Speaker verification: Advantages and limitations of a biologically inspired feature extractor

MAJA GAJIC

MASTER´S THESIS

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Speaker verification: Advantages and limitations of a biologically inspired feature extractor

Maja Gajic
maja.g98@gmail.com

Department of Electrical and Information Technology
Lund University

——

IntuiCell

Supervisor: Fredrik Edman

Examiner: Erik Larsson

June 28, 2022

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Speaker verification is the process of verifying the identity of a person based on voice. This process usually encompasses the following steps: The speech signal is mapped into features using a feature extractor, these features are then classified using a post processor. The most common features used in speaker verification today are STFT, MFBs, and MFCCs, that are different spectral representations of the speech signal. Recently, a biologically inspired feature extractor called the cuneate nucleus (CN) model, that outputs CN features, was created. The main goal of this Master thesis is to find an optimal ANN post processor for the CN features. Testing different models on both conventional features and CN features concluded that a CNN model and a LSTM model were most suitable. The performance result concluded that the CN features and STFT performed well on noisy data but worse on clean data compared to the MFCCs and MFBs. A statistical analysis of the features was conducted using cross correlation, average activity and entropy. The analysis concluded that the inherent dynamical properties of the CN features and STFT make the training process of an ANN difficult, and therefore performance on clean data is poor. On the other hand these dynamical properties is what allows the features to perform well on noise. In comparison, the MFCCs and MFBs have the opposite inherent properties and this allows them to have state-of-the-art performance on clean data but poor performance on noise data. This in turn means that a conventional ANN post processor can only provide limited performance for CN features, and that other post processor methods need to be developed to reach beyond that limit.

Acknowledgements

I want to thank everyone at IntuiCell for helping me during my thesis and for giving me the opportunity to do this project. I want to give a special thanks to my supervisors Udaya and Edvin for supporting me throughout this project and for always being willing to help out. I also want to thank my main supervisor Fredrik for giving helpful insights in my report writing.

I want to thank my family and friends for their encouragement and support during my time at university.

Popular Science Summary

Speaker verification is the process of identifying the person speaking. It is a widely studied field and usually deep neural networks are used. This work aims at doing speaker verification inspired by biology.

We humans can easily distinguish between different voices, and we can often identify a person speaking with only hearing their voice. Further our ability to identify a person does not notably diminish in noise conditions. This is an amazing ability to have and it is often taken for granted.

Speaker verification is the process of identifying a person by the use of their voice, today this is a hard task to accomplish especially in noise conditions. An example of speaker verification is when we use voice command on our mobile devices. Assume that two people that are sitting next to each other, both use hands free voice commands option on their phones. If someone activates their voice command by for example saying "Hey, Google!" only that persons phone should answer. Otherwise if someone uses the activation phrase in a crowded area you could potentially have multiple mobile devices answering you, which would be inconvenient. To deal with this, phone companies usually require you to say the activation phrase a couple of times before you can start using voice command. When you are repeating these phrases you are actually actively training a deep neural network to be specialised at identifying your voice. This is why the scenario above does not occur in real life.

Deep neural networks are most often used in the field of speaker verification today. And these networks function in the following way: They process frequency information of the raw speech signal, and based on that try to determine who is speaking. But, theses methods are not true to how our ear and brain picks up and process voices. In this work instead of using frequency information for the networks to process, the information output from a biologically inspired model was used. This biologically inspired model picks up frequency patterns for the deep neural network to process.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

To be able to distinguish between different speakers is usually a simple task for humans, but for computers this task is difficult. Speaker recognition is the process of identifying a person by his or hers voice. Today, this is a wide area of research and an example of use is voice command on a mobile device. Most smartphones have the option to activate their voice assistant hands-free by for example saying "Hey Google". The problem with this is that you do not want your voice assistant reacting when someone else is trying to activate their voice assistant. So, in other words, the assistant needs to be able to tell if it is the owner of the phone speaking or somebody else.

Recently, the company IntuiCell has made developments in feature extraction technology used in speaker recognition. The new feature extractor is based on a biologically inspired model, and has not before been observed in the field. For this reason, this master thesis work aims to test and evaluate different post processors given the new feature extractor. The post processors that will be tested and evaluated are be based on artificial neural networks (ANNs). To get a better overview of how the post processor can be designed, a literature study was performed of the following neural network architectures: long short-term memory (LSTM) networks, recurrent neural networks (RNNs), convolutional neural networks (CNNs) and deep neural networks (DNNs). The aim was to gain an understanding of how these network architectures work and why, and in what way different post processors perform better than others. The aim of the study is also to evaluate if the post processors are suitable to use for the speaker recognition problem.

When the literature study was completed, the focus of the thesis was firstly to implement the suitable ANN post processors, and secondly the post processors were trained on both conventional features, that are introduced in chapter 3.2, and on the features developed by IntuiCell described in chapter 3.1. The post processor models and features were during and after training evaluated in different ways.

Chapter 2
Background

The field of speaker recognition can be divided into two categories; speaker verification and speaker identification. This master thesis work is focused on speaker verification which is the process to confirm a claimed identity by using voice biometrics. In other words speaker verification is a classification problem. The common way of solving this classification problem is to first do a feature extraction of the raw speech signal, then these features are used as an input to a post processor. The post processor creates a speaker embedding, which is a vector, containing high level speaker biometrics. The embedding can be compared to a "voice fingerprint" for a specific person. The final classification is based on these embeddings.

There are two main ways of creating speaker embeddings. The first way, which is the traditional way of doing speaker verification, is to create an embedding called the i-vector. This i-vector is usually extracted from a Gaussian mixture model (GMM). After the GMM/i-vector model is created a template i-vector is created for the target person. This template i-vector is saved and later used to be able to confirm a persons identity. When the GMM/i-vector model is later used for verification tasks, an i-vector is created for every new utterance, an utterance is a short segment of speech which can be divide up into equally spaced frames, these frames usually represent one time step (Bai, & Zhang 2021). The template i-vector is then compared to the i-vector that was just created, often by measuring how similar the two i-vectors are. Usually, the cosine similarity is used to measure this similarity, see equation (3.5). This traditional GMM/i-vector is based on statistical methods and does not use an ANN. The traditional method of using an i-vector extractor is not usually used today in speaker verification since other methods using ANNs have surpassed the traditional method's performance. About two decades ago ANNs were first used in the field of speaker recognition. The ANNs have revolutionised the speaker recognition field, yielding better results than any traditional method. The embedding technique is also used when performing speaker verification with an ANN post processor and is the other standard way to create embeddings. The embedding is usually extracted from the last hidden layer of the ANN model and is called the d-vector or x-vector, depending on the architecture type of the ANN. The cosine similarity is also usually used to measure how similar two embeddings are, a more thorough explanation of the d-vector and x-vector can be found in chapter 3.3.

Chapter 3

Theory

The theory is divided into two main parts, first a explanation of the CN features and conventional features is given, and second there is a description of the different ANN architectures.

3.1 CN features

The CN features are two dimensional, where one dimension is in space and the other in time and are constructed using a unique ANN architecture. This architecture is unique since it follows a biologically inspired model based on (Andersson 2021). This model attempts to mimic the process of neurons encoding signals from tactile inputs. The CN features are created according to the following steps:

1. Take the short time Fourier transform (STFT) of the raw speech signal.
2. This time frequency signal is then used as input to a biologically inspired feature extractor based on an ANN architecture. The input is fed into the feature extractor one time step at a time. This point is explained in more detail below.
3. The output from the feature extractor is the CN feature that will be used as input for the ANN post processor.

The feature extractor consists of a single layer of nodes. There are two types of nodes in this layer, first there are excitatory nodes and second there are inhibitory nodes. The excitatory nodes amplify key frequencies in a positive way and the inhibitory nodes amplify key frequencies in a negative way. All excitatory nodes produce an output which is the feature itself. Each inhibitory node is connected to every other inhibitory node, but not to itself, and each inhibitory node is also connected to every other excitatory node. In other words the inhibitory nodes do not produce an output themselves. Therefore, the feature size for each time step will be the number of excitatory nodes. The input to the feature extractor is the STFT, all nodes are connected to all frequencies of the STFT by synapses that have weights. The nodes then learn what the key frequencies are by an unsupervised learning model. Certain frequencies are deemed important by a node if they correlate, the total output of the node then gets stronger than the separate signals together. Each node tends to be connected to different correlated frequencies. For one node most synapses connected to the frequencies will die, meaning that the weight for those synapses will be close to zero. This means that only a few of the frequencies correlate per node (Andersson 2021).

The weights in the CN model get updated by the following rule

$$w_{t+1,i} = w_{t,i} + r \cdot \int_0^{t_{max}} (y_t - K_{LPT}\overline{y_t}) \cdot \max\{f_{t,i}w_{t,i} - K_{LAT}\overline{f_{t,i}w_{t,i}}, 0\}dt \quad (3.1)$$

Where $w_{t,i}$ is the weight of synapse i at time t, and r, K_{LPT} and K_{LAT} are constants, y_t is the output of the node that synapse i is connected to, and $f_{t,i}$ is the frequency being fed into synapse i at time t. The weight update in equation (3.1) measures the correlation between the output of the node and the input of the synapse i. In other words, the CN model does not learn according to traditional supervised learning. In Figure 3.1, an example of the CN feature can be seen, the CN model has 100 excitatory and 33 inhibitory neurons. In the Figure the output from the excitatory neurons are visible, in a three seconds window. Here the irregular shape between time steps 100 and 150 is a vowel.

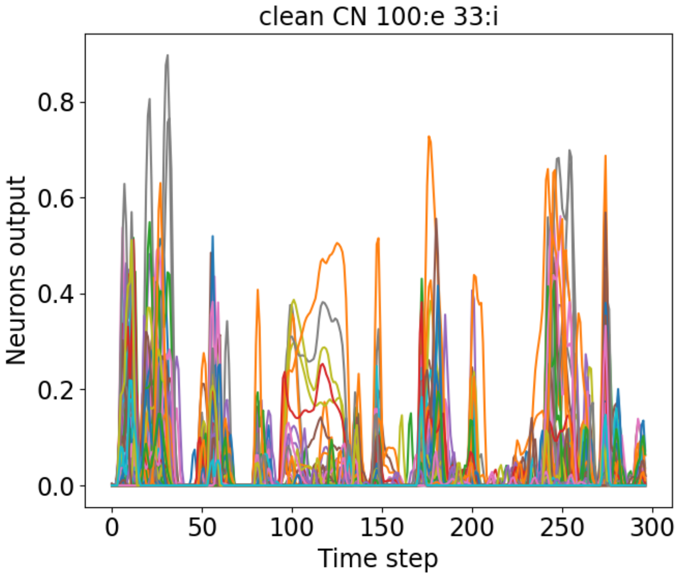

Figure 3.1: An example of the CN feature, here each line is a neuron output, where the x-axis is time, and the y-axis is neuron activity.

3.2 Conventional features

The features ordinarily used as inputs to an ANN post processor are, raw speech signals, spectrogram, Mel-filter bank energy features (MFBs), Mel-frequency cepstral coefficients (MFCCs). Where MFBs and MFCCs are the most commonly used (Bai, & Zhang 2021). An overview of how the features are related to each other is shown in Figure 3.2. Below each feature type is described more in depth.

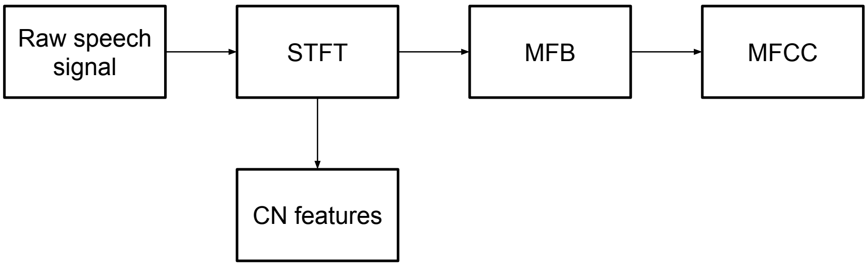

Figure 3.2: A graph showing the transforms applied to create the certain features.

3.2.1 Raw speech signal

The one dimensional raw speech signal is sometimes used as input to a post processor. Often the input features for ANN models are hand-crafted meaning that some sort of transform was applied to the raw speech signal. But by using the raw speech signals as input to the ANN model directly, features are allowed to develop by themselves within the model. The advantage with using the raw speech signal as input to the ANN model is that hand-crafted features can very easily miss something.

When used as an input to a CNN the analysis done after training shows that the CNN preforms filtering similar to a Fourier transform (Muckenhirn, Magimai-Doss, & Marcel 2018). Because of this the raw speech signal was not considered as an input feature for the chosen ANN models.

3.2.2 STFT

Since the STFT is the base for the other features, see Figure 3.2, it was deemed interesting to see how well a Fourier transform would hold to the other features. Therefore the absolute value of the STFT was used as input to the ANN models. The STFT was also used as input to have a base to compare with, and to see how the other features behave in comparison.

3.2.3 MFBs

The Mel-frequency scale is a frequency unit that is more adapted to how humans perceive sound and is therefore often used in speaker recognition in the form of Mel-filer bank energy features. For example, the human ear is more perceptive of changes in the lower frequencies than in the higher frequencies, and this is reflected in the MFBs since the lower frequency bins are narrower than the higher frequency bins (Andersson 2021). Here, bining is the process of data reduction, depending on the data resolution one wants, a number of bins is selected and distributed over the range of data. Usually, 20-40 bins or filter banks are used to create the MFBs. These bins have a shorter span for lower frequencies and higher span for higher frequencies. Except for the increasing bin width the MFB is quite similar to the STFT signal. The MFBs were used as inputs to the ANN models.

To compute the MFBs the power spectrum needs to be multiplied with the Mel-filter banks for every utterance. In other words, as one also can see in Figure 3.2, the MFBs is a transformation of the STFT. The power spectrum is calculated in the following way. A pre-emphasis filter is applied to the raw wave form, for instance this helps with the numerical stability of the Fourier transform. The pre-emphasis filter is on the form $y(t) = x(t) - \alpha x(t-1)$ where x is the raw wave form for at time point t and α is a constant in the range $(0, 1)$. After the pre-emphasis filter is applied the utterance are sliced into overlapping frames. For each frame a window function and then the STFT is applied. Finally, the power spectrum is computed for each frame by taking the absolute value of the STFT and squaring it then dividing by the number of points in the frame. The power spectrum is usually stored in a (number of frames $\times \frac{n_f}{2} + 1$) matrix, where n_f is the number of STFT points. After this is done the Mel-filter banks are computed in the following way. A lower and upper bound of the frequency in Hertz (Hz) is chosen, often this is based on what frequencies the human ear can perceive. These linear frequencies f are then transformed in to Mel-frequencies m which uses log scale, $m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$. Then if n_m is the number of Mel-filter banks that are wanted, then n_m equally spaced points are calculated in between the lower and upper Mel-frequency bound. Now, there are $n_m + 2$ equally spaced points in the Mel-frequency scale. These points are then transformed back into Hz, meaning that the points will not be equally spaced any more. These frequencies are rounded to the nearest STFT bin by taking $f_b = \left\lfloor \frac{(n_f+1)\cdot f}{s} \right\rfloor$, where n_f is the number of STFT points, f are the frequency points in Hz, and s is the sample rate. The Mel-filter banks are then calculated as

$$H(k) = \begin{cases} 0, & k < f_b(l-1) \\ \frac{k - f_b(l)}{f_b(l) - f_b(l-1)}, & f_b(l-1) \leq k < f_b(l) \\ 1, & k = f_b(l) \\ \frac{f_b(l+1) - k}{f_b(l+1) - f_b(l)}, & f_b(l) < k \leq f_b(l+1) \\ 0, & k > f_b(l+1). \end{cases} \tag{3.2}$$

Where k are frequencies between the upper and lower bound in the same scale as f_b, and $l = [1, \ldots, n_m]$. For every l, $H(k)$ will yield a triangular shaped filter bank. From equation (3.2) it can be seen that for every l the triangle will have its peek at $k = f_b(l)$ and its endpoints at $k = f_b(l \pm 1)$, since the distance between every adjacent point in f_b gets wider the triangular filter banks will get wider as well, see Figure 3.3. Normally, these Mel-filter banks are stored in matrices of size $(n_m \times \frac{n_f}{2} + 1)$, so that each triangle gets stored in one row. The MFBs are then calculated as the product between the power spectrum matrix and the Mel-filter bank matrix transposed, yielding a (number of frames $\times n_m$) matrix. The resulting output is log transformed before it is used.

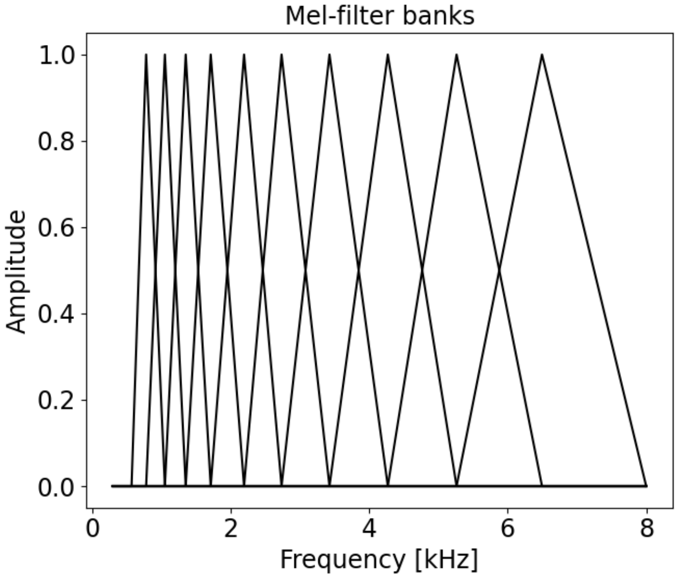

Figure 3.3: Ten Mel-filter banks with sampled frequency from 300 Hz to 8000 Hz.

3.2.4 MFCC

The MFCC features is a transformation of the MFBs as one can see in Figure 3.2, and they are obtained when the discrete cosine transform (DCT) is applied to the MFBs. This is done since the MFBs are correlated since the Mel-filter banks overlap, see Figure 3.3, and the DCT helps decorrelate the signal. The DCT is defined as

$$y_k = 2f \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k(2n+1)}{2N}\right), \text{ where} \tag{3.3}$$

$$f = \begin{cases} \sqrt{\frac{1}{4N}}, & k = 0 \\ \sqrt{\frac{1}{2N}}, & \text{otherwise,} \end{cases} \tag{3.4}$$

and $k = 0 \ldots N-1$. Sometimes the so called Δ and $\Delta\Delta$, which are the first and second order derivative of the MFCC, can be incorporated in the MFCC feature. Although, since Δ and $\Delta\Delta$ are linear transformation of the MFCC an ANN should be able to reproduce this, but still many ANN implementation utilise the MFCCs and their derivatives as inputs (Andersson 2021). The MFCCs were used as input to the ANN models without the Δ and $\Delta\Delta$ coefficients.

3.3 ANN architectures

There are many types of ANNs and not all may be suited to have the CN features as input. For this reason different types of ANN architectures were studied and from this study it was determined if the architectures are suitable to use when the CN features are used as input. The chosen ANN models are going to be tasked with learning to distinguish between different speakers using standard classification. The output layer from the network will contain the same number of nodes as the number of speakers in the training data set. Each node in the output layer will then represent one speaker, therefore the person connected to the node that outputs the highest value will be classified as the person speaking. During training the weights of the network will be optimised based on this output result. However, as mentioned in chapter 2 when the classification is done during testing, embeddings called d-vector or x-vector are used (Bai, & Zhang 2021). The d-vector is a frame level embedding and is created from the output from the last hidden layer of the ANN. After training a template d-vector of the target person is saved and then during testing a d-vector is created for every frame in an utterance, these d-vectors are averaged to create a single embedding. How close the template d-vector and test utterance d-vector are, is what decides how the person in question gets identified. Usually, the cosine similarity, which is cosine of the angle between the two d-vectors, is used (Variani, Lei, McDermott, Lopez Moreno, & Gonzalez-Dominguez 2014). The cosine similarity is calculated using the scalar product:

$$\cos(\theta) = \frac{a \cdot b}{||a||_2 \cdot ||b||_2}. \tag{3.5}$$

Where θ is the angle in between the two embeddings a and b. The x-vector is an advancement of the original d-vector, the difference is that the d-vector is a frame level embedding and the x-vector is an utterance level embedding. To create a x-vector first an embedding is created for each frame in an utterance using a one dimensional CNN (see chapter 3.3.3), then a statistical pooling layer is used to create a utterance level feature. This utterance feature is then usually fed through fully connected feed forward network, the x-vector is extracted from one of these layers. The ANN is trained as a whole to solve the classification problem. During testing a test utterance x-vector is compared to template x-vector in the same way as described for the d-vector (Snyder, Garcia-Romero, Povey, & Khudanpur 2017).

Speaker verification methods using ANNs can be categorised in two ways, stage-wise and end-to-end. The stage-wise method can be divided into a front-end and back-end stage. In the front-end the embeddings are extracted while in the back end a similarity score is calculated and compared with a threshold. Usually, if the score is higher than the threshold the two signals belong to the same person and otherwise not. The loss function used during training is usually the categorical cross entropy loss function. The end-to-end method compares a pair of input voices during training and produces a similarity score between these directly determining if they belong to the same person or not. The major difference between these two methods is the choice of loss function. Although, in literature there is sometimes a confusion between these two, since the end-to-end system is often called an embedding extractor. The reason for this name is that during testing an embedding has to be extracted and compared with a template embedding, these two embeddings are used as input to an independent back-end. If during training similarity scores are produced then the ANN model is regarded as an end-to-end method. With end-to-end methods there are three main challenges, deciding on a loss function, similarity metric, and constructing training pairs (Bai, & Zhang 2021).

Most of the post processors in speaker verification that uses a ANN structure have more than one layer of nodes. This, by definition, means that all the post processors are deep neural networks (DNNs).

3.3.1 RNNs

The traditional RNN network is similar to a feed forward network except that it also has backward connections to nodes in the same or previous layers. These backward connections have the same properties as forward connections, in other words they have a weight connected to them. The decision of where to put the backward connection is optional. RNNs are often used to solve problems that have time dependencies (Goodfellow, Bengio, & Courville, 2016). An example of a simple perceptron with a backward connection can be seen in Figure 3.4. In this figure we also see the network being unfolded in time two time steps. Here x is the input to the node and h is the output from the node. There are two weights, one forward weight W and one backward weight U. The same weights U and W are used for every time step.

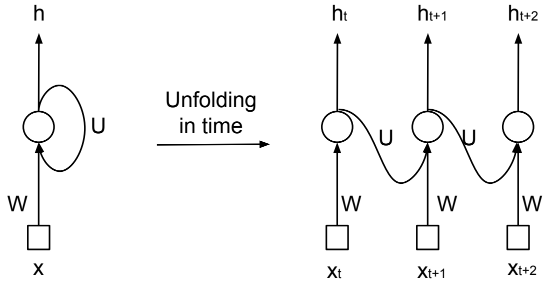

Figure 3.4: (Left) A perceptron which has a feedback connection. (Right) The same architecture as in (Left) but unfolded two time steps.

The output h_t is calculated as

$$h_t = \phi(x_t W + h_{t-1} U) \tag{3.6}$$

where ϕ is the activation function of the node. The node of an RNN can be seen in Figure 3.5.

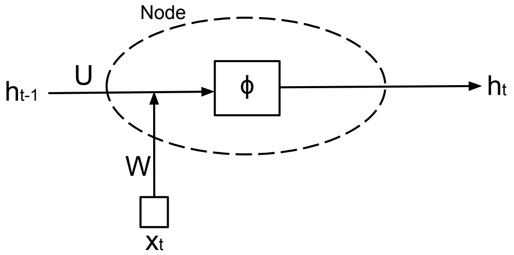

Figure 3.5: Inside a node of a recurrent neural network, where ϕ is the activation function of the node.

One of the main problems with the traditional RNN is that it is bad at capturing long term dependencies between samples. The reason for this is because the gradients need to be propagated through many time steps which often causes them to vanish, this is known as the vanishing gradient problem. To handle this problem a different kind of node was designed. The network that implements this kind of nodes are called long short-term memory (LSTM) network (Goodfellow, Bengio, & Courville, 2016). Since long term time dependencies could be needed to model the different features a LSTM network might be needed, and for this reason we look at them separately. Although, since the traditional RNN also models time dependencies it seems reasonable to compare the RNNs performance to the LSTM. Therefore the RNN architecture was chosen as one of the ANN models.

3.3.2 LSTMs

The LSTM network was designed to better handle long term dependencies between samples. A LSTM node is different then an ordinary node since it has a memory of previous inputs. The LSTM node has two outputs, the node value h_t, and an internal memory value c_t, where t denotes the time dependency (Ye, & Yang 2021). These values are recurrently fed to the LSTM node for each new time step input, see Figure 3.6. In the figure one can see a perceptron with a LSTM node and two values recursively being fed back into the node, and the network being unfolded two time steps. The difference between this network and the RNN in Figure 3.4 is that two values are being sent back into the LSTM node. To avoid confusion the weights have not been marked in Figure 3.6 since the LSTM network has more weights then the RNN.

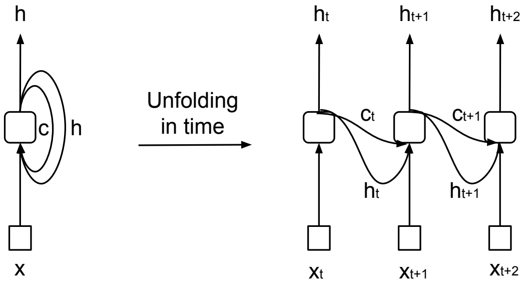

Figure 3.6: (Left) A perceptron with a LSTM node with two feedback connections. (Right) The same architecture as in (Left) but unfolded two time steps.

Inside the LSTM node there are three gates, the input gate i_t, forget gate f_t, and output gate o_t that help calculate the node value h_t, and the internal memory value c_t. The the node value and the internal memory are calculated as follows

$$c_t = c_{t-1}f_t + \tilde{c}_t i_t, \tag{3.7}$$

$$\tilde{c}_t = \tanh(x_t W^c + h_{t-1}U^c), \text{ and} \tag{3.8}$$

$$h_t = \tanh(c_t)o_t, \tag{3.9}$$

where

$$i_t = \sigma(x_t W^i + h_{t-1}U^i), \tag{3.10}$$

$$f_t = \sigma(x_t W^f + h_{t-1}U^f) \text{ and} \tag{3.11}$$

$$o_t = \sigma(x_t W^o + h_{t-1}U^o), \tag{3.12}$$

and $\sigma(a) = \frac{1}{1+e^{-x}}$ is called the sigmoid function, and $W^{c,i,f,o}$ and $U^{c,i,f,o}$ are weights (Ye, & Yang 2021). One setback of using a LSTM network is that it has four times more trainable weights per node then a traditional RNN. The LSTM node is depicted in Figure 3.7. To avoid confusion the weights specified in the above equations have not been marked in Figure 3.7, although one can easily see where the weights should be placed. In Figure 3.7, one can see the two values c_{t-1} and h_{t-1} being fed into the node, and the two values c_t and h_t being outputted. The Figure depicts a schematic interpretation of equations (3.7)-(3.12). The rectangular boxes in the figure are activation functions, in other words there are four small networks inside of the LSTM node. The circular shapes in the figure are point-wise operations, where × means multiplication, + means addition, and tanh is the hyperbolic tangent function, $\tanh(a) = \frac{e^{2x}-1}{e^{2x}+1}$.

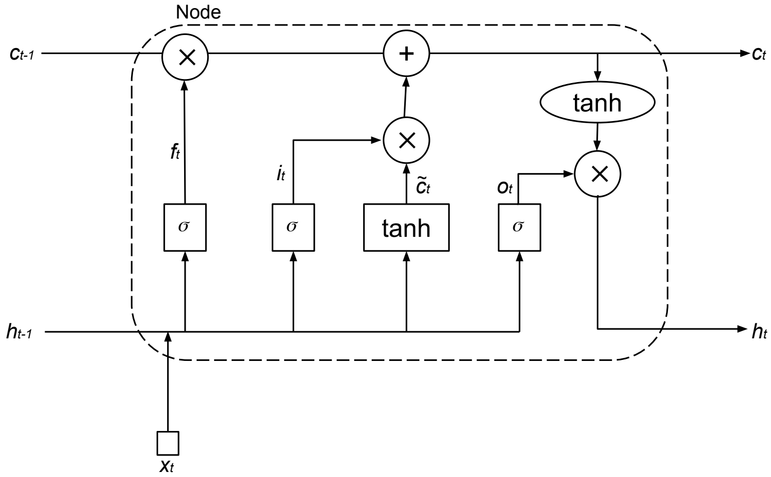

Figure 3.7: The inside of a LSTM node where σ and tanh are activation functions of the node.

Since a LSTM network is good at modelling time dependent data it was interesting to consider a LSTM network. It would also be of interest to compare the results with a traditional RNN to see if the traditional RNN is good enough.

3.3.3 CNNs

The CNN provides translation invariance, meaning that for two dimensional features in space and in time, it can locate speaker specific features without disrupting the timeline (Ye, & Yang 2021). For this reason the CNN architecture seems suitable to use. The CNN is weight efficient compared to other networks since one kernel in one layer uses the same weights for the entire input. A kernel in this case is a matrix, containing trainable weights, that moves over the input multiplying the weights with the specific sub region of the input. For each step the kernel makes there is one output. Another aspect of the CNN is that it has sparse weight connections, since only a small subset of the input is used to create the output in the next layer. A convolutional layer in a CNN consists of a convolution with the the input of the layer and the kernel, then an activation function is applied to the resulting convolution, and finally a pooling layer is applied (Goodfellow, Bengio,

& Courville, 2016). The kernel in the convolutional layer does not necessarily have to consist of one single block, it can be dilated by a fixed amount of steps.

The time delay neural network (TDNN) is a one dimensional CNN. The differences between a one and two dimensional CNN is the size of the kernel. For a two dimensional CNN the kernel is often two dimensional but the size of the kernel is smaller then the input size. So, for example, if a input has size $m \times n$ and the kernel is $i \times j$ then $i < m$ and $j < n$ meaning that the kernel has to move in two directions to cover the whole input. A one dimensional CNN has a kernel that only moves in one direction, meaning that if the input is of size $m \times n$ and the kernel is of size $i \times j$ then either $i = m$ or $j = n$ (Goodfellow, Bengio, & Courville, 2016). If the one dimensional CNN would be used in speaker verification then the kernel would move along the time axis. Results from the literature study showed that the CNN is the most used post processor in the field of speaker verification, both one and two dimensional CNNs are commonly used. The one dimensional CNN is the basis of the architecture that is used to create the x-vector, for this reason the one dimensional CNN architecture is suitable to use in speaker verification.

3.3.4 Layers

An ANN consists of layers often most of these layers implement different types of nodes with trainable weights connected to them, but this is not necessarily the case. Some layers have nodes that use fixed weights to try to reduce the size of the output from the previous layer or to confine the output values from the previous layers. Two type of layer that uses fixed weights are batch normalisation (BN) layer and statistical pooling layer. The BN layer is used to normalise the outputs from a layer. The normalisation is done over an entire utterance. In other words the mean and the standard deviation is calculated for an entire utterance and then the mean is subtracted from the entire batch and the the difference is divided by the standard deviation. This effectively confines the values that the layer outputs within a range, often contributing to the networks learning faster. The statistical pooling layer calculates the mean an the variance of an utterance and concatenates them, effectively reducing an utterance to a single vector representation. This is usually done when x-vectors are created.

3.3.5 Supervised learning model

During the training of an ANN post processor a supervised learning model is used. The weighs are updated using the Adaptive moment estimation (Adam) which is a method for minimising the loss function of a neural network. The method is based on gradient decent but also has momentum terms. The loss function to be minimised is the categorical cross entropy function

$$E(w) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{c} d_{n,i} \ln y_{n,i}. \tag{3.13}$$

Where c is the number of classes, N is the number of samples in a batch, $d_{n,i}$ is the true output, and $y_{n,i}$ is the network output.

3.3.6 Performance measure

In speaker verification the standard way of evaluating network performance during the testing stage is using the equal error rate (EER). The EER is a measure of when the number of false positives and false negatives is equal. The EER is defined as

$$\mathrm{EER} = \frac{\mathrm{FPR} + \mathrm{FNR}}{2}, \text{ if } \mathrm{FPR} = \mathrm{FNR}. \tag{3.14}$$

Where FPR is the false positive rate and FNR is the false negative rate. The EER is measured in percent.

3.3.7 Adaptation of the CN features

Two different CN models were used in this thesis, at first the old CN model or CN model V_1 was used, later this model was modified to CN model V_2. Based on the results of the ANN performance with the conventional features and the CN features from CN model V_1 used as input, and the feature analysis study (see sections 5.1 and 5.2.1), the CN features were modified in ways to try and improve the results for these features. There were three following types of changes made to the CN features:

1. The first modification was made to the CN model itself. In Figure 3.8 one can see part the weights of CN model V_1 as a heat map. Each row represents the weights that go from the input frequencies to an excitatory neuron. Here the colouring means: white is a high value, red is a medium value, blue is a low value, and black is zero. The weights for CN model V_1 are spare, meaning that most of the weights are zero or low (black or blue), and some few weights are medium or high (red or white). The modification done changed hyper parameters of the CN feature extractor so that the weights would pick up upon larger frequency bands. The resulting weights for this modified CN model, called CN model V_2, can be seen in Figure 3.9 where the colouring is the same as before. In CN model V_2 there are more medium and high weights (red and white) than in CN model V_1.

2. Upon observing the weights in Figure 3.9 one can see that the they resemble Mel-filter banks in the following way: Each excitatory neuron picks up a frequency band, and that band often has medium weights at the edges and high weights at the centre, giving the weights a some what triangular shape. These bands tend to overlap, looking at the weights between several neurons. This is a similar to the Mel-filter banks in Figure 3.3. Given this resemblance the CN features of CN model V_2 were log filtered since this is also done to the MFBs. To learn how the opposite would affect the performance the CN features for CN model V_2 were also filtered with the power of two. To see if the filtering had the same effect on the CN features of CN model V_1 the filters were also used on the CN features from CN model V_1 as well.

3. To additionally try to improve performance, the level of inhibitory neurons were increased and decreased. The number of inhibitory neurons tried were

11, 33, and 66 with 100 excitatory neurons. This was done for both CN model V_1 and V_2, and these new features with different level of inhibitory neurons were also log / power filtered.

To summarise there were a total of 18 different CN features tested and evaluated. Nine were from the old model and nine were from the new model. For each model those nine features can be divided into three triplets, where each triplet has the same number of inhibitory neurons and the difference within one triplet being the filter applied to it, either being none, log, or power. These features will not be given specific names, but it should be clear from context which feature is meant. The performance result of the CN features from CN model V_1 and CN model V_2 are represented in chapter 5.2.2, the sections before and after that in the result chapter are strictly on the conventional features and the non filtered CN features from CN model V_2 with 100 excitatory neurons and 33 inhibitory neurons.

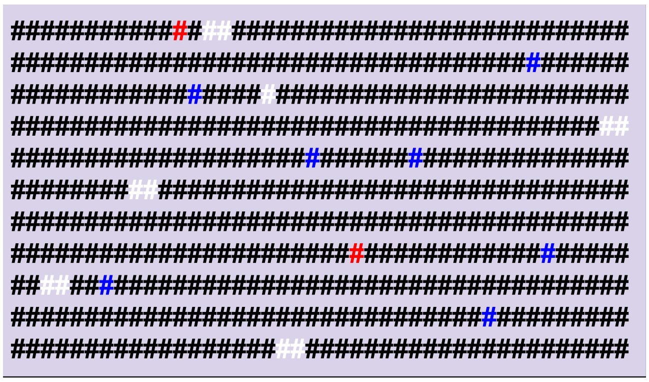

Figure 3.8: The weights of CN model V_1, here one can see that the weights are more sparse compared to the new model in Figure 3.9

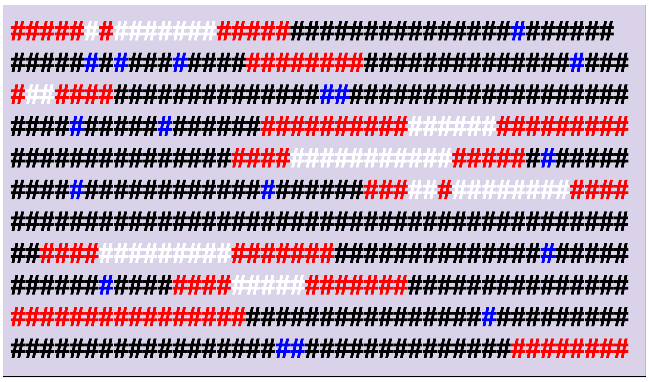

Figure 3.9: The weights of CN model V_2, here one can see that the weighs pick up larger frequency bands compared to the old model in Figure 3.8

Chapter 4
Methodology

Considering that the goal of this Master thesis work is to evaluate what type of information ANNs learn from specific features and understanding what type of information certain features offer, the ANNs chosen for implementation were, a one dimensional CNN based on (Snyder, Garcia-Romero, Sell, Povey, & Khudanpur 2018), and a LSTM network based on (Marchi et al. 2018). To further understand and analyse the advantages and shortcomings of the LSTM network a traditional RNN similar to the LSTM network was implemented. The CNN, and LSTM were chosen since they are the traditional networks used for speaker recognition and the more recent network architectures are based on them. For example, the CNN that will be implemented is based on the original x-vector model, and the LSTM model is based on Apples implementation on speaker verification used for Siri (Marchi et al. 2018). The more recent models were not chosen for implementation since they are more complex then the traditional models an would make evaluating the networks unnecessarily difficult. Details of the chosen model architectures can be found in section 4.1. All ANN models are stage-wise models. In this chapter "CN feature" refers to the non filtered CN features from CN model V_1 with 100 excitatory neurons and 33 inhibitory neurons.

The LSTM model, based on (Marchi et al. 2018), is optimised for MFCC features, and did not preform well for the CN features. To be able to do analysis of what a LSTM network learns from CN features, the LSTM network was regularised to get better performance for CN features. A RNN model regularised in the same way as the LSTM model was also created. Results from both the regularised and unregularised LSTM/RNN is presented in chapter 5.

The features that were used as input to the ANN models were the CN features, the MFCCs, the MFBs, and the STFT. All of the input features are two dimensional in time and space. These features consists of three seconds utterances, where each utterance is divided into about 300 frames or time steps. Depending on what type of input feature is used each frame consists of a number of data points equal to the number of points in space. A channel consists of all of the n:th data point for all frames. So for example in the STFT a channel is equal to a bin, or for the CN feature a channel is equal to one neuron output. The number of channels is equal to the number of points in space. The input size to the ANN models is the number of frames times the number of channels. The number of frames per utterance and the number of channels can be seen in Table 4.1 for every feature type used. In the table one can also see the window length and window step taken to calculate each feature.

Table 4.1: The number of frames per utterance and the number of channels. The window length and window step is also presented in seconds (s).

Feature type	Frames	Channels	Window length [s]	Window step [s]
CN	297	100	0.025	0.01
MFCC	299	20	0.023	0.01
MFB	299	40	0.023	0.01
STFT	302	62	0.023	0.01

The size of the MFCCs, MFBs and STFT features was chosen based on the standard used in the speaker verification field. To find the optimal number of excitatory neurons different amounts of excitatory neurons were tested on the regularised LSTM model. The EER result is shown in Figure 4.1, the number of excitatory neurons used is 50, 100, 150, and 200. The number of inhibitory neurons is a third of the number of excitatory neurons motivated by (Rongala, & Jörntell 2021). According to these findings the optimal number of excitatory neurons is 100, and the the optimal number of inhibitory neurons is 33.

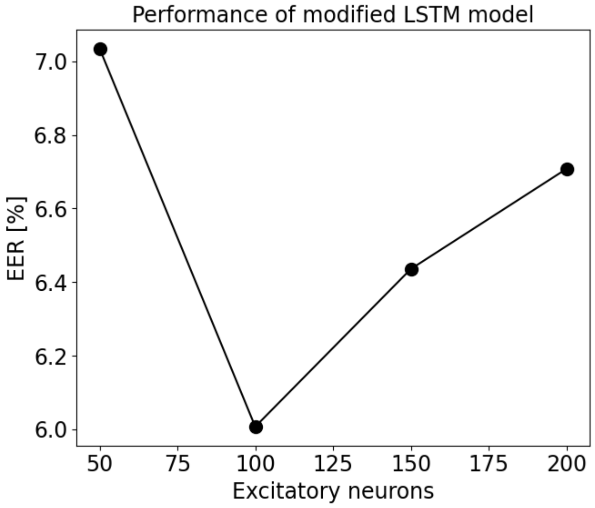

Figure 4.1: The EER for the regularised LSTM model for the clean test data set, for 50, 100, 150, and 200 excitatory neurons. The number of inhibitory neurons used is a third of the number of excitatory neurons.

4.1 ANN models

4.1.1 One dimensional CNN model

The one dimensional CNN is based on the same network that created the original x-vector (Snyder et al. 2018). It has five convolutional layers with BN after each convolutional layer. The details of the convolutional layers can be found in Table 4.2. All convolutional layers use the rectified linear unit (ReLu, $f(a) = \max(0, a)$) as activation function. The convolutional layers all have a stride of one. After the last convolutional and BN layer there is a statistical pooling layer, that calculates the mean and variance of an utterance (in time) and then concatenates the two resulting vectors. So, after the convolutional layers and the statistical pooling layer an utterance of about 300 time steps is reduced to a single vector consisting of the mean and variance of the utterance. After the statistical pooling layer there are two fully connected layers with ReLu activation functions. The number of nodes of these fully connected layers are 512 and 300 respectively. The speaker embedding (x-vector) is extracted from the last fully connected layer. The output layer has the same number of nodes as the number of speakers and uses the softmax activation function, defined as

$$f(a) = \frac{e^{a_i}}{\sum_{j=1}^{K} e^{a_j}}, \tag{4.1}$$

where i is the current node and K is the number of speakers.

Table 4.2: The architecture of the convolutional layers. After each convolutional layer a BN layer is applied (not shown in the table).

Convolutional layer	Nbr of filters	Kernel size	Dilation
1	512	5	1
2	512	3	2
3	512	3	3
4	512	1	1
5	1536	1	1

4.1.2 Regularised & unregularised LSTM & RNN models

The unregularised LSTM network is based on Apples paper about speaker verification for Siri from 2018 (Marchi et al. 2018). The first layer of the network consists of 512 LSTM nodes. The output from this layer is generated from the last frame in an utterance. The output from earlier frames in the utterance is only used to create the three different gates. In Figure 3.6 this would mean that if the features had three frames only the h_{t+2} output would be fed to the next layer. This means that the the first layer of the network effectively reduces an utterance to a single output vector. The activation function of the LSTM layer is the tanh function. After the LSTM layer comes a BN layer. The next layer in the network is a fully connected linear layer with 128 nodes with a linear activation function. The speaker embedding is extracted from this linear layer. After the linear layer comes a BN layer. The output layer has the same size as the number of speakers in the data set and has a softmax activation function, see equation (4.1).

The regularised LSTM network has the same general structure as the unregularised LSTM model. The differences are the following; instead of 512 LSTM nodes there are 64 LSTM nodes. The batch normalisation layer after the LSTM layer is removed. The output from the LSTM layer is fed directly into the linear layer. Dropout is also used on the linear layer, with a dropout probability of 30 %. The differences between the two models can be seen in Table 4.3.

Table 4.3: The differences between the unregularised and regularised LSTM/RNN models.

Unregularised	**Regularised**
512 recurrent layer	64 recurrent layer
BN layer	No BN layer
linear layer	linear layer
No dropout layer	30 % dropout layer
BN layer	BN layer
softmax layer	softmax layer

The traditional RNN implementation is similar to the LSTM network above, except that instead of the first layer (with 512 LSTM nodes) there are 512 ordinary nodes with a simple feedback connection as illustrated in Figure 3.4. Although, as in the LSTM network only the output generated from the last frame in an utterance in the RNN layer will be fed in to the next layer. The regularised RNN model is similar to the regularised LSTM model, in this case the LSTM layer is replaced with 64 ordinary nodes with a simple feedback connections. The differences between the two models can be seen in Table 4.3.

4.1.3 Hyper parameter settings

The following is true for all five networks. The Adam optimisation algorithm is used to update the weights with TensorFlow's default parameter settings, so the learning rate is $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. The loss function to be optimised is the categorical cross entropy function (see equation (3.13)), with the number of categories being the same as the number of speakers during training. The Adam optimisation method and the categorical cross entropy function was chosen since these were used in the original models and they are almost always used in all stage-wise implementations. A mini batch size of 32 is used, and only clean data is used to train the networks. Early stopping is used, the network stops training if the validation loss has stagnated for five epochs. Otherwise, the network is allowed to train for 200 epochs.

4.1.4 Testing of ANN model

When the ANN model has finished training, the model is saved from the input layer to the embedding layer. This embedding model is then tested on a data set that only contains new speakers. Five utterances from each speaker is used to create template embeddings for every speaker. The rest of the utterances are transformed to embeddings. The similarity score is computed, with the cosine similarity, between the current embedding and a template embedding (see equation (3.5)), if the score is above a certain threshold then the embeddings are considered to belong to the same person otherwise not. This threshold is not predetermined and needs to be optimised. The optimisation process calculates the similarity scores for each template embedding and utterance embedding, then every true positive and true negative, and false positive and false negative is recorded. If the false rejection rate is is larger than the false acceptance rate then the threshold gets lowered and vice versa. If the difference of the false rejection rate and the false acceptance rate is lower 0.1 then the optimal threshold value has been considered reached. After this, the speaker verification model is complete.

4.1.5 Data sets

Training data set

The training data set consist of data without noise and consists of 97 thousand samples where 80 % is used for training and 20 % is used for validation. There are a total of 921 different speakers in this data set.

Test data sets

The trained ANN models are tested on seven different data sets. One of the test sets named "test" is a data set without any noise. The data set "test" contains 40 new speakers, not included in the train data set. The other six test sets are modifications of the original "test" data set, the modification being that different types and levels of noise are superimposed on the original "test" set. The level of the noise is measured with the signal to noise ratio (SNR), in decibel (dB), defined as $\text{SNR} = 10 \cdot \log_{10}\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right)$, where P_{signal} and P_{noise} is the average power of the signal and the noise. The noise that is added is active noise, and can for example be other people talking in the background or a washing machine in the background. Three of the six data sets with noise contain only people talking in the background at different SNRs, which are 20 dB, 10 dB, and 0 dB. This three types of data sets are named as the "chatter" data sets. Then the other three of the six data sets with noise contain all types of active noise, with SNRs being 20 dB, 10 dB, and 0 dB, the data sets are named "active".

4.2 Statistical analysis of feature types

Three different types of feature analysis was performed, listed below:

1. Pairwise cross correlation for zero lag between feature channels, for all features, the result was then averaged. This was done for both the clean test data set and the chatter SNR 10 dB data set.
2. The average activity for each feature was measured by taking the sum over all channels at each time step and then taking the average of the result, this was done for the clean test data set and the chatter SNR 10 dB data set.
3. Calculating the entropy within the features by calculating how probable a channel output is at each time step. The result was then averaged. This was done for both the clean test data set and the chatter SNR 10 dB data set.

Each of these feature analysis methods will be explained a bit more in depth in the following sub sections.

4.2.1 Cross correlation

Assume that c_i is the ith channel of a feature and that $i = [1, n]$ where n is the number of channels that the feature has. Then the pairwise cross correlation $C_{i,j}$ of the ith and jth channel of this feature is calculated as

$$C_{i,j} = \frac{c_i^T \cdot c_j}{\sqrt{(c_i^T \cdot c_i) \cdot (c_j^T \cdot c_j)}}. \tag{4.2}$$

The cross correlation is the scalar product of the two channels, the denominator is the square root of the auto correlation for the two channels and normalises the output.

Calculating and interpreting correlation for the MFCCs

It is important to understand the general shape of the different feature types to be able to interpret the correlation result. In Figure 5.6 to Figure 5.9 one can see the different features for the same sample. Looking at Figures 5.6 and 5.7 one can see that both CN feature and the STFT have outputs larger than zero and that many channels are active at the same time. This will cause the correlation for these features to be somewhat high. In Figure 5.9 we can see an example of the MFB, here we see that the general shape of the feature is such that all channels generally have a value that is above zero or below zero making the total correlation high. According to the theory the MFBs are log transformed before they are used. Applying the log transform means that all the values of the MFB that were between zero and one before the log transform will be mapped into a value between minus infinity and zero. A MFB having a value between zero and one before log transformation means that there generally is silence. So all the outputs of the MFBs that has a value less then zero (post log transformation) is equivalent to there being silence. This probably one of the down sides of the MFBs since the negative amplitude is generally as large as the positive, giving a larger than necessary correlation result. In Figure 5.8, we can see that the MFCCs also take on both positive and negative values, and most of the time there are channels that are positive and negative at the same time, unlike the MFBs. This shape of the MFCC feature results in some of the pairwise correlations being negative and some being positive, and so these results when summed will cancel each other gaining a correlation mean close to zero. The negative output of the MFCCs is not to be confused to the negative MFB outputs, since they mean different things. A negative MFCC value still means that there is activity there. According to theory, the MFCCs are obtained from the DCT of the MFBs, see equations (3.3)-(3.4). Where the DCT returns spectrum of the MFBs, but only with respect to cosine. So, an negative value corresponds to the MFB being shifted π radians, compared to the cosine in equation (3.3). In other words a negative value means that there is activity in the same frequency as if the value was positive. So, when calculating the correlation for the MFCCs they have to be filtered with the absolute value, so that the feature only has non negative outputs, otherwise the result will be miss leading. This was done in the cross correlation results in chapter 5.2.1

4.2.2 Average activity

Assume that $a_{t,i}$ is the output of channel i at time t then the activity A of one feature is calculated as

$$A = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} a_{t,i}. \tag{4.3}$$

Where T is the number of time steps in the feature, and n is the total number of channels in one feature.

4.2.3 Entropy

Assume that $p_{t,i}$ is the probability of $a_{t,i}$ occurring at time t on channel i, then the entropy E for one feature is calculated as

$$E = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} p_{t,i} \cdot \log_2(p_{t,i}). \tag{4.4}$$

Where T is the number of time steps in the feature, and n is the total number of channels in one feature. A high entropy generally means that the predictability of the feature type is less certain, and a lower value means that the predictability is more certain.

Chapter 5

Results

5.1 ANN performance

In Figures 5.1 - 5.5 the EERs for the different test data sets is plotted for all models and features. In Figure 5.1 one can see that the CNN model manages to perform well on all features, where the MFCCs are performing best on the clean data and for active SNR 0 dB. The STFT feature performs best on active noise for SNR 10 and 20 dB, and on all chatter data sets. In the figure one can also see that the CN features have the highest EER on the clean data set, but performs better than some other features on the noise data sets.

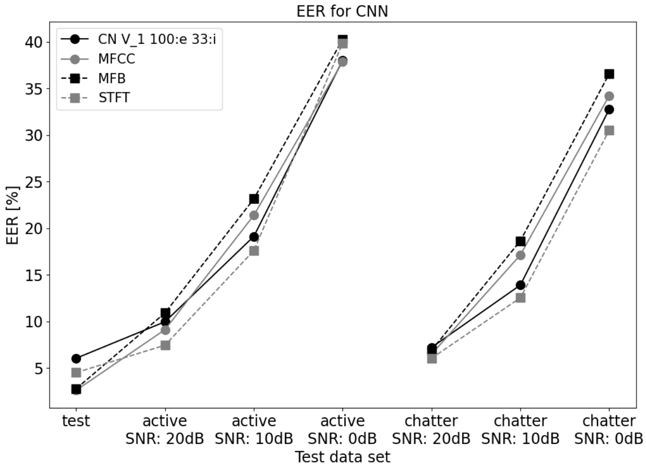

Figure 5.1: The EER results for the CNN for the different features and test sets.

In Figure 5.2 one can see that the performance of the unregularised LSTM model for the different features vary. Here one can see that the unregularised LSTM model does not have good performance for the CN features and the STFT. The MFBs are performing best for all test sets except on the active noise with SNR 0 dB test set where the MFCCs perform best.

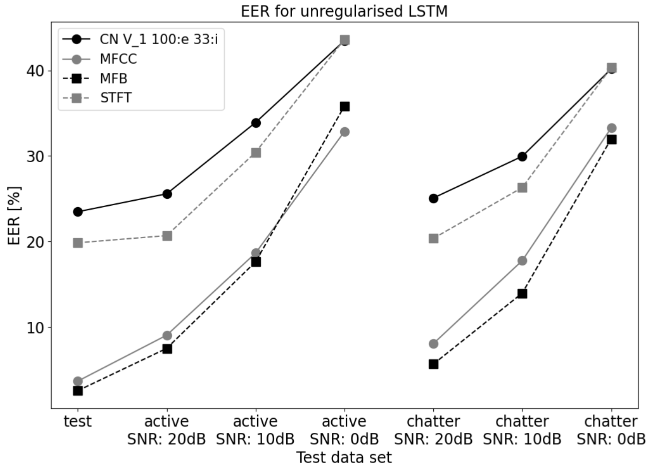

Figure 5.2: The EER results for the unregularised LSTM for the different features and test sets.

In Figure 5.3 one can see the EER for the regularised LSTM model are in the same order for all features, where the MFBs are performing best on the clean data set and chatter data set with SNR 20 dB. On active noise with SNR 10 and 20 dB, and chatter SNR 0 and 10 dB the STFT seems to be performing best, while for chatter SNR 0 dB the MFCCs seems to be performing best. One can also see the the CN features have similar relative performance as for the CNN model.

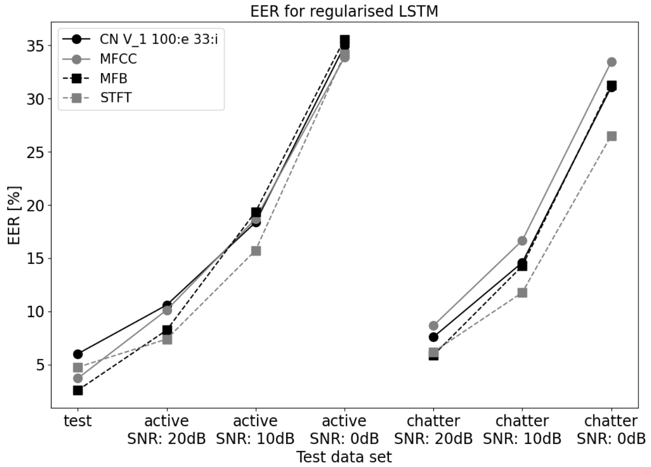

Figure 5.3: The EER results for the regularised LSTM for the different features and test sets.

In Figure 5.4 one can see the EER results for the unregularised RNN model, the results vary quite a bit from feature to features, with the MFCCs performing best on all data sets. Here one can see that the unregularised RNN model is performing poorly having an overall worse EER than all other previous models.

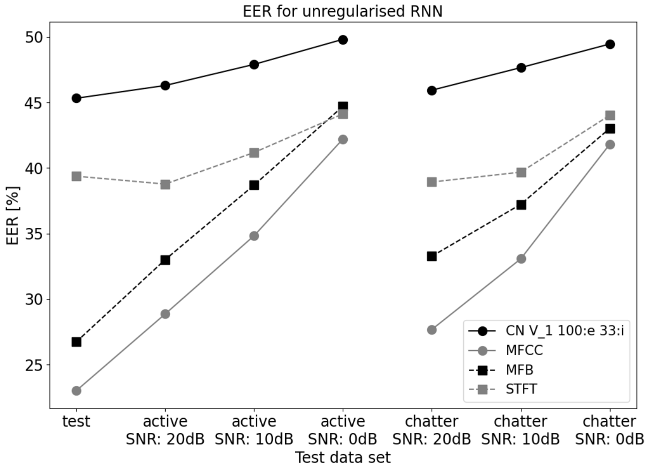

Figure 5.4: The EER results for the unregularised RNN for the different features and test sets.

The regularised RNN model can be seen in Figure 5.5, where the MFBs seem to be performing best on the active noise data sets and on chatter 0 dB, and on the clean data set. On chatter the STFT seems to be performing best for SNR 20 dB and 10 dB. Both RNN models are behaving poorly compared to the other models.

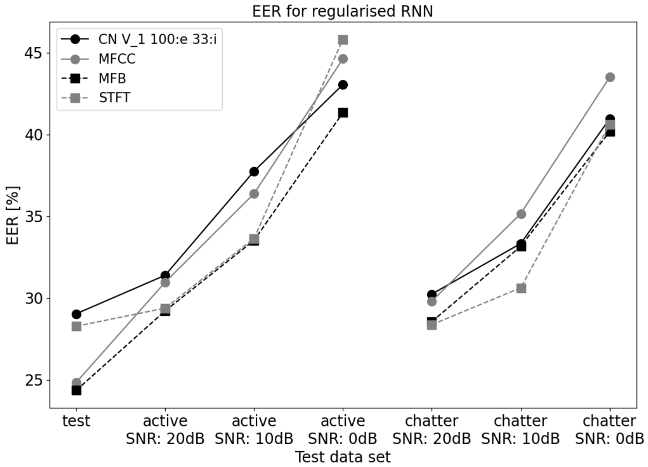

Figure 5.5: The EER results for the regularised RNN for the different features and test sets.

5.2 Statistical analysis of feature types

In Figures 5.6 - 5.9 one can see all the different feature outputs for the same clean three second sample. Looking at the different features in the Figure one can see that the CN feature and the STFT look similar, while the MFCCs and MFBs do not quite have a similar resemblance to the STFT.

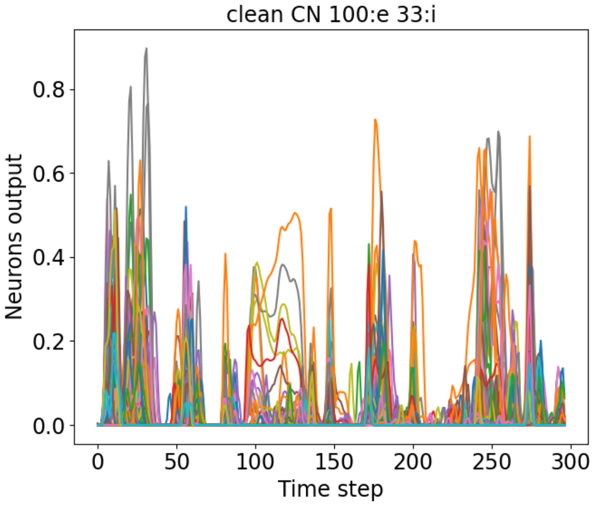

Figure 5.6: An example of the CN feature.

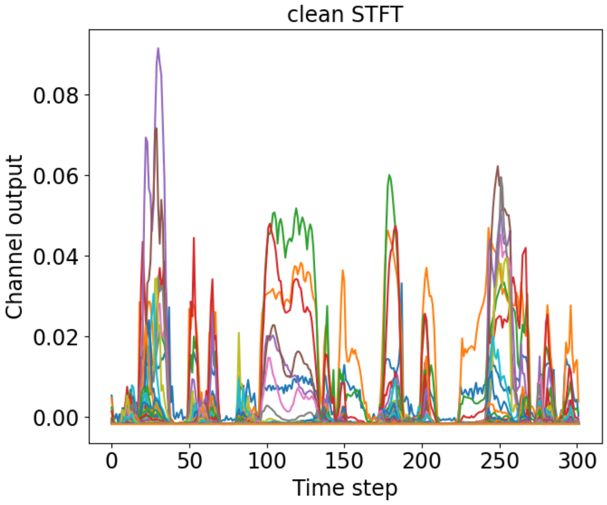

Figure 5.7: An example of the STFT feature.

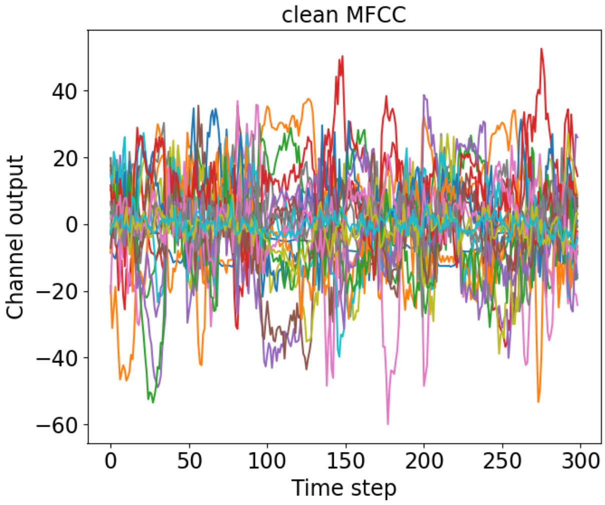

Figure 5.8: An example of the MFCC features.

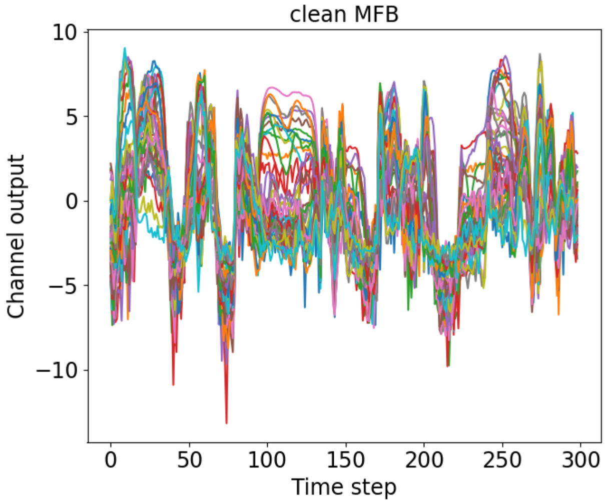

Figure 5.9: An example of the MFBs.

5.2.1 Correlation

In Figure 5.10, one can see the mean and standard deviation of the pairwise cross correlation of the clean test data set and for the chatter SNR 10 dB test data set, for all features. The correlation on the clean data set have the following results: From the figure one can see that both the CN feature and the STFT have a mean correlation between 0.3 and 0.4. While the MFCCs have a correlation result of about 0.63 and the MFBs have correlation result of about 0.51. From these correlation results one can say that the CN features and STFT have a similar correlation measure and that the MFCCs and MFBs have similar correlation measure. The CN features and the STFT also have medium correlation meaning that the outputs of the different channels have overlapping outputs. The MFCCs and MFBs have the the highest correlation meaning that the channels overlap more within a sample in comparison. The variance is the smallest For the MFCC meaning that the amount of overlap stays the same for all MFCC features. The variance is largest for the MFBs meaning that the amount of feature overlap varies a lot between samples. For the correlation on the chatter SNR 10 dB data set one can see that this correlation is always lower for the conventional features compared to the correlation on the clean test data set, but for the CN features this correlation is slightly higher.

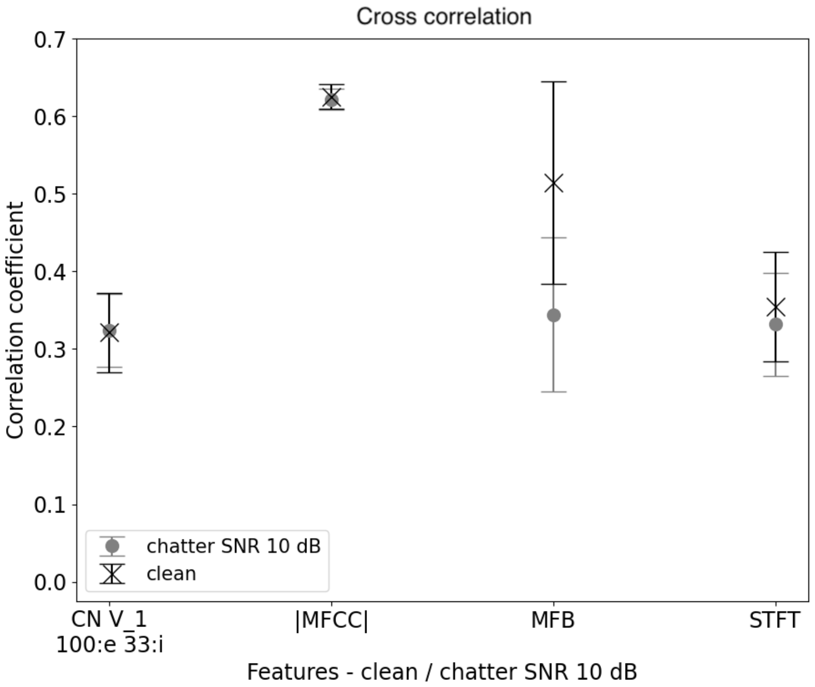

Figure 5.10: The pairwise cross correlation for zero lag. Here the mean and standard deviation of all samples in the respective test set is shown.

5.2.2 Adaptation of the CN features

Performance of CN model V_1 and V_2 with and without filtering

In Figure 5.11 one can see the correlation and performance result of CN model V_1 with the feature modifications described in chapter 3.3.7 in point 2. The performance is shown for the CNN model and the clean data set and the chatter SNR 10 dB data set. Here one can see that the correlation on the clean data grows for each feature on the x-axis. The features filtered with the power of two have the lowest correlation, the no filtered features have the next to highest correlation and the log filtered features have the highest correlation. The EER on the clean test data set is the highest for the power of two filtered features, next to lowest on the no filtered features, and lowest on the log filtered features. Here one sees a relationship between the correlation on clean data and performance on clean data, where a higher correlation gives lower EER and vice versa. Although, this is not true for data with noise, since one can see that for example the log filtered features have the highest EER on the data with noise. The correlation on the noise test data set does not give any indications of how the performance on noise should behave. On the clean data the EER improved with one percent point.

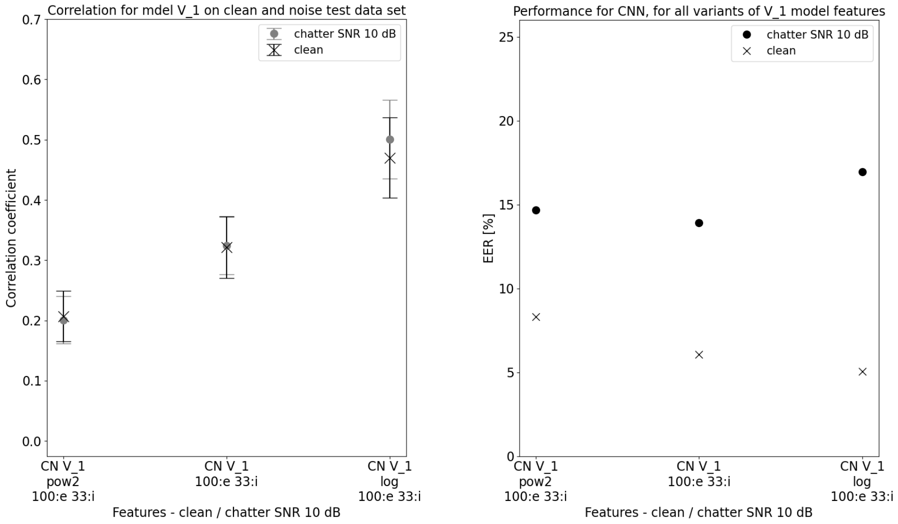

Figure 5.11: Results from CN model V_1 with 100 excitatory neurons and 33 inhibitory neurons, for both clean data and chatter SNR 10 dB. (Left) The cross correlation of the features. (Right) EER performance of the features.

In Figure 5.12 one can see the correlation and performance results of CN model V_2 with the feature modifications described in chapter 3.3.7 in point 2. The performance is shown for the CNN model. In this Figure one can make the same observations as in Figure 5.11. From both figures one can see that CN model V_1 has better performance results than CN model V_2. Here the performance of the regularised LSTM model is not shown since it had similar results.

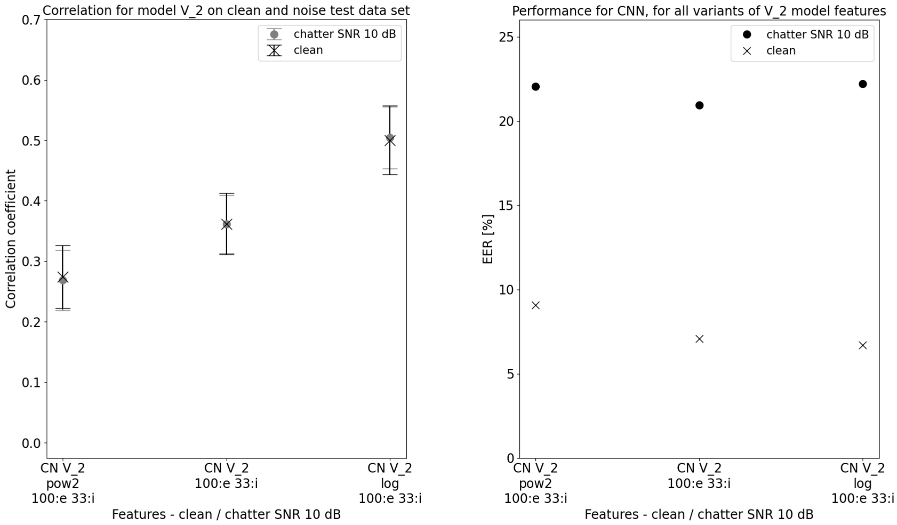

Figure 5.12: Results from CN model V_1 with 100 excitatory neurons and 33 inhibitory neurons, for both clean data and chatter SNR 10 dB. (Left) The cross correlation of the features. (Right) EER performance of the features.

Changing amount of inhibitory neurons

In Figure 5.13 one can see the performance result of CN model V_1 with the feature modifications described in chapter 3.3.7 in point 3. The performance is from the CNN model on the clean data set and the chatter SNR 10 dB data set. For all features that have the same amount of inhibitory neurons one can observe the same behaviour as in Figure 5.11. The same can be said for correlation (which is not shown). One thing to note in the figure is that the amount of inhibitory neuron does not change performance, it neither changes the correlation. This is true for both CN model V_1 and CN model V_2 for both the CNN model and regularised LSTM model. For this reason the the performance for other CN models and ANN models is not shown.

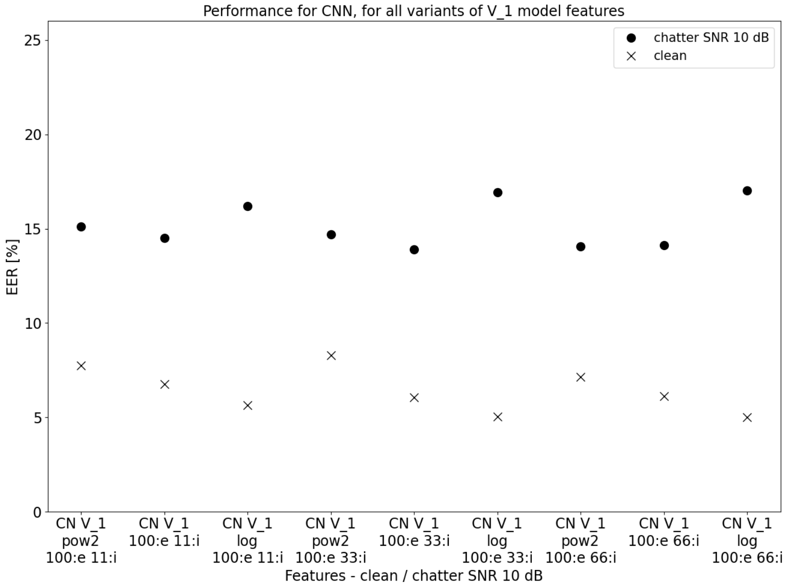

Figure 5.13: The performance of CN model V_1 with 100 excitatiory neurons and 11, 33, 66 inhibitory neuron, for both clean data and chatter SNR 10 dB.

5.2.3 Average activity

In Figure 5.14 one can see the average activity of the CN features from CN model V_1, MFCCs, MFBs and and STFT, and the standard deviation. The activity is calculated for the clean test data set and the chatter SNR 10 dB data set. Here, one can see that the CN features have high average activity and a large variance. The large variance implies that different samples of the features vary in the amount of total activity they output. One can also see that the activity is higher for the noise data set and lower for the clean data set. The MFCCs and MFBs have zero activity with zero variance, this means that for every sample the feature extractor puts out a constant amount of activity. The STFT also has low activity and variance, although higher than the MFCC an MFBs. Here the variance of the activity is a more important result than the mean, since it tells how much the energy changes from sample to sample.

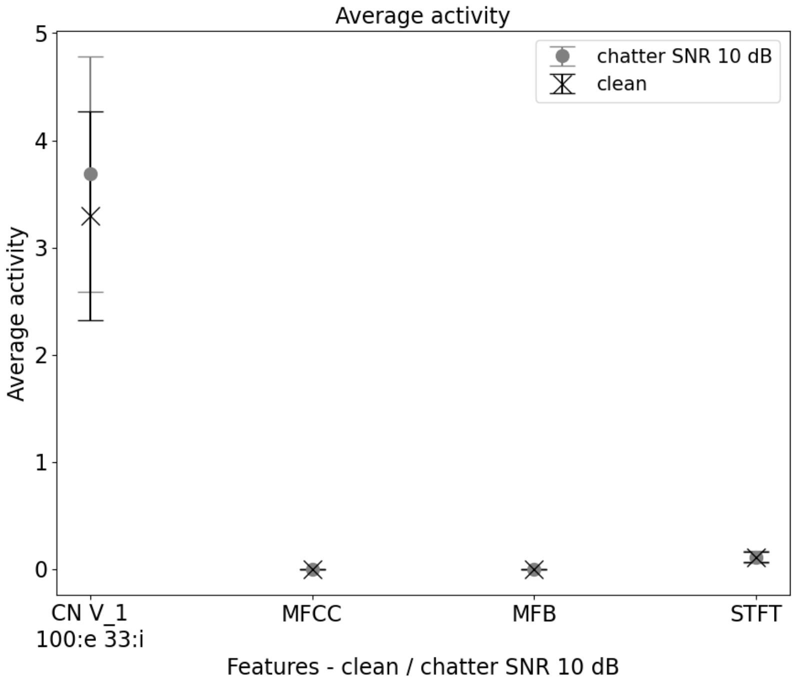

Figure 5.14: The activity of different features for both the clean test data set and the chatter SNR 10 dB data set.

5.2.4 Entropy

In Figure 5.15 one can see the mean entropy of the CN features from CN model V_1, MFCCs, MFBs, and STFT, and their standard deviation. The entropy is calculated for the clean test data set and the chatter SNR 10 dB data set. Here, one can see the the MFCCs and MFBs have a higher entropy then the other features, and that they have the lowest variance than the other features. A higher entropy means that there is more uncertainty of the next outcome. The low variance indicates that the uncertainty is the same for each sample. The CN features and the STFT have lower entropy then the other feature, meaning that there is a higher certainty of what will happen next. This certainty varies from sample to sample. One can also see that the entropy is higher for the noise data indicating that there is less certainty of what happen next. Here, the variance is more important than the mean since it tells how much the certainty varies from sample to sample.

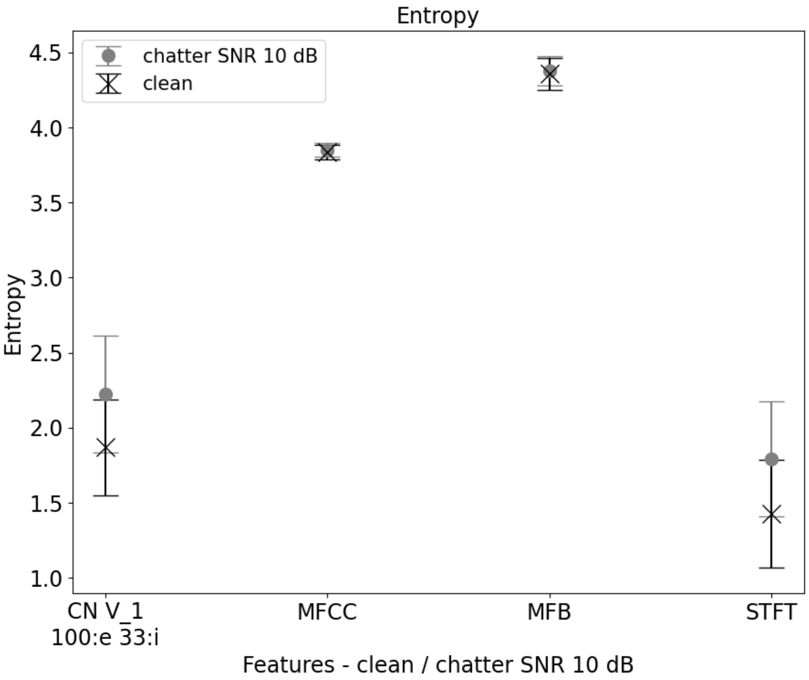

Figure 5.15: The entropy of different features for both the clean test data set and the chatter SNR 10 dB data set.

Chapter 6
Discussion

6.1 ANN performance

From the performance results in Figures 5.1 - 5.5, we found that the CNN model and regularised LSTM model perform better than the two RNN models and the unregularised LSTM model. The reason for the RNN models performing poorly is due to their inherent difficulty of capturing long term dependencies (further briefed in Section 6.1.1). We can see in Figure 5.2 that the unregularised LSTM model does not perform well for CN features and STFT. The unregularised LSTM models poor performance on CN features and STFT is due to overfitting (further briefed in Section 6.1.2). For these reasons the two RNN models and the unregularised LSTM model are not suited to be post processors to the CN features. The CNN model and regularised LSTM model perform better on all the different features, therefore we only use these two models to further analyse other parameters in this thesis study. From Figures 5.1 and 5.3, we can see the performance result of the CNN model and the regularised LSTM model. We observe that these models performed better with MFCCs and MFBs for clean data, compared to the CN features and STFT. In contrast, the models performed better with CN features and STFT for noise data, compared to the MFCCs and MFBs.

Since the CNN model and regularised LSTM model had similar performance, it was decided that analysing and understanding the information content in the features was more important for boosting performance than trying to decide which of these models is more optimal through hyper parameter optimisation. Realising this, a statistical feature analysis work started.

6.1.1 RNNs and long term dependencies

From Figure 5.4 and Figure 5.5 we can see that the traditional RNN is performing poorly. One reason for this is that the vanishing gradient problem causes the models to be bad at capturing long term dependencies, see chapter 3.3.1. This result is not particularly surprising and is the reason why LSTM networks are used instead of RNNs in state-of-the-art speaker verification approaches.

6.1.2 Unregularised LSTM model overfitting

In Figure 5.2 we can see that the unregularised LSTM model is overfitting to the CN features and the STFT since the EER is higher for these features compared to the MFCC and MFBs. This was also seen during training. Overfitting means that the unregularised LSTM overtrains on the training data set, resulting in a good training performance but bad validation and test performance. One reason for why overfitting occurs is due to a too large ANN model, and a simple solution is to reduce the size of the network. A large network is more prone to overfitting since it has more parameters that can fit the background noise of the training data set.

6.2 Statistical analysis of feature types

6.2.1 Correlation

The correlation results from Figure 5.10 show that the MFBs have a higher average correlation and also the highest variance compared to the other features, (on the clean test data set). The reason for this is that when the Mel-filter banks are created, there are overlaps between the bins (Figure 3.3). This overlap causes the bins to be artificially correlated. The CN features and STFT have similar correlation level, for both clean and noise test data sets. These results suggest that the CN features and the STFT can be comparable in terms of information content they output to the post processors.

From Figure 5.10 and 5.1, we see that a feature that is overlapping between different channel outputs (has a higher correlation on clean data), tends to perform better on clean data than features that overlap less. Based on these results a hypothesis was formed that if the correlation of the CN features on clean data were to increase, in other words if the feature overlapped more between different channel outputs then the performance on clean data should increase. The reason why a broader feature output is to be preferred to a narrower feature output is that a broader feature can compensate for the natural variance that occurs when a person speaks. Although, a feature cannot be too broad since then the feature will not be specific enough to be able to distinguish between different people. To test the hypothesis, different modifications were made to the CN feature to try and increase the correlation on clean data, and then they were tested on the different ANN models to see if this changed the performance.

6.2.2 Adaptation of the CN features

To be able to make the CN features broader, we explored different model configurations differing by; final weight distributions, output activity filtering, & number of inhibitory neurons.

Performance of CN model V_2

First, CN model V_1 was modified. From Figure 3.8, we can see that the weights of CN model V_1 are sparsely distributed. While looking at the weights of CN model V_2 (Figure 3.9), we can see that each excitatory neuron picks a range of frequency bands; leading to features with relatively high correlation, similar to the MFBs. The performance did not increase compared to the CN features created by CN model V_1 even if the correlation did increase. The CN model V_2 performed about one percentage point worse compared to the CN model V_1, on clean data, for the CNN model (Figure 5.11 & 5.12). In the same figures we can see that the chatter SNR 10 dB data set, the performance of CN model V_2 got worse by seven percentage points, for the CNN model. The correlation however did increase for CN model V_2 compared to CN model V_1 by 0.04 percentage points (clean test data set).

Filtering of CN output features

Another method, log filtering the output CN features of CN model V_1 and V_2, was also tested. The log filtering increased performance and the correlation on clean data compared to non filtered CN features. The log filter has a smoothing effect and makes the outputs broader than they were before filtering. In other words applying log filtering would give a higher correlation because of the smoothing effect. The same filtering approach was used in the state-of-the-art models that use MFBs. The correlation hypothesis seems plausible considering the performance and correlation results for the CN features of CN model V_1 and CN model V_2 with and without log filtering.

In order to test the effect of decreased correlation the CN features were filtered with the power of two to make the feature output narrower. This filtering would make every output sharper in turn decreasing the correlation. These modification yielded in results that strengthen the hypothesis (a decrease in correlation worsens the performance). The performance and correlation results for the CN features of CN model V_1 and CN model V_2 with and without filters (log & power) can be seen in figures 5.11 and 5.12 respectively. In both figures we see that the cross correlation index increases with each filter type as expected, we can also see that the EER of the clean data set also decreases for the respective features. However, the performance on the chatter SNR 10 dB data set is not following the same behaviour as the performance on the clean data set. Here the log filtered features have the highest EER, the power filtered features the next highest EER and the non filtered features the lowest EER. This behaviour in the features is the same as we saw in Section 6.1, where a specific feature type either has relatively better performance on clean data and relatively worse performance on noise data compared to the other feature types or it is the other way around. So here we see that the log filtered CN features have the same behaviour as the MFCCs and MFBs in both correlation and performance results. We also see that the power filtered CN features have the same behaviour as the non filtered CN features and the STFT in both correlation and performance. This relationship is interesting since a higher correlation on the clean data seems to indicate that the performance will improve for clean data, but the performance on the noise data set will worsen, and vice versa.

CN model V_1 performed better overall compared to CN model V_2, especially on the noise data set. This indicates that CN model V_1 is better than CN model V_2. The log filtered CN features of CN model V_1 improved the EER with one percentage point on the CNN post processor, compared to the non filtered CN features on clean data. Although, this results are still worse than for any other conventional feature.

Changing amount of inhibitory neurons

In order to improve the EER, the CN features of CN model V_1 and CN model V_2 were modified further. This was done by increasing and decreasing the number of inhibitory neurons (the number of excitatory neurons were always 100). The numbers of inhibitory neurons tested were 11, 33, and 66 for CN model V_1 and V_2. These features were also tested after log and power filtering. All the results, for the same CN model and number of inhibitory neurons, confirm the hypothesis, in that a higher correlation tends to improve the EER, see Figure 5.13. In these results we can also see that the log filtered CN feature has the best performance on clean data compared to the other CN features, but that they perform the worst under noise conditions. The power filtering decreases the performance of the clean data set, where these features performed the worst, but they do not perform the worst on the noise data. A final remark on the performance is that the amount of inhibitory neurons do not seem to affect the performance of the CN feature. The number of inhibitory neurons does not change the correlation result significantly either, raising the question of how they affect the CN model and if they are needed for speaker verification.

6.2.3 Average activity

The activity results in Figure 5.14 shows that the MFCC and the MFBs always have the same activity for each sample, because of the zero variance. This is true for both data with and without noise. This means that no matter what type of input was given to the feature extractor, its total output activity will always be the same, this is true even for silence. This could be one reason for the higher cross correlation index for these two features than compared to the CN features and STFT (Figure 5.10). The average activity for the STFT was similar to that of the MFCCs and the MFBs. From Figure 5.14, we can then also say that the CN features are fundamentally different than the MFCCs and MFBs, since the amount of activity in the CN features is not fixed. Here we can observe that the CN features have a higher activity for input data with noise compared to clean data, indicating that the level of activity of CN output features are dependent on the auditory input data.

6.2.4 Entropy

Figure 5.15 presents the entropy measure of the different feature extractor outputs. The MFCC and MFBs have a higher entropy and a lower variance than both the the CN features and STFT. The larger entropy value means that there is less certainty in the next output of these feature extractors, and vice versa for a smaller value of entropy. In the same figure, we see that the entropy values for the CN features and the STFT have a lower entropy and a higher variance than both the MFCCs and the MFBs. However, the large variance of the entropy indicates that the entropy changes for each sample, meaning that there actually is less certainty of the next outcome, and this makes these features harder to train on for a conventional ANN. This is reflected in the training of the ANN post processor were the CN features and STFT always took longer to train to get as good validation results as for the other features. This could also explain why traditional ANNs undertrain / overtrain on CN features.

6.3 Conclusion

We conclude that the main findings of this Master's thesis work below:

1. The CNN model and the regularised LSTM model performed the best for CN features compared to the other ANN models. The CN features and STFT performed better on noise and worse on clean compared to the MFCCs and MFBs.

2. Features with lower cross correlation index on clean data will lead to poor performance on the clean data, but this will also improve performance on noise data, and vice versa.

3. Features with high variance in average activity and entropy is difficult for a conventional ANN to train on, and the performance on clean data will be poor for these features. However the fact that there are fewer limitations on the activity and entropy allows the noise to be modelled in unique ways, and this facilitates for the post processor to perform better on noise data. The opposite is also true.

4. Because of the dynamics in the CN features, a conventional ANN will only reach a certain level of performance. Going forward, the above points need to be considered when building a biologically inspired post processor that can utilise the dynamics of the CN features to achieve better speaker verification performance.

References

[1] Andersson, E. (2021). *Speaker Recognition using Biology-Inspired Feature Extraction.* Master thesis. Lund: Lund University. https://lup.lub.lu.se/student-papers/search/publication/9059963

[2] Bai, Z., & Zhang X-L. (2021). *Speaker Recognition Based on Deep Learning: An Overview.* https://doi.org/10.48550/arxiv.2012.00931

[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press. https://www.deeplearningbook.org/

[4] Marchi, E., Shum, S., Hwang, K., Kajarekar, S., Sigtia, S., Richards, H., Haynes, R., Kim, Y., & Bridle, J. (2018). *Generalised discriminative transform via curriculum learning for speaker recognition.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 5324-5328, doi:10.1109/ICASSP.2018.8461296.

[5] Muckenhirn, H., Magimai-Doss, M., & Marcel, S. (2018). *Towards Directly Modeling Raw Speech Signal for Speaker Verification Using CNNS.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 4884-4888, doi:10.1109/ICASSP.2018.8462165.

[6] Snyder, D., Garcia-Romero, D., Povey, D., & Khudanpur, S. (2017). *Deep Neural Network Embeddings for Text-Independent Speaker Verification.* https://danielpovey.com/files/2017_interspeech_embeddings.pdf

[7] Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., & Khudanpur, S. (2018). *X-Vectors: Robust DNN Embeddings for Speaker Recognition.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 5329-5333, doi:10.1109/ICASSP.2018.8461375.

[8] Rongala, B. U., & Jörntell H. (2021). *Rich dynamics caused by known biological brain network features resulting in stateful networks.* https://doi.org/10.48550/arXiv.2106.01683

[9] Variani, E., Lei, X., McDermott, E., Lopez Moreno, I., & Gonzalez-Dominguez, J. (2014). *Deep neural networks for small footprint text-dependent speaker verification.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 4052-4056, doi:10.1109/ICASSP.2014.6854363.

[10] Ye, F., & Yang, J. (2021). *A Deep Neural Network Model for Speaker Identification.* Applied Sciences. 2021; 11(8):3603. https://doi.org/10.3390/app11083603

Printed by Tryckeriet i E-huset, Lund 2022

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2022-889
http://www.eit.lth.se