how much context should be included, to achieve the best performance. Investigations in this report, will use 15 frames moving forward. There is no guarantee that this choice is the best, when combined with the other free parameters. An exhaustive search of all combinations of free parameters is beyond the scope of this report.

The improvements achieved by including more context, moves the performance of the single hidden layer models closer to the performance of the MATLAB system. It could even be argued that their performance are equal, though they value mistakes differently. The MATLAB

system has a higher precision, while the neural networks have a higher sensitivity. Further improvement, achieved by including more context, could give neural networks the edge over the MATLAB system.

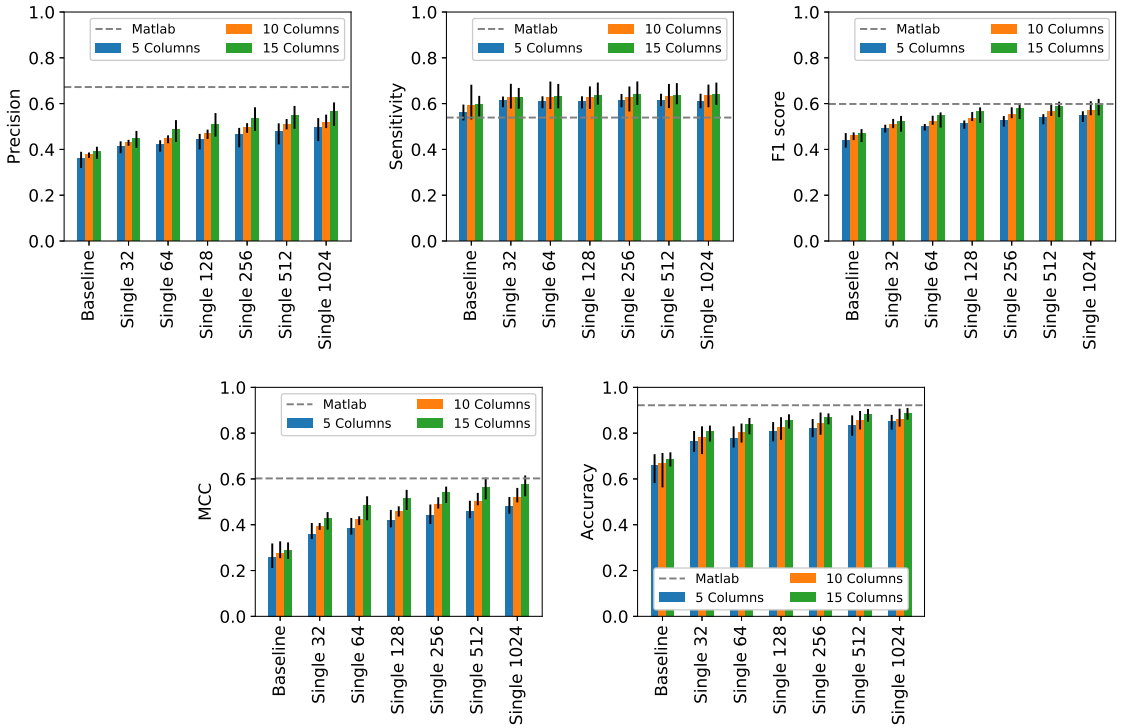


Figure 4.4: Plot of the metrics, for each of the three choices of included context, namely 5, 10 and 15 frames in each input. Including more data, results in better performance. The number of included frames are from left to right: 5, 10, 15.

4.4 Processing of the reference channel

So far, the clean reference channel is presented to the network models, in a way that allows the network to process it, however it find suitable.

There is no straight forward way of checking, how the neural network compares the clean reference with the dirty channels, or if it uses the clean channel for anything at all. The system's performance, may improve if the comparison between the reference channel and the dirty channels is forced before all preprocessing of the inputs. In this section, four ways of processing of the clean channel are proposed and compared. The processing methods are:

- Including the reference as a channel. This is simply the default processing that is presented in this report already.

- No reference channel. By removing the reference channel, the default processing method can be compared to not including the clean reference at all.
- Absolute difference between reference and the two channels separately. This forces the network to use the difference between the clean and dirty channels.
- A correlation coefficient comparing the clean and dirty channels over time. The correlation coefficient is like the absolute difference, but in the sense that it will reveal the similarity of the signal for each frequency bin.

The three new ways of processing the reference are explained in its own subsection

4.4.1 Removing the reference

One way of assessing the impact that the original way of including the clean reference channel has, is to simply remove it and compare the two approaches. The comparison is relevant because it helps to understand to which extent the clean reference benefit the default system. Every input has its clean reference channel removed. The two dirty channels remain. Thus every input, $\mathbf{X}$ will have size $C-1 \times F \times T_f$, where $C = 3$, F is 129, and T_f is 15 since this choice of included context resulted in the overall best performance of the single hidden layer networks.

4.4.2 Absolute difference between the reference and dirty channels

An intuitive way that any system would use the clean reference, is to directly compare it with each of the dirty channels. A simple approach is to use the absolute difference between each of the corresponding entries of frequency and time. Thus two channels of absolute difference is included in the input. The difference channels are defined as

$$D_{i,j,k} = |X_{i,j,k} - X_{3,j,k}| \tag{4.1}$$

where $D_{i,j,k}$ is the absolute difference of the j'th frequency bin at time step k between the clean channel and i'th dirty channel.

By appending the two difference channels to the inputs, the resulting shape is $C+1 \times F \times T_f$, where only one extra channel is present since the clean reference channel is removed from each of the inputs.

4.4.3 Correlation between the reference and dirty channels

As an alternative to using the absolute difference between each of the entries, a correlation coefficient is also proposed. The correlation coefficient compares each of the rows in each of the dirty channels, with the corresponding row in the clean reference. Let $\mathbf{a}_{i,j}$ be the j'th

row from the i'th dirty channel from a given input $\mathbf{X}$, and let $\mathbf{b}_j$ be the j'th row from the clean reference channel in the same input $\mathbf{X}$. The correlation between $\mathbf{a}_{i,j}$ and $\mathbf{b}_j$ can be expressed as [8]

$$\rho_{i,j} = \frac{(\mathbf{a}_{i,j} - \boldsymbol{\mu}_{\mathbf{a}_{i,j}})^T(\mathbf{b}_j - \boldsymbol{\mu}_{\mathbf{b}_j})}{\sqrt{||\mathbf{a}_{i,j} - \boldsymbol{\mu}_{\mathbf{a}_{i,j}}||_2^2 \cdot ||\mathbf{b}_j - \boldsymbol{\mu}_{\mathbf{b}_j}||_2^2}} \tag{4.2}$$

where $\rho_{i,j}$ is the correlation between the j'th row of the i'th dirty channel and the corresponding row in the clean reference. The vector $\boldsymbol{\mu}_{\mathbf{a}_{i,j}}$ is the estimated mean across the entries of $\mathbf{a}_{i,j}$ using all inputs in the training set. Similarly, $\boldsymbol{\mu}_{\mathbf{b}_j}$ is estimated from all inputs in the training set. The correlation coefficient should be interpreted as a measure of how similar the dirty channels and the clean reference are across time, for each frequency. The correlation coefficients are appended to each input as additional dimensions before the preprocessing step.

4.4.4 Comparing the ways of processing the clean reference

Figure 4.5 shows the performance of the single hidden layer networks, with the reference included as described in the previous subsections. Interestingly, there is not much difference in performance between the methods of including the reference, which means that the inclusion of the reference does not matter much if at all. This result seems counter intuitive since one would expect the additional information about the reference to be useful. Further investigations, could try different approaches to including the reference, in efforts to find a way that leads to better performance.

An argument that supports the results is that when artefacts are present, they are typically present in only one of the hearing aid channels, but not the other. Hence, it is possible that the network uses a combination of the channels as it own reference and that the additional clean reference is only redundant information that contributes to the number of dimensions. It should be investigated, whether there is a basis for the argument or not.

The fact that the neural network models perform comparable to the MATLAB system *without* being given the reference is interesting, since it frees the model from being dependant on recording or knowing the reference. This potentially allows users to use the system on data that is recorded in environments where the reference is unknown. It could also allow for "online" detection of artefacts in hearing aids, giving a continuous state of the feedback cancellation system.

In the following sections, models are trained using the data, where the clean reference is removed from inputs.

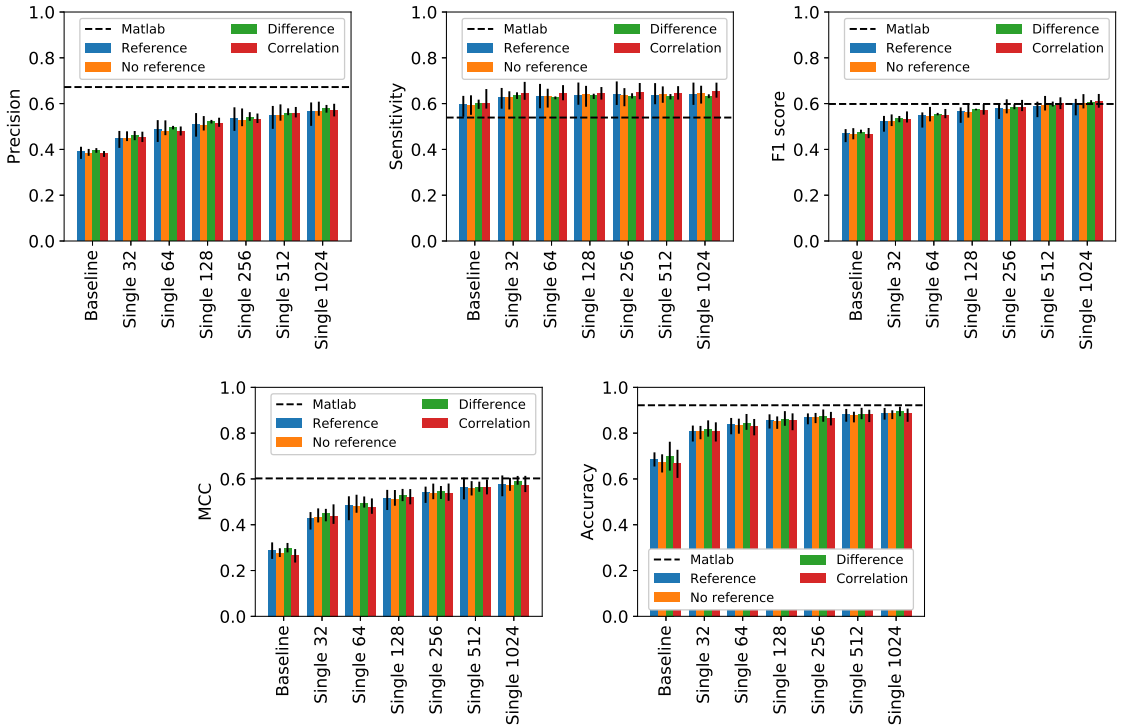


Figure 4.5: Plot of the metrics, for each of ways of including the reference. Each way of processing leads to similar performance. The ways of including the reference are from left to right: Reference, no reference, difference, correlation.

4.5 Multilayer feedforward neural networks

A natural extension of single hidden layer networks are models with multiple hidden layers. The investigation of models with a single hidden layer is one dimensional, since the number of neurons in the hidden layer is the only free variable. With multiple layers, there are more free parameters. It is, however, beyond the scope of this report to search all combinations of even two layer networks as extensively as in the single hidden layer case.

Instead of searching all combinations of layers, we selectively fix the first layer's size and try many choices of neurons for the second layer. Networks with more than two layers are delimited from, with a reason that is revealed by the results.

The first family of two layer neural networks that are investigated, are networks where the first layer has a fixed width of 1024. The number of neurons in the second layer is varied from 32 to 1024 neurons. This family of networks, builds upon the results in the single hidden layer case. Using 1024 nodes in the first layer gave the best performance. By building upon this architecture, we hope to see an increase in performance.

Figure 4.6 shows the key metrics, for all of the two layer networks, where the first layer has a fixed width of 1024 nodes. The additional layer, does not lead to a significant increase in performance.

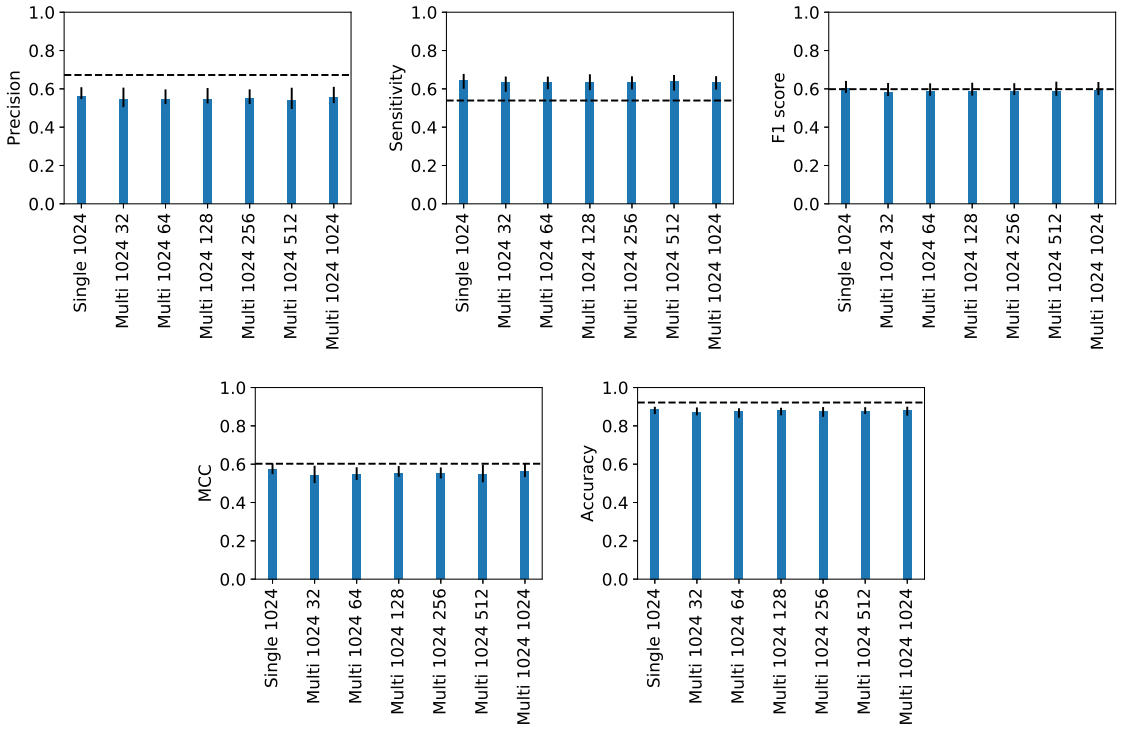


Figure 4.6: Metrics for the two layer neural networks, where the first hidden layer has 1024 neurons. The second layer has a varying number of neurons, from 32 to 1024.

Even though the more complex models do not provide better results than the single hidden layer models, it is still possible that the same performance can be achieved with fewer weights arranged in two layers. Figure 4.7 shows the performance of two layer networks, where the first layer has a fixed width of 32 neurons and the second layer varying from 32 to 1024 neurons. Again, there is no significant gain in performance by introducing a second layer.

4.6 Further balancing of classes

The class specific metrics might reveal where there is room for improvement. Figure 4.8 shows the average confusion matrix of the single hidden layer network with 1024 nodes. Notice that the class specific metrics for the howl are poor compared to the other classes. This suggests that training data imbalance still needs to be addressed.

When comparing the confusion matrix and the class specific metrics for the neural network

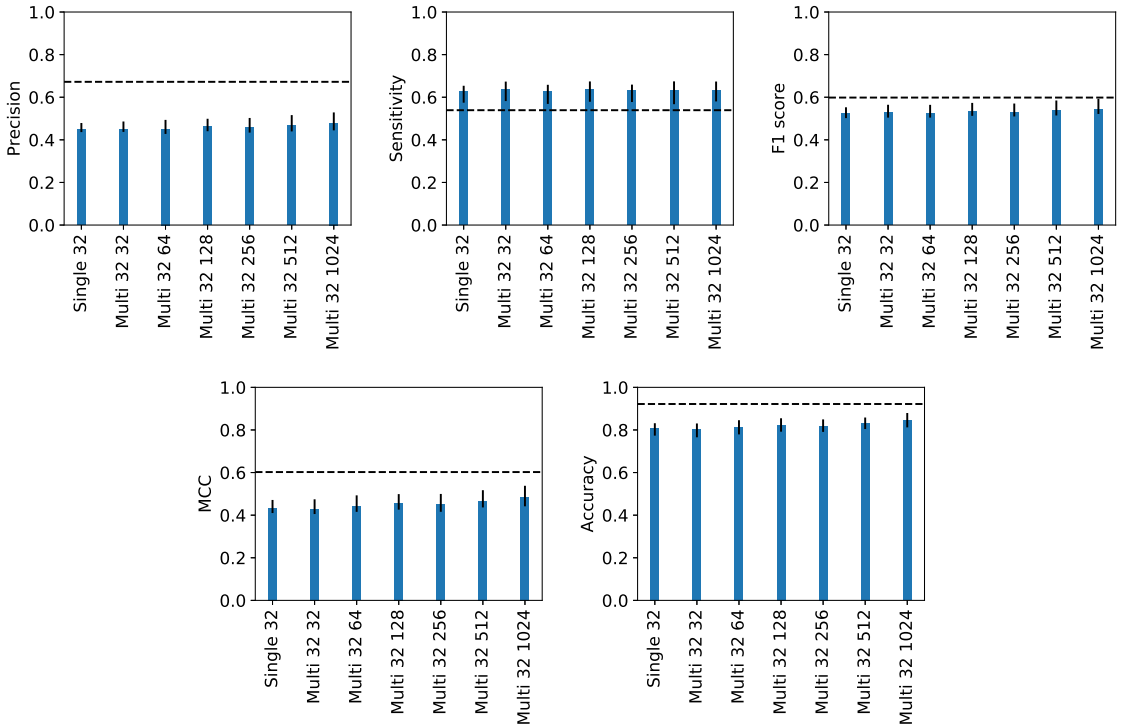


Figure 4.7: Metrics the two layer neural networks, where the first hidden layer has 32 neurons. The second layer has a varying number of neurons, from 32 to 1024.

to the MATLAB system's metrics, a striking similarity is, that both systems struggle with the howl class. An immediate thought is that it simply is hard to classify howl inputs, and that the performance is actually good, considering the underlying problem. This thought is definitively put to rest, when looking at the network's performance on the training data set. Figure 4.9 shows the confusion matrix and relevant metrics, for the single layer network with 1024 neuron, when evaluated on the training data set after 150 epochs. It is obvious that the model complexity allows for classification with a sufficient performance. This supports the results from the investigation of multi layer networks. The single layer networks are complex enough to classify the training data set.

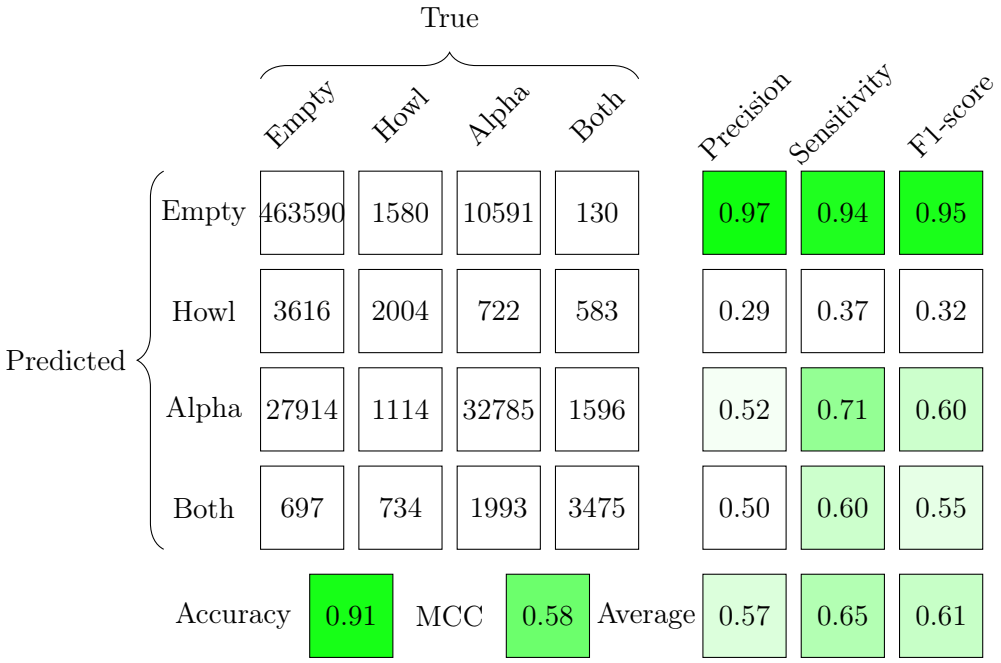


Figure 4.8: Confusion matrix and class specific metrics of the single hidden layer network with 1024 hidden neurons. The class specific metrics related to howl class is significantly poorer than the other classes' metrics.

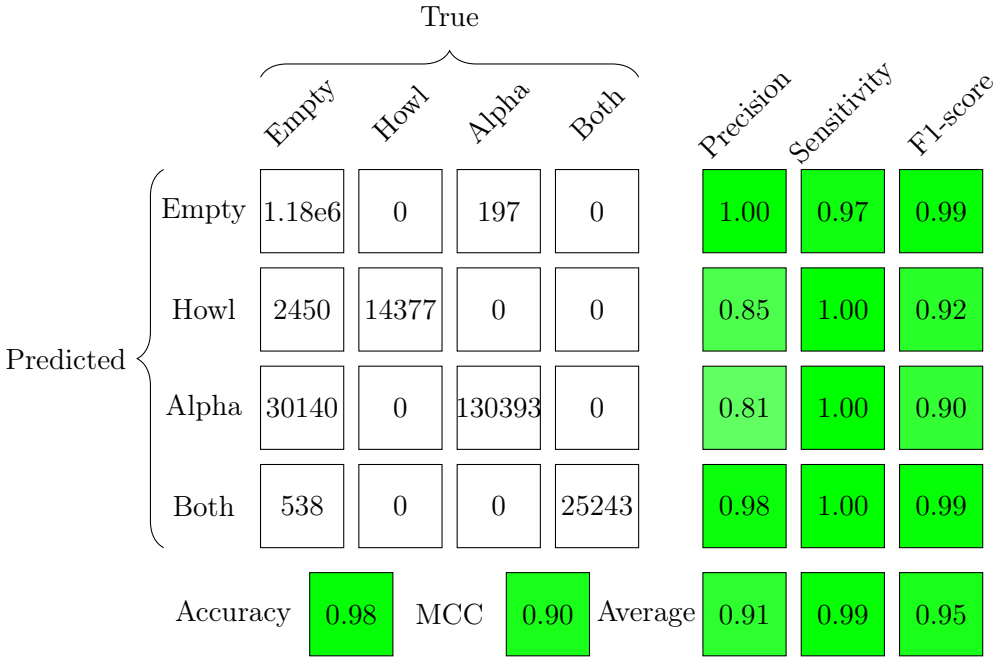


Figure 4.9: Confusion matrix of the single hidden layer network with 1024 hidden nodes, when evaluated on the training data set after 150 epochs. The metrics are near perfect.

Another approach to balancing is undersampling [22]. By removing some of the inputs originating from the empty class, the imbalance of the data set is reduced. Figure 2.2 shows the proportion of classes in the total data set. By removing 90% of the 'empty' inputs, the proportions of the 'empty' class and the 'alpha' class become comparable. Inputs are only removed from the training data set. The test and validation data sets remain unchanged.

Figure 4.10 shows the confusion matrix and relevant metrics, for the single hidden layer network with 1024 neurons, when 90% of inputs from the empty class are removed from the training data set. There is no significant difference with respect to the metrics comparing the model with and without undersampling applied to tthe training data set.

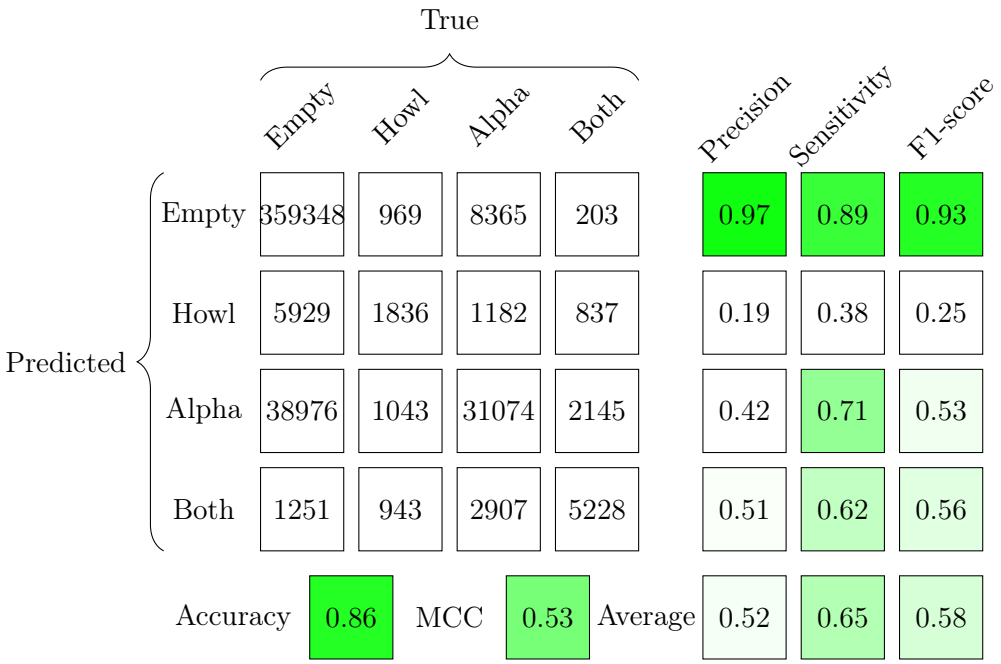


Figure 4.10: Confusion matrix of the single hidden layer network with 1024 hidden nodes evaluated on the test data set, after being trained using a training where 90% of empty samples are removed.

The class specific metrics reveal that improvement with respect to the howling class has the greatest potential. Increased model complexity and undersampling does not seem to be the answer. In future work investigations into the data set could cover augmentation of the howling examples, as is common practice in fields like image processing [23][22]. Augmentation would provide new training data, that potentially can improve the performance of networks with respect to the howling class.

4.7 Comparing the best network with the the MATLAB system

Many parameters have been investigated in this chapter. It is found, that including more context leads to better performance. It is found that including the clean reference does not affect the performance significantly. The use of multiple hidden layers in the feedforward neural network models, also does not increase the performance.

The metrics related to the network with the best performance is shown in Figure 4.8. In terms of the proposed metrics, the performance is comparable with the MATLAB system. The MATLAB system generally has a higher precision than the neural network, but also has a lower sensitivity than the neural network model in general. With respect to the MCC metric, which measures a systems performance overall, the MATLAB system is slightly better than the neural network model.

The way of comparing the models, in terms of confusion matrices, disregards the timely relation between concurrent inputs. It is relevant to investigate how the neural network models perform in terms of consistently predicting the same class over time, since the application of detecting artefacts is formulated originally, as annotating an artefact with a start time and duration.

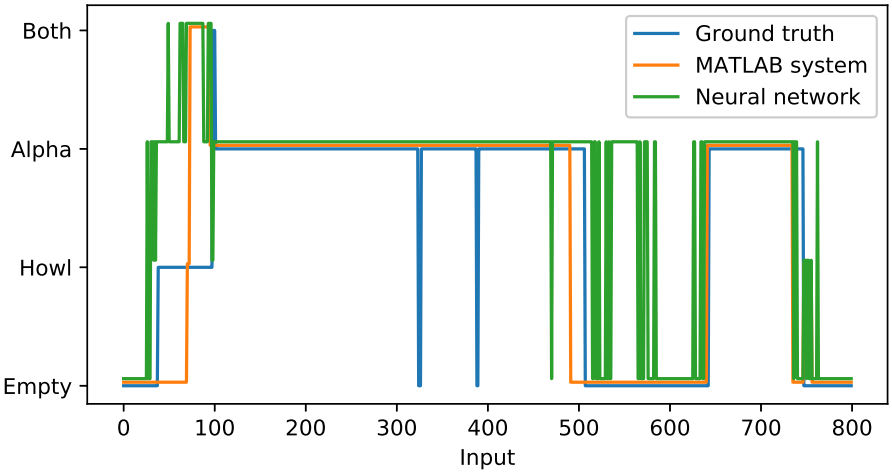


Figure 4.11: Plot of classified inputs over time. The best single hidden layer neural network with 1024 nodes is compared to both the MATLAB system and the ground truth.

Figure 4.11 shows a plot of the predicted classes over time for the single hidden layer neural network with 1024 hidden neurons, the MATLAB system and the ground truth. The section of inputs shown in the plot is selectively chosen from the test data set, to show some of the key characteristics when converting the classification task back to the annotation format. In

the signal section shown in Figure 4.11, both systems predict a class relatively consistent over time. The neural network output oscillates in some short periods of time. Especially at the start and the end of an artefact. Such oscillations do not make physical sense in the application at hand. Artefacts have some minimum length, for it to be audible or detectable.

The oscillations can be removed using various signal processing. An experimental example is a median filter, which removes some of the very short spikes. Further work could include an exhaustive investigation of post processing of the predicted classes of consecutive input in efforts to both increase the performance in terms of the metrics, but also the performance of the neural network models in terms of their use in the actual application.

To motivate the investigation, Figure 4.12 shows the same plot of predicted classes of inputs over time, but where the predictions of the neural network is processed by a median filter with size 9. Most of the oscillation has been removed, which gives an arguably better performance from a visual perspective.

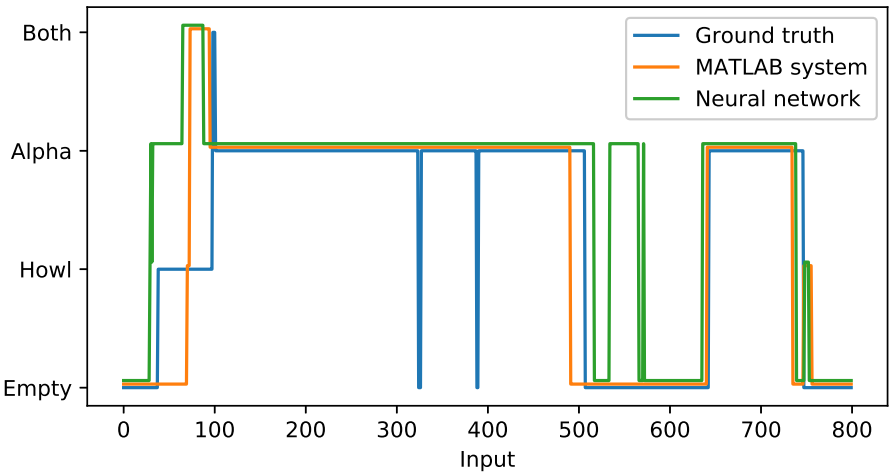


Figure 4.12: Plot of classified inputs over time. A median filter with size 9 is applied to the neural network series of predictions.

Conclusion 5

Acoustic feedback in hearing aids can cause annoyance to hearing impaired users. State of the art hearing aid systems use a feedback cancellation system that consists of an estimated acoustic feedback path that cancels the feedback. If the estimated path is not close enough to the true acoustic feedback path, howling occurs. State of the art hearing aids also include a spectro-temporal modulation system, which sometimes replaces the howling with a less intrusive, but still annoying artefact. Detection of these artefacts in recorded audio from hearing aids is essential for evaluation of the feedback cancellation system.

A suitable input and output is defined for a system that is capable of classifying, whether at a time instant there is either one of the artefacts, both of the artefacts, or no artefacts present.

Feed forward neural networks are proposed as suitable models for the classification task. A cost function that takes imbalanced data sets into account is presented, and a preprocessing step consisting of whitening and dimensionality reduction via PCA is proposed. Accuracy, precision, sensitivity, F_1 score and Matthews correlation coefficient are used as metrics for evaluation of the trained models.

Single hidden layer feedforward neural network models with a varying number of neurons in the hidden layer are trained and evaluated using K-fold cross validation. The models are compared, using the proposed metrics, and an artefact detection system developed by Oticon. The comparison shows that a single hidden layer network with 1024 hidden neurons has comparable performance to the existing detection system.

An investigation of the included context in each input showed that including more frames leads to better performance. The investigation was limited by the maximal number of included frames due to time constraints, thus better performance might be achieved if even more context is included in each input.

It is found that inclusion of a clean reference in each input does not lead to better performance. This potentially gives the network models more flexibility than the existing detection system, since the neural network can be used to evaluate hearing aid audio that is recorded in environments where the clean reference is unknown. The network can also be used "online" in hearing aids, giving a continuous state of the feedback cancellation system.

Class specific metrics indicate that improvement with respect to classification of howls has the greatest potential. Undersampling of the majority class in the training set did not lead to better performance.

Future work 6

Along with useful results, many interesting paths for future work are uncovered in the report.

Class balancing has received some attention, for example through the balanced cost function, and simple undersampling of the majority class in the training data set. The results suggest that improvements with respect to the howl class has the greatest potential. Future work could try addressing the imbalance of classes in the training data set for example through augmentation of howl samples.

The investigation of included context revealed that inclusion of more frames leads to better performance. The investigation was however limited by the choices of included frames. The maximum number of included frames, which was tested, was 15. Further work could try including even context, in efforts to further increase the performance of models.

In the last part of the numerical investigations the neural network models are compared with the existing detection system, with respect to classification over time. The results revealed that post processing of the neural networks predictions of concurrent inputs may improve the performance, both in terms of the presented metrics, but also the performance of the neural networks in the context of the application of annotating audio signals.

Bibliography

[1] Meng Guo. "Analysis, Design, and Evaluation of Acoustic Feedback Cancellation Systems for Hearing Aids: - A Novel Approach to Unbiased Feedback Cancellation". English. PhD thesis. 2013. ISBN: 978-87-7152-001-9.

[2] H. Nyquist. "Regeneration theory". In: *The Bell System Technical Journal* 11.1 (1932), pp. 126–147.

[3] Alexandra Winkler, Matthias Latzel, and Inga Holube. "Open Versus Closed Hearing-Aid Fittings: A Literature Review of Both Fitting Approaches". eng. In: *Trends in Hearing* 20.0 (2016-02-12). ISSN: 2331-2165.

[4] Pepe Gil-Cacho et al. "Regularized Adaptive Notch Filters for Acoustic Howling Suppression". English. In: *Proceedings of 17th European Signal Process. Conf. (EUSIPCO '09)*. Online proceedings. EURASIP, 2009.

[5] Toon van Waterschoot and Marc Moonen. "Comparative evaluation of howling detection criteria in notch-filter-based howling suppression". In: *Journal of the audio engineering society* 58.11 (2010), pp. 923–940.

[6] Meng Guo and Bernhard Kuenzle. "On the use of spectro-temporal modulation in assisting adaptive feedback cancellation for hearing aid applications". In: *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, Oct. 2017. DOI: 10.1109/acssc.2017.8335456. URL: https://doi.org/10.1109/acssc.2017.8335456.

[7] Meng Guo et al. "Extension and Evaluation of a Spectro-Temporal Modulation Method to Improve Acoustic Feedback Performance in Hearing Aids". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Apr. 2018. DOI: 10.1109/icassp.2018.8461534. URL: https://doi.org/10.1109/icassp.2018.8461534.

[8] Steven Kay. *Intuitive Probability and Random Processes Using MATLAB*. Berlin, Heidelberg: Springer-Verlag, 2007. ISBN: 0387241574.

[9] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0131988425.

[10] J. B. Allen and L. R. Rabiner. "A unified approach to short-time Fourier analysis and synthesis". In: *Proceedings of the IEEE* 65.11 (1977), pp. 1558–1564.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[12] Ossama Abdel-Hamid et al. "Convolutional neural networks for speech recognition". In: *IEEE/ACM Transactions on audio, speech, and language processing* 22.10 (2014), pp. 1533–1545.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[14] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. "Image processing with neural networks—a review". In: *Pattern recognition* 35.10 (2002), pp. 2279–2301.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[16] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[17] M. Kampffmeyer, A. Salberg, and R. Jenssen. "Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2016, pp. 680–688.

[18] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[19] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information Processing and Management* 45.4 (2009), pp. 427–437. ISSN: 0306-4573. DOI: `https://doi.org/10.1016/j.ipm.2009.03.002`. URL: `http://www.sciencedirect.com/science/article/pii/S0306457309000259`.

[20] J. Gorodkin. "Comparing two K-category assignments by a K-category correlation coefficient". In: *Computational Biology and Chemistry* 28.5-6 (Dec. 2004), pp. 367–374. DOI: `10.1016/j.compbiolchem.2004.09.006`. URL: `https://doi.org/10.1016/j.compbiolchem.2004.09.006`.

[21] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.1 (Jan. 2020). DOI: `10.1186/s12864-019-6413-7`. URL: `https://doi.org/10.1186/s12864-019-6413-7`.

[22] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural Networks* 106 (Oct. 2018), pp. 249–259. DOI: 10.1016/j.neunet.2018.07.011. URL: https://doi.org/10.1016/j.neunet.2018.07.011.

[23] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (July 2019). DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.