Continuous Speech Recognition Using Long Term Memory Cells

A thesis presented to
the faculty of
the Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Master of Science

Aby Abraham
December 2013
© 2013 Aby Abraham. All Rights Reserved.
This thesis titled
Continuous Speech Recognition Using Long Term Memory Cells

by

ABY ABRAHAM

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

Janusz Starzyk
Professor of Electrical Engineering and Computer Science

Dennis Irwin
Dean, Russ College of Engineering and Technology
ABSTRACT

ABRAHAM, ABY, M.S., December 2013, Electrical Engineering

Continuous Speech Recognition Using Long Term Memory Cells

Director of Thesis: Janusz Starzyk

The thesis proposes a continuous speech recognition model using neural network structure which was inspired by long term memory model of human cortex. Speech recognition model extracts and selects the finest representation of speech signal using mel-frequency cepstrum coefficients. The extracted features are fed to neural network with long term memory (LTM) cells which learns the sequence. The LTM cells have the capability to address three main issues of sequence learning including error tolerance, significance of elements and memory decaying which are used to tune the LTM model with parameters that depends on the environment it learns.

To validate the model, two datasets have been used - spoken English digits and spoken Arabic digits in speaker dependent mode. The parameters of LTM model have been optimized based on the environment.

The results show that the LTM model with fine tuning of parameters is 97% accurate in recognizing the spoken English digits datasets, and 99% accurate in recognizing spoken Arabic digits datasets.
DEDICATION

The thesis is dedicated to my parents without whom I would not have achieved anything in my life.
ACKNOWLEDGMENTS

First, I would like to offer my sincere thanks to Dr. Janusz Starzyk for being my advisor and helping me through all my research with his support and advice. He has been the source of encouragement guiding and mentoring me throughout the research. I would also like to dedicate my sincere thanks to Vu Anh Nguyen who helped me throughout the research period with the LTM model and helped me understand complex algorithm in the model. I would like to thank Dr. Savas Kaya, Dr. Wojciech Jadwisienczak and Dr. Martin J. Mohlenkamp for being a part of my thesis committee in spite of their busy schedule. I would like to thank my parents Rajan P Abraham and Lisa Mathew for all their unconditional love and trust they had in me. Next to my sisters Deepa Simon, Anu Anna Thomas and brother in laws Simon Baby and Thomas Jacob for all the love and support and motivation they have given me throughout my career. I would also like to thank my wonderful friends Jayakrishnan Muraleedharan Nair, Parthiban Rajan, Sai Goutham Koya, Kevin Peris, Pradeep Cheriyan, Mithun Roy, Renil Antony, Arun Murali and H inbound G for being with me during good and hard times and for all their support and motivation. I would take this opportunity to thank Mr. Bryan Jordan for all his help and support throughout my program in Ohio University.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Dedication</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>5</td>
</tr>
<tr>
<td>List of Tables</td>
<td>8</td>
</tr>
<tr>
<td>List of Figures</td>
<td>9</td>
</tr>
<tr>
<td>Chapter 1. Introduction</td>
<td>10</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>10</td>
</tr>
<tr>
<td>1.2 Research Goal</td>
<td>14</td>
</tr>
<tr>
<td>1.3 Thesis Organization</td>
<td>16</td>
</tr>
<tr>
<td>Chapter 2. Signal pre-processing</td>
<td>18</td>
</tr>
<tr>
<td>2.1 Feature Extraction</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Mel Frequency Cepstral Coefficient (MFCC) Algorithm</td>
<td>19</td>
</tr>
<tr>
<td>2.2.1 Framing and Windowing</td>
<td>22</td>
</tr>
<tr>
<td>2.2.2 Fast Fourier Transform (FFT)</td>
<td>26</td>
</tr>
<tr>
<td>2.2.3 Mel-scale filter bank</td>
<td>27</td>
</tr>
<tr>
<td>2.2.4 Discrete Cosine Transform</td>
<td>28</td>
</tr>
<tr>
<td>Chapter 3. Neural network structure for spatio temporal long term memory cell</td>
<td>30</td>
</tr>
<tr>
<td>3.1 Error Tolerance</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Significance of Elements</td>
<td>32</td>
</tr>
<tr>
<td>3.3 Activation Decay</td>
<td>33</td>
</tr>
<tr>
<td>3.4 LTM Cell Organization</td>
<td>33</td>
</tr>
<tr>
<td>3.4.1 LTM Structure</td>
<td>34</td>
</tr>
<tr>
<td>3.4.2 Error Tolerance</td>
<td>39</td>
</tr>
<tr>
<td>3.4.3 Significance Estimation</td>
<td>40</td>
</tr>
<tr>
<td>3.5 Storage Mechanism of LTM Cells</td>
<td>41</td>
</tr>
<tr>
<td>3.6 LTM Sequence Recognition Algorithm</td>
<td>41</td>
</tr>
<tr>
<td>3.7 Illustration of LTM Model with simple sequence</td>
<td>42</td>
</tr>
<tr>
<td>Chapter 4. Experiments with speech datasets</td>
<td>48</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Activation of neurons at $t=1$ when stored sequence is passed as input</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>Activation of neurons at $t=2$ when stored sequence is passed as input</td>
<td>44</td>
</tr>
<tr>
<td>3</td>
<td>Activation of neurons at $t=3$ when stored sequence is passed as input</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>Activation of neurons at $t=4$ when stored sequence is passed as input</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>Normalized value of activation of SN neurons</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>Activation of neurons at $t=1$ when sequence (ABDC) is passed as input</td>
<td>45</td>
</tr>
<tr>
<td>7</td>
<td>Activation of neurons at $t=2$ when sequence (ABDC) is passed as input</td>
<td>46</td>
</tr>
<tr>
<td>8</td>
<td>Activation of neurons at $t=3$ when sequence (ABDC) is passed as input</td>
<td>46</td>
</tr>
<tr>
<td>9</td>
<td>Activation of neurons at $t=4$ when sequence (ABDC) is passed as input</td>
<td>46</td>
</tr>
<tr>
<td>10</td>
<td>Normalized value of activation of SN neurons</td>
<td>47</td>
</tr>
<tr>
<td>11</td>
<td>Configuration Arabic speech data for MFCC extraction</td>
<td>48</td>
</tr>
<tr>
<td>12</td>
<td>Parameter range for LTM optimization</td>
<td>50</td>
</tr>
<tr>
<td>13</td>
<td>Validation results for spoken English datasets</td>
<td>50</td>
</tr>
<tr>
<td>14</td>
<td>Phonetic representation of Arabic digits</td>
<td>55</td>
</tr>
<tr>
<td>15</td>
<td>PA and SR of Arabic spoken digits</td>
<td>59</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Envelopes connecting formants of speech</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2</td>
<td>MFCC algorithm block diagram</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Framing and overlapping</td>
<td>23</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Original signal and windowed version</td>
<td>24</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Generalized Hamming window</td>
<td>25</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Effects of Hamming window</td>
<td>26</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Variation of Mel frequency with linear frequency</td>
<td>27</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Mel filter banks</td>
<td>28</td>
</tr>
<tr>
<td>Figure 9a</td>
<td>LTM Cell</td>
<td>34</td>
</tr>
<tr>
<td>Figure 9b</td>
<td>LTM Cell Structure</td>
<td>34</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Computing activation of primary neurons</td>
<td>36</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Computing activation of secondary neurons</td>
<td>37</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Secondary neurons update</td>
<td>39</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Sensitivity of PA and SR to varying eta in spoken English digits</td>
<td>51</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Sensitivity of PA and SR to varying gamma in spoken English digits</td>
<td>52</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Sensitivity of PA and SR to varying tau in spoken English digits</td>
<td>53</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Sensitivity of PA and SR to varying eta in spoken Arabic digits</td>
<td>56</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Sensitivity of PA and SR to varying gamma in spoken English digit</td>
<td>57</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Sensitivity of PA and SR to varying tau in spoken English digits</td>
<td>58</td>
</tr>
</tbody>
</table>
CHAPTER 1. INTRODUCTION

1.1 Background

Most natural skills of human being are trained from the time we were born. These training techniques are very complex yet they occur naturally without realizing the complexity of the process. Speech is such a skill, and it is the most natural mode of communication for people. The organs that control the speech of human being, such as the vocal tract and articulators, can be influenced by various factors from gender to the emotional state of the person [1]. Hence vocalization in different people can have variations in pitch volume, quality, pronunciations, articulations speed etc. Speech recognition becomes very complex when one wants to consider all these features together with the background noise and echoes associated with the environment.

Human brain is well trained and it can easily recognize this complex speech. We are so comfortable with speech, that instead of using conventional keyboard and mouse, we now try to interact with computers using speech [1]. Various engineering models were introduced in the field of speech recognition since 1950’s and have shown tremendous progress in this field. As a result of that, several applications were developed such as voice dialing for phones, interactive voice response for business purposes, voice translation services etc. Despite the numerous possibilities in this field, the technology has not been developed to recognize speech in par with the human brain.

Human brain is able to process speech information with the help of large number of parallel connected neurons and the synaptic weights associated with it. Different computational models were developed since 1940’s in order to produce brain-like speech
processing based on neural networks. Some of these models are based on artificial neural networks [1]. Neural networks have been very efficient in learning complex patterns or sequence recognition. The more training of the neurons the more pronounced and better developed will be the strength of interconnection weights associated with it.

An intelligent system is able to detect and reproduce a particular order in which an event occurs [2]. To do this an efficient method for sequence learning and detection is needed. Sequential learning method is one of the most dominant forms of learning in humans as well as animals [3]. Sequential learning is an important factor in various fields including planning, natural language processing, speech recognition, DNA sequencing etc. Various approaches to sequential learning are implemented based on the field to which it is applied. To improve the existing models we need to combine different existing techniques and develop more accurate algorithm for comparing the sequences.

To determine whether a particular sequence is accurate based on the conditions provided is one of the difficulties of sequence learning approach. Recognition process in sequence learning is made possible with training of different types of computational models with sequences [12].

As stated earlier all natural sequential data are mostly dynamic and hence temporal dependencies between the consecutive elements of a sequence should also be considered while recognizing a sequence. This temporal dependency is very crucial in human intelligence and hence temporal learning becomes one of the most important processes in neural network learning.
Spatio–temporal word is defined in both time and in space. Some systems, such as static systems, do not change its state with the change of time. A dynamic system however depends on the current input as well as the previous state of the system [10], [11]. These dynamic systems are called temporal processing systems. Spatio-temporal sequence learning, anticipation and prediction are components of human perception and intelligence [12]. Over the past decade there have been many models which focused on temporal sequence learning [12].

Recurrent neural networks (RNN) [13], [14] is one of the most dominant and effective methods in sequential data processing. With the help of the recurrent network connections in RNN, model can learn and calculate important dynamic information of the sequential actions of the data. The traditional RNN network training was based on back propagation through time (BPTT). Due to the gradient-descent algorithm in RNN, the error signals may not last long. As a result, the model was not sufficient for long time-lag sequences [15]. As a solution to this, another model was proposed in [15] called long short term memory (LSTM). In LSTM the error signals were able to withstand a long period of delay time with help of constant error carousel (CEC) by which it was successful in learning and prediction of longer sequences.

Wang and Arbib [2] introduced another method of temporal sequence learning. They proposed special types of neurons which were able to recognize sequences with similar properties. They also proposed another model in [16], which overcomes the drawback of storage capacity of the STM by introducing chunking mechanism.
In [10], the model used for natural learning of sequence was explained. The neural networks are arranged at different levels, so called hierarchical level structure in which, the winner neuron of each level is then passed on to the next level of the structure. Algorithm used to determine the winner is WTA (winner takes all). Compared to the old way of expressing time as an additional parameter within sequences, this model has indirectly represented time with influence on the sequence itself. This model also uses a method called prediction-mechanism in which the network learns either through one-shot learning or using an error in its own prediction. It has provided effective results for the four-level models explained in 10]; however, much more computational resources may be required for a larger level network model. The problem with the model is that exact match of sequence is required for the neuron to be considered as a winner which makes this model practically not that effective [11]. When the input sequence is somewhat distorted, the exact match is not obtained which will in turn consider the distorted sequence as a new sequence.

As an extension to the above model, the same authors in [11] proposed a LTM model which can be used for building neural networks. The model proposes a combined structure of short term memory (STM) and LTM cells for processing the inputs. The LTM cells are arranged hierarchically in several levels and input to each LTM is from a STM. The LTM cells are modeled in such a way that they select only certain features of the input sequence; hence, even if the inputs are distorted, the features of the inputs are needed for the LTM to distinguish the sequence. Also, the WTA algorithm decides the winner LTM in each level. The model is compared with the Hidden Markov Model
(HMM) for text recognition. The model can be used for various sequence-dependent learning and can have various applications in the real world.

To summarize, all the computational models proposed in literature are not strong enough to process multi-dimensional and real valued data such as speech, vision etc. Improvement of such models is important for the expansion of sensory based demonstrations and motor control of embodied intelligent systems.

1.2 Research Goal

Human perceptions and intelligence is based on spatio temporal sequences and speech signal is one of them. In this thesis a continuous speech recognition model is developed based on the neural network structure proposed in [17]. This model is based on spatio temporal learning and recognition and was inspired by long term memory of the human cortex. It is structured in such a way that it can process real valued and multidimensional sequences.

This model has much improved architecture than [10] and [11]. Three improvements to the current model are suggested. One of them is the introduction of error tolerance within the LTM and also between the LTM cells, which optimizes the recognition part. Another improvement is the addition of significance of elements within the LTM, which in turn improves the LTM activations. The third addition is the delay factor which helps the activation of a particular LTM for some time. This gives the LTM enough time to learn, and if input sequence is deviating away from stored sequence in LTM, the activation will start decreasing. Thus the LTM is able to remember as well as forget a sequence depending on the frequency of use and sequence importance.
Dynamically, its activation level can increase or decrease depending on the match of the stored and observed sequences. This avoids confusions in the activation of LTM and makes recognition more optimized.

The basic step of speech recognition is conversion of speech signals into features in digital form which represents the speech signals. In feature extraction process the dimensionality of the input speech need to be reduced; this is done by extracting specific features from the speech which vary from person to person. The peaks in a speech spectrum carry the identity of the speech signal. These peaks are called formants. Formants are extracted from speech signal and used for further analysis and processing. Speech signals are time-varying signals, which when analyzed for a small interval of time will provide stable information [7]. Hence, spectral analysis can be used to characterize speech signals. There are various methods or procedures for parametrically representing speech signal such as linear predictive analysis (LPC), Perceptual linear prediction (PLP) and Mel scale Cepstral analysis (MEL) etc.

Whenever we have two speech signals of the same word or sentence, the signal waveform will never be the same. It can vary depending on various features such as length, amplitude, background noise and sample rate. However, the perceptual information regarding the speech will remain constant in both speech signals [7]. This extraction of the perceptual information of speech signal is one of the basic steps of speech recognition system. Mel-frequency Cepstral coefficient, which is one of the most widely accepted methods is used in this thesis for analysis [8].
In this thesis the perceptual information from speech signals is extracted and fed to the LTM model. Each sequence can be stored into a single LTM cell. Whenever each word from a continuous speech is encountered, LTM which represents the particular word gets maximum activation. Hence as a continuous speech plays along we could see a series of LTM activations.

The research goal of thesis is to implement speech recognition model using the LTM cells and validate the model using spoken English datasets and spoken Arabic datasets which has utterances of digits as a single word. The scope of the thesis is that as the LTM model can be played against different length sequences, accurate recognition of words will in turn be implemented for continuous speech recognition. Experiment and analysis based on these datasets are performed to demonstrate accurate recognition of LTM model for each word.

1.3 Thesis Organization

This thesis is organized in to five chapters. Chapter 2 presents the details of the preprocessing steps for the speech recognition systems. In this chapter the feature extraction process using Mel-frequency Cepstral coefficient (MFCC) is explained in detail which include details about Mel-scale, Fast Fourier transform (FFT), framing and windowing, Mel-scaled bank filters and Cepstral coefficients.

Chapter 3 discusses the details of the neural network structure for LTM cell which is used for the speech recognition; this included the LTM cell organization, learning and sequence recognition algorithm. This chapter also explains how the LTM is able to process real valued and multidimensional sequence by addressing three critical problems
in sequential learning namely error tolerance, significance of the sequence of elements, and memory forgetting.

In Chapter 4 the merit of speech recognition model based on LTM is validated using two data sets, Spoken English digits and Spoken Arabic digits. These are introduced to the LTM cells after the preprocessing procedures. The details of the criteria which are used for analysis of data which include performance accuracy and separation ratio are explained in this chapter. Performance analysis of data which is done by varying different parameters of the LTM is also explained in this chapter.

Chapter 5 concludes the work done with the explanation of the main results obtained from the thesis.

Chapter 6 discusses the future work in the neural network model and methods for computation optimization.
2.1 Feature Extraction

In human body the speech production system has mainly three cavities which are nasal, oral and pharyngeal which together form the main acoustic filter. When we speak, the form and shape of the vocal and nasal tracts continuously changes with time, which creates time varying frequency responses. As air from the lungs travels through the tracts, the frequency spectrum is shaped by frequency selectivity of these tracts. The resonance frequency of the vocal tracts forms the formants. The shape and the dimension of the vocal tract have influence over the formants. For each individual the opening and closing of the cord is unique which in turn defines the feature and the individuality of the particular speech.

These features are major players in the speech recognition systems as they carry the identity of a particular word or a continuous sentence. Performance of a speech recognition system highly depends on how successfully the features are extracted from the speech signal. The features extracted should have the information which should be quantifiable, should not be changed by the background noise or any medium through which it transmits, and should be consistent with each speaker since in this thesis speaker dependent model is considered.

In speech processing, to extract the formants from the speech signal, we need to obtain an envelope which connects all the formants together and forms a smooth curve. This is shown in Figure 1.
Several signal processing algorithms are available to convert this speech waveform to parametric representation. Linear predictive analysis (LPC), Perceptual linear prediction (PLP) and Mel scale Cepstral analysis (MEL) are some of them. In this thesis Mel-frequency Cepstral analysis is used for extraction of features.

2.2 Mel Frequency Cepstral Coefficient (MFCC) Algorithm

MFCC is the most commonly used feature extraction technique in automatic speech recognition system. There are many other feature extraction algorithms such as; linear predictive coefficients, linear predictive cepstral coefficients and human factor cepstral coefficient. In this thesis MFCC is used because of its high accuracy, extraction of the most important features from the speech and more-over it is the most common and efficient algorithm used in speech recognition.
Each speech signal contains signals of varying frequencies. Research shows that human ear does not follow a linear scale. Each frequency observed in the input signal is converted to a special kind of spectrum, known as Mel-scale [7], [8]. To accomplish this in MFCC analysis, linear filters are spaced at low frequencies and logarithmical filters are spaced at higher frequencies [7].

Once the Mel frequency wrapping of the signal is done, this Mel spectrum is transformed into time. Log of the Mel spectrum provides us with information about the local spectral properties. These are converted into time domain by taking the discrete cosine transform (DCT). This gives the Mel frequency cepstral coefficients [7], [8]. The overall flow of data in MFCC processor is shown in the block diagram in Fig. 2.
The accuracy and the efficiency of feature extraction are possible only with proper preprocessing of the speech signals. The preprocessing steps include digital filtering and speech detection. Pre emphasis filters are used to reduce the high dynamic range due to the added additive noise in the digitized speech. This is achieved by a first order FIR high pass filter.The filter equation in time domain is given as

\[
Y[n] = x[n] - a \cdot x[n - 1]
\]  

Where \(x[n]\) is the sampled input and \(0.9 \leq a \leq 1.0\). In z domain the transfer function is given by
Usual voice characteristics produce low frequencies higher in amplitude than high frequencies. During pre-emphasis stage the filter will shape the voice signal to produce equal amplitude of lows and high frequencies before it is passed on to the next stage. The important fact is that the filter will boost high frequencies which will make information from higher formants available to further stages of speech recognition.

2.2.1 Framing and Windowing

Speech signal is a quasi-stationary in short interval of time. Hence time varying speech signals are analyzed in very short intervals of time in the range of 10-30 ms. This will help to apply block processing techniques such as FFT or DFT to analyze speech signal [26]. Hence most of the signals processing models are applied in this short interval of time during which signals are time invariant. To achieve this, signal is divided in to several frames and every single frame is analyzed in a short time rather than analyzing the complete signal at once. For each frame selected, a hamming window is applied which will obscure some information from the start and end part of the frame. In order to avoid this we use overlapping of the frames. Overlapping size is called hop size. The hop size will be 1/3 to 1/2 of the frame size. Overlapping will help in adding the information at the beginning and at the end of the window to the extracted features. Fig.3 shows overlapping windows applied to the signal.

\[ H[Z] = (1 - \alpha) \cdot (z^{-1}), \quad 0.9 \leq \alpha \leq 1.0 \]
While implementing windowing to a particular frame it is understood as multiplication of the window function by the signal in the frame to get a weighted value of the signal. The window to be selected depends on various other conditions. In this thesis we have used hamming window. While multiplying with the window function a convolution of the Fourier transform and the speech signal is obtained and more over it reduces the sudden change of amplitude at end points by attenuating the amplitude at both end of the extraction interval. Fig.4 shows how a window function is applied to the signal to obtain the weighted value of signal.
Figure 4: Original signal and windowed version
Figure 5: Generalized hamming window

Hamming window is the most commonly used for speech processing systems and it is defined as

$$W_H(n) = 0.54 - 0.46 \cos \left( \frac{2\pi n}{N-1} \right), \quad 0 \leq n \leq N - 1$$  \hspace{1cm} (3)$$

Where $N$ = total number of samples per frame.
2.2.2 Fast Fourier Transform (FFT)

During spectral analysis we can see that different tones in a speech signal matches different energy distribution in frequency domain. Hence in order to get magnitude of frequency response of each frame we apply Fast Fourier transform to each frame.

When FFT is applied to a particular frame it is based on the assumption that the signal confined within the frame is periodic and continuous. FFT can still be applied to the frame but it might give adverse effects in the frequency response due to the discontinuity at beginning and end of the frame. As explained earlier this can be avoided by using hamming window to multiply each frame and thereby increasing the continuity at the starting and the end points. The Fig. 6 shows how the effects of multiplying the speech signal with a hamming window.

![Figure 6: Effects of hamming window](image)
Figure 6 shows a sine wave with some noise is generated as the signal. We can see that the multiplying the signal with the hamming window increases the continuity at the starting and end point of the frequency response and as a results the peaks of the frequency response are more distinct and sharp.

2.2.3 Mel-scale filter bank

Each frequency measured in Hz is converted to corresponding Mel-scale. For a given frequency \( f \) (Hz), Mel-scale can be calculated using the following formula.

\[
\text{Mel} (f) = 2595 \times \log_{10}(1 + f/700)
\]  

(4)

Figure 7: Variation of Mel frequency with linear frequency [28].
In order to obtain the desired spectrum the magnitude frequency response is multiplied by a set of 40 triangular band pass filters. These are called Mel Filter banks. They are designed to reproduce similar band pass filtering occurring in the auditory systems. These filters are positioned in such a way that they are linear from 0 Hz to 1 kHz and beyond that they are logarithmic [24].

As shown in Fig. 8, Mel filter banks are created by placing triangular band pass filters with an overlap of 50%. These triangular band pass filters are generated with Mel frequency as the center of the triangle.

**2.2.4 Discrete Cosine Transform**

Discrete cosine transform is applied to the log of energy obtained from the filter banks to get the information in the time domain. The result is called the Mel frequency
cepstral coefficients. The cepstral illustration of speech signal will offer good information about the local spectral properties of the signal for given period of time. Comparing to Inverse Fourier transform (IFFT) DCT is preferred because it is less complex and more efficient.

MFCC coefficients are calculated using the formula below:

$$C_n = \sum_{k=1}^{K} (\log(S_k))[n(k - 0.5)\pi/K]$$

where $K$ represents the number of MFCC coefficients extracted, 

$n = 1, 2, 3, ..., K$

$S_k = $ energy obtained from filter banks.

The first coefficient, $C_0$ is not usually considered since it contains only the power of the signal. Zero$^{th}$ coefficients can be considered as group of average energies of each frequency bands in the signal which does not have any speaker related information, but analysis and experiments in [25] shows that energy information is useful in cepstral analysis for speech recognition models. Some researchers also use the delta and delta-delta values of the coefficients which are the first and second derivatives of the MFCC coefficients, for more improved performance of speech recognition systems. Thirteen MFCC coefficients are often used by most of the researches and so in this thesis also thirteen MFCC coefficients are extracted from the speech signals which are used to train and test the LTM cells.
CHAPTER 3. NEURAL NETWORK STRUCTURE FOR SPATIO TEMPORAL LONG TERM MEMORY CELL

As stated in Chapter 1, speech recognition model in this thesis is based on the neural network model developed by Vu Anh Nguyen, Janusz A. Starzyk, Wooi –Boon Goh and Daniel Jachyra and presented in the paper titled “Neural Network Structure for Spatio-temporal Long-term Memory” [17]. This model is an improvement of their earlier models [10],[11] in a way that it has more advanced ways of dealing with real valued and multi-dimensional data.

The earlier model in [10] has dealt with many concerns associated with LTM based sequential memory including competition, anticipation and one shot learning. The LTM which stores the sequence was arranged in hierarchical model similar to the cortical mini columns inside of a human brain [1]. This architecture helped to learn complex sequences in a natural way. The network uses feedback connections similar to [19] for anticipating the next element of the sequence and, learning of the given sequence is activated only if the prediction is incorrect. The measure of the complexity of the sequence learned by the model depends upon the separate subsequences which should be less than the memory volume delivered by the hierarchical structure.

In their model [11] authors improved model from [10] with the addition of a similarity matching mechanism. This mechanism had error tolerance for variations in test sequences including delayed sequences, order mismatch and also the variation of the start and end of the sequences. The efficiency of the learning mechanism by the network has
been improved since learning mechanism is initiated only if there is a mismatch between the test sequence and the stored sequence.

The model used in this thesis was developed in [17] and preserves all the features of models from [10] and [11] with addition of certain features such as error tolerance, significance of elements and activation decay mechanisms to improve its ability to perform more accurate recognition of real valued multi-dimensional data. The three main contributions to this model is error tolerance within the LTM cell, significance factor between the elements in LTM cell and also addition of activation decay mechanism in the LTM hierarchy.

3.1 Error Tolerance

There are two types of error tolerances that can be analyzed in a sequential learning mechanism. One is inter-element level and other one is intra-element level. The inter-element type error is due to the various uncertainties between the successive elements of the input sequence whereas the intra-element is the alterations in the content of the input. These types of error were not dealt with in the previous models. As the input sequences are multidimensional and varying continuously these errors become more complex.

According to [17], with the knowledge of statistical spatio temporal differences, the intra-element error tolerances can be estimated for content of each element. The advantage of this analysis is the fact that the tolerance estimation from a single sequence can be used to train the neural network and can recognize test sequences with less error. Even though the error tolerance are supposed to be calculated with the statistical analysis
of different samples of a particular sequence there are many situations in which a particular system has to function after a single observation or learning of a sequence. For instance a robot should learn a particular path after traversing through it for the first time [18]. In another example, an agent in an episodic memory organization [20] will be able to learn sequence of events after a single observation with this method of intra element tolerance calculation.

In [11] the inter-element tolerance is used by the sequence recognition algorithm, whenever a test sequence is encountered. While testing the LTM cells, compare the test sequence with the ideal one and compete between each other to be the winner. The activation of the LTM cells varies depending on the variation between the elements. The maximum activation of the LTM cell is obtained only for the stored sequence. Any variation from the stored sequence can cause a decrease in the activation. The maximum activation of LTM is analytically derived and used to normalize the activation. This method enables the testing of different length sequences against the stored sequence.

3.2 Significance of Elements

Significance is another important feature of the LTM model presented in [17]. In a computational model an agent will try to improve his recognition process by picking up only the prominent subset of input data. In this model significance of each feature is analyzed and assigned based on the statistical variation of features depending on the specific applications. Significance analysis assesses the significance of each element which is in contrast to previous models [10], [11] in which all elements are given equal significance. The significance analysis helps the LTM cells to revolve around the highly
significant elements of a particular time-domain sequence. Significance of individual feature change in time depending on their variability. The activation of the LTM cells is now modulated with the valued significance of each element. Other measure of significance can be used depending on application. For instance significant elements of the observed scene can be used using LTM cells for scene recognition. In speech recognition, significance can be assigned to entire words or grammatical elements of a complex sentence.

3.3 Activation Decay

Addition of the activation decay mechanism in the LTM hierarchy is to sustain the activation for sufficient period of time for the LTM to build time association in the networks and also to predict. Another advantage of the activation decay mechanism is that as the sequence deviates from the LTM cell, the activation should decay rapidly to avoid any confusion in sequence recognition. The importance of the memory activation decay for sequential neural networks was discussed in [10], [11].

3.4 LTM Cell Organization

This section explains how LTM is structured to store and recognize spatio temporal sequences.
3.4.1 LTM Structure

Fig. 9a represents the basic block diagram of a LTM cell. The LTM takes test sequence as the input and gives graded match signal.

The sequential neural network used for the sequence recognition is interconnection of several LTM Cells. Each LTM cells is structured in such a way to store a sequence so that whenever a test sequence or matching sequence is encountered the closeness to the stored sequence is determined with a graded signal [17]. The entire network topology as shown in Fig. 9b can be divided into 4 different layers. The input layer, primary layer, intermediate layer and the secondary layer.
3.4.1.1 Input Layer

All the inputs to the neural network are passed through the input layer. These inputs can deliver information about the specific environment the neural network is dealing with; these can be information from any sensory units or even can be signals from various LTM cells which are at the lower level in the network structure. As mentioned in chapter 2, the input layer of speech recognition model explained in this thesis comprises of thirteen MFCC coefficients which are extracted from speech signals. As noted in [17] at a particular time \( t \) the input vector delivered to the network can be

\[
I(t) = \{ I_i(t) \mid i = 1 \ldots F \} \tag{6}
\]

where \( F \) is the total number of inputs.

3.4.1.2 Primary Layer

Primary layer consists of primary neurons which are used to compute similarity factor between the trained sequences along with input sequences from input layer. The trained sequences are stored as synaptic weights between connections of input layer and the primary layer [17].

These synaptic weights are represented in [17] as

\[
W = \{ w_{ij} \mid i = 1 \ldots F, \ j = 1 \ldots L \} \tag{7}
\]

Where \( F \) is the total number of input neurons and \( L \) is the total number of primary neuron.

Pattern recognizers such as a multi-layer perceptron are employed by the primary layer so as to calculate the above mentioned similarity factor. Radial Basis Function (RBF) is used in this model in order to perform this function [17].
In [17] the output of the mth primary neuron at a time step $t$ is denoted as $y_m^{PN}$.

The computation of $y_m^{PN}$ is shown in Figure 10.

\[
y_m^{PN}(t) = \exp \left[ -\frac{1}{F} \sum_{i=1}^{F} \left( \frac{w_{im} - l_i(t)}{\delta_{im}} \right)^2 \right]
\]

(8)

where $\delta_{im}$ is defined in [17] as the tolerance which represents the statistical variation of each element in a sequence.

As shown in figure the primary neuron (represented as $R_m$) calculates the output from the primary layer $y_m^{PN}(t)$ and passes it to the next level which is the intermediate layer. Each neuron computes its updates concurrently [17] such that LTM recognition can be implemented in hardware cells for fast concurrent processing.
3.4.1.3 Intermediate Layer

The intermediate layer consists of L number of intermediate neurons. As Nguyen explains in [17] output of the m<sup>th</sup> neuron of the intermediate layer is the sum of output of m<sup>th</sup> primary neuron at time step t which is calculated using (8) and the (m-1) secondary neuron at previous time step (t -1) (which is explained in the next section). The significance factor (Φ) (significance estimation is explained in the later section) is added as weighted partner of the output of the primary layer. Intermediate neurons are also updated concurrently [17].

The output of intermediate layer which is denoted as $y_m^H(t)$ is given as

$$y_m^H(t) = [\Phi_m y_m^{PN}(t) + \hat{y}_{m-1}^{SN}(t)]^+ \quad (9)$$

where $\hat{y}_{j}^{SN}(t)$ is given by
\[ \hat{y}_j^{SN}(t) = f_j \left( y_j^{SN}(t-1) \right) \forall j \] (10)

\( \hat{y}_j^{SN}(t) \) represents the secondary neuron activation of the \( m^{th} \) neuron in the secondary layer after the decaying function \( f_j(\cdot) \) \( (j = 1 \ldots L) \) is applied, and where \( [x]^+ = x \) if \( x \geq 0 \) and 0 otherwise.

\[ f_j(x) = x - \gamma \varnothing_{j+1}, (j = 1 \ldots L, \gamma \in (0,1]) \] (11)

In which \( \gamma \) is the decaying factor.

Equation (11) is decaying function used in [17], unlike non-linear decaying model which is used in similar to psychological phenomenon, linear decaying behavior is implemented in this model.

3.4.1.4 Secondary Layer

Unlike primary and intermediate layer secondary layer neurons updates are computed incrementally. The \( m^{th} \) secondary neuron update at a time step \( t \) is denoted by \( y_m^{SN}(t) \).

\[ y_m^{SN}(t) = \max\{\hat{y}_m^{SN}(t), y_m^H(t), y_{m-1}^{SN}(t)\} \] (12)

- \( \max \{\cdot\} \) represents the point wise maximum function.
- \( \hat{y}_m^{SN}(t)_{t=0} = 0 \). In which \( \gamma \) is the decaying factor.
Figure 12: Secondary Neuron Update

As shown in the equation (12) secondary neuron update is the result of the maximum function of three different signals.

1. The decayed activation from the previous step $\hat{y}_m^{SN}(t)$ which is calculated by applying decay function in equation (11).

2. The activation of the $m^{th}$ intermediate neuron $y_m^H(t)$ which was calculated after attaining the signal from the $m^{th}$ primary neuron.

3. Matching grade between the presented signal and the subsequence up to $(m-1)$ elements of the stored sequence $y_{m-1}^{SN}(t)$.

3.4.2 Error Tolerance

Local variations of features of the elements are used to estimate the uncertainty measure of the system. In order to normalize the matching between the LTM cell and input vector uncertainty measured is used as the tolerance value $\Delta$. This approach is very much suitable for spatio temporal sequences since the local variation of features will vary with time, which in turn will give more information for uncertainty analysis.
3.4.3 Significance Estimation

The significance estimation is based on statistical analysis of the features of each element. In a given LTM cell the mean and the standard deviation of the \( i \)th feature of an element is denoted as \( \mu_i \) and \( \sigma_i \) [17]. The mean and standard deviation can be calculated based on the following equations [17]

\[
\mu_i = \frac{1}{L} \sum_{j=1}^{L} w_{ij}
\]

\[
\sigma_i = \sqrt{\frac{1}{L-1} \sum_{j=1}^{L} (w_{ij} - \mu_i)^2}
\]

where \( \mu_i \) is the mean of the \( i \)th feature

\( \sigma_i \) is the standard deviation of the \( i \)th feature.

\( w_{ij} \) is the synaptic weights of the stored sequence.

\( L \) is the total length of the sequence.

Significance estimation at feature level is estimated as denoted by equation [17]

\[
r_{im} = 1 - \exp\left\{ - \frac{(w_{im} - \mu_i)^2}{\eta \sigma_i^2} \right\}, m = 1, \ldots, L
\]

In which \( \eta \) is the tuning parameter.

Significance estimation of LTM cell at element level is computed as shown below [17]

\[
\phi_{m} = \sum_{k=1}^{F} \frac{r_{km}^2}{F}, m = 1, \ldots, L
\]

From equation (15) we could see that feature level significance \( r_{im} \) is an element of \( \{0, 1\} \) for all values of \( i \) and \( j \) hence \( \phi_m \) is an element of \( \{0, 1\} \) for all values of \( j \) [17].
From equation (16) it is seen that higher significance are assigned to elements which are statistically different from mean value and low significance is assigned to elements whose value are closer to the mean values [17].

3.5 Storage Mechanism of LTM Cells

Whenever an input signal is given to the LTM Cells, each LTM cell competes among them to become best match for the given input signal. WTA (Winner Take All) algorithm is used to determine the winner [17]. If the match of the winner LTM is less than a threshold value, the winner is considered as the best matching sequence for the given input.

If value is above the threshold, a learning signal is activated which in turn results in learning a new sequence [17]. A new LTM cells is dedicated to learn and input sequence is stored as synaptic weights W between the input and the primary layer as explained in the previous sections.

This mechanism is called one shot learning. It is used in many neural networks to increase the learning effectiveness of the system. The threshold value which triggers the learning signal is set by itself in an intelligent system by its interaction with the environment [17]. In our model the learning rate of the model is set high so each sequence is stored in a separate LTM Cell.

3.6 LTM Sequence Recognition Algorithm

In [17], algorithm for the sequence recognition of model used in the thesis is explained in detail. In [17] it is explained that in model, each input of the testing sequence is fed to LTM cell at which it compares the test sequence with its stored
sequence and gives an output which will be a grade of degree of match between input and the LTM cell. The winner LTM is determined by the WTA algorithm as explained in the previous section.

In [17] sequence recognition algorithm introduces delay factor and equivalent counters, which maintain activation of secondary neurons for further more steps. The disturbances and delay of the inputs can be compensated with the addition of delay factor to the model.

3.7 Illustration of LTM Model with simple sequence.

The working of LTM model is explained with an example of simple sequence recognition. Consider a simple sequence ABCD which has a length of four. Each element in the sequence contains two features. A has features as \{0.2, 0.8\}, B has features as \{0.6, 0.4\}, C has features as \{0.4, 0.6\} and D has features as \{0.8, 0.2\}. The LTM organization contains four primary, intermediate and secondary neurons and two inputs. Length of sequence \(L = 4\) and no of features \(F = 2\). The parameters used for training are delay factor tau (\(\tau = 4\)), significance factor eta (\(\eta = 2\)) and decay factor gamma (\(\gamma = 0.2\)). The significance factor component is estimated as \(\phi_1 = 0.49, \phi_2 = 0.072, \phi_3 = 0.072, \phi_4 = 0.49\). Maximum activation of the LTM is obtained when the stored sequence is played against the LTM. The stored sequence here is ABCD. The activation of each primary neuron, intermediate neuron and secondary neurons when stored sequence ABCD is given as input is as shown below. \(m\) represents the position of neuron.
Table 1: Activation of all neurons at t = 1 when stored sequence is passed as input

<table>
<thead>
<tr>
<th>m</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN Activation</td>
<td>0.4908</td>
<td>0.0482</td>
<td>0.0201</td>
<td>0.0276</td>
</tr>
<tr>
<td>IN Activation</td>
<td>0.3926</td>
<td>0.03379</td>
<td>0.005673</td>
<td>0</td>
</tr>
<tr>
<td>SN Activation</td>
<td>0.3926</td>
<td>0.3926</td>
<td>0.3926</td>
<td>0.3926</td>
</tr>
</tbody>
</table>

As shown in the table 1 first column the IN activation is less than the PN activation because as per equation (9) it is the sum of PN weighted by the significance degree of component and the decayed activation of previous SN at previous time step. Since $y_0^{SN}(t) = 0$ by convention, the IN activation of first neuron at $t = 1$ is less than the PN activation of first neuron.

In table 3 column 2 we can see that the PN activation is less than the IN activation. This is because the sum of decayed SN activation of same neuron at previous time step ($t = 2$) and weighted significance of PN is greater than the PN activation. The SN activation as shown in equation (12) is the maximum value of decayed activation of current neuron, current activation of intermediate neuron and the SN activation of the previous neuron. Depending on how the input deviates from the stored sequence the maximum value switches between the three which is selected as the SN activation.

For instance consider table 3 column 2. To calculate the SN activation of $m = 2$ we have from (12) $y_m^{SN}(t) = \max\{y_m^{SN}(t), y_m^H(t), y_{m-1}^{SN}(t)\}$ $y_2^{SN}(t)$ is calculated as decayed activation of SN at $m = 2$ at the previous time step. Decay function is given by equation (11).
\[ \hat{y}_2^{SN}(3) = f_j(y_2^{SN}(1)) = f_j(y_2^{SN}(3 - 1)) = y_2^{SN}(2) - \gamma \Phi_{2+1} \]

\[ \hat{y}_2^{SN}(3) = (0.4504) - ((0.2) \times 0.072) = 0.4360 \]  

(a)

other values in equation (12) are obtained from table 3

\( y_2^{IN}(3) = 0.4120 \)  

(b)

\[ y_2^{SN}(3) = y_1^{SN}(3) = 0.3637. \]  

(c)

Hence SN activation of \( m = 2 \) at \( t = 3 \) is \( \max\{\text{(a)}, \text{(b)}, \text{(c)}\} = 0.4360 \) (as shown in table 3)

| Table 2: Activation of neurons at \( t = 2 \) when stored sequence is passed as input |
|-----------------|-----|-----|-----|-----|
| \( m \)         | 1   | 2   | 3   | 4   |
| PN Activation   | 0.3277 | 0.0722 | 0.0524 | 0.1367 |
| IN Activation   | 0.2295 | 0.4504 | 0.4307 | 0.4312 |
| SN Activation   | 0.3782 | 0.4504 | 0.4504 | 0.4504 |

| Table 3: Activation of neurons at \( t = 3 \) when stored sequence is passed as input |
|-----------------|-----|-----|-----|-----|
| \( m \)         | 1   | 2   | 3   | 4   |
| PN Activation   | 0.0975 | 0.04824 | 0.0722 | 0.3565 |
| IN Activation   | 0   | 0.4120 | 0.5082 | 0.7089 |
| SN Activation   | 0.3637 | 0.4360 | 0.5082 | 0.7089 |
Table 4: Activation of neurons at $t = 4$ when stored sequence is passed as input

<table>
<thead>
<tr>
<th>$m$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN Activation</td>
<td>0.0129</td>
<td>0.01436</td>
<td>0.05249</td>
<td>0.4908</td>
</tr>
<tr>
<td>IN Activation</td>
<td>0</td>
<td>0.3636</td>
<td>0.4740</td>
<td>0.9009</td>
</tr>
<tr>
<td>SN Activation</td>
<td>0.3493</td>
<td>0.4215</td>
<td>0.4740</td>
<td>0.9009</td>
</tr>
</tbody>
</table>

Table 5: Normalized value of final activations of SN neuron for (ABCD)

| SN activation(Normalized) | 0.43 | 0.5  | 0.78 | 1    |

We can see that the highest activation of the LTM is obtained at 4$^{th}$ secondary neuron as 0.9009. When a test sequence with a distorted input is passed on to the LTM activation will be less than the activation of stored sequence. When a sequence ABDC with elements swapped from stored sequence, the LTM model the activations is as shown below

Table 6: Activation of neurons at $t = 1$ when sequence (ABDC) is passed as input

<table>
<thead>
<tr>
<th>$m$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN Activation</td>
<td>0.4904</td>
<td>0.0482</td>
<td>0.02012</td>
<td>0.0276</td>
</tr>
<tr>
<td>IN Activation</td>
<td>0.3926</td>
<td>0.03379</td>
<td>0.005673</td>
<td>0</td>
</tr>
<tr>
<td>SN Activation</td>
<td>0.3926</td>
<td>0.3926</td>
<td>0.3926</td>
<td>0.3926</td>
</tr>
</tbody>
</table>
Table 7: Activation of neurons at $t = 2$ when sequence (ABDC) is passed as input

<table>
<thead>
<tr>
<th>$m$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN Activation</td>
<td>0.0975</td>
<td>0.0482</td>
<td>0.0722</td>
<td>0.3565</td>
</tr>
<tr>
<td>IN Activation</td>
<td>0</td>
<td>0.4264</td>
<td>0.4504</td>
<td>0.6510</td>
</tr>
<tr>
<td>SN Activation</td>
<td>0.3782</td>
<td>0.4264</td>
<td>0.4504</td>
<td>0.6510</td>
</tr>
</tbody>
</table>

Table 8: Activation of neuron at $t = 3$ when sequence (ABDC) is passed as input

<table>
<thead>
<tr>
<th>$m$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN Activation</td>
<td>0.0129</td>
<td>0.0143</td>
<td>0.0524</td>
<td>0.4908</td>
</tr>
<tr>
<td>IN Activation</td>
<td>0</td>
<td>0.3781</td>
<td>0.4645</td>
<td>0.8431</td>
</tr>
<tr>
<td>SN Activation</td>
<td>0.3637</td>
<td>0.4120</td>
<td>0.4645</td>
<td>0.8431</td>
</tr>
</tbody>
</table>

Table 9: Activation of neurons at $t = 4$ when sequence (ABDC) is passed as input

<table>
<thead>
<tr>
<th>$m$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN Activation</td>
<td>0.3277</td>
<td>0.0722</td>
<td>0.0524</td>
<td>0.1367</td>
</tr>
<tr>
<td>IN Activation</td>
<td>0.2295</td>
<td>0.4215</td>
<td>0.4500</td>
<td>0.5030</td>
</tr>
<tr>
<td>SN Activation</td>
<td>0.3493</td>
<td>0.4215</td>
<td>0.4500</td>
<td>0.8431</td>
</tr>
</tbody>
</table>
Table 10: Normalized value of final activations of SN neuron for (ABDC)

| SN activation (Normalized) | 0.43 | 0.722 | 0.9380 | 0.9380 |

From above tables we can see that as the input sequence is distorted the activation of secondary neuron which is the output of the LTM decreased to 0.8431. Hence as the distortion in input sequence increases the activation of LTM decreases and maximum activation occurs only for the stored sequence.
CHAPTER 4. EXPERIMENTS WITH SPEECH DATASETS

In this thesis, experiments and analysis have been performed on two different datasets. As stated earlier all the analysis has been performed with speaker dependent mode. In both databases each recording has one word per recording and it is an isolated word. The first database consists of spoken English digits from 1-10 recorded by 5 different people. Each digit was spoken 30 times by the each person. Hence a total of 1500 data are available for training and testing purposes. This dataset was collected using a normal voice recorder and is used only for validation purposes.

The Arabic dataset for the experiment was collected by the Laboratory of Automatic and Signals, University of Badgi-Mokhtar Annaba Algeria. It consists of 88 speakers with a total of 8800 utterances. Each speaker has repeated each digit from 0 – 9 10 times. (10 digits x 10 Repetitions x 88 Speakers). Each of the spoken digits is time series of 13 MFCC coefficients which are extracted as explained in the Section 2. Each of the 13 MFCC coefficients are computed based on the following conditions as shown in table 11 below.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Arabic Spoken Digits database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling rate</td>
<td>11025 Hz, 16 bits</td>
</tr>
<tr>
<td>Window applied</td>
<td>Hamming</td>
</tr>
<tr>
<td>Filter Pre-emphasized</td>
<td>Value of α as 0.97 so H[z] = 1 - 0.97z⁻¹</td>
</tr>
</tbody>
</table>
Each of the above said sequence data is stored in separate LTM cell with a tag which represents the digit. The parameters of LTM model need to be optimized in order to obtain expected results. The parameters include delay factor (τ), significance factor (η) and decay factor (γ). The local standard deviation was calculated with a window size τψ.

For the quantitative analysis of the LTM, two different criteria were taken in to consideration,

1. Prediction Accuracy (PA),
2. Separation Ratio (SR).

PA is defined as number of correct prediction of test sequences divided by the total number of sequences [17]. Each correct recognition is evaluated based on the tag associated with each LTM. If winning LTM obtained has same tag as the given test sequence given it is considered as a correct recognition,

\[
PA = \frac{\text{Total number of correct recognitions by network}}{\text{Total number of test sequences presented to network}}
\]

SR is defined as the ratio between the activation of best and second best LTM whenever network predicts accurately [17]. This measurement will provide us with information about strength of the predictions made,

\[
SR = \frac{\text{Activation of the best LTM}}{\text{Activation of second best LTM}}
\]

To find optimum parameters for environments, two of the optimal parameters were fixed while varying the other one. By varying each parameter PA and SR of each value was calculated and plotted.

Table 12 shows the range in which the parameters are varied.
Table 12: Parameter range for LTM optimization.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay factor (τ)</td>
<td>[0, 2, 4, …… 20]</td>
</tr>
<tr>
<td>Significance factor (η)</td>
<td>[0, 2^n, 2^i, 2^j, …… 2^n]</td>
</tr>
<tr>
<td>Decay factor (γ)</td>
<td>[0.0, 0.1, 0.2, 0.3 … 1.0]</td>
</tr>
</tbody>
</table>

4.1 Analysis of Spoken English Digits

Analysis of spoken English digits was performed using the LTM model. For validation purposes test was conducted by selecting particular number of classes at a time. During validation test of the Spoken English Digits, it was found that as the number of classes in a data set increases the prediction accuracy of the model starts decreasing. In this dataset total number of classes is 10. Table 13 shows the results of validation test for different number of classes.

Table 13: Validation results of Spoken English Dataset.

<table>
<thead>
<tr>
<th>Number of Classes</th>
<th>PA</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>#classes(#run / # training instances)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 (30/90)</td>
<td>1</td>
<td>1.2252</td>
</tr>
<tr>
<td>5 (30/150)</td>
<td>1</td>
<td>1.150206</td>
</tr>
<tr>
<td>8 (30/240)</td>
<td>0.9985</td>
<td>1.143071</td>
</tr>
<tr>
<td>10 (30/300)</td>
<td>0.9667</td>
<td>1.10152</td>
</tr>
</tbody>
</table>

Each dataset environment need to be tuned to LTMs optimum parameter to obtain efficient results. In order to find the optimum parameter for a selected number of classes,
two of three parameters are fixed and third parameter is varied. The average results for a selected class of 10 are plotted from Fig. 13 – Fig.19.

**Figure 13: Sensitivity of PA and SR to varying $\eta$ for spoken digits**

From Fig.13 we could see that there is an improvement in PA and SR with increase in the significance factor ($\eta$).
From Fig. 14 shows that PA starts decreasing after a 0.4 of gamma (γ). SR significantly improves at value of 1. But in such a case the PA will be very low which will result in weak decision making of the neural network.
Fig. 15 shows that the increase in delay after particular value can reduce the accuracy rate of model for spoken English digits database. Hence from the performance analysis, for optimum performance of spoken English database, the significance factor $\eta > 2^2$, decay rate gamma ($\gamma$) should be between $0.1 < \gamma < 0.3$ and delay factor should be $2 < \tau < 3$. 
4.2 Analysis of Spoken Arabic Digits

The Arabic is a semantic language which is fifth most widely used language in the whole world. It has its own scripts and characters and it is entirely different from English and other European languages. Arabic language has 34 phonemes. Phonemes are defined as the smallest element of a speech unit. They determine how words and sentences differ in their meaning [23].

In the 34 phonemes 6 of them are basic vowels and 28 are consonants. The phonemes in Arabic can be classified into mainly two classes which are pharyngeal and emphatic phonemes. This type of classification can be observed only in semantic languages such as Hebrew [23].

A vowel must be present in each Arabic syllable. The Arabic syllables can take the form of CV, CVC, or CVCC of which C represent consonant and V represent vowels. Short syllables are of the form CV and others are grouped as long syllables. These can be again grouped in to open and closed syllables in which open syllable ends with a vowel and a closed syllable ends with a consonant. From this classification of syllables it is clear that that each Arabic syllable will start only with a consonant [24].

All Arabic digits are polysyllabic words except zero. The Arabic digits 0 to 9 are Syfr, Wahid, Ethnan, Thalath, Arb, Khams’a, Syt’a, Sab’a, Thamany’a, Tes’a. The table below shows the details of phonetic explanation of each digit in Arabic.
Table 14: Phonetic Representation of Arabic Digits [23].

<table>
<thead>
<tr>
<th>Digit</th>
<th>Arabic Writing</th>
<th>Pronunciation</th>
<th>Syllables</th>
<th>IPA Representation</th>
<th>No. of Syllables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>واحد</td>
<td>wā-hid</td>
<td>CV-CVC</td>
<td>wa:-hîd</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>أمبابنتين</td>
<td>?ith-nâyn</td>
<td>CV-CVCC</td>
<td>?iθ-nî:n</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>ثلاثة</td>
<td>ðâ-ðâ-lâ-ðâh</td>
<td>CV-CV-CVC</td>
<td>ða-lâ-ða-h</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>أربعة</td>
<td>'aâr-bâ-fâh</td>
<td>CVC-CV-CVC</td>
<td>?ar-ba-fâh</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>خمسة</td>
<td>khâm-sâh</td>
<td>CVC-CVC</td>
<td>xam-sâh</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>سبعة</td>
<td>sâf-tâh</td>
<td>CVC-CVC</td>
<td>sît-tâh</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>ثمانية</td>
<td>sub-bâh</td>
<td>CVC-CVC</td>
<td>sab-bâh</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>تسعة</td>
<td>thâ-má-né-yah</td>
<td>CV-CV-CV-CVC</td>
<td>ða-má-nî-jâh</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>صفر</td>
<td>tês-âh</td>
<td>CVC-CVC</td>
<td>tîs-fâh</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>şēfr</td>
<td>CVCC</td>
<td>şîfrah</td>
<td>1</td>
</tr>
</tbody>
</table>

4.3 Experiment design and analysis

The Arabic spoken digits data set as explained before has 8800 utterances of Arabic digits from 0 – 9. In the experiment, only speaker dependent mode is considered, so the dataset is divided in to two equal sets of 4400 each. One part is used for training the LTM and second part is used for testing LTM.

In order to find the optimum parameters for Arabic spoken digit data set, performance analysis have been done in a similar way as done in Spoken English Data set. Cross validation of Spoken Arabic digits was performed to obtain optimal parameters using 110 samples of each digit. Performance analysis has been done by fixing two of the optimal parameters and varying the third one. The optimal parameters obtained through cross validation are Eta ($\eta$) = 4, decay rate gamma $\gamma$ = 0.2 and delay factor should be $\tau$ = 4. The plot in Fig.20 – Fig.22 shows the performance analysis of the spoken Arabic digits.
Fig. 16 shows that the sensitivity factor (eta) has very slight effect on the Arabic Spoken digit database. PA and SR varied slightly with variation of eta.

Figure 16: Sensitivity of PA and SR to varying ETA of Arabic spoken digits
Fig. 17 shows similar behavior to that of the Spoken English Digits. After a particular value of gamma the PA reduces significantly and mean while SR shoots up at value 1.
Fig. 18 shows that the delay rate tau should be greater than 5 for accurate prediction of the spoken Arabic database.

Based on the performance analysis we found that the optimal parameters that we have obtained through two fold cross validation seems to be right. The prediction
accuracy and SR ratio of each digit was obtained based on the optimal parameters and listed in the Table 15.

Table 15: PA and SR of Arabic Spoken Digits.

<table>
<thead>
<tr>
<th>Digit</th>
<th>PR (in %)</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100.00%</td>
<td>1.5471</td>
</tr>
<tr>
<td>1</td>
<td>99.71%</td>
<td>4.1281</td>
</tr>
<tr>
<td>2</td>
<td>98.18%</td>
<td>5.9716</td>
</tr>
<tr>
<td>3</td>
<td>97.27%</td>
<td>7.3647</td>
</tr>
<tr>
<td>4</td>
<td>98.79%</td>
<td>8.8405</td>
</tr>
<tr>
<td>5</td>
<td>99.39%</td>
<td>10.6582</td>
</tr>
<tr>
<td>6</td>
<td>100.00%</td>
<td>12.21</td>
</tr>
<tr>
<td>7</td>
<td>100.00%</td>
<td>13.8429</td>
</tr>
<tr>
<td>8</td>
<td>99.70%</td>
<td>15.7312</td>
</tr>
<tr>
<td>9</td>
<td>99.70%</td>
<td>17.3263</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>99.274 %</td>
<td></td>
</tr>
</tbody>
</table>

4.4 Discussion

The analysis of both spoken English and Arabic digits gives an accuracy of 97% and 99% respectively with LTM model. The model sensitivity increases with increase in the number of classes. First observation is the effect of parameters in LTM model proposed in [17] clearly affects the prediction accuracy and can be used for the fine tuning of the whole network. Prediction accuracy improves with the variation of significance ($\eta$) which clearly shows its effect in the LTM model. Another observation is that with the increase in number of training sequences in spoken Arabic digit compared to Spoken English Digits, the LTM model improved its performance with an increase in
prediction accuracy. It was also observed that the parameter needs to be changed with the environment in which the LTM model was applied.

In all the analysis above each word is stored in an LTM. When a continuous speech signal is played against the LTM model each LTM will produce maximum activation at the point when the corresponding word stored in the LTM is encountered.
CHAPTER 5: CONCLUSION

In this thesis I have described the speech recognition based on the LTM model proposed in [17]. The speech recognition model has a pre-processing procedure which converts the digital speech signal into Mel scale. Once the wrapping of the frequency into mel-scale is complete it is transformed in to 13 MFCC coefficient using Mel-scale analysis. These 13 MFCC coefficients contain the essential information of the speech signal which serves as input to the LTM model.

The LTM model is based on [17] which have a neural network organization for multidimensional sequences. It has an effective algorithm for the recognition system with emphasis on significance of the elements, error tolerance as well as memory decaying mechanism. The LTM model compares the input with the stored sequence in the LTM and gives out matching grade with the stored sequence. The accuracy and efficiency of LTM model can be varied with tuning of parameters tau (τ), gamma (γ) and eta (η) which represents the delay factor of LTM, the memory decay factor of LTM and also the significance factor of the elements respectively.

The merits of the speech recognition system were demonstrated with two sets of databases, Spoken English digit dataset and Spoken Arabic digit datasets. The results show that speech recognition model has produced an accuracy of 97% - 99% with the given datasets. With improved algorithm to reduce the number of computations in the LTM, the proposed speech recognition system is suitable for various applications in the modern world.
CHAPTER 6: FUTURE WORKS

As a future research the LTM model can be aligned in such a way that similar sequences can be stored in a single LTM. With the help of sequence merging similar sequences from various speakers can be considered as a single LTM. This merging will help speech recognition model to be more accurate with speaker independent capability.

Efficiency of computational model can be improved with new algorithms which reduce the number of computations in the LTM. With reduction in computation time it can be implemented in hardware to perform real time speech recognition.
REFERENCES.


APPENDIX A: MATLAB CODE FOR MFCC CALCULATIONS

% mfcc - Mel frequency cepstrum coefficient analysis.
% [ceps,freqresp,fb,fbrecon,freqrecon] = ...
% mfcc(input, samplingRate, [frameRate])
% Find the cepstral coefficients (ceps) corresponding to the
% input. Four other quantities are optionally returned that
% represent:
% the detailed fft magnitude (freqresp) used in MFCC calculation,
% the mel-scale filter bank output (fb)
% the filter bank output by inverting the cepstrals with a cosine
% transform (fbrecon),
% the smooth frequency response by interpolating the fb
% reconstruction
% (freqrecon)
% -- Malcolm Slaney, August 1993
% Modified a bit to make testing an algorithm easier... 4/15/94
% Fixed Cosine Transform (indices of cos() were swapped) - 5/26/95
% Added optional frameRate argument - 6/8/95
% Added proper filterbank reconstruction using inverse DCT - 10/27/95
% Added filterbank inversion to reconstruct spectrum - 11/1/95
%
% (c) 1998 Interval Research Corporation

function [ceps,freqresp,fb,fbrecon,freqrecon] = ...
    mfcc(input, samplingRate, frameRate)
    global mfccDCTMatrix mfccFilterWeights

[r c] = size(input);
if (r > c)
    input=input';
end

% Filter bank parameters
lowestFrequency = 133.3333;
linearFilters = 13;
linearSpacing = 66.66666666;
logFilters = 27;
logSpacing = 1.0711703;
fftsize = 512;
cepstralCoefficients = 13;
windowSize = 400;
windowSize = 256;               % Standard says 400, but 256 makes more sense
    % Really should be a function of the sample
    % rate (and the lowestFrequency) and the
    % frame rate.
if (nargin < 2) samplingRate = 16000; end;
if (nargin < 3) frameRate = 100; end;

% Keep this around for later....
totalFilters = linearFilters + logFilters;
% Now figure the band edges. Interesting frequencies are spaced % by linearSpacing for a while, then go logarithmic. First figure % all the interesting frequencies. Lower, center, and upper band % edges are all consecutive interesting frequencies.

freqs = lowestFrequency + (0:linearFilters-1)*linearSpacing;
freqs(linearFilters+1:totalFilters+2) = ...
    freqs(linearFilters) * logSpacing.^(1:logFilters+2);

lower = freqs(1:totalFilters);
center = freqs(2:totalFilters+1);
upper = freqs(3:totalFilters+2);

% We now want to combine FFT bins so that each filter has unit % weight, assuming a triangular weighting function. First figure % out the height of the triangle, then we can figure out each % frequencies contribution
mfccFilterWeights = zeros(totalFilters,fftSize);
triangleHeight = 2./(upper-lower);
fftFreqs = (0:fftSize-1)/fftSize*samplingRate;
for chan=1:totalFilters
    mfccFilterWeights(chan,:) = ...
        (fftFreqs > lower(chan) & fftFreqs <= center(chan)).* ... 
        triangleHeight(chan).*(fftFreqs-lower(chan))/(center(chan)-
        lower(chan)) + ...
        (fftFreqs > center(chan) & fftFreqs < upper(chan)).* ... 
        triangleHeight(chan).*(upper(chan)-fftFreqs)/(upper(chan)-
        center(chan));
end
%semilogx(fftFreqs,mfccFilterWeights')
%axis([lower(1) upper(totalFilters) 0 max(max(mfccFilterWeights))])

hamWindow = 0.54 - 0.46*cos(2*pi*(0:windowSize-1)/windowSize);
if 0                    % Window it like ComplexSpectrum
   windowStep = samplingRate/frameRate;
   a = .54;
   b = -.46;
   wr = sqrt(windowStep/windowSize);
   phi = pi/windowSize;
   hamWindow = 2*wr/sqrt(4*a*a+2*b*b)* ... 
       (a + b*cos(2*pi*(0:windowSize-1)/windowSize + phi));
end

% Figure out Discrete Cosine Transform. We want a matrix
% dct(i,j) which is totalFilters x cepstralCoefficients in size.
% The i,j component is given by
% cos( i * (j+0.5)/totalFilters pi )
% where we have assumed that i and j start at 0.
mfccDCTMatrix = 1/sqrt(totalFilters/2)*cos((0:(cephstralCoefficients-1))' * ... 
               (2*(0:(totalFilters-1))+1) * pi/2/totalFilters);
mfccDCTMatrix(1,:) = mfccDCTMatrix(1,:) * sqrt(2)/2;

% imagesc(mfccDCTMatrix);

% Filter the input with the preemphasis filter. Also figure how
% many columns of data we will end up with.
if 1
    preEmphasized = filter([1 -.97], 1, input);
else
    preEmphasized = input;
end

windowStep = samplingRate/frameRate;
cols = fix((length(input)-windowSize)/windowStep);

% Allocate all the space we need for the output arrays.
ceps = zeros(cepstralCoefficients, cols);
if (nargout > 1) freqresp = zeros(fftSize/2, cols); end;
if (nargout > 2) fb = zeros(totalFilters, cols); end;

% Invert the filter bank center frequencies. For each FFT bin
% we want to know the exact position in the filter bank to find
% the original frequency response. The next block of code finds the
% integer and fractional sampling positions.
if (nargout > 4)
    fr = (0:(fftSize/2-1))'/(fftSize/2)*samplingRate/2;
    j = 1;
    for i=1:(fftSize/2)
        if fr(i) > center(j+1)
            j = j + 1;
        end
        if j > totalFilters-1
            j = totalFilters-1;
        end
        fr(i) = min(totalFilters-.0001, ...
            max(1, j + (fr(i)-center(j))/(center(j+1)-center(j))));
    end
    fri = fix(fr);
    frac = fr - fri;
    freqrecon = zeros(fftSize/2, cols);
end

% Ok, now let's do the processing. For each chunk of data:
%   * Window the data with a hamming window,
%   * Shift it into FFT order,
%   * Find the magnitude of the fft,
%   * Convert the fft data into filter bank outputs,
%   * Find the log base 10,
%   * Find the cosine transform to reduce dimensionality.
for start=0:cols-1
    first = start*windowStep + 1;
    last = first + windowSize-1;
    fftData = zeros(1,fftSize);
fftData(1:windowSize) = preEmphasized(first:last).*hamWindow;
fftMag = abs(fft(fftData));
earMag = log10(mfccFilterWeights * fftMag');

ceps(:,start+1) = mfccDCTMatrix * earMag;
if (nargout > 1) freqresp(:,start+1) = fftMag(1:fftSize/2)'; end;
if (nargout > 2) fb(:,start+1) = earMag; end
if (nargout > 3)
    fbrecon(:,start+1) = ...
        mfccDCTMatrix(1:cepstralCoefficients,:)'* ...
        ceps(:,start+1);
end
if (nargout > 4)
    f10 = 10.^fbrecon(:,start+1);
    freqrecon(:,start+1) = samplingRate/fftSize * ...
        (f10(fri).*(1-frac) + f10(fri+1).*frac);
end
end

% OK, just to check things, let's also reconstruct the original FB
% output. We do this by multiplying the cepstral data by the transpose
% of the original DCT matrix. This all works because we were careful
to
% scale the DCT matrix so it was orthonormal.
if 1 & (nargout > 3)
    fbrecon = mfccDCTMatrix(1:cepstralCoefficients,:)'* ceps;
    % imagesc(mt(:,1:cepstralCoefficients)*mfccDCTMatrix);
end;
function Status = trainCVSR(TrFn,TolMode,FeatSelMode)
% function trainCVSR This function chunks the input sequences into short LTM sequences, estimates the tolerance of LTMs and stores the LTMs into storage folder.
% @param TrFn The input training sequence file name that contains a data matrix X (\in \mathbb{R}^{FF \times TT}) and the corresponding label lab.
% @param TolMode It specifies the tolerance estimation methods. See Header.m.
% @param FeatSelMode It specifies the feature selection methods. See Header.m

%% initialize all the constant
load('LTMConf.mat');

if (~exist(TrFn,'file'))
    Status = 0;
    return;
end;

%% Load the input: Data stored as matrix X and corresponding label 'lab'
load(TrFn,'-mat','X','lab');
RawSeq = X; % Data
RawIdx = lab*ones(size(RawSeq,2)); % Label

%% chunking sequence into short sequences based on label or predefined size
fprintf('Chunking the sequence into LTM cells...
');
[LTMsize TrSe TrLab] = ChunkSequence(RawSeq,RawIdx,tt,frOverlap);
fprintf('ok
');

%% feature selection
fIdxSe = cell(1,LTMsize);
if FeatSelMode == FEAT_RANK_ALL % this is most commonly used.
    fprintf('Selecting all the features...
');
    fIdx = 1:ff;
    for i=1:LTMsize
        fIdxSe{i} = fIdx;
        tmp = TrSe{i};
        TrSe{i} = tmp(fIdxSe{i},:);
    end;
    fprintf('ok
');
elseif FeatSelMode == FEAT_RANK_RAND
    fprintf('Selecting features by by random sampling...
');
    fIdx = sort(randi(maxff,ff,1)); % select nFeat out of maxNFeat
    for i=1:LTMsize
        fIdxSe{i} = fIdx;
        tmp = TrSe{i};
        TrSe{i} = tmp(fIdxSe{i},:);
end;
fprintf('ok\n');
elseif FeatSelMode == FEAT_RANK_MD
    fprintf('Selecting features by most distant ranking\n');
    for i=1:LTMsize
        fIdxSe{i} = rankMD(TrSe{i}',ff);
        tmp = TrSe{i};
        TrSe{i} = tmp(fIdxSe{i},:);
    end;
fprintf('ok\n');
elseif FeatSelMode == FEAT_RANK_QR
    fprintf('Selecting features by QR factorization...\n');
    for i=1:LTMsize
        [Q R E] = qr(TrSe{i}',0);
        fIdxSe{i} = E(1:ff);
        tmp = TrSe{i};
        TrSe{i} = tmp(fIdxSe{i},:);
    end;
end;

%% estimate the tolerance
% fprintf('Estimate tolerance: \n');
TrTolSe = cell(1,LTMsize);
for i =1:LTMsize
    fprintf('LTM %d: ',i);
    if TolMode == TOL_MODE_LOCAL % by spatial envelope fitting
        TrTolSe{i} = EstTolSpatEnv(TrSe{i},GaussEnvParam);
    elseif TolMode == TOL_MODE_GLOBAL % by standard deviations
        stddev = std(TrSe{i},0,2);
        TrTolSe{i} = repmat(stddev,1,size(TrSe{i},2));
    else % by constant
        TrTolSe{i} = tolerance*ones(size(TrSe{i}));
    end;
end;
save(ToleFn,'TrTolSe');

%% append with existing LTM cells
nCell = 0;
if AppendLTM_Flag ==1
    load(NLTMCellFn,'nCell');
end;

%% save training data in LTM
% fprintf('Store LTM cells by one-shot learning');
for i=1:LTMsize
  fprintf('.');
  w = TrSe{i};
  sw = EstSigLTM(w,n_M);
  tw = TrTolSe{i};
  labw = TrLab(i);
  fIdx = fIdxSe{i};
  if (LTMSave_Flag)
    save(sprintf('%s/%s%d.mat',LTMStoDir,LTMStoPref,nCell+i),'w','tw','sw','labw','fIdx','-mat');
  end;
end;

%% update the number of LTM cells
nCell = LTMsize+nCell;
save(NLTMCellFn,'nCell');
% fprintf('ok\n');

Status = 1;
function \[TMatch \quad SRMatch \quad RunLab \quad TrueLab\] = testCVSR(TeFn,MaxActMode,DispMode)
\% \function testCVSR This function finds the maximum match of LTM sequences
\% with normalization to the test data sequences(s). This
\% @param TeFn The input test sequences. The format is given in trainCVSR
\% @param MaxActMode If 0, the maximum activation of each LTM is computed
\% and stored in the file LTMMMaxAct.m. If 1, the maximum activation of each
\% LTM is loaded from the file LTMMMaxAct.m.
\% @param DispMode 1 if display mode is on, 0 if display mode is off
\% @return TMatch The returned matching activations of the LTM cells
\% @return SRMatch The returned separation ratio (See the paper).
\% @return RunLab The returned winning labels throughout the test phase.
\% @return TrueLab The returned true labels (used for performance validation)

%% initialize all the constant
gamma=0.0; % since gamma is a predefined function of matlab.
load(sprintf('LTMConfig/LTMConf.mat'));

%% load the LTM cells
\% fprintf('Load the LTM cells...\');
if LTMSave_Flag == 1
    \% load the number of cell
    load(NLTMCellFn,'nCell');
    LTMsize = nCell;
    W = cell(LTMsize,1);
    TW = cell(LTMsize,1);
    SW = cell(LTMsize,1);
    FIDX = cell(LTMsize,1);
    TrLab = zeros(LTMsize,1);
    for i=1:LTMsize
        load(sprintf('%s/%s%d.mat',LTMStoDir,LTMStoPref,i),'w','tw','sw','labw','fIdx');
        \W{i} = w;
        TW{i} = tw.^(-1);
        SW{i} = sqrt(sum(sw(:,1:end-2).^2)/size(sw,1)); \% compute on the fly
        FIDX{i} = fIdx;
        TrLab(i) = labw;
    end;
end;
\% fprintf('ok\n');

%% estimate maximum activations of LTM cells
maxSGNF = zeros(LTMsize,1);
if (MaxActMode==1)
    fprintf('Estimate the norm of activation');
    for jj=1:LTMsize
        fprintf('.');
        wx = W{jj};
        twx = TW{jj};
        swx = SW{jj};
        Amax = ones(1,size(wx,2))*del;
        S = zeros(1,size(wx,2));
        for k=1:size(wx,2)
            tst = wx(:,k);
            [Amax,S]=ContVecSeqMaxEst(wx,twx,swx,tst,S,Amax,del,gamma);
        end;
        maxSGNF(jj) = S(end);
    end;
    MaxAct = maxSGNF;
    save(sprintf('%s/LTMMaxAct.mat',ConfFolder),'MaxAct');
    fprintf('ok
');
else
    load(sprintf('%s/LTMMaxAct.mat',ConfFolder));
    maxSGNF = MaxAct;
end;

%% reset the potential
% fprintf('Reset activation of LTM cells...');
if LTMActClear_Flag == 1
    SS = cell(LTMsize,1);
    AMX = cell(LTMsize,1);
    for ii=1:LTMsize
        SS{ii} = zeros(1,size(W{ii},2));
        AMX{ii} = ones(1,size(W{ii},2))*del;
    end;
end;
% fprintf('ok
');

%% load the input
load(TeFn,'-mat','X','lab');
Test = X';
T = size(Test,1);

%% start testing
TMatch=zeros(T,LTMsize);
matchingLTM = zeros(T,1);
for jj=1:LTMsize
    % Precompute primary excitations - for faster speed
    PrimExc = exp(-0.5*(Test.^2*TW{jj}.^2 - 2*Test.*(W{jj}.*TW{jj}).^2)+repmat(sum((W{jj}.*TW{jj}).^2),T,1))/ff).*repmat(SW{jj},T,1);
    % Update the LTM cells
    for kk=1:T
        [matchingLTM(kk),SS{jj},AMX{jj}]=ContVecSeqRec(...
        PrimExc(kk,:),SW{jj},SS{jj},AMX{jj},del,maxSGNF(jj),gamma);
end;
TMatch(:,jj) = matchingLTM;
end;

%% report the results
[dummy Idx] = sort(TMatch(end,:), 'descend');
RunLab = TrLab(Idx(1));
TrueLab = lab;
% compute the separation ratio.
ListLab = TrLab(Idx);
Idx = find(ListLab==ListLab(1),1);
if (~isempty(Idx))
    SRMatch = min(dummy(1)/(dummy(Idx)+10e-8),50);
else
    SRMatch = 1;
end;
SRMatch;
if DispMode == 1
    close all;
    figure(1),
    plot(TMatch);
    title('Increasing activation of various LTM cells');
xlabel('Iterative step of the dataTest sequence');
ylabel('LTM cell activation');
    pause;
end;
APPENDIX D: MATLAB FUNCTION FOR SIGNIFICANCE ESTIMATION

```
function SW = EstSigLTM(Wx, GW)
% @param Wx The input sequence
% @param GW The Gaussian width factor
% The significance of each element x \in W is calculated by the
% normalized Euclidian similarity.

[FF TT] = size(Wx);
SW = zeros(FF, TT+2);

% Calculate the significance for each feature
FeatMean = mean(Wx, 2); % mean
FeatStdDev = std(Wx, 0, 2); % std dev
SW(:,:,1) = 1.0 - exp(-((Wx - repmat(FeatMean, 1, TT))./repmat(FeatStdDev, 1, TT)).^2/GW) FeatMean FeatStdDev;
```
APPENDIX E: MATLAB FUNCTION FOR CONTINUOUS VECTOR RECOGNITION ALGORITHM

function [Amax,S]=ContVecSeqMaxEst(wx,twx,swx,input,S,Amax,del,gamma)
% This function is used to calculate the maximum activation of the LTM
% based on the significance-induced excitation. The maximum activation
% is used to properly normalize the LTM's activation during testing.

% @param wx The connection weight of the LTM cell
% @param twx The tolerance weight of the LTM cell
% @param swx The significance weight of the LTM cell
% @param del   The delay
% @param gamma Decay rate
% @param MaxAct Accumulated maximum effect of significance
% @return S The state of the LTM cell
% @return Amax The delay counter of the LTM cell

[ff tt] = size(wx);

% compute a primary excitation vector
[A, SigA] = Match(input,wx,twx,swx);

A = max([0 S(1:end-1)]-gamma*SigA + A,0);
S = max(0,S-gamma*[SigA(2:end) 0]); % linear decay

Sxt=[0 S 0];

for k=1:tt
    if Sxt(k+1)>max(Sxt(k),A(k)) && Amax(k)>0
        Amax(k)=Amax(k)-1;
    else
        Amax(k)=del;% Reset the delay
        Sxt(k+1)=max(Sxt(k),A(k));
    end;
end;
S=Sxt(2:tt+1);
APPENDIX F: MATLAB FUNCTION FOR MATCH BETWEEN TEST AND
SEQUENCE VECTOR

function [A, SEx] = Match1(X,Wx,TWx,SEx)
% function Match between unit test tst with the sequence wx
% @param X test unit vector of size 1xFF
% @param Wx The current LTM sequence of size FFxTT
% @param TWx The tolerance for the LTM sequence of size FFxTT
% @param SEx The significance vector for each LTM sequence element
[FF, TT] = size(Wx);
A = exp(-0.25*sum(((repmat(X,1,TT)-Wx)./TWx).^2)/FF).*SEx;
APPENDIX G: MATLAB FUNCTION FOR ESTIMATION OF TOLERANCE

function EstTol = EstTolLocalStd(W,EnvParam)
% \\ fn EstTol = EstTolLocalStd(W,L)
% Spatial Envelope fitting with cubic function
% @param W The input sequence
% @param L The length of the sequence
% @param EnvParam The envelope parameters: The first entry contains the
% half-size of the Gaussian envelope, the second entry contains the
% standard deviation of the Gaussian envelope.
% @return The optimized tolerance of each data point.
% 
% @author Nguyen Vu Anh
% @version 3.0

warning off all;

Disp_Flag = 0  ;  % Set for displaying the optimal curve

% the dimension and the length of sequence
FF = size(W,1);
TT = size(W,2);

D = W;

tau = EnvParam(1);  % size of the envelope
if TT>=EnvParam(1)
   D = [D(:,tau:-1:1) D D(:,end:-1:end-tau+1)];
else
   D = [D(:,end:-1:1) D D(:,end:-1:1)];
end;

% estimated D
D_h = zeros(FF,TT);
M_h = zeros(FF,TT);

% For each feature, we estimate the feature uncertainty by fitting each
% feature to the ground level
M_h(:,1) = mean(D(:,1:1+2*tau),2);
D_h(:,1) = var(D(:,1:1+2*tau),0,2);
B=1/(1+2*tau);
D_Sqr = B*D.^2;
for k=2:TT
   M_h(:,k) = M_h(:,k-1)+B*(D(:,k+2*tau)-D(:,k-1));
end;
M_hSqr = M_h.^2;
for k=2:TT
   D_h(:,k) = D_h(:,k -1)+D_Sqr(:,k+2*tau)-D_Sqr(:,k-1)-M_hSqr(:,k)+M_hSqr(:,k-1);
end;
if (Disp_Flag == 1)
    % Plot only the optimal curve
    j=1;
    close all;
    figure(1),
    plot(D(j,tau+1:end-tau),'-b');
    hold on;
    plot(sqrt(D_h(j,:)),'.r');
    title(sprintf('Curve fitting.'));
    xlabel('t');
    ylabel('||D||');
    pause;
end;
EstTol = D_h.^0.5;