SPEECH ACTIVITY DETECTION IMPLEMENTATION IN HEARING AIDS USING DEEP BELIEF NETWORK

By

EITHAR ALFATIH ABDALRAHMAN ELBASHEER

INDEX NO. 124033

Supervisor

Prof. Mustafa Omer Nawari

A REPORT SUBMITTED TO

University of Khartoum

in partial fulfillment of the degree of

B. Sc (HONS) Electrical and Electronic Engineering

(SOFTWARE ENGINEERING)

Faculty of Engineering

Department of Electrical and Electronic Engineering

October 2017
DECLARATION OF ORIGINALITY

I declare this report entitled “SPEECH/ NONSPEECH CLASSIFICATION IN HEARING AIDS USING NEURAL NETWORK” is my own work except as cited in references. The report has been not accepted for any degree and it is not being submitted currently in candidature for any degree or other reward.

Signature: _______________________
Name: _______________________
Date: _______________________

Acknowledgment

First and foremost, all praise and thanks to Allah for providing me health, will and strength to complete this work. Words are not enough to express my gratitude for all his blessings.

I am really thankful to Professor Mustafa Nawari, our supervisor, who have always been there for us whenever we needed help and never hesitated to provide us with the information and resources we needed.

I will also take this opportunity to thank Olla Abdelmoneim, my graduation project partner, for all the information I got from her and the amazing learning journey we had. Wish her successful life.
Dedication

To the one that has always been there for me, the one that made me who I am today, and the woman that I see as my idol and my forever supporter, my mother, thank you so much, god protects you.
To my father for being my hero and the source of comfort …..
To my sisters and brothers for the emotional support…
To all of my friends……
Abstract

Hearing aids are small devices, hearing impaired and deaf people wear to compensate their hearing losses.

Most of the affordable available hearing aids nowadays are designed only on one environment and for certain hearing characteristics.

In our work, we proposed an intelligent hearing aid that is able to adjust to changing environment and it can be designed according to the patient hearing characteristic.

Speech activity detection is implemented using deep belief networks. This step is used so that different processing steps will be done to the audio signal before it reaches the human ears.

Two different features have been used to train the DBN and a comparison was made between them. MFCC and LPCC features were used and gave similar output results. A choice of using LPCC features is made because of they are easy to implement.

المستخلص

السماعات الطبية هي اجهزة صغيرة توضع على اذن من يعانون من مشاكل في السمع خصوصا الصم منهم.

تتعويض النقص في السمع.

اغلب السماعات الطبية المتاحة حاليا باسعار في متناول الجميع, تعاني من مشكلتين اساسيتين وهما ان تصميمها يتم في بيئة محددة ولخواص سمعية معينة, حيث يصعب ارتدائها لكل الناس كما انها تصبح مزعجة عند تغيير البيئة. وقد أوضحت الدراسات ان اغلب ممن يعانون من مشاكل في السمع لا يستعملون السماعات بسبب الارتعاج عند تغيير البيئة.

كان ولأثر الكشف عن الحديث في موجات الصوت من اهم واكثر الخطوات فعالية في تطبيقات الصوت, وكذلك طرق مختلفة لتطبيق هذه الخطوة, منها وامها النظم الذكية والشبكات العصوبية.

في هذا المشروع نستعرض طريقة لتصميم سماعة ذكية تناسب تغييرات البيئة المحيطة ومصممة خصيصا حسب الشخص المريض وفقا لخواصه السمعية. يعتمد هذا التصميم على نوع يسمي الشبكات العصوبية العملية, والذي يستقبل خواص الصوت الداخل كمدخلات ويستخدمها في تحديد ما اذا كان يحتوي على حديث او لا. تم تمت مقارنة نوعين من الخواص وتم الحصول على نتائج مقارنة جدا. تم اختيار الترميز التنبؤي الخطي لسهولة تصميم خواصه.
TABLE OF CONTENTS:

List of figures

FIGURE 2.1: A TYPICAL DIGITAL HEARING AID ................................................................. 12
FIGURE 2.2 A SINGLE NEURAL NETWORK NODE COMPARED TO A HUMAN NEURON .......... 14
FIGURE 2.3 COMPARISON BETWEEN NON-DEEP NEURAL NETWORK AND A DEEP NEURAL
   NETWORK .......................................................................................................................... 15
FIGURE 2.4 CNN .................................................................................................................. 15
FIGURE 2.5 RNN ................................................................................................................ 16
FIGURE 2.6 DBN .................................................................................................................. 16
FIGURE 2.7 TIMELINE OF WHEN THE DIFFERENT NEURAL NETWORKS HAVE BEEN CREATED... 17
FIGURE 2.8 POPULARITY OF DNN FRAMEWORKS ............................................................ 18
FIGURE 3.1 PLOT OF MEL FILTERBANK AND WINDOWED POWER SPECTRUM .................. 25
FIGURE 3.2 LPCC PLOTS ..................................................................................................... 26
FIGURE 3.3 GBM AND RBM ............................................................................................... 27
FIGURE 3.4 GIBBS SAMPLING METHOD ........................................................................... 31
FIGURE 3.5 ILLUSTRATION OF CD1 STEPS ..................................................................... 32
FIGURE 3.6 STACKS OF RBM ............................................................................................ 33
FIGURE 3.7 BACKPROPAGATION METHOD ........................................................................ 34
FIGURE 3.8 IMPLEMENTED CLASSES AND THE RELATIONSHIP BETWEEN THEM .......... 36
FIGURE 3.9 THE HEARING AID ......................................................................................... 39
FIGURE 4.1 PERFORMANCE OF THE MFCC DBN ............................................................. 36
FIGURE 4.2 PERFORMANCE OF THE LPCC DBN .......... ERROR! BOOKMARK NOT DEFINED.37
List of Tables

TABLE 2.1 DNN FRAMEWORKS ........................................................................................................... 17
TABLE 3.1 BRIEF COMPARISON BETWEEN MATLAB TOOLBOXES .................................................... 35
TABLE 4.1 TESTING AND TRAINING ERRORS ............................................................................................ 38
TABLE 4.2 RECONSTRUCTIVE RBM WEIGHTS (MFCC) ................................................................................ 38
TABLE 4.3 DISCRIMINATIVE RBM WEIGHTS (MFCC) .................................................................................. 39
TABLE 4.4 RECONSTRUCTIVE AND GENERATIVE RBM BIASES (MFCC) .................................................. 39
TABLE 4.5 RECONSTRUCTIVE RBM WEIGHTS (LPCC) ................................................................................. 40
TABLE 4.6 DISCRIMINATIVE RBM WEIGHTS (LPCC) .................................................................................. 40
TABLE 4.7 RECONSTRUCTIVE AND GENERATIVE RBM BIASES (LPCC) 41

List of Abbreviations

ADC  Analog to Digital Converter
BP   Back-propagation
CD   Contrastive Divergence
CNN  Convolutional Neural Network
DAC  Digital to Analogue Converter
DBN  Deep Belief Network
DCT  Discrete Cosine Transform
DFT  Discrete Fourier Transform
DNN  Deep Neural Network
DRBM Discriminative Restrictive Boltzmann Machine
DSN  Deep Stacking Network
DSP  Digital Signal Processing
EER  Equal Error Rate
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAR</td>
<td>False Alarm Rate</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>GBM</td>
<td>General Boltzmann Machine</td>
</tr>
<tr>
<td>GRU</td>
<td>Gated Recurrent Unit</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>IDFT</td>
<td>Inverse Discrete Fourier Transform</td>
</tr>
<tr>
<td>LED</td>
<td>Linear Energy-based Detector</td>
</tr>
<tr>
<td>LLR</td>
<td>Log-Likelihood Ratio</td>
</tr>
<tr>
<td>LR</td>
<td>Likelihood Ratio</td>
</tr>
<tr>
<td>LPC</td>
<td>Linear Prediction Coefficient</td>
</tr>
<tr>
<td>LPCC</td>
<td>Linear Predictive Cepstral Coefficient</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long/Short Term Memory</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel-Frequency Cepstral Coefficient</td>
</tr>
<tr>
<td>MR</td>
<td>Miss Rate</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>OOP</td>
<td>Object Oriented Program</td>
</tr>
<tr>
<td>PCR</td>
<td>Persistent Contrastive Divergence</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>RBM</td>
<td>Restricted Boltzmann Machine</td>
</tr>
<tr>
<td>SAD</td>
<td>Speech Activity Detection</td>
</tr>
<tr>
<td>SM</td>
<td>State Machine</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>VAD</td>
<td>Voice Activity Detection</td>
</tr>
</tbody>
</table>
Chapter One: Introduction

1.1 Overview

This chapter presents an overview of the thesis including a problem statement, the background and motivation for this study, objectives of this project and the structure of the thesis.

1.2 Problem Statement

As the world’s population is increasing, there is a dramatic increase in the number of cars, factories and other noisy sources. This makes it difficult for us to hear each other in the streets without shouting and more difficult to enjoy our favourite music and the sound of singing mockingbirds. As this is a challenge for people with normal hearing characteristics it is a serious problem for hearing impaired people. Many methods are proposed to make intelligent hearing aids using artificial intelligence especially neural networks.

1.3 Background and Motivation

As the world is getting noisier and noisier with time, more people are experiencing hearing losses, which leads to them being incapable of learning, perception and communication with others. Isolation, loneliness and depression are well known mental problems must of hearing impaired people suffer from.

This what motivated people to spend more time and work to create hearing aids. The fisrt hearing aid was created in the 17\textsuperscript{th} century. Started as ear trumpets, hearing aids revolted so quickly. Currently available ones are digital and are available in different sizes and prices.

There are still technical limitations in the hearing aids. Most of people who needs hearing aids do not wear it constantly. One of the reasons is that most of the available affordable hearing aids were designed only on one environment and don’t adapt to the changes in the surrounding environment. Intelligent hearing aids in good prices are required to overcome this problem.
1.4 Objective

The objective of this project is to build an intelligent and affordable hearing aid that is designed according to the patient hearing characteristics and able to adapt to the changing environment.

1.5 Thesis Layout

Chapter Two: contain a brief background about different neural networks and basic hearing aids design. It also contains a literature review about SAD and implemented hearing aids.

Chapter Three: describes the methodology used and the design steps followed.

Chapter Four: contains the results of the system and discusses the behavior of the entire system.

Chapter Five: concludes our work and shows what we hope to accomplish in the future.

Chapter Two: Literature Review

2.1 Introduction

Hearing aids are devices used by people with hearing impairments to compensate for the performance of their ears. Even though this device has helped a lot and provided a great service to many, it still has its drawbacks. These include wasteful power, and manual adjustment to the environment.

Speech/non-speech classification has been applied in hearing aids to overcome these drawbacks.[2] SAD saves resources by disabling the hearing aid when there is no speech. Moreover, SAD dramatically improved the noise reduction process.
For this application, we can use more than two classes: speech-in-quiet, speech-in-noise, music and others. Though our main interest is in the information contained in the speech, and for power saving reasons we will use two classes.

People with hearing impairment have weaker speech perception than normal people especially when speech is unclear because of interference with noise. Our proposed hearing aid can solve this problem by identifying the speech even in very noisy environments. This way our hearing aid will help them evolve from people with disabilities to people with proper hearing or maybe even super hearing.

2.2 Digital Hearing Aid

Hearing aids are used to improve hearing and therefore speech comprehension of people who suffer from hearing loss. The hearing loss can be due to damage to the small sensory cells in the inner ear called hair cells.

The purpose of a hearing aid is to magnify sound vibrations entering the ear. Surviving hair cells detect the larger vibrations and convert them into neural signals that are passed to the brain. The greater the damage to the hair cells, the more severe the hearing loss and the greater hearing amplification is needed. If the inner ear is exceedingly damaged a hearing aid would be ineffective.

The introduction of digital technology has drastically changed the way hearing devices process and deliver sound. The development of digital hearing aids drove the industry forward, making hearing aids more effective and more adaptable than ever before. A DSP device converts incoming sound waves into digital signals, which more efficiently replicates the incoming sounds. Because digital hearing aids are able to manipulate the sounds in their digital form by filtering, distorting and amplifying, they can be individually programmed to work more proficiently in a large range of settings. For example, digital aids can work in a crowded and noisy place by filtering out unwanted background noise and amplifying speech sounds, which creates a more natural and convenient sound environment. [3]

A typical hearing aid consists of the following components:
2.3 Speech Activity Detection

Speech activity detection also known as voice activity detection is a method used to detect the presence or absence of human speech in a sound signal.

Speech is a one-dimensional function of time. It is an audio signal which differs from person to person. Its different features include pitch, speed, language, accents and style in general. In
Chapter Two

addition to the features mentioned above, environmental noise interferes in the audio signal as well which makes speech processing a difficult task.

One way to overcome this difficulty is through implementing Speech Activity Detection. SAD also known as voice activity detection is a process of identifying speech segments in audio signal. It is a method used to detect the presence or absence of human speech in a sound signal, by doing this it eliminates noise signals. [4]

In view of the importance of SAD, it should be widely used as a necessary preprocess to speech processing applications such as: speech coding, speech recognition, speech understanding, speech verification, language translation, speech modification and speech enhancement and many others.

In the case of absence of speech, it deactivates the speech processing application. This way SAD saves resources and makes the latter speech processing applications easier, faster and generally more efficient.

Another significant implementation is using it in digital hearing aids. A process of noise extraction precedes the amplification process. If this is not done efficiently it might change the signal and moreover it will waste power. Therefore, using SAD in hearing aids improves there performance.

2.4 Neural Network[5]

Neural networks are a set of algorithms, modelled loosely after the human brain, that are designed to recognize patterns. They extrapolate sensory data through instrument perception, labelling or clustering raw input. The patterns they recognize are contained in vectors, into which all real-world data must be translated. The data could be images, sound, text or time series.

A neural network consists of an input layer, a hidden layer and an output layer. The layers are made up of nodes. A node is where computation takes place. The node is loosely designed to imitate a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines the input data with a set of coefficients, or weights. These input-weight products are
summed and the sum is passed through a node’s activation function, to determine whether the signal should progress and how much further it progresses to affect the ultimate outcome.

The input layer receives the inputs and features that the neural network is trying to cluster or classify. In the hidden layer is where the actual work takes place and the output layer presents to us the result of the clustering or classification.

2.4.1 Types of Neural Network[4][5]

There is classical neural network and then there is deep neural network. Non-deep neural network consists of an input layer, a hidden layer and an output layer. DNN is a stacked neural network it is composed of several hidden layers.
Figure 2.3 Comparison between non-deep neural network and a deep neural network

Different types of DNN include:

- CNN- Convolutional Neural Network is the brainchild of Yann LeCun and are often used to classify images, cluster them according to their similarities and perform object recognition within scenes. They are DNNs with two kinds of special layers: convolution layer and pooling layer.

- RNN- Recurrent neural networks are especially useful for processing sequential data such as sound, time series (sensor) data or written natural language. They differ from regular feedforward networks because they include a feedback loop. Feedforward networks accept one input at a time, and produce one output. Recurrent nets do not face the same one-to-one constraint. Typical RNNs later developed into LSTM and then GRU. GRU is fairly new and its trade-offs have not been completely explored yet.
**DBN- Deep Belief Network** is a neural network that consists of more than one Restricted Boltzmann Machine. RBM and DBN were conceived by Geoffrey Hinton. RBM is an algorithm useful for dimensions’ reduction, regression, collaborative filtering, classification, feature learning and topic modelling. It was invented by Geoff Hinton. It is a shallow two-layer neural network made up of a visible input layer and a hidden output layer. They are the building blocks of DBN.
Chapter Two Literature Review

- DSN- Deep Stacking Network or Deep Convex Network is different from regular DNNs. It is a deep set of individual networks.

![Timeline of neural networks](image)

Figure 2.7 Timeline of when the different neural networks have been created.

2.4.2 Programming languages for DNN
Deep learning algorithms can be written in a multiple number of programming languages. Those include Python, MATLAB, C++, Java, R, Lua and others. There are libraries, toolkits and toolboxes that help making the programming easier so that the programmers do not have to start building a neural network from scratch. Table 2.1 shows some of them.[7][8]

Table 2.1 DNN Frameworks

<table>
<thead>
<tr>
<th>Framework</th>
<th>Main Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theano</td>
<td>Python</td>
</tr>
<tr>
<td>Caffe 2</td>
<td>C++</td>
</tr>
<tr>
<td>Deeplearning4j</td>
<td>Java</td>
</tr>
<tr>
<td>Torch</td>
<td>Lua</td>
</tr>
<tr>
<td>MXNet</td>
<td>R and Python</td>
</tr>
<tr>
<td>CNTK</td>
<td>Python</td>
</tr>
<tr>
<td>Caffe</td>
<td>Python</td>
</tr>
<tr>
<td>Tensorflow</td>
<td>Python</td>
</tr>
</tbody>
</table>
2.5 Algorithms and Techniques

Intelligent hearing aids contain an extra step of classifying the input signal to some classes so that amplification and other processes occur based on decision made. Firstly, SAD is performed. Different algorithms and techniques were used and proposed for SAD. In this chapter we will go through a few of them. We will go through non-neural network techniques, classical neural network, DNN in general and then RNN and DBN.

2.5.1 Speaker and Noise Independent Voice Activity Detection[9]

F. Germain et al designed a VAD system which is both unsupervised and robust. Their approach is based on non-negative matrix factorization. The mixture of sounds, speech and noise, was explicitly modelled. This has the advantage that if one has a reasonable general model for speech, then the methodology will work in any noise environment. Therefore, once the system is deployed, it is considered to be unsupervised from a user’s perspective. The used approach also has the advantage of being fully interpretable the features used for classification correspond exactly to the relative levels of the speech and noise if this model was to be used for separation.

Their method based on non-negative matrix factorization for performing voice activity detection requires no training data from the user and is robust to changes in the noise environment. In particular, this method is able to handle a variety of non-stationary noises at low signal-to-noise ratios.

Figure 2.8 Popularity of DNN framework
2.5.2 Voice Activity Detection for Speech Enhancement Applications [10]

E. Verteletskaya and K. Sakhnov proposed an algorithm that used a periodicity measure of the signal, full band energy computation and signal energy ratio between high and low frequency. Their algorithm’s performance was compared to linear energy-based detector (LED) algorithm. The authors method outperforms the LED algorithm. This paper has presented voice activity detection algorithms used to detect the presence or absence of speech components in an audio signal. The results show that their proposed VAD scheme is better than the LED algorithm. It is easy to recognize that the algorithm has low computational complexity, and can be easily integrated into speech coders and other speech enhancement systems.

2.5.3 GMM versus DNN [11]

Neville Ryant, Mark Liberman, Jiahong Yuan in their work GMM baseline system and DNN SAD were compared. They used the same dataset of 65 hours’ web video for training and testing. They grouped the data into four different environments. These environments were music present, noise present, singing present and clean. Four error metrics used for evaluation; error rate, miss rate, false alarm rate and equal error rate.

First in the GMM baseline system 13 MFCC features were extracted, normalized and concatenated with their first and second vector to form a 39-dimensional feature vector input.

The GMM classifier consisted of two classes speech and non-speech. The number of components per class was 128.

Two segmentation methods were used. The first makes frame-wise speech/non-speech decisions by comparison of the log-likelihood ratio (LLR) between speech and non-speech GMMs to a threshold. The threshold was chosen so that FAR and MR are equal and only report the corresponding EER. In the second segmentation scheme, frame-wise decisions were produced by Viterbi decoding of the GMM log-likelihoods with a 2-state (speech/non-speech) HMM.

Use of Viterbi decoding results in a 8.41% reduction in overall frame-wise error rate when compared to the other segmentation scheme. Overall and within each environment individually, false alarms appear less prevalent than misses.
Secondly in the DNN system 13 MFCC features were extracted. After normalizing and concatenating with preceding and following frames they ended up with 1053 component feature vector input.

The DNN’s architecture consisted of 1053 unit input layer, 3 hidden layers (each containing 512 Rectified Linear Units) and two units output layer. The network was trained by backpropagation for 50 epochs.

Two segmentation methods were used as was done in the GMM baseline system. The first makes frame wise speech/non-speech decisions by thresholding on the posterior probability of speech being present that is produced by the DNN. The threshold was again chosen so that FAR and MR are equal and only report the corresponding EER. In the second segmentation scheme, frame wise decisions were produced by Viterbi decoding with a 2-state HMM.

Frame-wise EERs for the DNN SAD system compared relatively to the GMM baseline a 50.86% reduction in overall EER was observed. While the Viterbi decoding results in 15.43% overall frame-wise error rate reduction.

2.5.4 DBN[12]

Kim et al used an algorithm of deep belief networks (DBN) with likelihood ratio (LR) for SAD. The main idea was calculating the likelihood ratio from the input signal which was assumed to follow Gaussian probability density function (PDF). The likelihood ratio was used as an input to the DBN which made the classification to speech or non-speech. For the training of the DBN Bernoulli-RBM was used. The DBN was implemented with 100 epochs for pre training and 200 for fine tuning. The dataset used was 230 seconds from a group of males and females. Then noise was directly added to the clean speech resulting in SNR of 15dB. This method provided performance enhancement of range (0.7, 1.2) under quasi-stationary noise conditions, (0.7, 2.6) for street noise and (1, 2.1) for office noise.
2.5.5 RNN[13]

T. Hughes et al presented a novel recurrent neural network (RNN) model for voice activity detection. In this model, in which nodes compute quadratic polynomials, outperformed a much larger baseline system composed of Gaussian mixture models (GMMs) and a hand-tuned state machine (SM) for temporal smoothing. All parameters of the RNN model were optimized together, so that it correctly weighted its preference for temporal continuity against the acoustic features in each frame. This RNN used one tenth the parameters and outperformed the GMM+SM baseline system by 26% reduction in false alarms, reducing overall speech recognition computation time by 17% while reducing word error rate by 1% relative.

They trained and evaluated several variations on the RNN architectures by changing the number of parameters and frames. Then compared to each other and GMM +SM. The best variation was obtained with 354 parameters and 10 frames.

2.6 Intelligent Hearing Aid

Hearing aids have undergone vast development in the past couple of years. They have come a long way from ear trumpets and they continue to evolve as technology advances. Lately hearing aids designers and engineers are moving towards artificial intelligence and using neural networks.

Neural Network is being used to assist in the selection of hearing aids to patients with a particular audiogram since it has become increasingly difficult as the number of available models on the market has grown considerably.[14]

Otoconsult NV has created fitting Fox. Fox is a fitting assistant which helps audiologists in programming cochlear implants. is the first fitting assistant that helps the audiologist in programming cochlear implants. [15]

DeLiang Wang made a breakthrough in speech separation after years of research. He used a deep neural network classifier to classify speech from noise. His method has not been yet embedded into hearing aids.[16]
2.7 The Proposed Idea

We would like to create a deep neural network to classify speech from non-speech. Our method is going to try to attempt this classification from a different perspective to see if we would obtain the same results or different ones and different accuracy. The network is meant to be simpler and easier to implement. Afterwards the neural network will be applied in a hearing aid to optimize its performance.
Chapter Three: Methodology and Design

3.1 Extracted features:

3.1.1 Mel Frequency Cepstral Coefficients [16] [17] [18]

While speaking humans generate sounds that are filtered by the vocal tract including teeth and tongue. This how different sounds come out to create speech. In order to detect the speech, we need to detect the envelope of its short time power spectrum. MFCC features accurately represent this envelope.

Calculating MFCC features starts with framing the input signal into short frames. 35 ms frames are used here. Then we calculate the power spectrum for each frame. This will be followed by a step of applying the mel filterbank to the power spectrum and then for each filter we calculate the sum of energies. We then take the logarithm of each filterbank energy. After that we calculate the discrete cosine transform (DCT) of the filterbank energies with keeping the coefficients 2-12 while discarding the rest.

The framing step is important for simplicity, because audio signals change constantly and by framing the signal we can assume that on short time scales the audio signal doesn't change much. The best way is always to frame the signal into 20-40 ms frames. If the frame is much shorter we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.

The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates (which wobbles small hairs), different nerves fire informing the brain that certain frequencies are present. Our periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame.

One of the problems is that the cochlea can’t differentiate between two closely spaced frequencies. That’s why we take clumps of periodogram bins and sum them up to get an idea of
how much energy exists in various frequency regions. This is performed by the Mel filterbank: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations. The Mel scale determines exactly how to space the filter banks and how wide to make them. Motivated by human hearing we need to take the logarithm of the filterbank energies.

Because our filterbanks are all overlapping, the filterbank energies are quite correlated with each other. To do this we compute the DCT of the energies and we only keep the first 12 of the 26 DCT coefficients. This is because the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes actually degrade the performance of the system.

**Implementation steps:**

Calling our time domain signal as \( S(n) \), after framing we get \( S_i(n) \) frames where \( i \) indicates the frame number. When calculating DFT, we get \( S_i(k), P_i(k) \) represents the power of frame \( i \).

\[
S_i(k) = \sum_{n=1}^{N} S_i(n)h(n)e^{-j2\pi kn/N} \quad 1 \leq k \leq K
\]

\[
P_i(k) = \frac{1}{N}|S_i(k)|^2
\]

We apply the triangular filters (26 standard) to the calculated power to estimate the Mel-spaced filterbank. We can convert from frequency to Mel scale as follows:

\[
M(f) = 1125 \ln(1 + f/700)
\]  \( \text{(1)} \)
3.1.2 Linear Predictive Cepstral Coefficients [19]

LPCC features have always been used in speech recognition applications. We first compute the spectral of the smoothed Auto-Regressive power. Then we compute the autocorrelation sequence using the Wiener–Khinchin theorem.

To calculate LPCC features, $x(n)$ is the time domain signal, $X(k)$ is the complex spectrum, $P(k)$ is the power spectrum $x(n)$ of $x(n)$ and $A(n)$ is the autocorrelation sequence of $x(n)$.

First we calculate Discrete Fourier Transform

$$DFT(x(n)) \rightarrow X(k)$$

The power spectrum is computed as follows:
Chapter Three  Methodology and Design

\[ |DFT(x(n))|^2 \rightarrow P(k) \]

Tacking the inverse DFT of log the power spectrum we can get the LPC coefficients:

\[ IDFT(\log(P(k))) \rightarrow C(n) \]

Figure 3.2 LPCC plots

3.2 Deep Belief Networks (DBNs) and Restricted Boltzmann Machines (RBMs) [17]

DBNs are deep networks and composed of multiple layers of RBMs. RBM is a Boltzmann machine where the connections between hidden and visible layers are disjointed.

The Boltzmann Machine is a concurrent network with stochastic binary units. This network has set of visible units \( v \in \{0, 1\} \) and a set of hidden units \( h \in \{0, 1\} \).

The energy of the joint configuration \( \{v, h\} \) in Boltzmann machine is given as follows:

\[
E(v,h) = -\frac{1}{2} v^T L v - \frac{1}{2} h^T J h - v^T W h \tag{1}
\]

Restricted Boltzmann Machines are used for simplicity where \( J=0 \) and \( L=0 \).
The energy of the joint configuration \(\{v, h\}\) in restricted Boltzmann machine with adding bias is:

\[
E(v,h) = -v^TWh - a^Tv - b^Th = \]

\[
= \sum_{i=1}^{g_v} \sum_{j=1}^{g_h} W_{ij} v_i h_j - \sum_{i=1}^{g_h} a_i v_i - \sum_{j=1}^{g_h} b_j h_j \tag{2}
\]

Where \(W_{ij}\) represents the symmetric interaction term between visible unit \(i\) and hidden unit \(j\), while \(a_i\) and \(b_j\) are bias terms for hidden units and visible units respectively. The network assigns a probability value with energy function to each state in visible and hidden units.
The joint probability distribution for visible and hidden units can be defined as:

$$p(v, h) = \frac{1}{Z} \exp(-E(v, h))$$  \hspace{1cm} (3.3)

Where $Z$ as partition function or normalization constant, is obtained by summing over all possible pairs of visible and hidden vectors.

$$Z = \sum_v \sum_h \exp(-E(v, h))$$  \hspace{1cm} (3.4)

We can always increase the probability a network assigns to an input vector by decreasing its energy while increasing other inputs energies this can be done by manipulating the weights and biases.

A very simple learning rule for performing stochastic steepest ascent in the log probability of the training data can be defined by this equation:

$$\Delta w_{ij} = \varepsilon (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$  \hspace{1cm} (3.5)

Where the angle brackets are used to denote expectations under the distribution specified by the subscript that follows.

To find the corresponding biases we use this equation:

$$\Delta a_i = \varepsilon (\langle v_i \rangle_{data} - \langle v_i \rangle_{model})$$  \hspace{1cm} (3.6)
\[ \Delta b_i = \in (<h_j>_{data} - <h_j>_{model}) \quad (3.7) \]

Knowing that hidden units are independent in parallel we can calculate their probabilities of turning on given an input vector this way:

\[ P(h_j = 1|v) = \mathcal{G}(b_j + \sum_i v_i w_{ij}) \quad (3.8) \]

The same is for calculating the corresponding visible units when hidden units are given:

\[ P(v_i = 1|h) = \mathcal{G}(a_i + \sum_j h w_{ij}) \quad (3.9) \]

Knowing that \( \mathcal{G}(x) \) is the logistic sigmoid function \( \mathcal{G}(x) = 1/(1+\exp(-x)) \) we can compute \( <v_i h_j>_{data} \) without any difficulties. Computing \( <v_i h_j>_{model} \) is difficult and many methods has been introduced to calculate it.

### 3.3 Sampling Methods:

According to equation (3.8) we can define the gradient as the log \( (v) \) as follows:

\[ \emptyset = \log P(v) = \emptyset^+ - \emptyset^- \quad (3.10) \]

\[ \emptyset^+ = \log \sum_h \exp(-E(v, h)) \quad (3.11) \]
\[ \phi^- = \log Z = \log \sum_v \sum_h \exp(-E(v,h)) \quad (3.12) \]

Positive and negative gradients can be computed as follows:

\[ \frac{\delta \phi^+}{\delta w_{ij}} = v_i P(h_j = 1 | v) \]

(3.13)

\[ \frac{\delta \phi^-}{\delta w_{ij}} = P(h_j = 1, v_i = 1) \]

Comparing equation (3.5) and (3.12) we find that calculating \( <v_i h_j>_{\text{data}} \) is similar to calculating the positive gradient, that is why we call it the positive phase. Similarly, \( <v_i h_j>_{\text{model}} \) is called the negative phase.

While we can easily calculate the positive phase by finding \( P(h_j = 1 | v) \) for the visible unit \( v \), it is difficult for the negative phase to be calculated. Many learning methods are proposed to train DBN that differ in the sampling of the negative phase.

### 3.3.1 Gibbs sampling method

In this method, we start with random values for visible units and update the corresponding values for hidden units using equation according to equation (3.8). After that we calculate visible units using hidden units as inputs according to equation (3.9). We repeat these steps for a long time until we get good results. We initialize the chain by setting the values of visible units to be the same as the input vector.
3.3.2 Contrastive Divergence (CD)

Gibbs sampling takes so much time and if we know that the initial weights are bad it will be inefficient to spend all this time in the learning. G. Hinton discovered a shortcut that helped in introducing the Contrastive Divergence method [17].

In this method, we need only do one full update for visible and hidden units that means we calculate hidden units from visible units and then visible units will be recomputed from hidden units.

The figure below illustrates \( CD_1 \) method. By repeating \( CD_1 \) \( K \) times we can get \( CD_K \). Although \( CD_1 \) doesn’t calculate the gradient in a perfect way its results are acceptable [17].
Figure 3.5 Illustration of CD1 steps

Improving CD sampling method to avoid its disadvantages could be done by initializing the visible units using last chain state in the last update step instead of using training data [20]. This will give us an improved method called Persistent Contrastive Divergence (PCD), which we used in our work.

DBN consists of a stack of RBMs, where the activation values of the hidden units of each RBM will be supplied as input training data for the learning of the following RBM in the stack. After this pre-training, we fine-tune the error by performing back-propagation through the whole network. Figure 3.6 shows a DBN model where each RBM model performs a nonlinear transformation on its input vectors and produces as output, the vectors that will be used as input for the next RBM model in the sequence.
3.4 Backpropagation

Figure 3.6 illustrates the steps needed to implement the backpropagation algorithm.
Chapter Three

Methodology and Design

3.5 (DeeBNet V3.0) toolbox [22]

A DeeBNet V3.0 is an object oriented MATLAB toolbox which provides packages, classes and functions for implementing DBNs. There are many implemented classes for sampling, creation of RBMs, defining DBNs and other functions.

This toolbox has great abilities for implementation of DBNs. The toolbox supports different sampling methods (e.g. Gibbs, CD and PCD), different RBM types (generative and discriminative).
### Table 3.1 Brief comparison between MATLAB toolboxes

1. Object-oriented programming
2. Discriminative Restricted Boltzmann Machine
Figure 3.8 Implemented classes and the relationship between them
Chapter Three  Methodology and Design

3.5.1 Basic classes:

ValueType is an enumeration class to define the type of units for both RBM and DBN. The types can be binary, Gaussian or probability.

As we have two types of RBMs we have RbmType class that determines whether the RBM is discriminate or generative.

The RbmParameters class is used to define all parameters of an RBM such as weight matrix, biases, learning rate, etc.

In the DataClasses package we have DataStore class that has functions for normalizing and shuffling the data. It also has a function to cut the training data into data for training and another for validation. We can also plot the data using DataPlot class.

For sampling, different classes are implemented. Gibbs class uses Gibbs learning rule for training and is a parent of CD class and PCD class.

3.5.2 RBM classes

The most important classes include GenerativeRBM class that is used to define generative RBMs. It provides functions like train, getFeature, generateData, reconstruct Data, etc.

Train takes DataStore object and based on its training, validation and testing data it modifies the RBM parameters. The training will stop based on the value of maximum epochs. The getFeature method, samples the hidden units from visible units according to the predefined sampling method. On the other hand, generateData samples visible units from hidden units. To reconstruct the input data reconstructData is used. This function uses the extracted features from the visible units to reconstruct the input vector.

To define discriminative RBMs DiscriminativeRBM class is used. It has the same methods as in the GenerativeRBM class added to them two methods generateClass and predictClass methods. Given a class label the generateClass can generate different data. The predictClass method does exactly the opposite, it predicts the class of input data.
Chapter Three  
Methodology and Design

3.5.3 DBN classes

DBN consists of stack of RBMs. The addRBM method is used to add an RBM to the top of the stack. Other methods include train, getFeature, backpropagation, getOutput, plotBases, etc. The train method trains DBN, layer by layer. In other words, this method trains RBMs one after another and uses their extracted features for training in the next RBM.

The “getFeature” method is used to extract features from input data. This method extracts features layer by layer and returns hidden units’ activation values in last hidden layer as extracted feature.

The backpropagation method is used to fine-tune the free parameters of the network. It uses the neural network toolbox that is available in Matlab.

3.6 Our Dataset:

Our training set contains 2970 samples and it is divided into 48.88% speech and 51.12% noise. The testing dataset contains 1020 samples divided into 48.48% speech and 51.52% noise. Thirty sentences from the IEEE sentence database [23] were recorded in a sound-proof booth using Tucker Davis Technologies (TDT) recording equipment. The sentences were produced by three male and three female speakers. The IEEE database (720 sentences) was used as it contains phonetically-balanced sentences with relatively low word-context predictability. The thirty sentences were selected from the IEEE database so as to include all phonemes in the American English language. The sentences were originally sampled at 25 kHz and down sampled to 8 kHz.

Noise is artificially added to the speech signal as follows. The IRS filter is independently applied to the clean and noise signals. The active speech level of the filtered clean speech signal is first determined using the method B of ITU-T P.56 [24]. A noise segment of the same length as the speech signal is randomly cut out of the noise recordings, appropriately scaled to reach the desired SNR level and finally added to the filtered clean speech signal.
Chapter Three

Methodology and Design

Noise signals were taken from the AURORA database [25] and included the following recordings from different places: Babble (crowd of people), car, Exhibition hall, Restaurant, Street, Airport and Train station.

The long-term spectra of the above noises are given in [25]. The noise signals were added to the speech signals at SNRs of 0dB, 5dB, 10dB, and 15dB.

3.7 The Hearing Aid

We improved a simple efficient design for our digital hearing aids that is based on noise reduction, frequency, and amplitude filters. [26]

![Diagram of the hearing aid](image)

**Figure 3.9 The hearing aid**

According to the design described in figure (4.1), based on the result from the DBN classifier the audio signal will be passed for further processing before passing to the speaker if it contains speech signal otherwise no further processing is done.

The first step is to convert the analog signal into digital signal followed by Noise reduction. For the noise reduction, the noise is assumed to be modelled by white Gaussian noise and FIR MATLAB filter was used.
After that a frequency shaper is used. It amplifies the hard to hear frequencies and modifies other frequency regions. The frequency shaper needs the hearing characteristics of the patient in order to determine which frequency regions to amplify.

The last step in the sound manipulation is the Amplitude Shaper. We assume that the Frequency Shaper raises the frequencies that the user has difficulty hearing to sound pressure levels within his dynamic range of hearing. Therefore, all that our Amplitude Shaper has to do is check, bit by bit, that output power does not exceed a given saturation level (Psat). Since noise is concentrated in the low power levels as well, the filter also removes a significant amount of noise. Output power is equal to zero for levels below Psat.
Chapter Four: Results and Discussion

In this chapter we showed our work and the obtained outcome of training and testing our neural network. It shows the final parameters to obtain the best classification results. It show a proposed design for the digital intelligent hearing aids.

4.1 Results

The results obtained when Momentum = [0.5 0.4 0.3 0.2 0.1 0], Learning rate for the generative RBM = $1.000000000000000e-03$, and for the discriminative RBM it is 0.1 are as follows:

4.1.1 Error rate

The input to the network were LPCC and MFCC features. We applied each features separately and compared the results. We also the outputs are shown in table 4.1 for both networks. We did a further comparison as we tested the network once against stationary noise and then against noisy speech with different SNR values. The results are shown in table 4.1.

<table>
<thead>
<tr>
<th>Number of input features</th>
<th>Training error before BP</th>
<th>Training error after BP</th>
<th>Testing error before BP</th>
<th>Testing error after BP</th>
<th>Testing Error for SNR 0</th>
<th>Testing Error for SNR 5</th>
<th>Testing Error for SNR 10</th>
<th>Testing Error for Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCC</td>
<td>0.0034</td>
<td>3.367e-04</td>
<td>0.0061</td>
<td>0.001</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.087</td>
</tr>
<tr>
<td>LPCC</td>
<td>0.5152</td>
<td>0.0</td>
<td>0.5112</td>
<td>0.001</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0061</td>
</tr>
</tbody>
</table>
4.1.2 Neural network training performance

When trained and tested with the MFCC features the network the performance of the network was faster compared with the LPCC features, the difference is shown in the following figures:

*Figure 4.1 Performance MFCC features as input DBN*
4.1.3 Parameters
When supplied with The MFCC features the DBN RBMs’ parameters turned out as follows:

- Reconstructive RBM weights are shown in table 4.2.
- Discriminative RBM weights are shown in table 4.3.
- Reconstructive RBM visible and hidden layer biases are in table 4.4.
- Discriminative RBM visible layer and hidden layer biases are in table 4.4.
Table 4.2 Reconstructive RBM weights (MFCC)

<table>
<thead>
<tr>
<th>Visible Layer Nodes</th>
<th>Hidden layer nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.232595</td>
<td>2.942096</td>
</tr>
<tr>
<td>-0.13676</td>
<td>3.529409</td>
</tr>
<tr>
<td>-0.09227</td>
<td>-2.06233</td>
</tr>
<tr>
<td>0.280478</td>
<td>4.223146</td>
</tr>
<tr>
<td>-0.06177</td>
<td>5.069977</td>
</tr>
<tr>
<td>6.588595</td>
<td>-3.77946</td>
</tr>
<tr>
<td>-0.54325</td>
<td>-4.52762</td>
</tr>
<tr>
<td>-0.7411</td>
<td>-3.95794</td>
</tr>
<tr>
<td>6.617206</td>
<td>4.317389</td>
</tr>
<tr>
<td>-0.67044</td>
<td>4.980335</td>
</tr>
<tr>
<td>2.753979</td>
<td>-3.72483</td>
</tr>
<tr>
<td>-3.306</td>
<td>-4.77365</td>
</tr>
<tr>
<td>2.455187</td>
<td>-3.74589</td>
</tr>
<tr>
<td>-2.19845</td>
<td>-2.34989</td>
</tr>
<tr>
<td>2.335898</td>
<td>2.728517</td>
</tr>
<tr>
<td>-2.38958</td>
<td>-3.59119</td>
</tr>
<tr>
<td>2.983514</td>
<td>3.013306</td>
</tr>
<tr>
<td>-4.1472</td>
<td>-4.75782</td>
</tr>
<tr>
<td>3.282119</td>
<td>2.753979</td>
</tr>
<tr>
<td>-1.81274</td>
<td>-3.306</td>
</tr>
<tr>
<td>3.248177</td>
<td>-3.41052</td>
</tr>
<tr>
<td>-2.06621</td>
<td>2.728517</td>
</tr>
<tr>
<td>1.207113</td>
<td>-5.02745</td>
</tr>
<tr>
<td>-3.22164</td>
<td>3.013306</td>
</tr>
<tr>
<td>1.223967</td>
<td>-3.95794</td>
</tr>
<tr>
<td>3.432566</td>
<td>4.317389</td>
</tr>
<tr>
<td>-0.27349</td>
<td>4.980335</td>
</tr>
<tr>
<td>-0.70788</td>
<td>3.95794</td>
</tr>
<tr>
<td>3.442729</td>
<td>-4.75782</td>
</tr>
<tr>
<td>1.924</td>
<td>2.753979</td>
</tr>
<tr>
<td>-2.27237</td>
<td>-3.306</td>
</tr>
<tr>
<td>1.883147</td>
<td>-3.41052</td>
</tr>
<tr>
<td>1.518992</td>
<td>2.728517</td>
</tr>
<tr>
<td>0.063412</td>
<td>-5.02745</td>
</tr>
<tr>
<td>1.544618</td>
<td>3.013306</td>
</tr>
<tr>
<td>0.088117</td>
<td>-4.75782</td>
</tr>
<tr>
<td>2.178636</td>
<td>3.432566</td>
</tr>
<tr>
<td>-1.68863</td>
<td>-0.27349</td>
</tr>
<tr>
<td>2.102686</td>
<td>-0.70788</td>
</tr>
<tr>
<td>-1.87145</td>
<td>3.442729</td>
</tr>
<tr>
<td>1.829406</td>
<td>1.924</td>
</tr>
<tr>
<td>0.345749</td>
<td>-2.27237</td>
</tr>
<tr>
<td>1.902315</td>
<td>1.518992</td>
</tr>
<tr>
<td>0.115598</td>
<td>0.063412</td>
</tr>
<tr>
<td>1.698054</td>
<td>-1.68863</td>
</tr>
<tr>
<td>-0.94347</td>
<td>-1.07749</td>
</tr>
<tr>
<td>1.397997</td>
<td>1.829406</td>
</tr>
<tr>
<td>-1.211</td>
<td>1.698054</td>
</tr>
</tbody>
</table>

Table 4.3 Discriminative RBM Weights (MFCC)

Table 4.4 Reconstructive and Discriminative RBM Biases (MFCC)

<table>
<thead>
<tr>
<th>Reconstructive RBM</th>
<th>Generative RBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.942096</td>
<td>3.529409</td>
</tr>
<tr>
<td>-2.06233</td>
<td>-2.51153</td>
</tr>
<tr>
<td>4.223146</td>
<td>5.069977</td>
</tr>
<tr>
<td>-3.77946</td>
<td>-4.52762</td>
</tr>
<tr>
<td>-3.95794</td>
<td>-4.65195</td>
</tr>
<tr>
<td>4.317389</td>
<td>4.980335</td>
</tr>
<tr>
<td>-3.72483</td>
<td>-4.77365</td>
</tr>
<tr>
<td>Visible layer biases</td>
<td>Hidden layer biases</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>-0.0054</td>
<td>-0.25058</td>
</tr>
<tr>
<td>5.699019</td>
<td>-0.25458</td>
</tr>
<tr>
<td>-0.8011</td>
<td>0.243446</td>
</tr>
<tr>
<td>0.042431</td>
<td>-0.25087</td>
</tr>
<tr>
<td>-1.71788</td>
<td>0.247864</td>
</tr>
<tr>
<td>1.097953</td>
<td></td>
</tr>
<tr>
<td>-2.09241</td>
<td>-0.25458</td>
</tr>
<tr>
<td>2.594185</td>
<td>0.243446</td>
</tr>
<tr>
<td>-0.9944</td>
<td>-0.25087</td>
</tr>
<tr>
<td>1.492584</td>
<td>0.247864</td>
</tr>
<tr>
<td>0.068929</td>
<td></td>
</tr>
<tr>
<td>1.833715</td>
<td></td>
</tr>
<tr>
<td>0.373948</td>
<td></td>
</tr>
</tbody>
</table>

And then supplied with The LPCC features and the parameters turned out as follows:

- Reconstructive RBM weights are shown in table 4.5.
- Discriminative RBM weights are shown in table 4.6.
- Reconstructive RBM visible and hidden layer biases are in table 4.7.
- Discriminative RBM visible layer and hidden layer biases are in table 4.7.
Table 4.5 Reconstructive RBM weights (LPCC)

<table>
<thead>
<tr>
<th>Visible Layer Nodes</th>
<th>Hidden layer nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.145359</td>
<td>0.376614</td>
</tr>
<tr>
<td>-0.03235</td>
<td>0.129308</td>
</tr>
<tr>
<td>-0.03423</td>
<td>0.273005</td>
</tr>
<tr>
<td>0.007605</td>
<td>-0.01275</td>
</tr>
<tr>
<td>-0.08623</td>
<td>-0.09808</td>
</tr>
<tr>
<td>-0.0179</td>
<td>-0.05181</td>
</tr>
<tr>
<td>-0.11842</td>
<td>0.018765</td>
</tr>
<tr>
<td>0.075315</td>
<td>0.039236</td>
</tr>
<tr>
<td>-0.0205</td>
<td>0.001386</td>
</tr>
<tr>
<td>-0.07362</td>
<td>0.054855</td>
</tr>
<tr>
<td>0.061283</td>
<td>0.04283</td>
</tr>
<tr>
<td>0.128709</td>
<td>-0.02488</td>
</tr>
</tbody>
</table>

Table 4.6 Discriminative RBM Weights (LPCC)

<table>
<thead>
<tr>
<th>Visible Layer Nodes</th>
<th>Hidden layer nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.011105</td>
<td>-1.42072</td>
</tr>
<tr>
<td>-4.00946</td>
<td>1.372883</td>
</tr>
<tr>
<td>-0.12181</td>
<td>-0.09</td>
</tr>
<tr>
<td>0.121843</td>
<td>0.046835</td>
</tr>
<tr>
<td>-0.14728</td>
<td>-0.24763</td>
</tr>
<tr>
<td>-0.00441</td>
<td>-0.05916</td>
</tr>
<tr>
<td>-0.0385</td>
<td>-0.05176</td>
</tr>
</tbody>
</table>
### Table 4.7 Reconstructive and Generative RBM Biases (LPCC)

<table>
<thead>
<tr>
<th></th>
<th>Visible layer biases</th>
<th>Hidden layer biases</th>
<th>Visible layer biases</th>
<th>Hidden layer biases</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reconstructive RBM</strong></td>
<td></td>
<td></td>
<td><strong>Generative RBM</strong></td>
<td></td>
</tr>
<tr>
<td>Visible layer biases</td>
<td>0.441508</td>
<td>0.014815</td>
<td>0.441508</td>
<td>-0.18388</td>
</tr>
<tr>
<td>Hidden layer biases</td>
<td>-0.00762</td>
<td>-0.01071</td>
<td>-0.00762</td>
<td>-0.29663</td>
</tr>
<tr>
<td>0.091198</td>
<td>0.023161</td>
<td>0.091198</td>
<td>0.021608</td>
<td>-0.07606</td>
</tr>
<tr>
<td>-0.07606</td>
<td>0.021608</td>
<td>-0.07606</td>
<td>0.021434</td>
<td>0.021434</td>
</tr>
<tr>
<td>0.021434</td>
<td>0.009023</td>
<td>-0.00779</td>
<td>-0.00779</td>
<td>0.063186</td>
</tr>
<tr>
<td>0.063186</td>
<td>-0.0192</td>
<td>0.063186</td>
<td>-0.0192</td>
<td>0.011163</td>
</tr>
<tr>
<td>-0.0192</td>
<td>0.011163</td>
<td>-0.01024</td>
<td>-0.01024</td>
<td>0.006785</td>
</tr>
<tr>
<td>0.006785</td>
<td>-0.01672</td>
<td>0.006785</td>
<td>-0.01672</td>
<td></td>
</tr>
</tbody>
</table>

#### 4.1.4 Final Design

The flowchart in figure 4.3 describes our system and its subsystems.
Figure 4.3 Design Flowchart
4.2 Discussion

Speech activity detection (SAD) is an important step in most of the audio signal applications. Many features can be used for SAD with some trade-offs in the accuracy and efficiency.

The DBNs provide an ideal speech activity detection for both training with MFCC and LPCC features.

While training the DBN we were continuously changing the number of generative RBMs and their hidden layers until we got a good architecture that gave us good results.

We started training with only 500 samples and tested with 500 samples and we were gradually increasing the training dataset, we noticed that even when trained with small dataset the network performs really well.

The training dataset was represented with different noisy speech signals with different SNR values, and when tested against speech signals with high amount of noise it perfectly classified it. The network showed a small amount of error in the recognition of stationary noise.

The neural network’s testing error did not show a significant difference between training it with the MFCC and LPCC features. The only difference is in the time needed for the training, in the case of training with MFCC features the minimum error was reached at epoch 18 while it was at epoch 21 for training with LPCC features.

Since both features give an ideal SAD we will design the network with LPCC feature as it is easier in implementation.

As the result of using the speech activity detection as a pre-processing step the power consumption is reduced in the digital hearing circuit. This was accomplished because we don’t do any further processing and noise reduction for the input audio signal if it doesn’t contain speech signal.

The hearing aids will be designed according to the hearing characteristic of the patient. This is done in the frequency and amplitude shaping steps.
Chapter Five: Conclusion and Future Work

5.1 Conclusion

In this project we designed an efficient hearing aid that uses Speech activity detection (SAD) as a preprocessing step. The SAD step helped in reducing the power consumption and it also made it possible for the hearing aid to adapt to the changing environment without interference of the human being.

We used deep belief networks (DBN) to perform the SAD. The training dataset and the testing dataset are composed of clean speech, noisy speech and stationary noise. MFCC and LPCC features were used to represent the audio signals. Very accurate classification results were obtained.

Our proposed hearing aids will be fabricated according to the hearing losses and characteristics of the patient while maintaining the suitable size.

5.2 Future Work

To better the system and make it more intelligent we will increase the number of classes that will indicate the surrounding environments such as conferences, streets, offices or concerts, then according to the environment further processing for the audio signal will be done.

Another way to enhance it will be by separating the speech signal from the noise signal and apply different kind of filters and processing on each one.

We wish that we can also implement an artificial hearing aids that is able to learn the hearing characteristics of the human and enhance the audio signal accordingly.
References


Appendix A

A  MATLAB Codes

A.1  Test classification in DBN:

Clear
clc
res={};
more off;
addpath(genpath('DeepLearnToolboxGPU'));
addpath('DeeBNet');
% preparing the training and testing datasets
data = prepare_feat();
% validation set preparation
data.validationData=data.testData;
data.validationLabels=data.testLabels;
% classificative DBN
dbn=DBN('classifier');
% RBM1
rbmParams=RbmParameters(5,ValueType.binary);
rbmParams.samplingMethodType= SamplingClasses.SamplingMethodType.PCD;
rbmParams.performanceMethod='reconstruction';
rbmParams.maxEpoch=20;
dbn.addRBM(rbmParams);
% RBM2
rbmParams=RbmParameters(2,ValueType.binary);
rbmParams.samplingMethodType= SamplingClasses.SamplingMethodType.PCD;
rbmParams.maxEpoch=100;
rbmParams.rbmType=RbmType.discriminative;
rbmParams.performanceMethod='classification';
dbn.addRBM(rbmParams);
% train
ticID=tic;
dbn.train(data);
toc(ticID)
% test train
classNumber=dbn.getOutput(data.testData,'bySampling');
testErrorBeforeBP=sum(classNumber~=data.testLabels)/length(classNumber)
classNumber=dbn.getOutput(data.trainData,'bySampling');
trainErrorBeforeBP=sum(classNumber~=data.trainLabels)/length(classNumber)
% BP
ticID=tic;
dbn.backpropagation(data);
toc(ticID);
% test after BP
classNumber=dbn.getOutput(data.testData);
testErrorAfterBP=sum(classNumber~=data.testLabels)/length(classNumber)
classNumber=dbn.getOutput(data.trainData,'bySampling');
trainErrorAfterBP=sum(classNumber~=data.trainLabels)/length(classNumber)

A.2 Preparing the training and testing datasets

function [data] = prepare_feat()
% Creating an object to store train and test data and their labels
data=DataClasses.DataStore();
% Data value type is gaussian because the value can be considered a real
% value [0 +1]
data.valueType=ValueType.gaussian;
%dataFile=load('mfcc.mat');
dataFile=load('lpcc.mat');
data.trainData=dataFile.lpcc;
dataFile=load('training_labels.mat');
data.trainLabels=dataFile.training_labels;
dataFile=load('mfcc_test.mat');
data.testData=dataFile.mfcc_test;
dataFile=load('lpcc_test.mat');
data.testLabels=dataFile.lpcc_test;
dataFile=load('testing_labels.mat');
data.testLabels=dataFile.testing_labels;
end

A.3 Feature extraction functions

function lcep = lifter(cep,L)
    [N,D] = size(cep);
n = 0:D-1;
lift = 1 + (L/2)*sin(pi*n/L);
lcep = cep .* repmat(lift,N,1);
end

function fbank = msf_filterbank(nfilt,fs,lowfreq,highfreq,nfft)
    % compute points evenly spaced in mels
    lowmel = hz2mel(lowfreq);
    highmel = hz2mel(highfreq);
    melpoints = linspace(lowmel,highmel,nfilt+2);
    % our points are in Hz, but we use fft bins, so we have to convert from Hz to fft bin
    % number
    bin = 1+floor((nfft-1)*mel2hz(melpoints)/fs);
fbank = zeros(nfilt,nfft/2);
for j = 1:nfilt
    for i = bin(j):bin(j+1)
        fbank(j,i) = (i - bin(j))/(bin(j+1)-bin(j));
    end
    for i = bin(j+1):bin(j+2)
        fbank(j,i) = (bin(j+2)-i)/(bin(j+2)-bin(j+1));
    end
end

function hz = mel2hz(mel)
    hz = 700*(10.^(mel./2595) -1);
end

function mel = hz2mel(hz)
    mel = 2595*log10(1+hz./700);
end

function win_frames = msf_framesig(signal, frame_len, frame_step, winfunc)
    if size(signal,1) ~= 1
        signal = signal';
    end
    signal_len = length(signal);
    if signal_len <= frame_len % if very short frame, pad it to frame_len
        num_frames = 1;
    else
        num_frames = 1 + ceil((signal_len - frame_len)/frame_step);
    end
    padded_len = (num_frames-1)*frame_step + frame_len;
    % make sure signal is exactly divisible into N frames
    pad_signal = [signal, zeros(1,padded_len - signal_len)];

    % build array of indices
    indices = repmat(1:frame_len, num_frames, 1) + ... 
                repmat((0: frame_step: num_frames*frame_step-1)', 1, frame_len);
    frames = pad_signal(indices);

    win = repmat(winfunc(frame_len)', size(frames, 1), 1);
    % apply window
    win_frames = frames .* win;
end
function feat = msf_lsf(speech,fs,varargin)
    p = inputParser;
    addOptional(p,'winlen', 0.025,@(x)gt(x,0));
    addOptional(p,'winstep', 0.01, @(x)gt(x,0));
    addOptional(p,'order', 12, @(x)ge(x,1));
    addOptional(p,'preemph', 0, @(x)ge(x,0));
    parse(p,varargin{:});
    in = p.Results;

    frames = msf_framesig(speech,in.winlen*fs,in.winstep*fs,@(x)hamming(x));
    temp = lpc(frames',in.order);
    feat = zeros(size(temp,1),in.order);
    for i = 1:size(temp,1)
        feat(i,:) = poly2lsf(temp(i,:))';
    end
end

A.4 MFCC feature extraction

function mfccs = msf_mfcc(speech,fs,varargin)
    p = inputParser;
    addOptional(p,'winlen', 0.025,@(x)gt(x,0));
    addOptional(p,'winstep', 0.01, @(x)gt(x,0));
    addOptional(p,'nfilt', 26, @(x)ge(x,1));
    addOptional(p,'lowfreq', 0, @(x)ge(x,0));
    addOptional(p,'highfreq', fs/2, @(x)ge(x,0));
    addOptional(p,'nfft', 512, @(x)gt(x,0));
    addOptional(p,'ncep', 13, @(x)ge(x,1));
    addOptional(p,'liftercoeff', 22, @(x)ge(x,0));
    addOptional(p,'appendenergy',true, @(x)ismember(x,[true,false]));
    addOptional(p,'preemph', 0, @(x)ge(x,0));
    parse(p,varargin{:});
    in = p.Results;
    H = msf_filterbank(in.nfilt, fs, in.lowfreq, in.highfreq, in.nfft);
    pspec = msf_powspec(speech, fs, 'winlen', in.winlen, 'winstep', in.winstep, 'nfft', in.nfft);
    en = sum(pspec,2); % energy in each frame
    feat = dct(log(H*pspec'))';
    mfccs = lifter(feat(:,1:in.ncep), in.liftercoeff);
    if in.appendenergy
        mfccs(:,1) = log10(en);
    end
end
A.5 LPCC feature extraction

function feat = msf_lpc(speech,fs,varargin)
    p = inputParser;
    addOptional(p,'winlen', 0.035,@(x)gt(x,0));
    addOptional(p,'winstep', 0.01, @(x)gt(x,0));
    addOptional(p,'order', 12, @(x)ge(x,1));
    addOptional(p,'preemph', 0, @(x)ge(x,0));
    parse(p,varargin{:});
    in = p.Results;

    frames = msf_framesig(speech,in.winlen*fs,in.winstep*fs,@(x)hamming(x));
    feat = lpc(frames',in.order);
    feat = feat(:,2:end); % ignore leading ones
end

A.6 Hearing Aid

function y = hearingAidF(input,g,Psat,transitionV,newfile);
% y = hearingAidF(input,g,Psat,transitionV,newfile)
% Combines the filters of our digital hearing aid system.
% Returns a Matlab sound file of filtered signal.
% input - the input signal to the system. Should be a wave file.
% g - the maximum gain that will be applied to the signal
% Psat - the cut off power. The output power will not be higher than this
% transitionV - 4 element vector that has the values of where the gain changes
% to the next piecewise function
% newfile - desired name for the output sound file. It will be in .au format
[x,fs,nbits] = wavread(input);
xc = denoiseEm(x);                             % denoising filter
xf = applySkiSlope(xc,g,transitionV,fs);       % frequency shaping filter
y = powerCompress(xf, Psat,fs);               % amplitude shaping filter
x_length = length(x);
t=[0:1/fs:(x_length-1)/fs];
%soundsc(input, fs);
sound(y,fs);
auwrite(y,fs,nbits,'linear',newfile);
function y = denoiseEm(x);
% y = denoiseEm(x);
% method to denoise a given signal using wavelets
% x is the input Matlab sound file

%THR is the threshold, SORH is for soft or hard thresholding, KEEPAPP allows you to keep
%approximation coefficients
[thr,sorh,keepapp]=ddencmp('den', 'wv', x);

% returns a de-noised version xc of input signal x (our one-dimensional speech signal)
%obtained by wavelet coefficients thresholding using global positive threshold THR
%PERF0 and PERFL2 are L2-norm recovery and compression score in percentage.
[y, xwc, lxc, perf0, perfl2]=wdencmp('gbl', x, 'db3', 2, thr, sorh, keepapp);

function y = powerCompress(input, Psat,Fs);
% y = powerCompress(input, Psat,Fs)
% Takes in a a signal makes sure that the maximum power in any frequency
% is less than or equal to Psat. Also had some denoising capabilities, by
% zeroing out very low power frequencies.
% input - input Matlab sound file
% Psat - Saturation power
% FS - Sampling frequency of the input signal
x=input;
%x, Fs, Nb]=wavread(input);
len=Fs*0.1;
iter=floor(length(x)/len);
Plow=0.008;

for rg=0:1:iter;
    start=rg*len+1;
    en=rg*len+len;
    if rg*len+len>length(x)
        en=length(x);
    end
    clear signal X  X_pow Y_pow Y y z;
    signal=x(start:en);
    n = nextpow2(len);
    N = 2^n;
    X = fft(signal,N);
    X_phase=angle(X);                  % Save the old phase information
    X_pow = abs(X)/N;
    Y_pow = X_pow;

Y=zeros(N,1);
for k=0:N/2
    if Y_pow(k+1)<Plow              % Take out noise
        Y_pow(k+1)=0;
        Y_pow(N-k)=0;
    elseif Y_pow(k+1)>Psat         % Clip amplitudes higher than Psat
        Y_pow(k+1)=Psat;
        Y_pow(N-k)=Psat;
    end;
    Y(k+1) = Y_pow(k+1)*(cos(X_phase(k+1))+i*sin(X_phase(k+1)));
    Y(N-k) = Y_pow(N-k)*(cos(X_phase(N-k))+i*sin(X_phase(N-k)));
end;

y = real(ifft(Y,N));
z = y(1:en-start+1);
sig_out(start:en)=z;
end;

y = sig_out*2000;
%wavwrite(sig_out*1000,Fs,16,'output');