Loughborough University

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

creative commons

COMMONS DEED

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

Loughborough University

Pilkington Library

Author/Filing Title ..... HASKEY, S.J. .....

.....

Accession/Copy No.

Vol. No. ..... | Class Mark .....

LOAN COPY

0401992004

BADMINTON PRESS
UNIT 1 BROOK ST
SYSTON
LEICESTER. LE7 1GD
ENGLAND
TEL: 0116 260 2917

A Modified One-Class-One-Network ANN Architecture for Dynamic Phoneme Adaptation

by

Stephen Haskey

A Doctoral Thesis submitted in partial fulfillment of the requirements for the award of

Doctor of Philosophy of Loughborough University

May 1998

© by Stephen Haskey 1998

Loughborough University
Library
Date May 99
Class
Acc No. 040199200
K0641762

ABSTRACT

As computers begin to pervade aspects of our everyday lives, so the problem of communication from man-to-machine becomes increasingly evident. In recent years, there has been a concerted interest in speech recognition offering a user to communicate freely with a machine. However, this deceptively simple means for exchanging information is in fact extremely complex. A single utterance can contain a wealth of varied information concerning the speaker's gender, age, dialect and mood. Numerous subtle differences such as intonation, rhythm and stress further add to the complexity, increasing the variability between inter- and intra-speaker utterances. These differences pose an enormous problem, especially for a multi-user system since it is impractical to train for every variation of every utterance from every speaker. Consequently adaptation is of great importance, allowing a system with limited knowledge to dynamically adapt towards a new speakers characteristics. A new modified artificial neural network (ANN) was proposed incorporating One-Class-One-Network (OCON) subnet architectures connected via a common front-end adaptation layer. Using vowel phonemes from the TIMIT speech database, the adaptation was concentrated on neurons within the front-end layer, resulting in only information common to all classes, primarily speaker characteristics, being adapted. In addition, this prevented new utterances from interfering with phoneme unique information in the corresponding OCON subnets. Hence a more efficient adaptation procedure was created which, after adaptation towards a single class, also aided in the recognition of the remaining classes within the network. Compared with a conventional multi-layer perceptron network, results for inter- and intra-speaker adaptation showed an equally marked improvement for the recognition of adapted phonemes during both full neuron and front-layer neuron adaptation within the new modified architecture. When testing the effects of adaptation on the remaining unadapted vowel phonemes, the modified architecture (allowing only the neurons in the front-end layer to adapt) yielded better results than the modified architecture allowing full neuron adaptation. These results highlighted the storing of speaker information, common to all classes, in the front-end layer allowing efficient inter- and intra-speaker dynamic adaptation.

ACKNOWLEDGEMENTS

Described in this thesis is work that could not have possibly undertaken without the kind help, support and guidance of my supervisor, Dr. S. Datta. I have learnt many things from his approach to the subject and I offer my sincere thanks.

Thanks also go to all the members of staff from the Electronics and Electrical Engineering Department for their advice and friendship. Thanks also to the secretarial and technical staff who have assisted me during my time at Loughborough. Special thanks go to past and present members of the Signal Processing Group for their friendship and assistance.

I am also very grateful to all the past and present members of the campus radio station for their friendship and support and to the radio station itself, enabling me to wind down after work. I would like to thank all the close friends that I have made during my time at Loughborough, especially Mark, Adey, Chris and Jan. Finally special thanks also goes to my parents and close family for their support, without which I may never have been able to pursue a higher degree.

GLOSSARY

ACF _____ Autocorrelation Function

ANN _____ Artificial Neural Networks

BP _____ Back Propagation Algorithm

DFT _____ Discrete Fourier Transform

DTW _____ Dynamic Time Warping

ECG _____ Electrocardiograph

FFT _____ Fast Fourier Transform

HMM _____ Hidden Markov Model

LPC _____ Linear Predictive Coefficient

MLP _____ Multi-Layered Perceptron

OCON _____ One-Class-One-Network

OCR _____ Optical Character Recognition

PLP _____ Perceptual Linear Prediction

RMS _____ Root Mean Square

SNNS _____ Stuttgart Neural Network Simulator

TDNN _____ Time Delay Neural Network

TIMIT _____ Texas Instruments/Massachusetts Institute of Technology

TABLE OF CONTENTS

CHAPTER 3 : SPEECH DATA

CHAPTER 4 : NEURAL NETWORKS

CHAPTER 5 : NEURAL NETWORK ARCHITECTURE SELECTION

CHAPTER 6 : SPEAKER ADAPTATION LAYER

CHAPTER 7 : NEW ARCHITECTURE

CHAPTER 8 : CONCLUSION AND FUTURE WORK

CHAPTER 1

Introduction

1.1 Introduction

As computers begin to pervade aspects of our everyday lives, so the problem of communication from man-to-machine becomes increasingly evident. At present man-to-machine communication is restricted via clumsy peripherals such as the keyboard and mouse. These devices offer many problems for the growing number of people who, although not computer literate, are expected to communicate with these machines. Even when used by an experienced operator, these devices often act as a communication bottleneck. For this reason, in recent years, there has been concerted interest in speech recognition.

Speech is by far the most widely used and natural means of communication between people. It requires no additional training and by allowing the speaker to quite literally 'speak their mind' it has been shown that an average speaker requires only half the time to convey the same idea compared with that of the most experienced of typists [1]. Speech also offers a vast amount of freedom with numerous applications. It allows a user to communicate from a distance, via the telephone for example, a disabled person to control out-of-reach apparatus such as light switches or an operator to communicate with a device during any 'hands-busy eyes-busy' procedure such as whilst driving a car.

Although the idea of speech recognition may appear to be simple enough, offering numerous benefits for man-to-machine communication, its implementation is actually very problematic. These problems, which have been researched for over forty years, are mainly due to intra- and inter-speaker differences causing large variations from one utterance to the next. These variations are caused by physical differences in speaker's vocal apparatus, dialect characteristics and changes in intonation, rhythm and stress to alter an utterance's meaning.

It is impracticable to teach a speech recognition machine with every possible variation to allow any speaker of any age, gender, accent to speak freely and be understood. It is therefore necessary to normalise these variations by latching onto a speaker's vocal characteristics and use this information to improve the future recognition of utterances from the same speaker or from speakers with similar characteristics. Ideally, an initial single word could be used to obtain vocal-characteristic information giving rise to machines that could be adopted in any multi-speaker environment (public buildings and high streets for example) to verbally give and receive information, such as purchasing requests and location information.

Normalisation of speaker variations may be achieved by using an adaptive process within a speech recognition system, using new and past stored utterances to calculate any differences that may exist and using these differences to adapt the system's model. This adaptation for the majority of speech recognition systems only occurs initially for a new utterance or speaker, resulting in nothing more than further training of the classifier. However, even the same utterance from the same speaker can vary with time and so better recognition results can be achieved by constantly following any variations using dynamic adaptation. Dynamic adaptation enables a system to not only map itself towards a new dialect, age or gender but also to follow changes in a present speaker's mannerisms and mood. In addition to this, the vast majority of speech recognition systems with adaptation, apply an adaptive process to all the information held within an utterance, including both the word and speaker information. This is inefficient and unnecessary since it is only essential for a multi-speaker speech recognition system to adapt towards speaker variations. Therefore for this thesis a modified One-Class-One-Network Artificial Neural Network (ANN) architecture is considered that applies dynamic adaptation to speaker characteristics only, saving processing power, increasing recognition rates and hence increasing efficiency.

1.2 Thesis Objectives

In the search for a more efficient form of dynamic adaptation for multi-speaker speech recognition, this thesis sets out to fulfill three main objectives. These objectives are as follows :

(i) Develop an adaptation procedure that increases the convergence and reduces the processing time for each adaptation cycle.

(ii) Concentrate adaptation only on intra- and inter-speaker variables, i.e. speaker characteristics.

(iii) Allow adaptation towards a single speech class to improve recognition of remaining speech classes within the same vocabulary.

To fulfill these objectives the thesis presents a new adaptation procedure applied to a modified One-Class-One-Network (OCON) ANN which reduces the problem of intra- and inter-speaker variations in an efficient manner.

1.3 Thesis Outline

This thesis contains eight chapters which are organised as follows:

Chapter 2 presents the problems involved with speech recognition including a look at the production of speech, analysis techniques and speech classifiers and highlights the need for correct classifier selection for experimentation. There is also a quick insight into adaptation, its necessity and its application to speech recognition systems. As well as giving a general oversight of present speech recognition, Chapter 2 also includes some historical background from the earliest automated recognition systems in the early 1950s.

Chapter 3 presents an insight into the speech data used for training, adaptation and testing during all experimentation. Highlighting the importance of data selection, the chapter explains the selection of vowel phonemes and introduces their classification with respect to the mouths articulator movements. The TIMIT speech database, from which all the speech data was obtained, is introduced with its format and content explained. From the TIMIT

speech database, choices concerning the selection of speech data are made to create an experimental 'training set' and 'test set.' Finally a large section of the chapter deals with the pre-processing of the selected speech data, and its formatting into a manner that can be used for classification.

A review of neural networks is presented in Chapter 4. Starting with a brief insight into the biological model, the chapter discusses the concepts of parallelism and connectionist architectures to introduce the Artificial Neural Network (ANN). Using some historical background, several architectures, activation functions and learning rules are investigated. At this point Chapter 4 discusses and derives the Back-Propagation (BP) learning algorithm due to its comprehensive use in experimentation (Chapters 5 -7). Finally, the chapter evaluates some software packages that are available for the computer modelling of ANNs, leading to the selection of a package known as SNNS, the Stuttgart Neural Network Simulator, which has been used for all the experimental ANN modelling and is available from the Internet.

Using information from Chapter 4 concerning various activation functions, architectures and learning rules, Chapter 5 initially deals with the selection of an ANN configuration for experimentation. This configuration consists of a multi-layered perceptron (MLP) architecture, a sigmoidal activation function and the BP learning algorithm. The One-Class-One-Network (OCON) architecture is introduced with details of its history, theory and potential for computational saving. A comparative study between the OCON and the common MLP is initiated in Chapter 5 to investigate variations in recognition rates between the two architectures for adapted phonemes and the effect that this adaptation has on the remaining unadapted phonemes.

Chapter 6 deals with the need for adaptation in greater detail and introduces a modified OCON architecture which is believed to be able to segregate speaker information, allowing for more efficient adaptation. This new architecture, containing a new front-end adaptation layer, is explained in detail. To investigate the effectiveness of the modified OCON architecture, Chapter 6 employs two adaptation procedures involving the freezing and unfreezing of selected network weights. By concentrating mainly on intra-speaker effects,

speech data from a single speaker is employed. Chapter 6 also deals with changes to the SNNS software code which is necessary for implementation of the adaptation procedures.

Chapter 7 continues the theme of Chapter 6, by fully testing the newly modified OCON architecture for both intra- and inter-speaker adaptation. The intra-speaker adaptation procedure looks in detail at the effect that each vowel phoneme adaptation has on remaining vowel phonemes from within the same network. For the inter-speaker adaptation procedure, Chapter 7 initially describes the three discrete selection processes used to select the speakers for the test set. The Inter-speaker procedure then investigates the effect that adaptation towards a single speaker has on recognition rates of utterances from the remaining speakers.

The final chapter, Chapter 8, provides a conclusion and discussion for all the experimental research. This includes the effectiveness of the modified OCON architecture on intra- and inter-speaker results along with the fulfillment of the thesis' objectives. Finally, suggestions for further work are made regarding experimentation, improvements, and possible implementation ideas for the modified OCON architecture.

1.4 References

[1] A. Chapanis, "Interactive Human Communication," Sci. America., Vol.232, No.3, pp.36-42, 1975.

CHAPTER 2

Historical Background and Problem Formulation

2.1 Introduction

Automatic man-to-machine communication has been the objective of researchers since the earliest speech recognition device in the early 1950s [1]. With the advent of digital computing in the early 1960s the area of speech research has seen significant advances in processing, coding and classification techniques. However, even with today's technology and advances in signal processing, the aim of a large-vocabulary multi-speaker continuous-speech recognition system has constantly eluded us.

2.1.1 Speech Production

Speech can be viewed as a sequence of basic sound elements known as phonemes. There are approximately 40-50 of these abstract linguistic units in the English language, depending on dialect deviations, which vary with respect to the shape and length of the vocal tract as well as the position of the mouths articulators such as the tongue, jaw and lips. Using these vocal variables, phonemes can be split into three distinct categories, voiced, unvoiced and plosive, each of which can be further split to produce a total of twelve categories (see Figure 2.1). Throughout the text, all phonemes will be described using the ARPABET representation [2] as used in the TIMIT speech database [3]. Voiced phonemes are produced by forcing air from the lungs through the glottis which contains two vocal cords (see Figure 2.2). These vocal cords then vibrate, the speed of which is governed by both the vocal cord's tension and the flow of air between them. The quasi-periodic waveform, the fundamental frequency, travels via the larynx to the oral cavity which acts as a resonant chamber. The oral tract's length and variations in shape, due to movement of the mouths articulators, gives rise to

different resonant or 'formant' frequencies. These formant frequencies combine to produce the voiced phoneme sounds. Of all the mouths articulators, the tongue has the greatest effect on the voiced phonemes[4][5]. The tongue can be used to categorise the vowel phonemes with respect to the position of the tongue's hump within the oral cavity, front, middle and back.

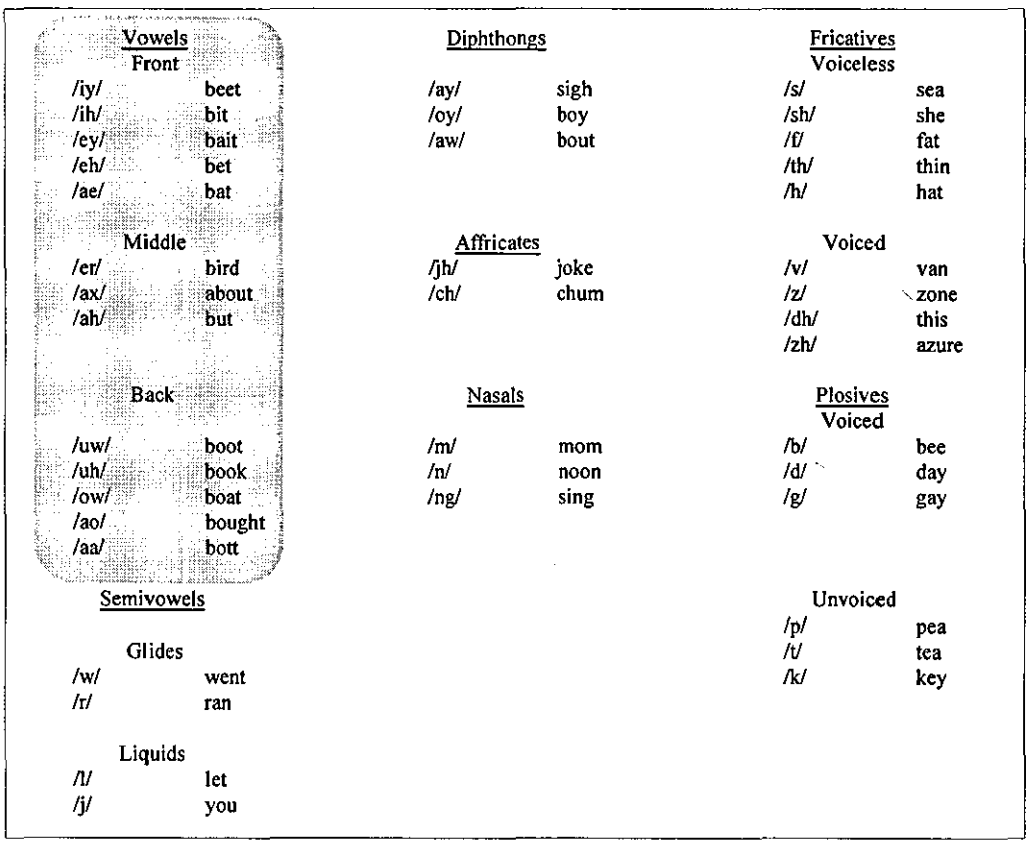


Figure 2.1 The English Phonemes

During the production of unvoiced phonemes, the vocal cords do not vibrate. The phoneme sounds are produced by forcing air through the glottis with the vocal cords open, through to the oral cavity during which a constriction is used to produce fricative and affricate sounds. Fricative sounds are produced by creating the constriction at some point within the oral tract causing turbulence to occur. Since this mostly occurs at the front of the mouth, the resonance of the oral tract has little or no effect. However, for affricate sounds, the constriction occurs at the glottis as the vocal cords are held slightly apart. Therefore the resonance of the oral tract influences the phoneme sound, the effect of which can be heard clearly during whispered speech.

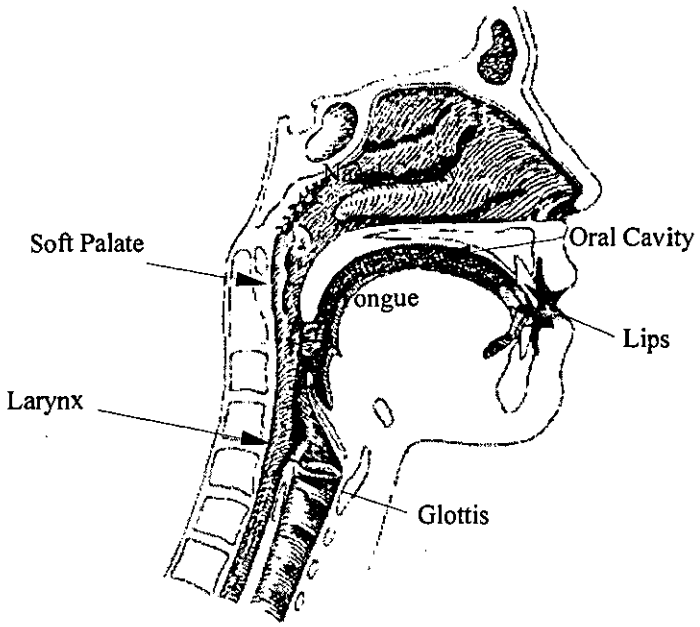


Figure 2.2 A Cross-section of the Human Vocal Apparatus

Plosive sounds are produced by blocking the oral tract with the tongue or the lips, allowing the air pressure to build up until it is suddenly released to create a transient excitation. This transient excitation can occur with or without vibration of the vocal cords to produce voiced or unvoiced plosive phonemes. Other categories of phoneme sounds are the nasals, which are produced by lowering the soft palate to couple the nasal cavity to the oral cavity, and the diphthong, whose sound changes from beginning to end due to tongue movement.

2.1.2 The Complexity of Linguistics

Although speech may appear to be the obvious tool for man-to-machine communication, this deceptively simple means for exchanging information is indeed extremely complex. As well as gender, age and dialect, the simplest of utterances contain a vast wealth of speaker information. Subtle variations in the intonation, rhythm, stress and pitch, allow a speaker to express their mood or change the meaning of a sentence. All of these intra- and inter-speaker variations add to the complexity of the speech signal making recognition that much harder. True speech recognition can be divided into two stages: sound recognition and understanding.

The first problem, sound recognition, involves classifying the sounds 'phonemes' that have been uttered by the speaker. Although speech can be described as a series of phoneme sounds, these sounds are not discretely joined to one another. It takes time for the mouth's articulators, vocal cords and soft palate to move from one phoneme sound configuration to the next and so the sound of each phoneme is influenced by the phonemes surrounding it. This merging of one phoneme into the next is known as co-articulation and can be clearly heard when comparing natural speech against a stilted utterance from an old-style speech synthesizer. The problem with co-articulation is that it gives rise to an effect known as allophonic variation where each phoneme may have many different allophones depending on the surrounding phonemes. This effectively adds to the number of classifiable speech sounds, increasing yet further recognition difficulty. There also exists the problem of dialect differences where the same phoneme is produced in a different manner, creating a different sound. This can be heard in certain English dialects during voiced plosives. As well as variations in the speech sounds themselves, one of the biggest problems for an everyday speech recognition system is coping with noise. Speech recognition systems are rarely used in clinical conditions with sound-proofed environments and high quality microphones. They are often used in busy environments where noise varies unpredictably. Some noises can be eliminated using filtering techniques to improve recognition performance (high frequency noise for example). However, noise such as the chatter of nearby speakers around a user can cause severe problems for a speech recognition system. Humans on the other hand are somehow able to decipher speech from a speaker even in an environment such as at a party. This phenomena is known as the 'cocktail party effect' and allows an individual to hear their name uttered from across the room under a barrage of utterances from other speakers.

Once the speech sounds have been correctly classified, their collective definition needs to be interpreted. This requires knowledge concerning the meaning (semantics), the rules of word formulation (morphology) and the rules of sentence formulation (syntax) [6]. However, problems first start when many expected phonemes during freely spoken speech are not spoken at all or are completely obliterated by surrounding sounds. Humans overcome this problem automatically by subconsciously predicting what the next sound or utterance may be, using syntax, morphology and semantic knowledge of the present utterance [7]. This can cause problems in itself, when the human brain expects to hear one sound and a similar sound

is uttered. This often occurs in words that rhyme and results in the listener being adamant that one word was uttered, when in fact the speaker uttered another. Unfortunately, some words not only rhyme but are phonetically identical. For example, when we have the phonetic combination '/t/ /uw/' it can refer to the word 'to', 'too' or 'two.' Also, we may have a situation when the correct word may be classified but its meaning confused. For example, if applying speech recognition to a word-processor and dictating a letter on punctuation, when the speaker utters the word comma, does the system write 'comma' or ','? All these problems highlight the recognition systems need for intelligence. In order to understand a sentence fully, it must not only understand the context so as to recognise the correct words and punctuation, but also the intonation and stress that can subtly change a phrase from a reply to a statement into a rhetorical question.

2.2 Speech Analysis

All areas of speech research, whether is be speech recognition, synthesis or coding, require some form of front-end signal processing. In the case of speech recognition, some form of preliminary analysis is necessary before classification can occur to remove any redundant information and extract the acoustic information necessary for recognition. This not only reduces the information rate but also helps to highlight subtle differences between classes that might otherwise be obscured. The area of speech analysis can be split into two broad categories: transform domain and time domain. Transform domain techniques include discrete Fourier transforms, fast Fourier transforms, cepstrum, wavelets and bandpass filtering. For time domain analysis, autocorrelation, zero-crossing and signal energy techniques are often used to extract essential information. In addition, a very powerful speech analysis technique used for both frequency and time domain analysis is linear prediction which represents a speech signal as a combination of linear predicted coefficients. All of these analysis techniques are applied to speech over a short time interval (10-30ms) since, for periods of this length, the non-stationary speech signal is assumed to be stationary. This is due to the limited rate at which the mouth's articulators can move, due to physical constraints, which create the changes in the speech signal. Consequently most analysis techniques split the signal into uniform segments or 'windows' (see Section 3.3). Before any

analysis procedures are applied to a speech signal, the signal is pre-emphasised to improve the definition of its frequency spectrum.

2.2.1 Pre-Emphasis

During natural speech there is a combination of a -12dB/octave trend due to the voiced excitation source and a +6dB/octave trend due to radiation from the mouth [8]. This results in a -6dB/octave trend, effectively reducing the speech signals amplitude by a factor of 16 for every doubling of the speech signal's frequency. Consequently the speech signal needs to be pre-emphasised (a +6dB/octave lift to compensate for the -6dB/octave roll-off) in order that important high frequency information is not lost. This is achieved using a high-pass filter on the digitised speech signal to reduce the dynamic range (i.e. flatten the speech spectrum) and is typically based on using :

$$y[n] = \chi[n] - \alpha\chi[n-1] \tag{2.1}$$

where y[n] is the current output sample of the pre-emphasis filter, χ[n] is the current input sample, χ[n-1] is the previous input sample and α (usually between 0.9 and 1) is a constant that determines the cutoff frequency of the filter.

2.2.2 Frequency-Domain Analysis

The earliest speech recognition system, built in 1952 [1], used bandpass filters to enable the first ten numeric digits to be classified. Constructed from extrinsic components, the recognition system crudely split the speech signal into its first two formants using two filters, a high and a low pass 900Hz filter. Information from these formants was used during the classification process, producing recognition rates between 97% and 99% for a single speaker. This filtering technique has since been employed by many systems [9][10][11], passing the speech signal through a whole bank of bandpass filters covering the speech bandwidth. Using the energy output from the bandpass filters, a pattern for each time frame

is created and sent to a matrix. This matrix, over several frames, creates a pattern for each spoken utterance allowing the recognition system to compare it with stored test patterns or templates to find the most likely match. In these very early systems, all the bandpass filters were linearly spaced. However, this soon changed to logarithmic spacing in an attempt to try and emulate the frequency response of the human auditory system.

The discrete Fourier transform (DFT), normally computed via the fast Fourier transform (FFT) is one of the most widely used techniques for evaluating the frequency spectrum of speech. It enables a speech utterance to be represented in terms of amplitude and phase as a function of frequency allowing spectral data from the vocal tract, such as formant information, to be used for classification. Due to the non-stationary nature of speech, DFTs are applied over a finite time period using short-time analysis windows. The short-time DFT of a signal *H[k]* is often defined as :

$$H[k] = \sum_{n=0}^{N-1} h[n] e^{-j2\pi nk/N} \qquad for \quad 0 \le k \le N - \qquad (2.2)$$

They were widely used for the frequency analysis of speech signals, but were slow for large values of N. This was due to the large number of calculations involved, N^2, making real-time analysis almost an impossibility. However in 1963 the fast Fourier transform (FFT), developed by Cooly and Tukey [12], reduced the number of calculations to $N\log_2 N$ whilst still obtaining exactly the same result. This was a significant step for frequency analysis since $N\log_2 N$ is dramatically less than N^2, especially when N becomes large. This reduction was possible because the FFT algorithm exploits the symmetry properties of the discrete-time complex exponential of the DFT, removing redundant calculations. The only restriction with the FFT algorithm is the value of N since for maximum efficiency, N must be a power of 2, i.e. $N = 2^m$ where m is an integer.

2.2.3 Time-Domain Analysis

The autocorrelation function (ACF) is a powerful technique for estimating the pitch-period of voiced speech. This is achieved by correlating the speech signal with a delayed copy of itself. The autocorrelation value, $R[k]$, of a stationary signal $x[n]$ for a time-shift of k samples is defined as

$$R[k] = \sum_{n=-\infty}^{\infty} x[n] \cdot x[n+k] \tag{2.3}$$

When applied to voiced speech, the autocorrelation function exhibits peaks at certain time-shifts corresponding to multiples of the pitch-period. This is because at these points, the speech signal is in phase with itself, giving high correlation values. However equation (2.3) only applies when the amplitude is known for all time and, for the summation to remain finite, the signal has a finite energy. As far as speech is concerned, it is not known for all time, so it is necessary to use a short-time auto-correlation function by isolating successive segments (frames) of the signal into windows.

A simple time-domain analysis technique which provides spectral information at a low computational cost is the zero-crossing rate. The zero-crossing rate is quite simply the number of times the signal waveform crosses the time axis, i.e. the amplitude changes sign. Although this technique can be used to obtain simple spectral information for an entire utterance it is more commonly used to determine the difference between a voiced and unvoiced segment of speech. This is possible since the random nature of unvoiced speech creates a higher zero-crossing rate than voiced speech.

Another time-domain technique which is used for discriminating between a voiced and unvoiced segment of speech is the signal energy function. By splitting a speech utterance into frames, using a simple rectangular window, and calculating the total squared values from each frame, the changes in the utterances energy with time are recorded. This highlights differences in the voiced and unvoiced segments since voiced speech generally contains more energy that unvoiced speech.

As well as the zero-crossing rate and signal energy function being used for speech signal analysis, they are also used for end-point detection. One of the greatest difficulties with speech recognition, especially if the speaker is in a noisy environment, is to detect the beginning and ending of an utterance. This was a problem addressed by Rabiner and Sambur [13] in the mid seventies who used an end-point detection algorithm based on the signal energy function and the zero-crossing rate of speech. It accounted for the level of background noise by adapting relevant thresholds on its decision criteria during the recording interval and because of its speed and efficiency was and still is widely used.

2.2.4 Linear Prediction

In the early seventies the technique of linear prediction was shown to be applicable to speech by Atal and Hanauer [14]. Used for both frequency and time-domain analysis, its a very powerful speech processing technique used in speech synthesis, recognition and coding systems [15]. The basic idea behind the method is that sample values of speech, $x[n]$, can be approximated as a linear combination of the past p speech samples. As the value of p increases then so the Root Mean Square (RMS) prediction error between the sampled and predicted signal falls. Mathematically, the linear predictor is described by the equation

$$\tilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k] \tag{2.4}$$

where $\tilde{x}[n]$ is the predicted sample at instant n and $a_1, a_2, \ldots, a_p$ are the Linear Predictor Coefficients (LPCs).

It is generally impossible to predict each signal sample exactly and this leads to a prediction error $e[n]$ at each sample instant:

$$e[n] = x[n] - \tilde{x}[n] \tag{2.5}$$

By minimizing the mean squared error between the actual speech samples and the linearly-predicted ones, the predictor coefficients can be determined by solving a set of linear

equations. However, finding the solution to a system of equations with many unknowns can be problematic, even if the equations are linear. Fortunately, two different methods exist for finding the solution of the system of equations. These are the covariance method and the more commonly used auto-correlation method. When using the autocorrelation method the system of equations created produces a symmetric matrix where all the diagonal elements are the same. This is known as a Toeplitz matrix and the very efficient Durbin[16]-Levinson[17] method, requiring much less computational effort, exists for solving this special system of equations. Due to the time varying properties of a speech signal it is necessary to calculate a new set of predictor coefficients every 10-30ms. Consequently a speech utterance is segmented into short frames, 10-30ms in duration, and the short-time autocorrelation function applied to each. However, before the short-time autocorrelation function can be applied to a framed speech signal $x[n]$ it must be multiplied by a soft window function. A soft window function is essential in order to reduce prediction error at the beginning and the end of the segment. Large prediction errors will arise at the start of the interval $(0 \le n \le p-1)$ since the predictor is effectively being required to predict the signal from samples which have arbitrarily been set to zero, while at the end of the interval $(N \le n \le N+p-1)$ it is endeavoring to predict zero signal from samples that are non-zero. The covariance method uses the same principle of windowing the speech, only it does not use a soft window. Although this means that it can give accurate estimates of prediction coefficients using a narrower window it is, unlike the autocorrelation method, not always guaranteed to produce a stable predictor.

Despite these different methods for obtaining the linear predictive coefficients they all approximate equally well at all frequencies which is not consistent with that of the human hearing system. Above 800Hz the spectral resolution of hearing reduces with frequency and for the volume levels in normal speech, hearing is more sensitive than in other areas of the audible spectrum. As a result, Hermansky[13] in 1990 proposed a new form of linear prediction known as Perceptual Linear Prediction (PLP) which mimicked this hearing response. The PLP technique incorporates three concepts from the psychophysics of hearing to derive an estimate of the human auditory system; the equal-loudness curve[18], the intensity-loudness power law [19] and the critical-band spectral resolution[20]. The use of these three concepts by PLP offered comparable performance with the conventional linear

prediction technique using a much lower order of predictive coefficients. Tests on a speaker-independent recognition system showed that a 5^{th} order PLP yielded a better recognition accuracy than a conventional 14^{th} order system. Although PLP is ideal for applications where the data rate (number of predictive coefficients) must be kept to a minimum, the method is computationally very costly and unnecessary for speech recognition simulation.

2.3 Speech Classifiers

Since the late 1960s many methodologies have been applied towards isolated/connected word and dependent/independent speaker recognition classifiers. However, regardless of their method and application, they are all based on the same principle, matching inputted speech patterns with stored reference patterns within the classifier. Initially a speech signal undergoes some type of processing to transform it into a sequence of feature vectors. It is then compared, using an appropriate distance measure[21][22], with a representative training set of reference patterns stored in the classifier during a training phase. This methodology of pattern matching, used by the first recognition systems (see Section 2.2.2), is now used by more modern classification systems such as Dynamic Time Warping (DTW), Hidden Markov Models (HMMs), Artificial Neural Networks (ANNs) and hybrids systems which are a combination of the former systems.

2.3.1 Dynamic Time Warping

Due to the inherent variability of speech, a speech pattern, even from the same speaker, can vary both locally and globally with respect to time. Hence it is necessary to time-align speech patterns in order to find a distance measure between them. The simplest form of time-aligning two patterns is Dynamic Time Warping (DTW)[23][24] which maps the time axis of an inputted speech pattern onto the time axis of an trained reference pattern, as shown in Figure 2.3. In order to align the input and reference speech patterns, the mapping function $\omega[n]$ needs to be determined. To achieve this, both the beginning and end-points of the two patterns have to first be detected. Once the beginning and end-points have been detected,

time-alignment of the two patterns can begin by solving the optimisation equation (2.6) to determine ω[n].

$$D = \min\left[\sum_{n=1}^{T} d\left[I[n], R(\omega[n])\right]\right] \tag{2.6}$$

where $d[I[n],R(\omega[n])]$ is the distance between frame n of the input speech pattern I and frame $\omega(n)$ of the reference pattern R, and D is the accumulated distance between speech patterns T and R over the optimal path $\omega(n)$.

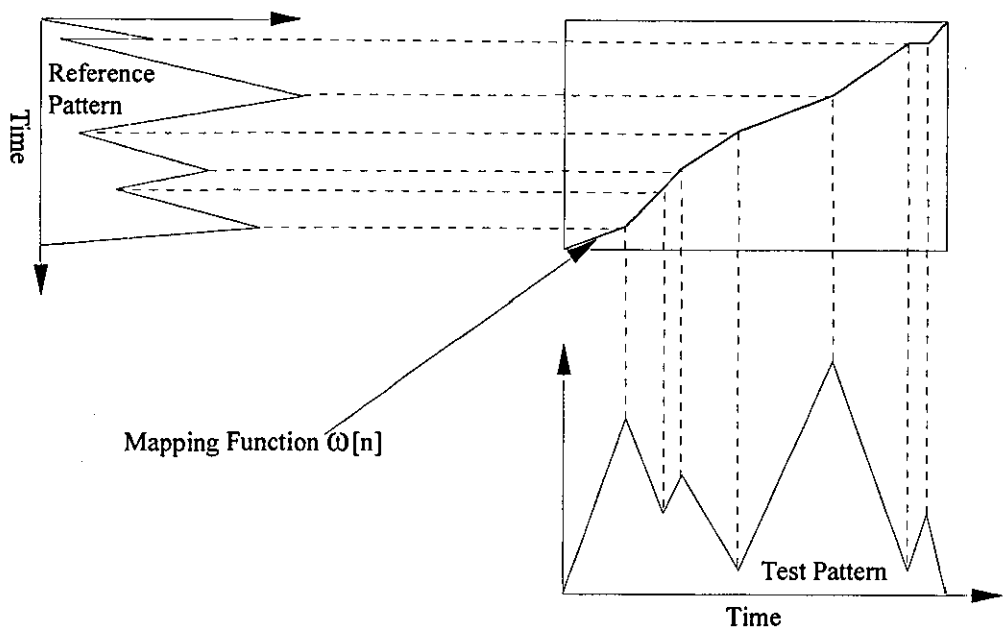


Figure 2.3 Graphical Representation of the Time-Aligning Procedure used by Dynamic Time Warping (DTW) to Align a Test and Reference Pattern.

Although DTW is a simple technique, it does have a few drawbacks. First DTW can have a heavy computational load making real-time application difficult, second is the failure of DTW to adequately exploit redundant information within a speech signal to aid recognition and finally is the problem of dealing with noisy inputs. However, these problems can be eased by implementing non-linear sampling when obtaining frame intervals and omitting highly correlated frames from a speech utterance.

2.3.2 Hidden Markov Models

Hidden Markov Models (HMMs) are different from DTW in that they do not directly compare input patterns with stored test patterns. Instead they create stochastic models from known utterances to create test patterns and compare the likelihood that the input pattern was generated by one of the models (test patterns). The basic theory of HMMs was published by Baum [25] in the late 1960's and early 1970's and was first used in speech processing by Baker [26] and Jelinek [27] in the 1970's. It was not until the mid-eighties that a profound understanding of the subject and its application was founded. The HMM structure consists of an underlying Markov chain containing inter- and intra-linked states (see Figure 2.4).

$\pi = (\pi_1 \ \pi_2 \ \pi_3 \ \pi_4)$

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \qquad B = [b_{jk}] = \begin{bmatrix} b_{11} & b_{21} & b_{31} & b_{41} & b_{51} \\ b_{12} & b_{22} & b_{32} & b_{42} & b_{52} \\ b_{13} & b_{23} & b_{33} & b_{43} & b_{53} \end{bmatrix}$$

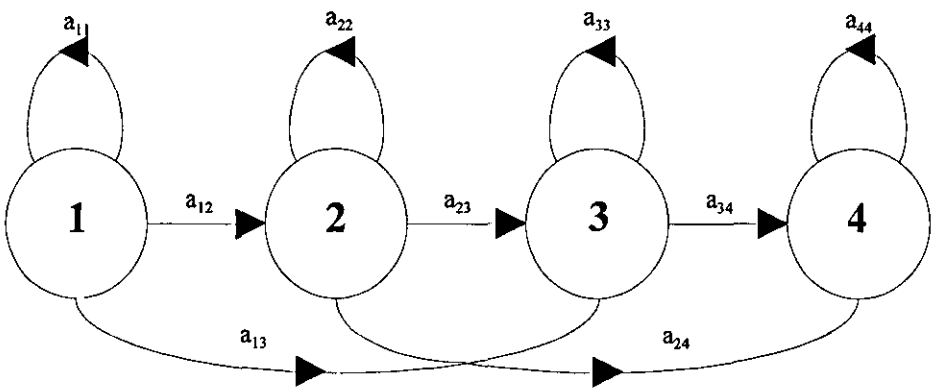


Figure 2.4 Graphical Representation of a 4-State Hidden Markov Model and its Control Matrices

At a discrete instant of time 't' the HMM is in one of the states and outputs a certain speech pattern. At the next time instant, t+1, the model moves on to another state, or loops back to the same state again, and produces yet another pattern from the state its in. The looping back allows for time fluctuations from the input pattern to the test pattern. This procedure is repeated until the complete sequence of patterns has been produced. In a sense a HMM gives a description of an utterance as a concatenation of sounds from each state. Three matrices

control the behaviour of the chain. The first matrix (π) controls the probability of which state the models starts. The next is the state transition matrix, A, which controls the probability of changing from one state to the next and finally matrix, B, which represents the probability distribution function of each state. The number of states, patterns in each state and interconnections vary from system to system although the basic idea and architecture remain the same.

2.3.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a family of architectures used for pattern matching. Although they are said to emulate the neural structure of the human brain, there are enormous differences between the brain and ANNs. The first attempts to model brain function were by McCullogh and Pitts back in 1943 [28]. However, it was not until 1986 when an adequate training algorithm was developed [29] that ANNs were used for serious classification procedures such as optical character recognition (OCR), Electrocardiograph (ECG) analysis and speech recognition. One of the most attractive features of ANN classifiers is their variety and architecture adaptability. ANNs offer a large potential for trying new architectures and for creating application specific classifiers. For this reason ANNs were selected for experimentation to fulfill the three research objectives of this thesis and so their history, development, implementation and limitations are detailed in Chapter 4.

2.4 Adaptation

Classification systems rely on a data set of trained reference patterns to classify an utterance from a speaker. However, the nature of speech means that utterances can contain a wealth of varied information, increasing their inter- and intra-speaker variability, and reducing their likelihood of a match with a reference pattern. Since it is not practical to train a classification system with every possible utterance from every possible speaker, some form of adaptation is necessary. Adaptation allows a system to dynamically map itself towards a new utterance, normalising speaker characteristics to ensure that a speech recognition system is only

sensitive to the phonetic information. Such adaptation can occur at either the pre-processing stage, which transforms the speech signal into a sequence of feature vectors, or the classification stage, which classifies the feature vectors into speech sounds. When adaptation is applied during the pre-processing stage the type of normalisation depends on the type of pre-processing involved. For example, if a system uses formant information for its recognition features then the upper and lower limits of the formant frequencies for each utterance can be normalised [30][31][32]. The array of adaptation techniques available for the classification stage is vast and, like the pre-processing adaptation, depends very much on the type of system being used. However, the basic concept is still the same, to normalise the incoming utterances making them insensitive to speaker characteristics. Despite the variety and large number of adaptation techniques, they can all be categorised into one of two groups; supervised or unsupervised.

During supervised adaptation, an external teacher is used to inform the recogniser of either a correct or incorrect classification. The external teacher can be in the form of a discrete input from the speaker such as a keyword or an adaptation data set of input-output pairs. This ensures that the recogniser always adapts towards the correct speech class, but this is often impractical. For unsupervised adaptation the procedure requires no teacher and the selection of a class to adapt towards is automated. This is faster and more practical than a supervised system but can lead to incorrect adaptations without the users knowledge of control. Further information on adaptation is detailed in Chapter 6.

2.5 Summary

There are many problems concerning the recognition of speech, mainly due to the variability and complexity of even the simplest utterance. However, adaptation offers an ideal opportunity to overcome many of these problems and achieve multi-speaker large-vocabulary continuous speech recognition. This chapter has described several analysis and classification techniques from the earliest of systems to the present day.

To fulfill the research objectives, a classification and speech analysis combination was required that would offer a wide scope of system adaptability and variability to allow the creation of a new adaptation technique. To create the classifier, ANNs were selected since they offer a large potential for trying out new ideas and creating novel architectures. For the speech analysis, linear prediction was chosen due to its simplicity and well documented behaviour. This was necessary since all the adaptation was occurring within the classifier, the main area of concern. An over complicated speech analysis technique may have clouded and confused new adaptation-procedure results.

The selection, preparation, and format of the speech data for all the experimentation is explained in the following chapter. It highlights the importance of data selection and the selection of the vowel phonemes from the TIMIT speech database. It then details all the pre-processing and the final formatting of the speech data, ready for introducing to the classifier. An introduction into ANNs, detailing their background, various architectures and learning rules, is given in Chapter 4.

2.6 References

[1] K.H.Davis, R.Biddulph and S.Balashek, "Automatic Recognition of Spoken Digits," J Acoustical Society of America," Vol 24, No6, pp 637-642, Nov 1952.

[2] J.E.Shoup, "Phonological Aspects of Speech Recognition," pp125-138, Trends in Speech Recognition, W. A. Lea, Ed., Prentice-Hall, Englewood Cliffs, NJ, 1980.

[3] DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM, Oct 1990. Web Address: //www.ldc.upenn.edu/

[4] L.Rabiner and J.B.Hwang "Fundamentals of Speech Recognition," Prentice Hall, NJ, 1993.

[5] P.B.Denes and E.N.Pinson, "The Speech Chain, The Physics and Biology of Spoken Language," Bell Telephone Laboratories, Incorporated, March 1972.

[6] An Introduction to Language V. Fromkin R. Rodman , Harcourt Brace Jovanovich College Publishers, 1993.

[7] G.A.Miller and S.Isard, "Some Perceptual Consequences of Linguistic Rules," Journal of Verbal Learning and Verbal Behavior, Vol 2, pp217-228, 1963

[8] F.J.Owens, "Signal Processing of Speech," The Macmillan Press Ltd, 1993.

[9] P.Denes and M.V.Mathews, "Spoken Digit Recognition using Time Frequency Pattern Matching," J Acoustical Society of America, Vol 32, pp1450-1455, Nov 1960.

[10] H.Dudly and S.Balashek, "Automatic Recogntion of Phonetic Patterns in Speech," J Acoustical Society of America, Vol 30, pp721-732, Aug 1958.

[11] H.F.Olson and H.Belar, "Phonetic Typewriter," J Acoustical Society of America, Vol 28, pp1072-1081, Nov 1956.

[12] J.W.Cooly and J.W.Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," Mathematics of Computation, Vol 19, pp297-301, 1963.

[13] L.R.Rabiner and M.R.Sambur, " An Algorithm for Determining the Endpoints of Isolated Utterances," The Bell system technical journal, pp297-315, 1975.

[14] B.S.Atal and S.L.Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Acoustic Wave," J Acoustical Society of America, Vol 50, pp637-655, 1971.

[15] Makhoul, "Linear Prediction," Procs IEEE, April 1975

[16] J.Durbin, "The Fitting of Time-Series Models," Rev. Inst. Int. Statist., Vol.28, No.3, pp.233-243, 1960.

[17] N.Levinson, "The Wiener RMS (root mean square) Error Criterion in Filter Design and Prediction," J. Math. Phys., Vol.25, No.4, pp.261-278, 1947.

[18] D.W.Robinson and R.S.Dadson, "A Predetermination of the Equal-Loudness Relations for Pure Tones," Br. J. Appl. Phys., Vol.7, pp.166-181, 1956.

[19] S.S.Stevens, "On the Psychophysical Law," Psychol. Rev., Vol.64, pp.153-181, 1957.

[20] M.R.Schroeder, "Recognition of Complex Acoustical Signals," Life Sciences Research Report, Vol.5, pp.324, 1977.

[21] A.H.Gray and J.D.Markel, "Distance Measures for Speech Processing," IEEE Trans., ASSP-24, Vol.5, pp.380-391, 1976.

[22] N. Nocerino, F.K.Soong, L.R.Rabiner and D.H.Klatt, "Comparative Study of Several Distortion Measures for Speech Recognition," Speech Communications, Vol.4, pp.317-331, 1985.

[23] F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans., ASSP-23, Vol.1, pp.67-72, 1975.

[24] C.S. Myers, L.R. Rabiner and A.E. Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition," IEEE Trans., ASSP-28, Vol.6, pp.623-635, 1980.

[25] L.E.Baum and T.Petrie, "Statistical Interference for Probabilistic Functions of Finite State Markov Chains," Ann. Math. Stat., Vol.37, pp.1554-63, 1966.

[26] J.K.Baker, "The DRAGON System - An Overview," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol.23, No.1, pp.24-9, 1975.

[27] F.Jelinek, "Continuous Speech Recogntion by Statistical Methods," Proc IEEE, Vol.64, pp.532-6, 1976.

[28] W.S. McCullogh and W. Pitts, "A Logical Calculas of the Ideas of Immanent in Nervous Activity," Bull, Math., Biophys., 3, pp115-133, 1943.

[29] R.P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, pp.4-22, April 1987.

[30] L.J.Gerstman, "Classification of Self-Normalized Vowels," IEEE Trans., AU-16, Vol.1, pp.78-80, 1968.

[31] E.P.Neuburg, "Frequency-Axis Warping to Improve Automatic Word Recognition," Proc. ICASSP-80, pp.166-168, 1980.

[32] H.Wakita, "Normalisation of Vowels by Vocal-Tract Length and its Application of Vowel Identification," IEEE Trans. ASSP-25, Vol.2, pp.183, 1977.

CHAPTER 3

Speech Data

3.1 Introduction

To obtain the desired results from a speech recognition system, usage of correct speech data is vitally important. The recording conditions of the speech along with the choice of speakers, their dialect region and gender can all dramatically influence the results. All these parameters, including the quantity of data, must be chosen specifically for the classification task at hand to ensure that the network exhibits a good generalisation. If a limited data set is chosen which doesn't give a good overall representation of the speech to be classified then the generalisation is poor, hindering the recognition of new utterances. However, if the data set is too large and over comprehensive then the generalisation is said to be too high causing problems with the convergence of the classifier during the training and adaptation phase. The goal of training a network is not for it to learn an exact representation of the speech data, but rather to build a statistical model of the process which generates each speech utterance wishing to be classified. It is therefore necessary to optimise the diversity and quantity of the speech in order to achieve the best generalisation.

Speech has many linguistic levels such as sentences, words and syllables. Most speech recognition systems perform at the 'word' level since words can be uttered discretely to avoid any co-articulation effects. This is fine with a limited vocabulary when a classifier need only be small to represent each 'word' class. However, when dealing with large vocabularies (1000+ words), the size of the classifier can become unmanageable. To avoid this problem it is best if we deal at the 'phoneme' linguistic level. Phonemes are the basic sound units of speech of which there are about 50 in the American-English language. This means that regardless of the vocabulary size it is only necessary to classify 50 sounds. Phoneme recognition systems are consequently the most promising approach to large vocabulary systems and so all experimentation will be focused at the phoneme linguistic level. The

experimentation is primarily concerned with intra- and inter-speaker dynamic adaptation. Therefore to emphasise any adaptation effects that may occur within the modified ANN architecture, experimentation needs to be concentrated on speaker-information rich phonemes. Phonemes can be categorised into 12 groups (see Figure 2.1). Of these 12 groups, vowel phonemes are the most spectrally well defined making them more easily and reliably recognised. Also, the vowel phonemes are produced solely by vibration of the vocal cords creating resonant frequencies (formants) in the vocal tract which are altered by the movement of the mouth's articulators. The nature of their production means they contain a wealth of speaker information making them ideal for testing intra- and inter-speaker adaptation effects.

There are 13 vowel phonemes in American-English speech and during experimentation it is necessary to not only test the effect on the adapted vowels but also the effect the adaptation has on the remaining unadapted vowels. This is a very labour intensive procedure and so it is favourable to reduce the phoneme groups further. The vowel phonemes can be split into 3 groups depending on the position of the tongue during their production. The variation in the cross-sectional area along the vocal tract determines the formants of each vowel phoneme. This cross-sectional area, particularly in the oral cavity, can be altered by the movement of the articulators, primarily the tongue. The tongue movement varies the formants in two ways; by the tongue-hump-position (i.e. front, middle, back) and by the degree-of-constriction (i.e. high, middle, low) the tongue hump causes (see Figure 3.1). As a result, the tongue plays a fundamental part in the production of the formants in the vocal tract that make the vowel sounds. Peterson and Barney [1] measured the first and second formants, F_1 and F_2 respectively, of 10 vowel phonemes uttered by a variety of both male and female speakers. The information they gathered was averaged out for each vowel phoneme and a plot of F_1 against F_2 constructed to obtain results for an average normalised speaker (see Figure 3.2). The plot created what is known as the 'Vowel Triangle' representing the extremes of the formant locations in the F_1-F_2 plane. The vowel triangle shows a pattern trend between the position of the vowel phonemes within the F_1-F_2 plane and the position of the tongue during their production. As the constriction of the tongue decreases so F_1 increases and as the position of the tongue-hump moves backwards through the oral cavity then so F_2 decreases.

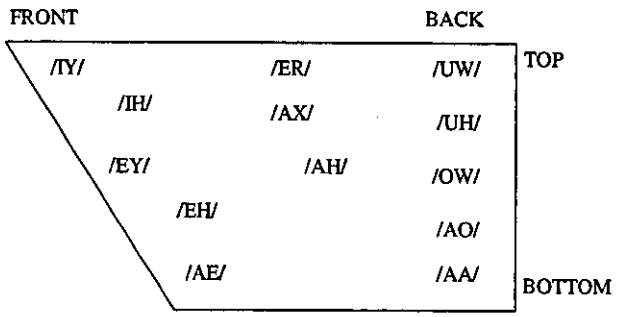


Figure 3.1 The Position of the Tongue-Hump in the Oral Cavity during the Production of the American-English Vowels.

Phonemes from the same tongue-hump group are positioned closely within the F_1-F_2 plane. With this strengthened by evidence from waveform plots and spectrograms showing acoustic similarities between vowels of the same tongue-hump position [2], the phonemes were grouped with respect to their front, middle and back tongue-hump positions. Using the ARPABET representation [3], the front vowel phonemes are /IY/, /IH/, /EY/, /EH/ and /AE/, the middle vowel phonemes are /ER/, /AX/ and /AH/, and the back vowel phonemes are /UW/, /UH/, /OW/, /AO/ and /AA/.

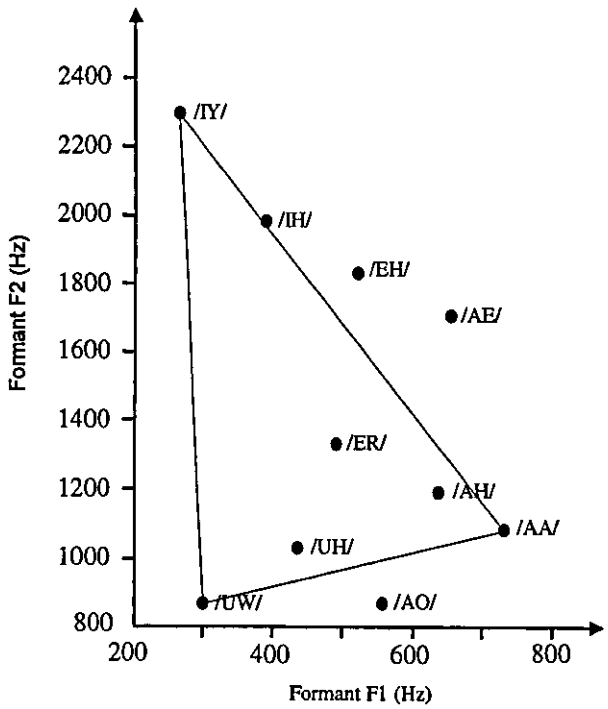


Figure 3.2 F_1 - F_2 Plot of Averaged American-English Vowel Speech Data from a Variety of both Male and Female Speakers showing the "Vowel Triangle."

3.2 The DARPA TIMIT Speech Corpus

All the speech data used in this thesis was obtained from the Texas Instruments/Massachusetts Institute of Technology (TIMIT) corpus of read speech available on a CD-ROM [4]. Sponsored by the Defense Advanced Research Projects Agency (DARPA), TIMIT contains speech from 630 speakers representing 8 major dialect divisions of American-English, each speaking 10 phonetically-rich sentences. Of these 630 speakers, 70% are male with their dialect region defined as the geographical area in the U.S. where they lived during their childhood years (age 2 to 10). The 10 phonetically-rich sentences uttered by each speaker consist of 5 phonetically-compact sentences, 3 phonetically-diverse sentences and 2 dialect sentences used to expose dialect variants between speakers. The speech data on the CD-ROM is divided into suggested Training and Test sets using the following criteria:

- Roughly 20 to 30% of the corpus should be used for testing purposes, leaving the remaining 70 to 80% for training.
- No speaker should appear in both the training and testing sets.
- All the dialect regions should be represented in both subsets, with at least 1 male and 1 female speaker from each dialect.
- The amount of overlap of text material in the two subsets should be minimised; if possible the training set and test set should have no sentence texts in common.
- All the phonemes should be covered in the test material; preferably each phoneme should occur multiple times in different contexts.

Although it was favourable during experimentation to use as much speech data as possible to obtain good generalisation, the 2 dialect sentences from each speaker used to expose dialect variants were omitted. This was because the same two sentences were uttered by every speaker in both the training and test sets, violating one of the suggested training and testing criteria.

All the experimentation was concerned with the effectiveness of adaptation on a new modified ANN architecture. It therefore would have made no sense using, initially, a wide variety of speakers from many dialect regions, since this would have made recognition difficult and left the possibility of clouding important subtle results. Consequently to

highlight the effects of any adaptation on the newly modified ANN architecture, all the speech data was selected from a single dialect region. The 7th dialect region, corresponding to the western geographical area of the U.S. was chosen for training and testing since it contained the most well-represented training set on the TIMIT CD-ROM. Also for exactly the same reasons used to select the dialect region, only speech data from 'male' speakers in dialect region 7 were used to train and test the new classifier. Male speakers made up ~70% of the TIMIT speech corpus giving 59 suitable male speakers for training and a choice of 15 male speakers for testing, all from the 7th dialect region. The 59 male speakers from the dialect region 7 training set on the TIMIT CD-ROM were collectively known throughout the experimentation as the 'Training Set'. The number of speakers in the 'Test Set' varied throughout the experimentation but were always made up from speech data belonging to the 15 male speakers in the dialect region 7 test set on the TIMIT CD-ROM.

3.3 Speech Pre-Processing

All the speech data was recorded in a noise-isolated recording booth and sampled at 16kHz with 16-bit quantisation. A sample of the TIMIT speech data is shown in Figure 3.3 which shows a frequency and time waveform plot of the utterance "She had your dark suit." As well as dialect and training/test set directories, the speech data was further catalogued into speaker directories. In order to identify the speaker of each utterance each directory was coded, where the first letter represented the speaker's gender (M for male and F for female), a further three letters represented the speakers initials and a single number was used avoid confusion between speakers with the same gender and initials. Within each speaker directory the recorded speech waveform files (.wav) had three associated transcription files (.txt, .wrd, .phn). The .txt file contained an account of the words the speaker uttered during each corresponding speech waveform. The .wrd and .phn files contained time-aligned transcriptions of the words and phonemes receptively. The time-alignment information referred to the boundaries of each word or phoneme in the spoken sentences allowing either the manual or automatic retrieval of selected sounds from any of the continuous utterances on the TIMIT CD-ROM.

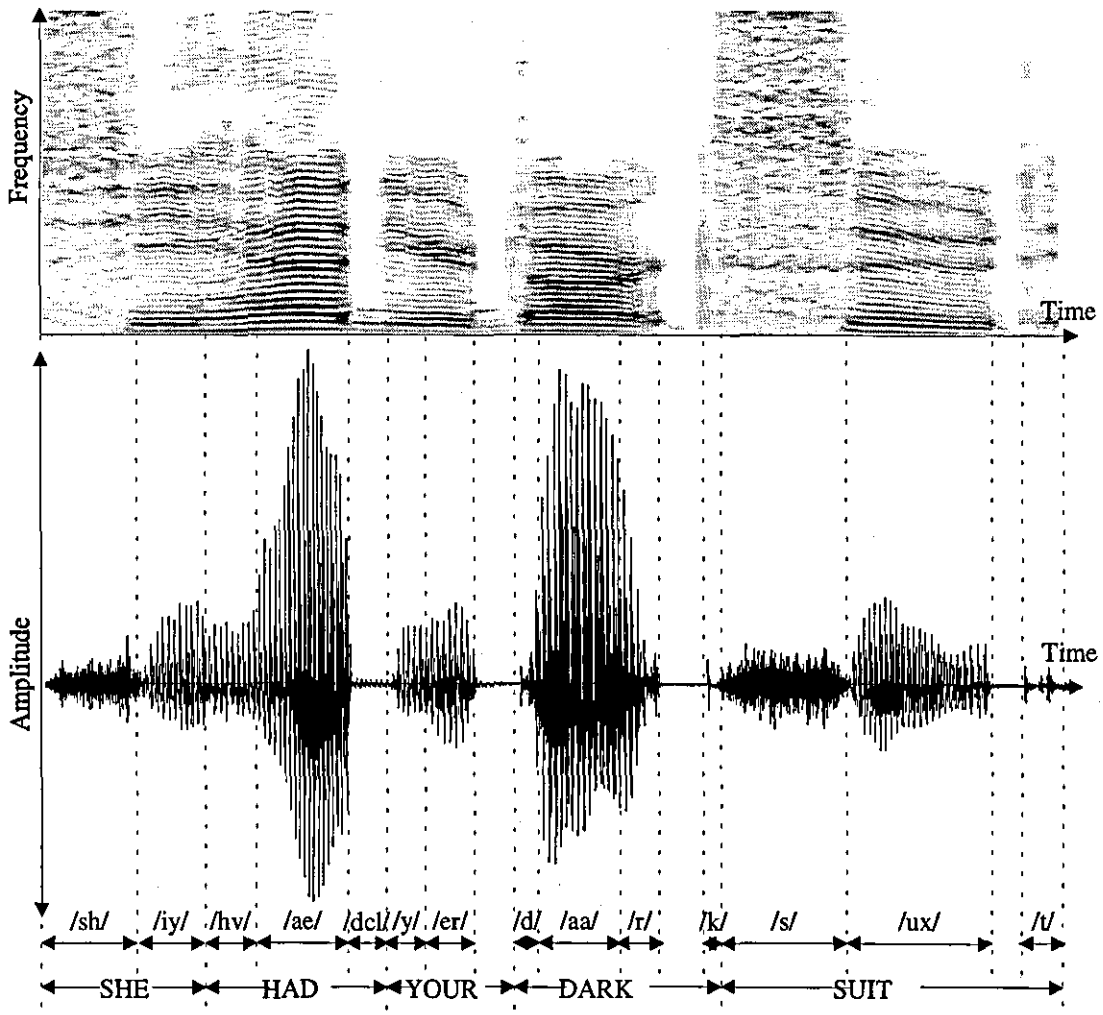


Figure 3.3 Spectrogram and Time-waveform of sampled TIMIT speech data

Due to the quantity of speech data, the retrieval of phoneme information was automated in terms of a look up table using 'Speech Tools' [5]. The 'Speech Tools' were a collection of software packages, complied for use on a UNIX machine, to specifically prepare and analyse TIMIT speech data. Using the .phn files, each of the desired phonemes were extracted from the corresponding speech waveforms to create new phoneme waveforms. Using batch files, this entire process was automated enabling the retrieval of all 13 American-English vowel phonemes from the 'Training Set' and 'Test Set'. However, it was at this stage that phoneme /UW/ was noted as having too few occurrences within the TIMIT database. It was therefore decided to omit phoneme /UW/ from all experimentation to avoid poor generalisation. The complete 'Training Set' was processed as one batch job whereas the 'Test Set' was processed, one speaker at a time, so as to keep phonemes from different speakers separated for experimentation purposes. For each phoneme class, the 'Speech Tools' automatically

concatenated all the relevant phonemes from the processed speech waveforms into a single phoneme waveform. Before any pro-processing of feature extraction could occur, each concatenated phoneme waveform had to be split back into single phoneme waveform files. This was achieved by writing a small software program which scanned the inputted concatenated phoneme waveform for specific periods of silence. Each of these periods of silence (strings of zeros in the file) were added by the 'Speech Tools' to separate each phoneme utterance. Between each period of silence, the waveform was copied to create a new single phoneme waveform. Due to the nature of speech, each of the phonemes were of differing duration. However the modified ANN architecture required each set of input data to be of equal length since the number of input nodes was fixed. To achieve this, each of the single phonemes were either cut off or had periods of silence added to them to achieve a fixed length phoneme file. Before this could be done, the ideal length of a phoneme file had to be investigated. If the file was too short then this would mean that vital phoneme information could be lost. If the files were too long, resulting in the majority containing some period of silence, then data resolution would be lowered. Over a short interval of time, ~10ms, speech is said to be stationary since during this time the mouths articulators have little time to move. However, for recognition of a vowel phoneme it is necessary to see how the waveform varies over a greater time period. For the phonemes from the TIMIT corpus, a period of ~70ms was deemed sufficient to allow recognition without the waveform becoming unnecessarily long. Therefore phonemes longer than this period were clipped and phoneme shorter were concatenated with zeros.

Before any features were extracted from the speech waveform, the signal needed to be pre-emphasised (see Section 2.2.1) to compensate for the -6dB/octave roll-off of voiced speech. By using an appropriate filter (i.e. equation (3.1)) the speech signal was given a +6dB/octave lift to ensure a similar dynamic range across the entire vocal spectrum. The filter used was :

$$y[n] = x[n] - 0.95[n-1] \tag{3.1}$$

One of the most common forms of speech pre-processing is linear prediction (see Section 2.2.4). Since all experimentation was primarily concerned with the performance of the modified classifier, linear prediction with its simple coding and well documented behaviour

was chosen as the most appropriate form of speech pre-processing. Linear prediction creates a set of linear predictor coefficients (LPCs) which can predict a speech signal reasonably accurately over stationary portions of speech. Since speech is only considered stationary for ~10ms, new predictor coefficients have to be generated to follow the varying nature of the speech signal. Consequently, the speech signal was split into overlapping windows of equal size and LPCs for each window generated. However, before LPCs were generated, the size and shape of the over-lapping windows had to be decided. With the speech signal sampled at 16kHz, it was important to ensure that each window represented at least 10ms. Hence, each window contained 256 samples equal to 16ms of speech. LPCs were created by predicting speech samples from a linear combination of past speech samples. The auto-correlation linear prediction method used in the experimentation (see Section 2.2.4) assumes that samples outside the finite window of speech are zero. This causes large prediction errors since the initial samples from the windows are trying to be predicted from samples that have been arbitrarily set to zero. Likewise the error can be large at the end of the interval because its trying to predict zero from samples that are non-zero. For this reason a window that tapered the segment of speech was needed, a 'soft window.' The Hamming window was chosen because it offered a good trade off between its tapered side lobes to reduce discontinuities at the edges and the large proportion of its energy held within the main lobe (see Figure 3.4 and equation 3.2). The hamming window is defined as :

$$y = \left\{0.54 - 0.46\cos\left(2\pi {}^{n}\!/_{N}\right)\right\}, \qquad 0 \le n \le N \qquad (3.2)$$

$$= \qquad , \qquad \text{otherwise}$$

To avoid missing any important features within the phonemes, the sliding windows along the speech waveform over-lapped halfway. Therefore, to achieve total time durations of ~70ms for each phoneme, with a time duration of 16ms for each window, 8 over-lapping windows were used giving a total duration of 72ms (see Figure 3.5).

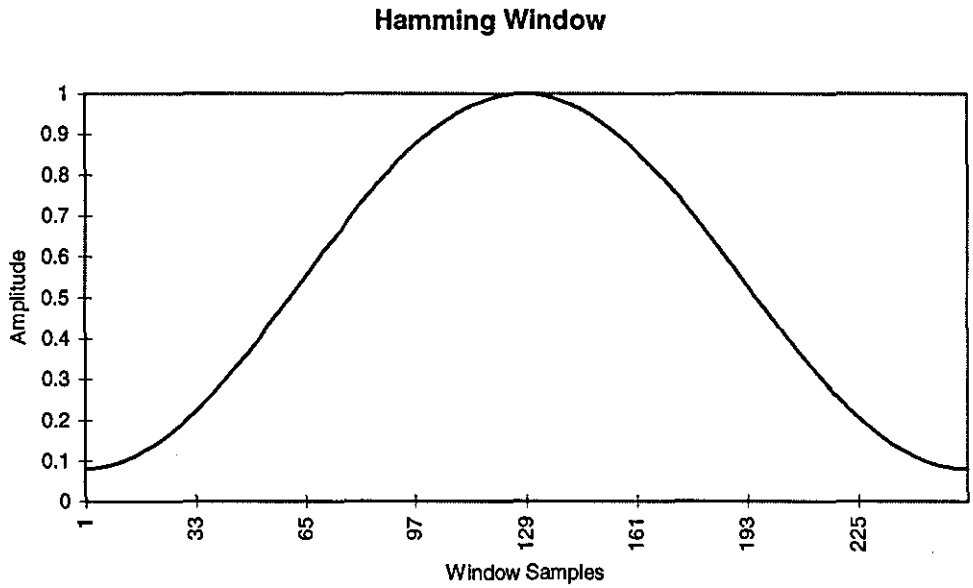


Figure 3.4 Graph showing the Envelope of the Hamming Window

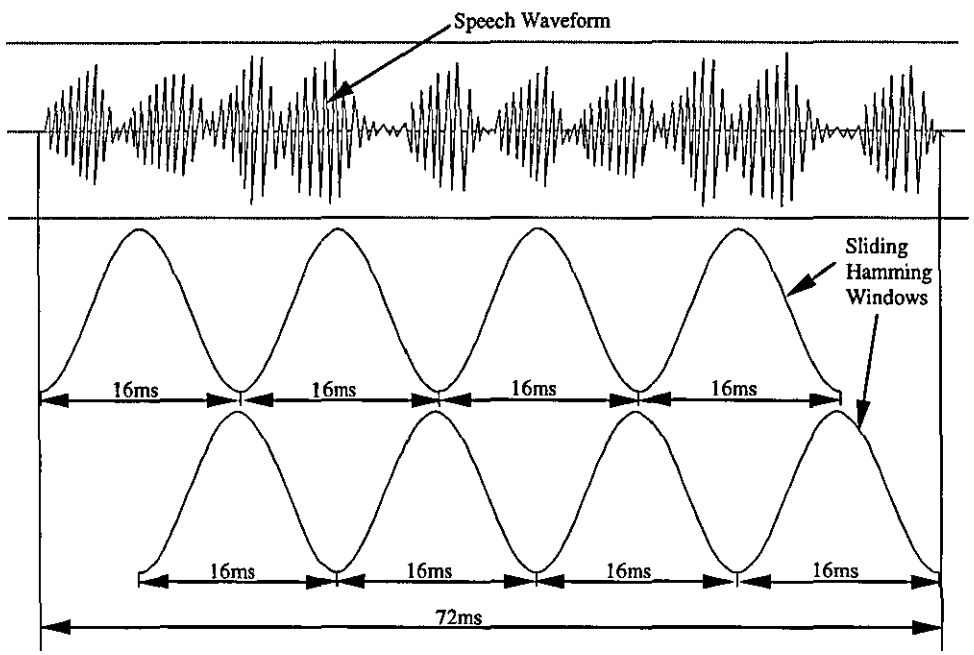


Figure 3.5 Graphical Representation of the Eight Hamming Over-Lapping Windows Sliding Along the Speech Waveform

After the phoneme speech waveforms were windowed, LPCs from each window were generated. The number of LPCs in each window reflected how accurately the coefficients matched the speech waveform. The idea behind linear prediction is to reduce the amount of speech data by eliminating unnecessary and redundant information. However, if too few LPCs are generated then important information is lost. Consequently there is a trade off between reducing the speech data and maintaining important information. As can be seen

from Figure 3.6 [6], as the number of LPCs increases so the RMS prediction error falls. The lower the RMS prediction error the more accurately the coefficients reflect the waveform. We therefore require a low RMS prediction error without the number of LPCs becoming too high.

The RMS prediction error initially falls quite dramatically for both the unvoiced and voiced speech. Vowel phonemes are produced solely by voiced excitation of the vocal cords and so only the curve for the voiced speech is of interest. After 12 LPCs, the curve is relatively flat and so having anything greater that 12 would be of no advantage, especially since 12 LPCs is adequate to reflect speaker characteristics within the waveform.

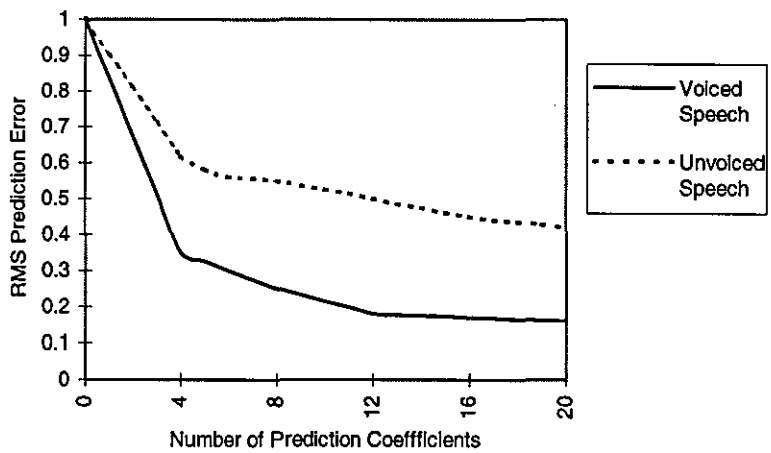


Figure 3.6 Variations of the RMS prediction error with the number of predictor coefficients, p.

Consequently, 12 LPCs for each window were chosen giving a total of 8x12=96 coefficients for each vowel phoneme. During the LPC procedure all the coefficients were normalised by dividing by the first. The first coefficient could therefore be eliminated since it was always equal to one. This left 11 LPCs for each window resulting in a total of 8x11=88 coefficients to be used by the classifier for training and testing.

3.4 Classifier Input-Data Format

Due to the selection of ANNs for creating the speech classifier, several software packages were investigated to model the ANN architecture (see Section 4.4). The package selected for experimentation was called the 'Stuttgart Neural Network Simulator' (SNNS) [7] and

required the data used for training, testing and adapting to be in a 'Pattern' format (Figure 6.4). A 'Pattern' file consists of a header containing the date, data type and quantity information, along with input information (the LPCs) and the desired outputs (which vowels the LPCs represent). The headers were generated using an SNNS software package called 'MKHEAD'. The headers were then appended to the LPC data along with desired output information.

3.5 Summary

The selection, generation and formatting of speech data for any recognition system is vital. Consequently great care was taken to select speech data for the experimentation that would yield clear results. To allow clear recognition and adaptation results, it was necessary to have speech data from speakers with a consistent gender and dialect. This reduced the amount of speech data available and so the main selection criteria for all aspects of the speech data was quantity. This led to the selection of dialect-region-7 male speakers. Despite this, a large training and test-set was still available with dialect region 7 representing geographically about a third of the entire United States. With all the speech data selected, processed and formatted, all emphasis is now given over to the classifier. Chapter 4 presents a review of neural networks classifiers, detailing their background, various architectures and learning rules. As mentioned in Section 3.4, Chapter 4 also investigates various software packages for modelling ANNs.

3.6 References

[1] GE Peterson and H.L. Barney, "Control Methods Used in a Study of the Vowels," J.Acoust. Soc. Am., vol. 24, pp.175-184, March 1952.

[2] L. Rabiner and B. Juang, "Fundamentals of Speech Recognition," 26-27, Ch2, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[3] J.E.Shoup, "Phonological Aspects of Speech Recognition," 125-138, Ch6 in Trends in Speech Recognition, W. A. Lea, Ed., Prentice-Hall, Englewood Cliffs, NJ, 1980.

[4] DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM, Oct 1990. Web Address: //www.ldc.upenn.edu/

[5] Speech Tools User Manual, Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology, August 1993.

[6] B.S.Atal and S.L.Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoust. Soc. Am., 50(2): pp637-655, August 1971.

[7] SNNS Stuttgart Neural Network Simulator User Manual, Version 3.1, University of Stuttgart, Institute for Parallel and Distributed High Performance Systems.

CHAPTER 4

Neural Networks

4.1 Introduction

Modern electronics are able to process and manipulate information both faster and more reliably than any human brain, and yet there are many scenarios where the human brain outstrips the most sophisticated machines with ease. Although operating millions of times slower than many computers, the human brain is able to deal with many involved tasks such as deciphering complex images and understanding continuous noisy speech. Using past experiences, it is also able to gain knowledge, aiding in the recognition and adaptation towards new information. These tasks and many others have all beaten the best machines and so there has been a concerted effort to try and understand the architecture and functionality of the human brain in an effort to emulate its behaviour and performance.

4.2 Biological Model

The human brain is an immensely complex structure containing an estimated 100 billion neurons all inter-connected, receiving and transmitting information via 1000s of synapses (the links between neurons). The neuron, a specialised cell that conducts nerve impulses, consists of a cell body, an axon and dendrites (see Figure 4.1). The exact way in which neurons interact with one another remains largely uncertain. However, we do know that a neuron sends its output to other neurons via its axon to weighted connections known as synapses which link to awaiting dendrites. A neuron sums all the signals (voltage potentials) received from its dendrites, both positive and negative, until a threshold value, the bias, is achieved and the neuron 'fires,' sending its signal to other neurons. This process occurs throughout the brain as millions of neurons simultaneously pass information around the

complex network, learning and adapting to new inputs by constantly altering the weighted connections and the biases.

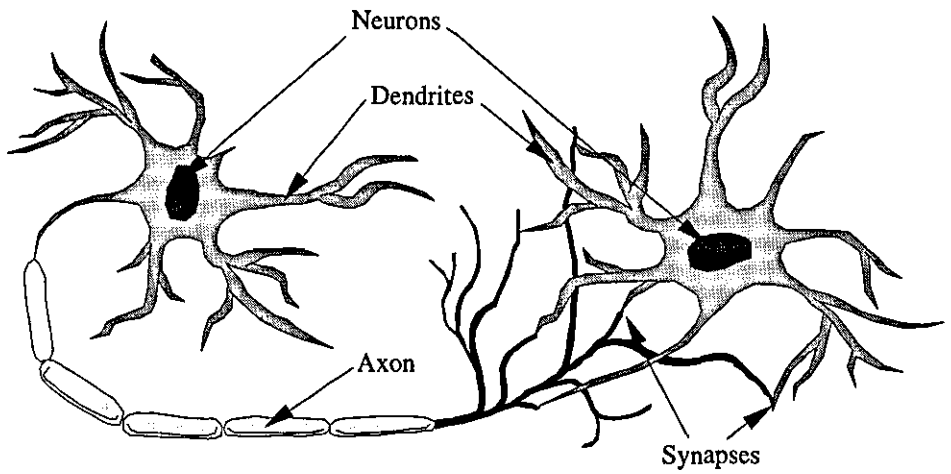


Figure 4.1 Graphical Representation of a Biological Neuron

4.3 Artificial Neural Networks (ANNs)

Due to the limited knowledge of the human brain's functionality and the restrictions of modern computational devices, the creation of an exact simulation has never yet been feasible. Consequently models of the human brain have always been simplistic, designed merely to introduce parallelism among highly interconnected nodes in order to perform non-linear transformations. By taking a rudimentary look at the anatomy of the biological model we can see the comparison with modern Artificial Neural Networks (ANNs) (see Figure 4.2).

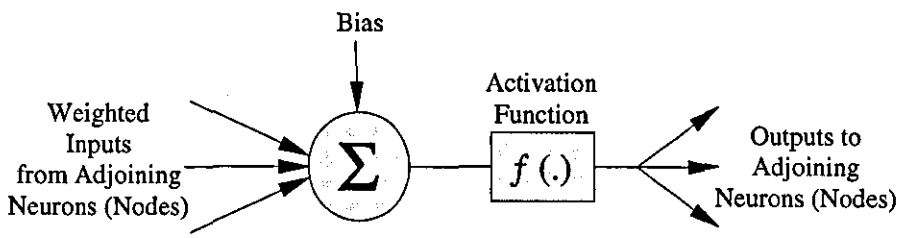


Figure 4.2 Simple model of an ANN

Figure 4.2 shows the neuron's inputs passing through the weighted connections (synapses) into the neuron cell, where all the influences are summed together. If the sum of the weighted inputs and the bias satisfies the activation function (i.e. the threshold is met) then the neuron fires, sending its signal to the inputs of the next adjoining neurons. These

artificial neurons (nodes) are usually, unlike the biological model, arranged into layers with the first and last layers acting as the ANN's inputs and outputs respectively. Each node often connects fully to every other node in an adjoining layer giving by far the most common and simplest arrangement of nodes in an ANN. However there have been, and still are many variations, offering differing architectures, activation functions and training methodologies. The first attempts to model brain function was by McCullogh and Pitts [1] back in 1943. Since then the development of the ANN has led a very chequered life with a paper by Minsky and Pappert in 1969 [2] reporting that two-layered networks were incapable of solving some simple problems. They further suggested that there was no reason to believe that networks with more than two layers would improve the situation, resulting in a huge decrease of interest in ANNs. It was not until 1986, when Rumelhart and Hinton [3] developed the back-propagation algorithm to enable the training of multi-layered networks, that ANNs began to show a sudden re-emergence of interest. Spurred on by its loose comparisons with the 'human brain,' this renaissance led to a vast array of architectures, training methodologies and application-driven networks being developed.

An ANN may be specified by 3 basic components; the activation function, architecture, and training method. The selection of each component plays a vital role in the behaviour of an ANN and depends very much on the application for which it is intended. Consequently, various choices from each component area were explored to produce a set configuration for the experimentation. Other than the architecture, the main criteria for selecting each component was its well documented usage and ease of implementation. This was to create a standard configuration, allowing the effects made by any architectural changes to be clearly highlighted.

4.3.1 The Activation Function

The activation function, $f(\alpha)$, is primarily a threshold regulator for firing a node in an ANN. Figure 4.3(a) shows how the activation function influences the combined weighted inputs and bias to produce the ANN node's output. The most basic activation function is the 'step function,' (Figure 4.3(b)), which acts as a discrete firing switch when the sum of the weighted inputs and the bias reach a threshold value h. Usually $h = 0$, and so the node fires when the

sum of the weighted inputs is equal to the node's bias. However, it is best if all the nodes have some non-linearity to enhance a network's approximation, classification and noise-immunity capabilities. The most often used non-linearity is the sigmoidal activation function because it is both continuous and differentiable, making its learning stage easier. Sigmoidal, meaning 'S' shaped and allows the outputs of a node to provide more than simply a classification; it can also be interpreted in terms of probabilities. Consequently, all nodes during the experimentation used the sigmoidal activation function.

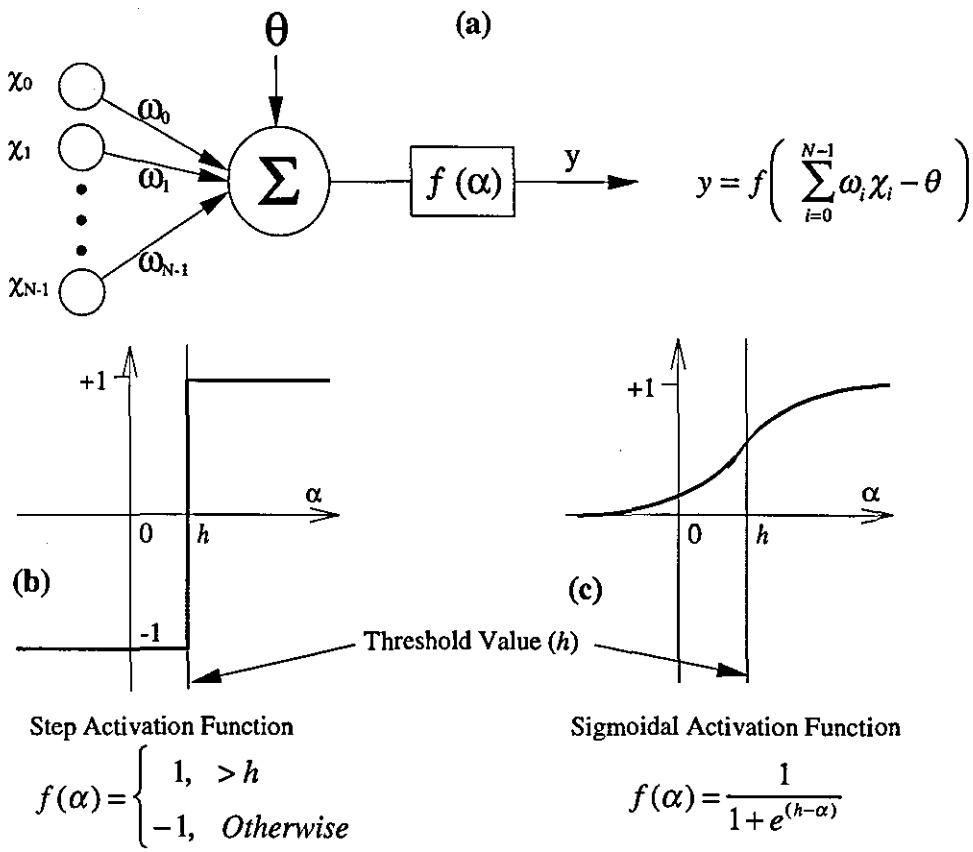


Figure 4.3 Graphical Representation of an ANN Node (a) Showing the Two Most Common Activation Functions $f(\alpha)$; the Step Activation Function (b) and the Sigmoidal Activation Function (c).

4.3.2 ANN Architecture

There are many kinds of ANN architectures which describe the way in which nodes are interconnected. Each application-driven architecture is designed specifically for the data type to be classified (binary or analogue-valued input), the training methodology (supervised or unsupervised) and the complexity of the classification task (single or multi-layered). There

exists three standard and well known architectures which can be used for static pattern classification; the recurrent network, the feed-forward network and the self-organising network. From these three groups of topologies, the Hopfield and Hamming 'recurrent' networks, the multi-layered perceptron (MLP) 'feed-forward' network and finally the Kohonen 'self-organising' network will be discussed.

Hopfield Network

The Hopfield network, illustrated in Figure 4.4, is a feed-back 'recurrent' network in which the inputs to each node include both inputs as well as outputs. Requiring binary inputs, the Hopfield network is ideal for systems such as OCRs that use discrete black and white pixels as input elements from ASCII text.

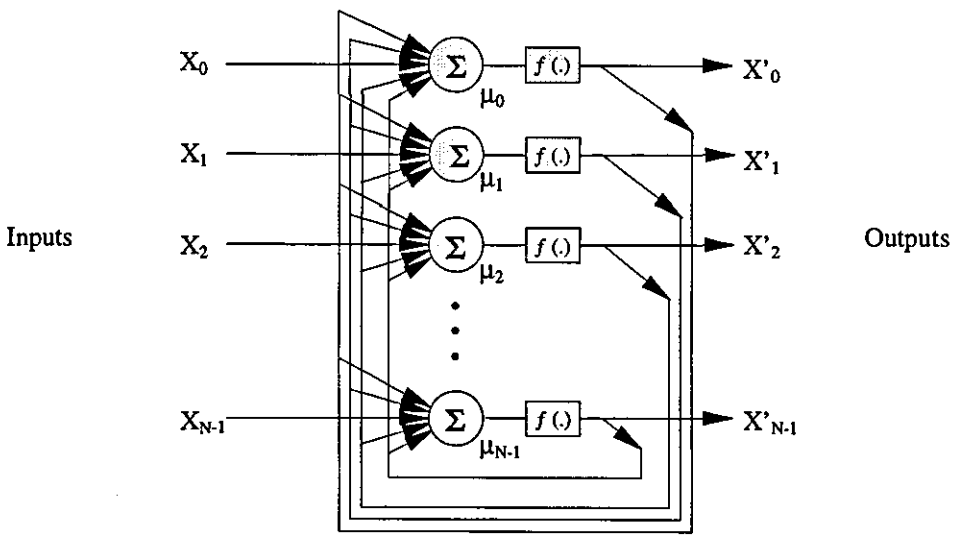


Figure 4.4 A Graphical Representation of a Hopfield ANN

Used as either an associative memory or a classifier, the network is trained using class patterns, each represented by either a +1 or -1 at each node input, X_0 - X_{N-1}. The network is initially trained for each class by converging the weights in the network, situated at every node input, using the Hopfield network algorithm [4]. When issued with an unknown pattern, the network produces an output, which during each indexed time step, is fed back into the input for another iterative pass. The weights within the network are constantly updated until they fully converge, no longer producing any changes between successive iterations. The pattern produced after this convergence is the networks final output. The

strength of this type of network is its ability, after several iterative passes, to clean up noisy versions of a class and produce the correct class pattern at the output. If used for classification purposes, this output can be compared with the stored classes to determine which class the output pattern matches. Although Hopfield networks do often perform well and are able to produce the correct outputs from noisy inputs, they do have two major limitations. Firstly the number of patterns that can be stored within the network is severely limited. If the number of classes begins to approach a sixth the number of nodes, then new fake patterns can be generated resulting in a 'No Match.' The second limitation is when two stored classes within the network are too similar. This can cause the network to converge inaccurately, producing an incorrect class pattern to appear at the output.

Hamming Network

The Hamming network [3] illustrated in Figure 4.5, is similar to the Hopfield network in that it requires binary inputs that for some selected nodes include both inputs and outputs. However, the network's output is quite different in that it does not produce a stored representation of the classified input, merely an indication of what the input is thought to be.

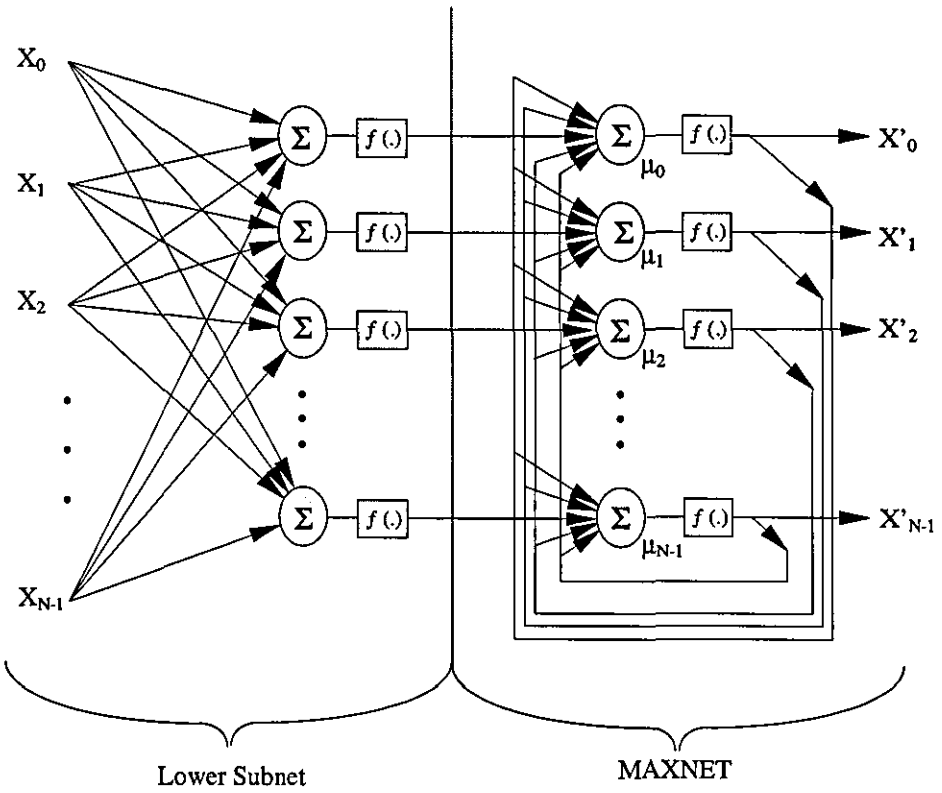


Figure 4.5 A Graphic Representation of a Hamming ANN

Initially the weights and thresholds are set in the lower subnet such that the output of the middle nodes are equal to N minus the Hamming distance for each training pattern where N is the number of input elements and the Hamming distance is the number of bits in the input which do not match the corresponding training pattern. Consequently a close match between an input pattern and a trained pattern results in the highest score at the corresponding output of the middle nodes. The upper subnet is a MAXNET which takes the outputs from the middle nodes and iterates until, at the output of the MAXNET, only one node is positive. Creating a 'Winner-Take-All' methodology, the single positive node corresponds to the classification of the input pattern. The Hamming network has an advantage over the Hopfield network in that it requires fewer connections and consequently fewer weights. In a network with 100 inputs and 10 classes, the Hamming network would only require 1,000 connections compared to ~10,000 for a Hopfield Network. Furthermore, the Hamming network does not produce new fake patterns when the number of classes is too great, resulting in a 'No Match' scenario.

Multi-Layered Perceptron

Multi-Layered Perceptrons (MLPs) are feed-forward networks with one or more layers of nodes between layers of input and output nodes as illustrated in Figure 4.6.

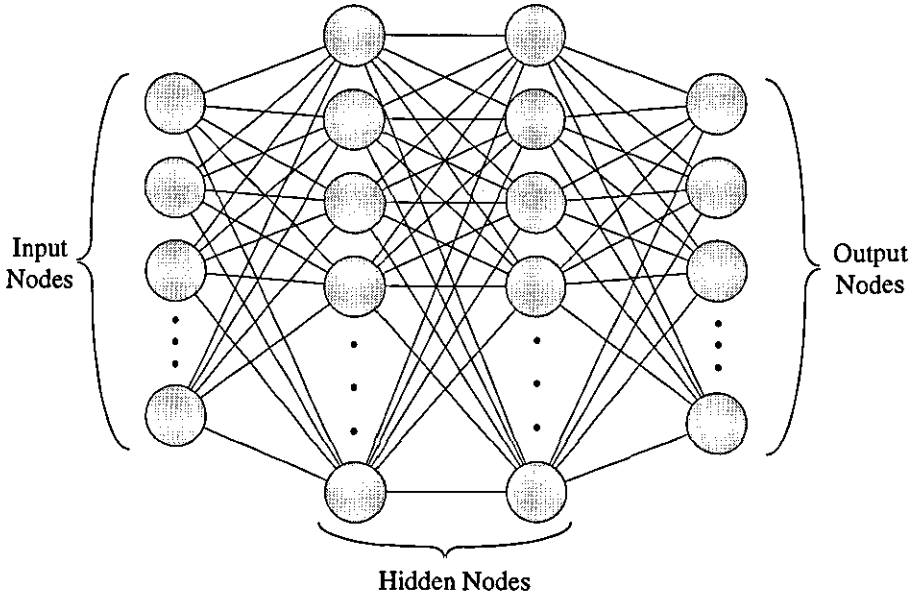


Figure 4.6 A Multi-Layered Perceptron (MLP) with each Node Representing a Processing Unit with Activation Function

These additional layers, containing hidden nodes, are connected via weights to the input and output nodes. The number of layers a MLP is said to contain, relates to the number of weight layers, not the number of node layers. Consequently an MLP with an input, output and single hidden layer will be referred to as having two layers since an input layer contains no weights. The power and versatility of an MLP lies with its use of non-linearities in each node. It is this non-linearity (activation function) that enables an MLP to create complex decision boundaries. If an MLP had linear nodes, then a single-layer network with suitably chosen weights would be able to perform exactly the same calculations. The complexity of the decision boundaries is governed not only by the non-linear nodes, but also by the number of layers (see Figure 4.7).

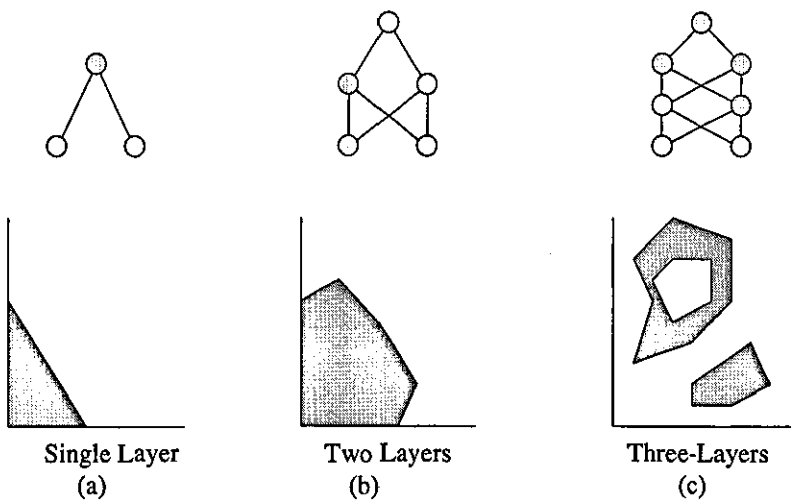


Figure 4.7. Types of decision boundaries which can be formed by networks having threshold activation functions and a various number of layers.

A single layer feed-forward network is only able to classify static input patterns into two classes by forming a half-plane decision region. As the number of layers increases, then so the complexity of the decision regions increases. However, networks with more than three layers are usually uncommon since three-layered networks should be able to produce any complex decision regions with arbitrary shapes, [3]. Most ANN architectures used for speech recognition are designed to classify static patterns. However, speech is inherently dynamic in nature and so a modified ANN structure, the 'Time Delay Neural Network, (TDNN)' was created [6]. The TDNN, figure 4.8, extends the input to each input node to include N adjacent speech frames, each with a time delay, D, of Δ seconds. This enables each input to cover a range of ΔN seconds, highlighting any phonemes variations in a speech signal.

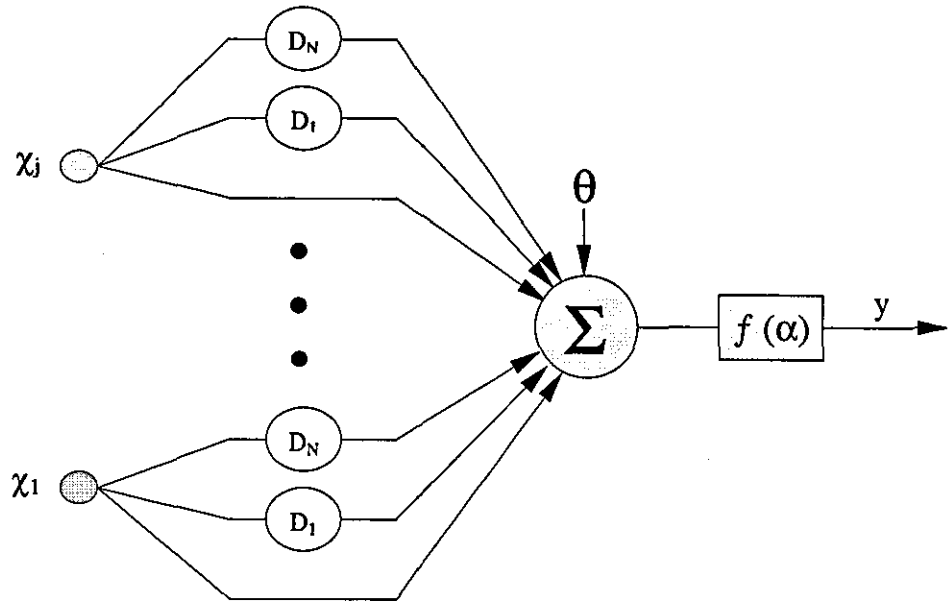


Figure 4.8 A graphical Representation of a Time-Delay Neural Network (TDNN) with varying time-delayed inputs, (0 - D_N).

Kohonen Network

The organisation of the neurons within the human brain, although genetically predetermined for low-level functions, often reflects the characteristics of external stimuli during learning. This principal of honing a neural architecture into a more efficient and input specific network was adopted by Kohonen with a self-organising feature map [7]. This self-organising network consists of input nodes connected via weights to output nodes, usually arranged in a two dimensional grid, figure 4.9.

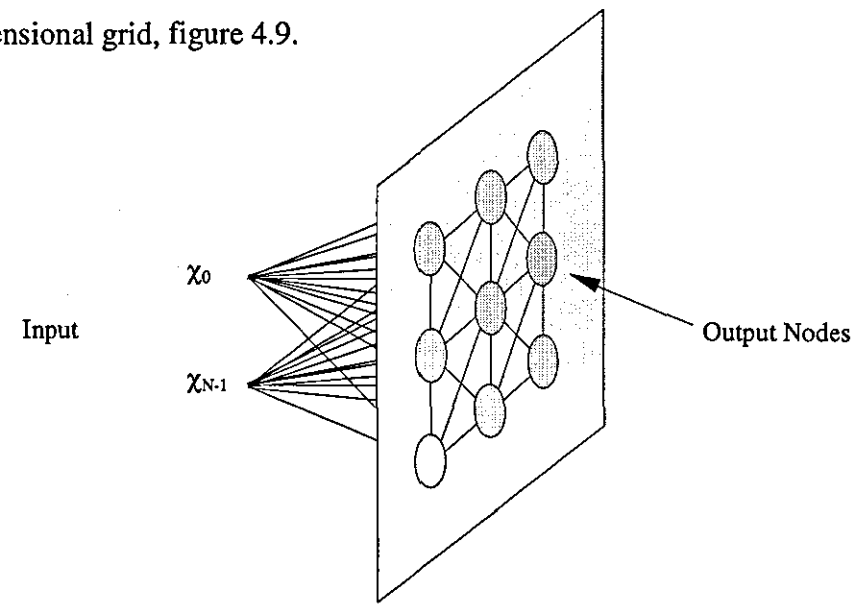


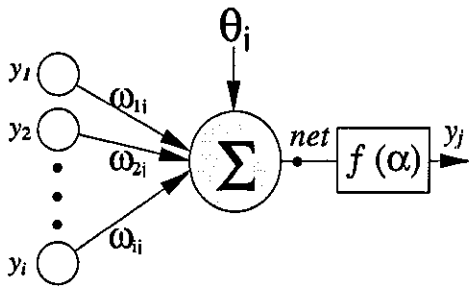
Figure 4.9 A Kohonen Self-Organising Network

During each iteration of the unsupervised training procedure, input patterns are fed into the network and the error distances between each input and corresponding node-weight calculated. The nearest node has all weights within its neighbourhood modified to enable the model to be more responsive to the applied input. After each training iteration, the initially large neighbourhood slowly decreases in size. Furthermore the update rate for the weights reduces with each iteration until it reaches zero, when the network is then able to associate between different classes using corresponding regions on the output grid. Because of the slow way in which the weights adapt during training and the fixed number of classes, the Kohonen network performs well with noisy inputs.

4.3.3 Training

The primary objective when training an ANN is not to learn an exact representation of the training data itself, but to build a statistical model of the process which generates the data. This is achieved by setting the weights and bias values within the ANN such that the network error is minimised for a representative training set. The network error is the sum of the differences between the desired and actual outputs. The algorithms used to achieve this goal vary depending on the ANN architecture, although by far the most common is the back-propagation (BP) algorithm used for training MLPs. In fact the success of the MLP is credited to Rumelhart [8], who in 1986 proposed the efficient back-propagation training algorithm for optimising the weights and thresholds of multi-layered networks.

The first stage of training an MLP is the initialisation of the weight and bias values with some small random values. After initialisation, supervised training begins with training pairs containing input/output information. The input patterns are presented to the network repeatedly and, on each presentation, the states of all nodes computed as the patterns pass through the network to the output nodes. For example let us a consider a fully connected MLP with each node using a sigmoidal activation function. For each node in the network the output y would be equal to :



$$y_j = 1/\left[1+\exp\left(-net_j\right)\right] \qquad (4.1)$$

where

$$net_j = \sum_i \chi_i \omega_{ij} - \theta_j \qquad (4.2)$$

where j is the number of nodes in the present layer and i is the number of nodes in the previous layer. The BP algorithm uses a gradient descent algorithm for updating the weights and bias values, based on the mean squared differences between the actual and desired output values. The mean squared error for a single input/output pattern is:

$$E = \frac{1}{2}\sum_j \left(t_j - y_j\right)^2, \qquad (4.3)$$

where t_j is the desired output value for the j^{th} output node.

The purpose behind the training is to alter the weight ω_{ij} and bias θ_j values within the network such that the error E for each input/output pattern is minimised.

In order to minimise E with respect to ω_{ij} and θ_j, we must apply the chain rule as follows :

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial \omega_{ij}} \qquad (4.4)$$

$$\frac{\partial E}{\partial \theta_j} = \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial \theta_j} \qquad (4.5)$$

Now we introduce the notation

$$\delta_j \equiv \frac{\partial E}{\partial net_j} \qquad (4.6)$$

Using equation (4.6) with (4.4) and (4.5) we obtain :

$$\frac{\partial E}{\partial \omega_{ij}} = \delta_j \cdot \frac{\partial net_j}{\partial \omega_{ij}} \qquad (4.7)$$

and

$$\frac{\partial E}{\partial \theta_j} = \delta_j \cdot \frac{\partial net_j}{\partial \theta_j} \tag{4.8}$$

Furthermore, from equation (4.2) we can obtain

$$\frac{\partial net_j}{\partial \omega_{ij}} = \chi_i \tag{4.9}$$

and

$$\frac{\partial net_j}{\partial \theta_j} = -1 \tag{4.10}$$

Next we want to calculate δ_j. Applying the chain rule again we get :

$$\delta_j = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \tag{4.11}$$

where $\partial y_j/\partial net_j$ is equal to the derivative of the activation function and $\partial E/\partial y_j$ is equal to the derivative of the mean squared error (4.3).

Fortunately the derivative of the sigmoidal activation function has a simple form

$$\frac{\partial y_j}{\partial net_j} = y_j\left(1 - y_j\right) \tag{4.12}$$

and

$$\frac{\partial E}{\partial y_j} = -(t_j - y_j) \tag{4.13}$$

However, equation (4.13) only applies if the node is in the output layer, otherwise the error is propagated back from the nodes in the upper layer, i.e.

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial y_j} \tag{4.14}$$

where k is the number of nodes in the next upper layer.

For the next upper layer we obtain

$$net_k = \sum_j y_j \omega_{jk} - \theta_k \tag{4.15}$$

Differentiating this equation with respect to y_j gives

$$\frac{\partial net_k}{\partial y_j} = \omega_{jk} \tag{4.16}$$

Using the notation from (4.6) we can get

$$\delta_k = \frac{\partial E}{\partial net_k} \tag{4.17}$$

Therefore if j is not an output node, a weighted sum of the partial errors can be back propagated from the nodes in the above layer to give

$$\frac{\partial E}{\partial y_j} = \sum_k \delta_k \omega_{jk} \tag{4.18}$$

Now that all the partial derivatives have been defined, the weights and bias values can be adjusted. However, some gain term η, usually set to between 0 and 1, must be used to control the adjustment of these values to avoid instability. This gives the changes in weight and bias values as

$$\Delta \omega_{ij} = -\eta \cdot \frac{\partial E}{\partial \omega_{ij}} \tag{4.19}$$

and

$$\Delta \theta_j = -\eta \cdot \frac{\partial E}{\partial \theta_j} \tag{4.20}$$

Since the BP algorithm is only a first order gradient descent technique, controlled convergence if often slow. It is also prone to being trapped in local minima on the error

surface, preventing the weights and bias values from settling. To avoid this problem, a momentum term can be added to encourage a decreasing network error to pass over any local minima. Equation (4.21) shows the BP algorithm with the momentum variable α.

$$\Delta\omega(t) = -\eta\frac{\partial E}{\partial\omega(t)} + \alpha\Delta\omega(t-1) \tag{4.21}$$

where t is the number of training iterations.

4.4 Neural Network Software

Although hardware offers the benefits of utilising the parallel processing power of ANNs, its rigid design makes it unsuitable for honing and testing new ANN architectures. Consequently software, although slower, offers the adaptability necessary when designing a new ANN. There are many ANN software packages on the market which offer a wide variety of modelling procedures, graphic interfaces, numerous training procedures and specialist ANN tools. When deciding on the software package to model the ANNs for experimentation, there were three main criteria used for selection. First the package had to permit the modelling of complex architectures, allowing the positioning of any nodes of any type to connect to any other. Second a graphical interface was necessary to allow the visualisation of the ANN since the construction of complex ANN architectures using a text-driven package can be difficult and prone to incorrect data entry. Finally, the software package needed to include a variety of training algorithms, permitting large amounts of speech data with result analysing tools to clearly display the ANN's behaviour. For experimentation three popular software packages were chosen for evaluation; MATLAB Neural Network Toolbox, Aspirin/MIGRAINES Software tools and the Stuttgart Neural Network Simulator (SNNS), and each checked against the above criteria.

4.4.1 MATLAB Neural Network Toolbox

MATLAB [8] offers a powerful environment in which a speech signal can be pre-processed, classified and analysed in one operation. This is because, as well as including the neural

network toolbox, MATLAB also includes signal processing and statistics functions. This allows great flexibility during any stage of the speech recognition process, making it ideal for experimentation. However, the MATLAB version available, Version 3.0, included only a crude neural network toolbox. Although it included all the common activation functions, architectures and learning algorithms, its scope for adapting these parameters to produce a customised complex network was limited. Since this investigation into the neural network toolbox the latest version, Version 4.2c, has overcome these restrictions with software that now allows all network properties to be easily customised and collected into a single network object.

4.4.2 Aspirin/MIGRAINES Software Tools

Designed in 1986, the software consists of two elements, Aspirin which is a declarative language used to describe arbitrarily complex neural networks and MIGRAINES which is an interface for evaluating and interacting with a neural network simulation. Using a high level language, Aspirin attempts to describe any network architecture and a number of simulation routines. This is possible, according to the manual [9], by allowing each node in a destination layer to be connected only to a subset of the nodes in the source layer. This description code is then complied and linked to the 'MITRE Interactive Graphical Research and Investigation Neural Network Emulation System' (MIGRAINES) package. MIGRAINES probes the generated ANN using available analysis tools allowing a researcher to visually illustrate its performance. However, the description code for the ANN was actually very restrictive, preventing the generation of many complex architectures. Unfortunately this was not explained in the supplied text, resulting in a lot of wasted time. It also had limited learning algorithms and the graphical interface was crude. Consequently, after much investigation, the Aspirin/MIGRAINES software was found to fulfill none of the required criteria

4.4.3 Stuttgart Neural Network Simulator

Since 1989 the Institute of Parallel and Distributed High Performance Systems at the University of Stuttgart has developed and constantly updated the Stuttgart Neural Network

Simulator (SNNS) [10]. The objectives of the project were to create an efficient and flexible simulation environment for research on ANNs. Consisting of two main components, the kernel performing all internal data operations and the XGUI graphical interface, SNNS allows the creation and manipulation of various ANNs in a clear visual way. Containing over 10 types of ANN architecture, 33 learning algorithms, 25 weight update functions and 24 weights initialisation functions in the 1995 version 4.1, SNNS is also very comprehensive. With on-line help and easy-to-use network creation tools, complex ANNs are created quickly and easily with careful thought given over to inexperienced users. The package, used under 'X-Windows,' offers several windows including a control panel, a network creation tool, a network error graph, weight display and information panel (see Figure 4.10). The package also included several analysing tools for statistically analysing the results and an inversion display which presented ideal input patterns necessary for specific network outputs. SNNS was clearly the best of the three packages chosen for evaluation and was therefore selected for the experimentation. However, once experimentation had begun, some of SNNS's limitations were experienced requiring re-writing of the kernel and XGUI source code (see section 6.3.3). The software also crashed occasionally in certain instances, but was an insignificant problem against the benefits of its versatility and performance.

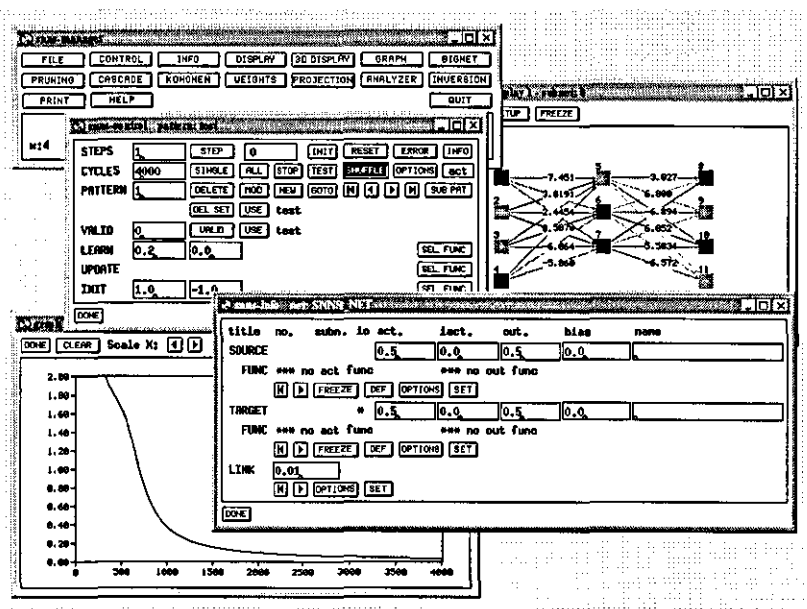


Figure 4.10 SNNS with some of the Windows available for Modeling, Training and Testing ANNs.

4.5 Summary

The selection of the architecture, activation function and learning rule plays a vital role in the behaviour of the network and the application for which it is meant. Several types of ANN components were investigated to enable the selection of the most suitable configuration for experimentation. The following chapter includes the selection process of the ANN configuration for experimentation and introduces the One-Class-One-Network architecture.

4.6 References

[1] W.S. McCullogh and W. Pitts, "A logical calculus of the ideas of immanent in nervous activity," Bull, Math., Biophys., 3, pp115-133, 1943.

[2] M. Minsky and S. Pappert, "Perceptrons," Cambridge, MA: MIT Press 1969.

[3] R.P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, pp.4-22, April 1987.

[4] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Natl. Acad. Sci, Vol 81, pp.2554-2558, April 1982.

[5] G.A. Carpenter and S. Grossberg, "Neural Dynamics of Category Learning and Recognition: Attention, Memory Consolidation and Amnesia," Brain Structure, Learning and Memory, AAAS Symp Series, 1986.

[6] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," Technical Report TR-1-0006, ATR Interpreting Telephony Research Laboratories, 1987.

[7] T. Kohonen, "Self Organization and Associative Memory," Springer-Verlag, Berlin, 1984.

[8] MATLAB Version 4.2c, The Math Works Inc. Address: The Math Works, Inc, 24 Prime Park Way, Natick, MA. 01760-1500

[9] Leighton, R.R, The Aspirin/MIGRAINES Software Tools, User's Manual, Release V5.0, The MITRE Corporation, 1991. Address: MITRE Signal Processing Lab, 7525 Colshire Dr, McLean, Va. 22102, USA.

[10] Stuttgart Neural Network Simulator, User Manual, Version 4.1, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart, 1995. Web Address: //www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html/

CHAPTER 5

Neural Network Architecture Selection

5.1 Introduction

Selecting the activation function, architecture and training of an ANN is a very application dependent procedure. For the purpose of experimentation, the ANN configuration had to fulfill certain criteria. The first was to select a common network that could be easily modified. It needed to be a common network to help ensure that the effects of any modifications could be clearly explained and documented. Also, due to the nature of the speech data, it had to be able to deal with analogue-valued inputs. This meant that ANNs that dealt solely with binary inputs could be eliminated. Finally the speech data was in the form of training pairs, containing both input and output information. This was purely through design so that the network could be adapted to a specific speech sound and the effects clearly noted. As a result, the training had to be supervised and so ANNs using unsupervised training algorithms were also eliminated from selection. Using the taxonomy tree [1] (see Figure 5.1) the remaining architecture that fulfilled all the criteria was the Multi-Layered Perceptron (MLP).

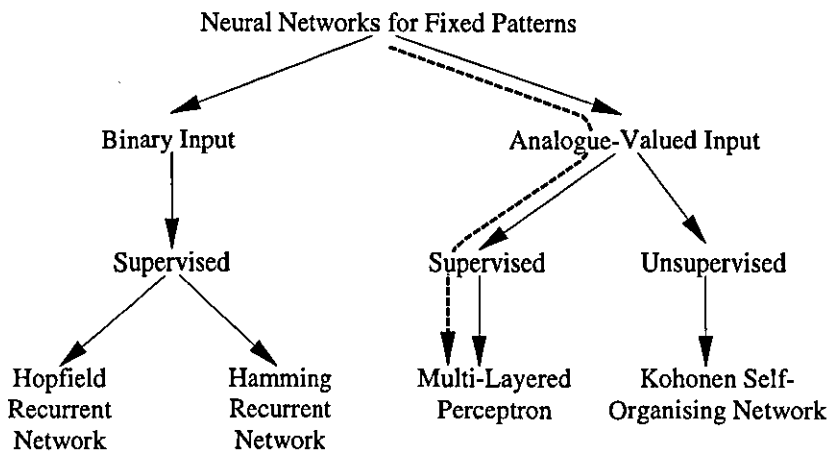


Figure 5.1 A Taxonomy tree showing four common ANN architectures

Feed-forward MLPs commonly use the well documented and easily applied back-propagation (BP) training algorithm. As a result the BP, along with the common sigmoidal activation function, was selected for experimentation, creating an ANN configuration that was well understood, adaptable and easy to model. This enabled the successful application and testing of new dynamic adaptation techniques.

5.2 Multi-Layered Perceptrons (MLPs)

There is no one solution when it comes to deciding on the number of nodes, layers and connections of an MLP architecture. However, there are some guidelines when creating an MLP for a specific application. The first of these application dependent parameters is the number of input, output and hidden nodes. One input node is necessary for each input data element and one output node for each class to be classified. Although some basic rules apply to the number of hidden nodes, a lot is down to experience, trial and error. This also applies to the number of hidden layers which is very much down to the complexity of the classification problem. The greater the number of the layers in an MLP, the more complex the decision boundaries (see Figure 4.7). Although too few layers make the decision boundaries over simplistic, too many layers can prevent the classification of a new acoustically similar utterance by making the trained network to input specific. Once the number of each node type has been decided, the connections between them is investigated. Most MLPs have fully-connected layers where the output of each node joins the inputs of every other node in a succeeding layer. However this is not always the case, since the connections can be used to split an ANN into subnets creating architectures such as the One-Class-One-Network (OCON).

5.3 The One-Class-One-Network (OCON)

A large fully-connected network can potentially contain many hundreds of neurons, each connected via weights to many others. This can make the training and adapting of such a network a long and difficult task. In addition, fully connected networks are prone to cross-class interference. Cross-class interference occurs when adapting towards a single class in a multi-class network, inevitably altering shared weights. As the network gets larger the

interference increases, drastically degrading the convergence rate of the shared weights due to the influence of conflicting signals. This can lead to, after adaptation towards a single class, the impaired classification for the remaining classes within the network. To eliminate these problems, I.C.Jou et al [2] proposed a new neural network architecture called the One-Net-One-Class. The same principle was later taken on by S.Y.Kung[3][4], who named the architecture the 'One-Class-One-Net' or the 'OCON' for short. The OCON is similar in design to that of a conventional MLP, only each class has its own dedicated subnet containing a single output neuron. This is illustrated in Figure 5.2. which shows a fully-connected network which is partitioned into three subnets by eliminating all the cross-class connections in the upper layer.

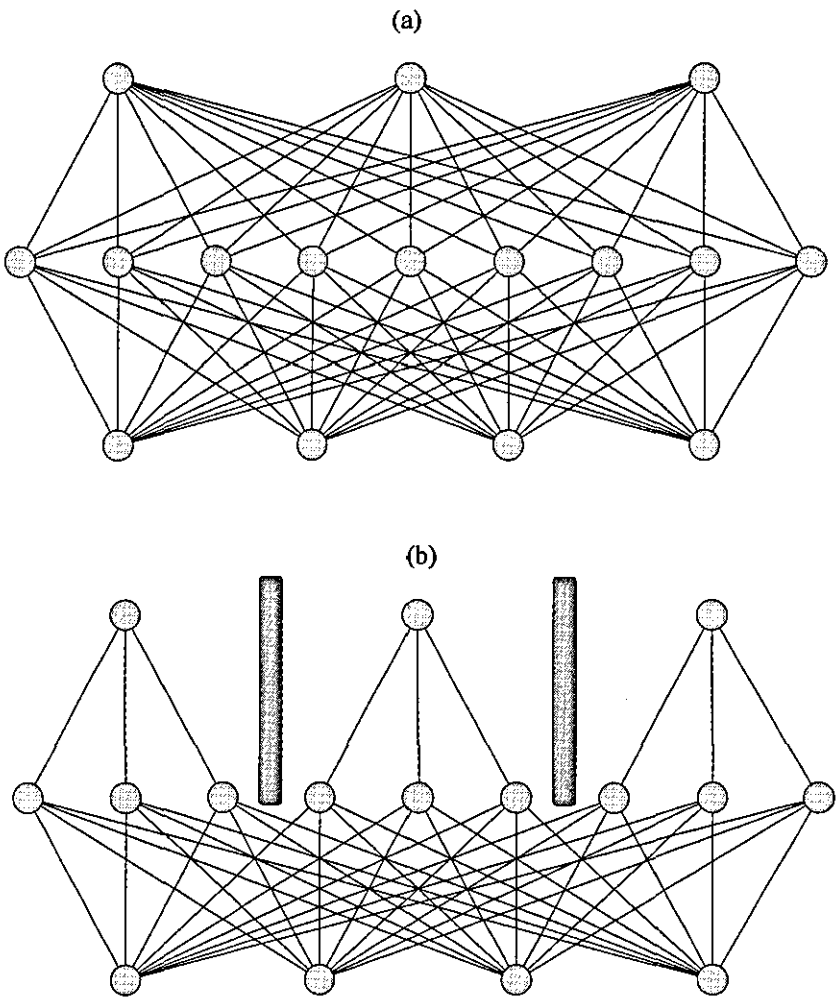


Figure 5.2 (a) A fully-connected MLP architecture. (b) A One-Class-One-Network (OCON) Neural Network Architecture

Each OCON subnet is specialised for distinguishing its own class from other patterns, resulting in fewer nodes being required in the hidden layers. Even if an OCON has the same

number of nodes as an MLP, it has fewer weights as highlighted in Figure 5.2 where the OCON has 45 weights compared with the MLP's 63. This consequently reduces both the training and recognition time, which becomes more significant as the number of output classes increases.

I.C.Jou first used the OCON architecture in 1991 for optical character recognition (OCR). Using 16 orientation angles to describe each written character, 36 alpha-numeric classes were used for recognition offering almost 10% improvement over a conventional network. Later S.Y.Kung [4] also applied the OCON architecture to OCR, achieving a training accuracy of 99.5% compared with 94% from a conventional MLP. Such architectures have also been used for texture classification, ECG analysis and speech recognition. Speech recognition systems, using OCONs, included the classification of mandarin speech syllables and isolated English words with a hybrid TDNN and OCON structure [5]. The OCON architecture has offered improved results in many areas of data classification although, in the field of speech recognition, has yet to be applied to phoneme speech recognition.

5.4 Comparative Study of MLP versus OCON

To test the performance of the OCON architecture on speech, a comparative study with the more conventional MLP was set. Although primarily concerned with the performance on an adapted class within a network, the effect the adaptation had on the remaining unadapted classes was also of importance. To emphasise any inter-class adaptation, the MLP and OCON architectures were represented by three networks each, each receiving a distinct group of acoustically similar vowel phonemes. These phoneme groups corresponded to the position of the tongue hump in the oral cavity during their production, 'front', 'middle', and 'back'. The vowel phonemes within the front, middle and back groups are shown in Figure 3.1. For each phoneme group, MLP and OCON networks (see Figure 5.3) were modeled using the Stuttgart Neural Network Simulator (SNNS), (see Section 4.4.3). All the networks contained the same number of input nodes, 88, dictated by the number of input coefficients representing each speech utterance (see Section 3.3). The total number of output nodes for each network was dependent on the phoneme group, five phoneme classes for the front and back and three phoneme classes for the middle. Finding the number of hidden nodes is often a case of trial and error, and for many applications is often a guess. Consequently, an educated guess of 15

hidden nodes was made for both the MLP and OCON networks. Finally the six networks, with every node using the sigmoidal activation function, were modelled with fully connected adjoining layers, except for the hidden and output layers of the OCON architecture as shown in Figure 5.3b.

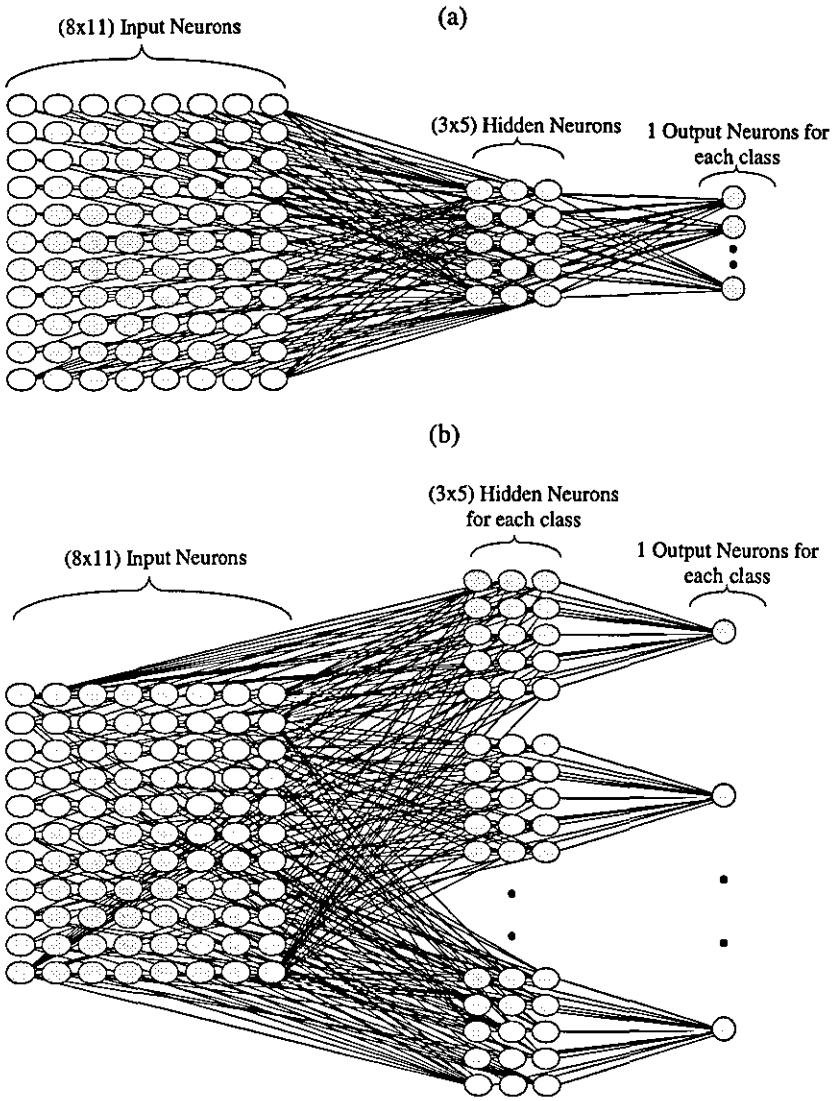


Figure 5.3 (a) Fully connected MLP architecture. (b) Fully connected OCON architecture.

Each network was trained with male TIMIT speech data from dialect region 7, the 'test set' (see Section 3.2). The weight and bias values within the networks were initially randomised and the standard back propagation algorithm used to train the networks towards the speech 'training set,' producing the six 'base-classifiers' necessary for the experimentation. The 'test set' comprised of the vowel phoneme data from the 15 male speakers of the TIMIT dialect region 7. Since there was only interest in intra-speaker effects and not inter-speaker

effects, all the speech data from every test speaker was amalgamated and categorised with respect to its phoneme content. The networks were then ready for adaptation and testing, but before that could occur, a single common back-propagation learning-rate for both the MLP and OCON networks had to be found.

Learning Rate

Each of the six base classifiers was adapted towards their relevant phonemes using speech data from all the male speakers within in the TIMIT test set. This was the same as the speech data to be used for the MLP versus OCON comparative study. Using the standard back-propagation algorithm, six learning rates from 0.1 to 2 were used to adapt towards each of the phoneme classes. Each phoneme class, one at a time, was presented to the relevant network and adapted for a total of 20 cycles. After each of the adaptation cycles, the recognition rate was recorded and after the twentieth cycle the network weights and bias' were reset to the initial base classifiers values ready for the next adaptation procedure. Average network-error results over the six base classifiers were calculated for all twenty adaptation cycles for each of the six learning rates, Figure 5.2. The graph shows an improvement in the rate of adaptation as the learning rate increased. However, above a learning rate of 0.5, this improvement reduced and the adaptation began to becomes unstable. Therefore to achieve the fastest adaptation without erratic results, a back-propagation learning rate of 0.5 was selected.

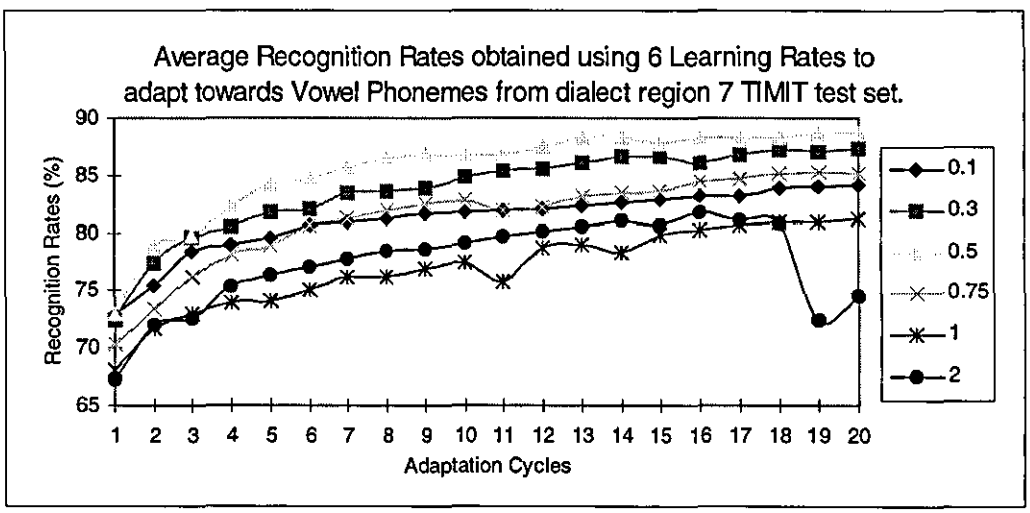


Figure 5.2 Graph showing the average recognition rate results from the six base classifiers using six back-propagation learning rates

It was appreciated that selection of the learning rates depends very much on the network it is being applied to and that a higher learning during other areas of experimentation could well have improved the convergence of a network. However, throughout all experimentation, the same learning rate was used for consistency as experimental results of concern were mostly comparative, not absolute.

After the learning rate was selected, the comparative study between the MLP and OCON began. Each of the six base-classifiers was adapted and tested using the 'test set.' Each network was adapted towards one of its relevant phoneme classes for a total of 100 cycles, during which 7 result snapshots were taken at 1, 3, 5, 10, 20, 50 and 100 cycles. Due to the non-linearity of network adaptation, the number of cycles between each result snapshot increased to produce a graph that offered a clear picture of the network's behaviour. The results taken at each snapshot were the recognition rates of both the adapted phonemes and the remaining unadapted phonemes within the same network. After adapting for 100 cycles towards each phoneme class, the weights and bias' within each network were reset to their initial base-classifier values ready for the next adaptation procedure involving another phoneme class. For the adaptation results, 24 graphs were produced showing comparative results between MLP and OCON networks on the adapted phonemes and the effects these adaptations had on the remaining unadapted phonemes. These are given in appendix A. From these 24 graphs, 6 further graphs were produced containing the averaged data with respect to their tongue-hump group for the adapted and unadapted phonemes (Figure 5.3(a) - (f)). As well as recognition rates, another area of interest was each network's convergence rate. The convergence rate for each of the 6 averaged data graphs was calculated by differentiating the recognition-rate data (calculating the distance between adjacent rates). However calculating the convergence rate in this way was viewed as being unrealistic since the closer that recognition rates reach the perfect goal of 100%, the greater the significance of recognition improvement.

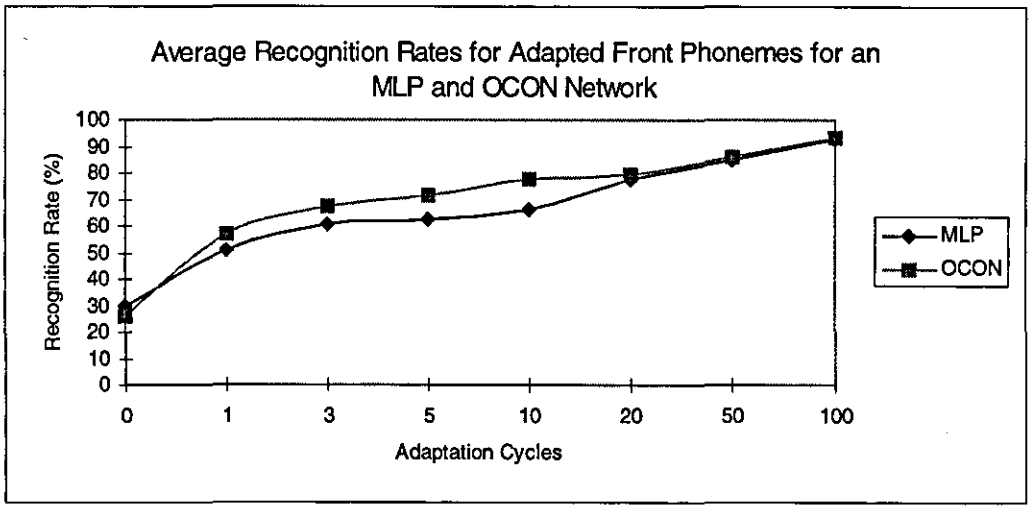
To reflect this, the convergence rate y was calculated using the following equation :

$$y = \left(\frac{100 - \chi_n}{100 - \chi_{n+1}} \right) - 1 \qquad (5.1)$$

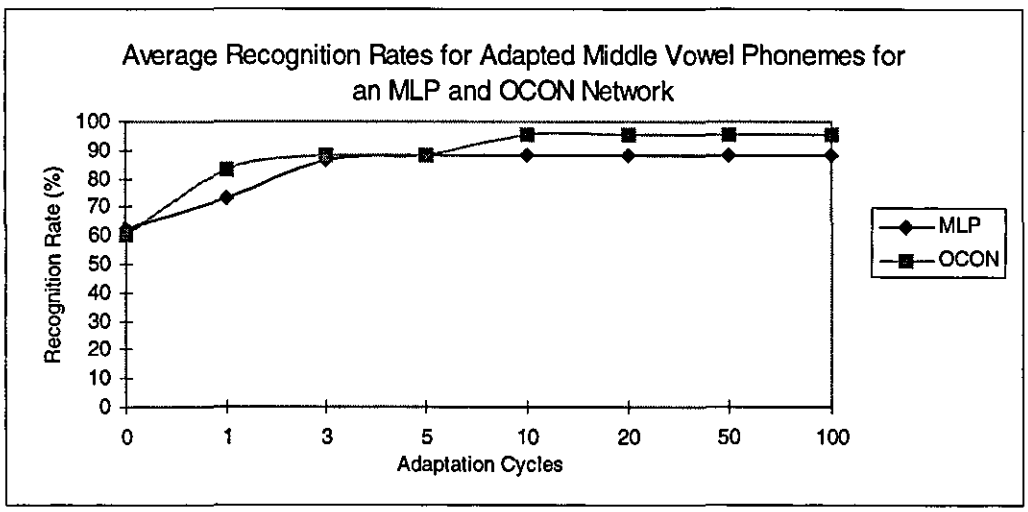
where χ_n and χ_{n+1} are two adjacent recognition rates. The term -1 in equation (5.1) was used so that positive values indicated positive convergence and negative values negative convergence. The 6 convergence rates graphs generated (Appendix A) were averaged to produce convergence graphs for all the adapted vowel phonemes (see Figure 5.4(a)) and all the unadapted vowel phonemes (see Figure 5.4(b)).

Figures 5.3 (a) - (c) show that for all three phoneme groups, the OCON networks show a clear improvement for the recognition rates of adapted vowel phonemes over the conventional MLP networks. On average, for all vowel phonemes, the experimentation shows a 12.3% increase in recognition rates for the OCON networks [6][7]. This result echoes the improvements shown in other data classification systems utilising OCON architectures [2][3][4][5]. Furthermore, the OCON architecture not only increases the adaptation rate but also reduces the processing time necessary for each adaptation cycle due to the reduction in network weights. This is shown in Figure 5.4(a) with the increased rate of convergence for each OCON network, offering a 273% increase against the MLP for adapted phonemes. However, the OCON architectures as they stand, deal badly with inter-class adaptation. Although the rates of convergence for both networks are roughly the same (see Figure 5.4(b)), Figures 5.3 (d) - (f) show that for all three phoneme groups, the OCON networks offer worse recognition rates for unadapted vowel phonemes over the conventional MLP networks. When the three Figures 5.3 (d) - (f) are combined, the average drop in recognition rates for the OCON networks, compared with the MLP networks, is 6.3%.

(a)



(b)



(c)

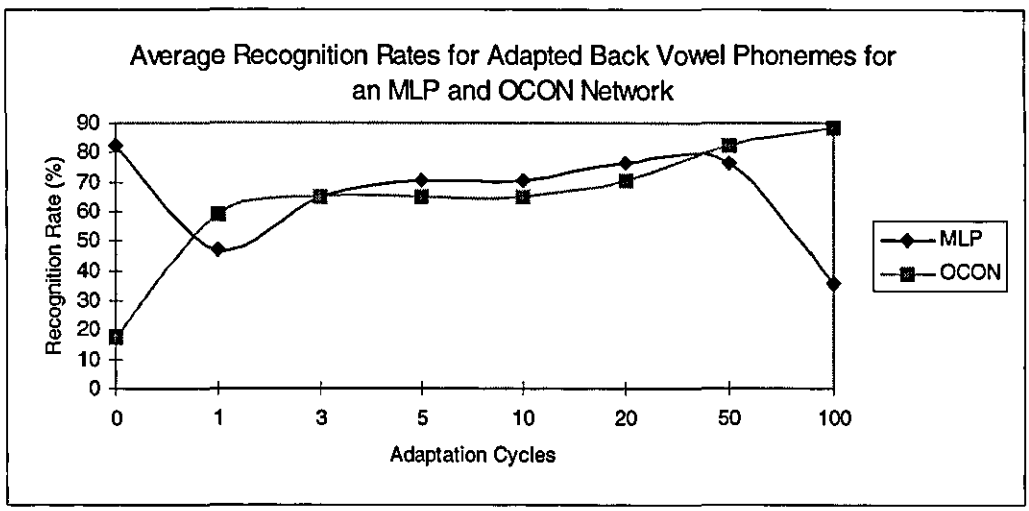
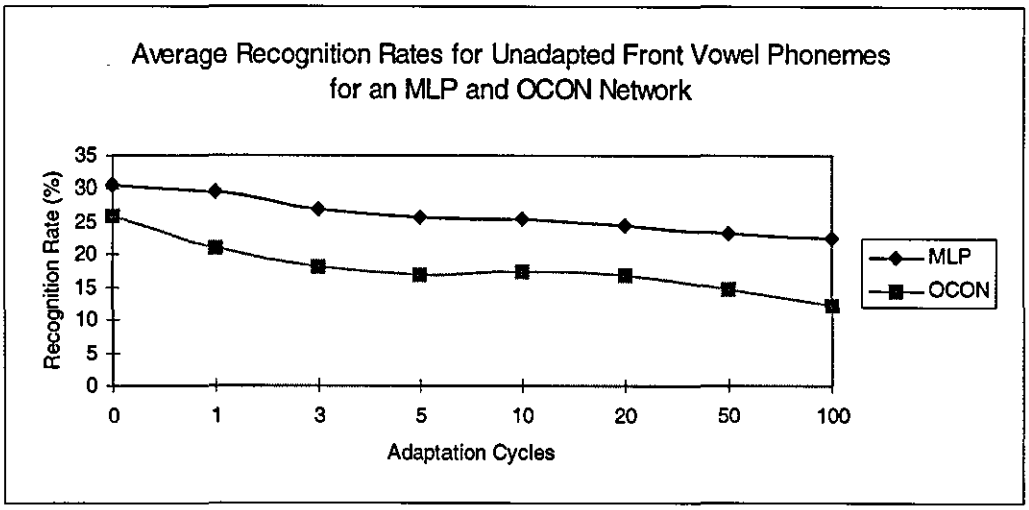
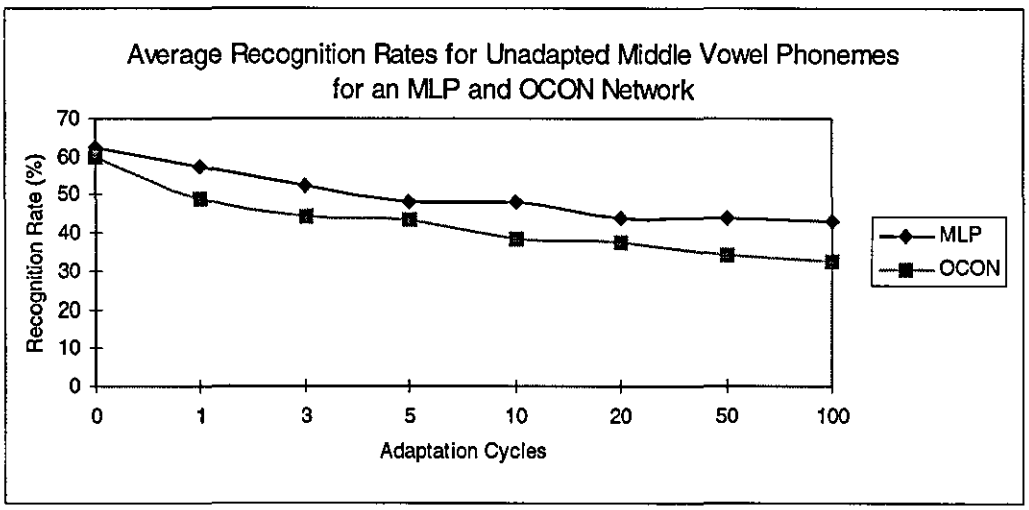


Figure 5.3(a) - (c) Graphs showing the average comparative MLP and OCON recognition rates for adapted vowel phoneme from all three tongue-hump groups.

(d)



(e)



(f)

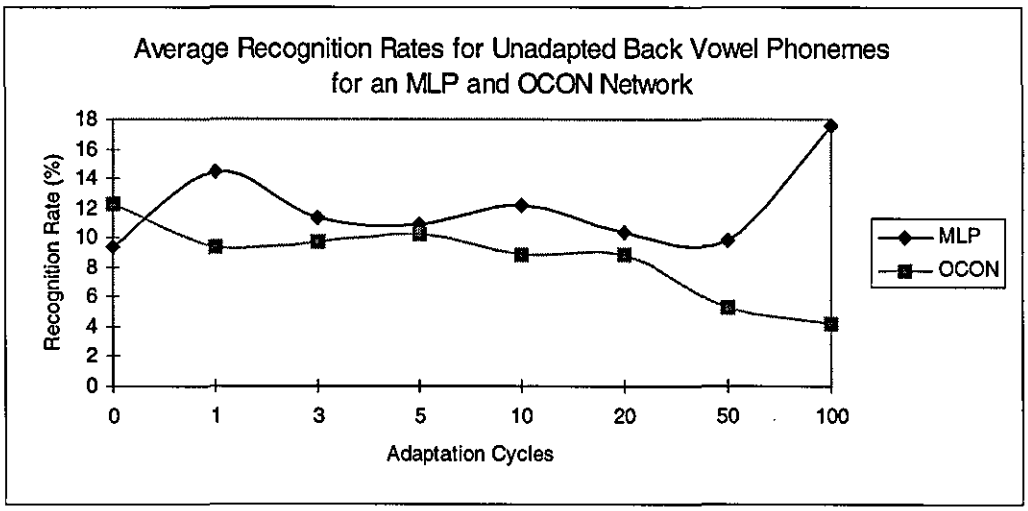


Figure 5.3(d) - (f) Graphs showing the average comparative MLP and OCON recognition rates for unadapted vowel phoneme from all three tongue-hump groups.

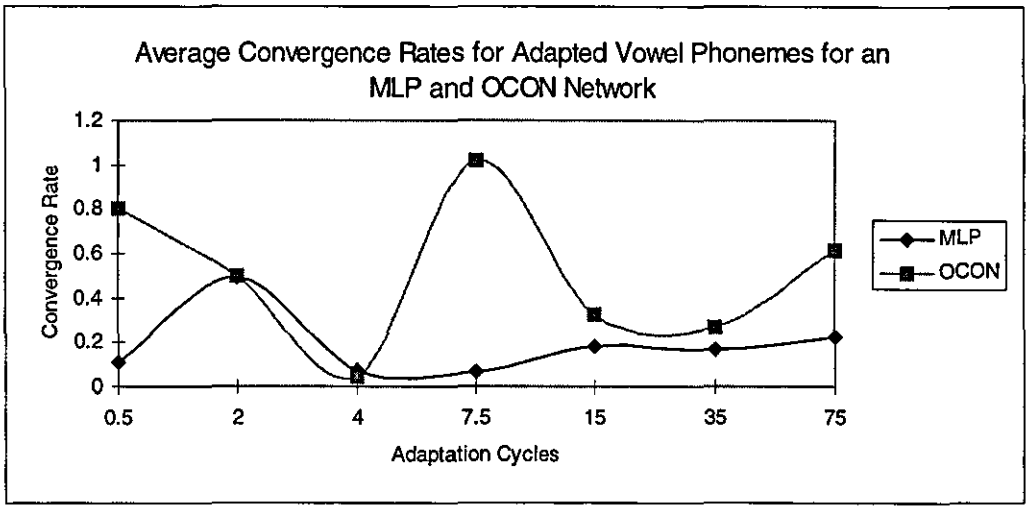


Figure 5.4(a) Graphs showing the averaged MLP and OCON convergence rates for all adapted vowel phonemes.

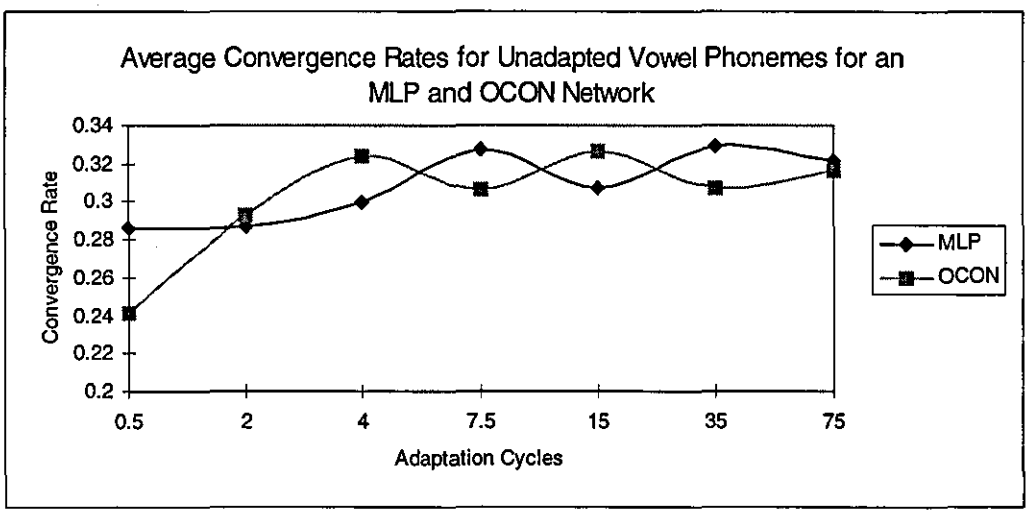


Figure 5.4(b) Graphs showing the averaged MLP and OCON convergence rates for all unadapted vowel phonemes.

5.5 Conclusion

As expected the OCON behaves better than the MLP when adapting and testing the same phoneme. This is primarily due to the individual networks in each OCON network being dedicated to each class. Not only are there fewer connections and hence weighted axes to train, but each network only has to deal with information concerning a single class. As a result the OCON fulfills the first of the three research objectives, not only reducing the processing time for each adaptation cycle, but also rapidly increasing the convergence rate. However, the OCON architecture as it stands, deals badly with inter-class adaptation. When adapting to a class, the OCON shows a lower recognition rate for the remaining phonemes in

the network compared to that of the MLP. This indicates that there must exist some common speaker information within all the classes in a network which isn't being exploited in the isolated networks of the OCON. Although in many applications cross-class interference can be a problem, MLPs compared to OCONs appear to use it to their advantage for inter-class adaptation. As a result an ideal network would be a hybrid OCON architecture containing isolated networks for improved single class adaptation but with some inter-class bonding to profit from any common speaker information. However it would be important that any hybrid OCON network should concentrate adaptation only on common speaker information as adaptation towards common class information could result in harmful cross-class interference.

5.6 Summary

The advantages of the OCON over the MLP has been clearly shown with improved recognition and convergence rates when adapting to any of the 12 vowel phonemes. However the OCON shows the problem of poor inter-class adaptation results due to the isolation of each sub-network. To address this problem and to fulfill the second and third research objectives a modified OCON architecture is investigated in Chapter 6.

5.7 Reference

[1] R.P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, pp.4-22, April 1987.

[2] I. Jou, Y. Tsay, S. Tsay, "Parallel Distributed Processing with Multiple One-Output Back-Propagation Neural Networks", Proc, Int Symp on Circuits and Systems, Singapore, pp.1408-1411, 1991.

[3] S.Y.Kung, J.S.Taur, "Decision-Based Neural Networks with Signal/Image Classification Applications," IEEE Transactions on Neural Networks, Vol.6, No.1, pp. 170-181, January 1995.

[4] S.Y.Kung, "Digital Neural Networks," Prentice Hall, Englewood Cliffs, NJ.

[5] J.N.Hwang and Hang Li. Interactive query learning for isolated speech recognition. In S.Y.Kung, F.Fallside, J.A. Sorensen, and C.A.Kamm, Neural Networks for Signal Processing, I, pp.513-522, Proceeding of the 1991 IEEE Workshop, Princeton, NJ, 1991.

[6] S.J.Haskey and S.Datta, "Dynamic Speaker Adaptation for Acoustically Similar Vowel Sounds using Sub-Cluster Neural Network," Conference and Workshop on New Ideas in Computing, Part 2, Coventry University, May 1997, pp41-44.

[7] S.J.Haskey and S.Datta, "Using Tongue-Hump-Position Information for Vowel Adaptation within a Subcluster Neural Network," Proc. of IEE Colloquium on Pattern Recognition, London 26 Feb 1997, pp 9/1 – 9/6.

CHAPTER 6

Speaker Adaptation Layer

6.1 Introduction

Within a single utterance lies an abundance of speaker information. This information inherently leads to intra- and inter-speaker variations which pose an enormous problem for recognition, especially for multi-speaker systems. It is impractical to train a recognition system with every variation of every utterance from every speaker. That is why adaptation within a system is of such importance. Adaptation allows a system, trained with a representative data set, to dynamically map itself towards a new speaker or towards a change in a present speaker's voice. By altering the model within a recognition system, a single utterance can aid in the total recognition of future utterances articulated in a similar way or by the same speaker.

It has been hypothesised that a listener interprets an utterance with reference to the acoustic space of the speaker to whom they have been listening [1]. This means that a listener normalises the vocal and dialectal characteristics of a new speaker in order to improve the recognition of further utterances. Therefore, for intra- and inter-speaker adaptation, the model needs only adapt with respect to the speaker's vocal characteristics, not the utterance itself. This speaker adaptation allows a recognition classifier to normalise incoming utterances with respect to the present speaker, improving recognition across the board.

Adaptation falls into two main categories, supervised and unsupervised. Supervised adaptation requires an external teacher to inform the recogniser of either a correct or incorrect classification before adaptation can continue. The external teacher can be in the form of an adaptation data set of input-output pairs or a discrete input from the speaker such as a key press or keyword. This form of adaptation guarantees that the recogniser always adapts towards the correct class. However, using input-output pairs can often be difficult in a practical system and discrete keywords from a speaker can themselves run the chance of not

being recognised. Unsupervised adaptation is more automated than supervised adaptation, requiring no external input. Using a winner-take-all methodology, an unsupervised classifier can decide which class has been correctly classified and consequently which class it needs to adapt towards. Although faster and easier to use than a supervised system, there is always the risk of misclassification and consequently adaptation towards a wrong class. During the process of experimentation, we will be merely interested in the effect of 'correct' adaptation on a classifier. Therefore all forms of adaptation will be supervised, using input-output data pairs as the external teacher.

In Chapter 5 we found that, compared with the conventional MLP, the OCON offered a higher recognition rate for an adapted vowel phoneme . However, due to the nature of the OCONs architecture, the vowel phoneme classes within the network were isolated from one another preventing any inter-class adaptation. This isolation led to poor recognition results for the remaining unadapted classes in each network. In an ideal system, adapting to a single class would improve the recognition of the remaining unadapted classes uttered by the same speaker. Such a system would then only require a single uttered class from each network for every vowel phoneme to be adapted. Therefore, to improve inter-class adaptation, it is necessary to build into the neural network architecture some common layer that will allow common inter-class information (speaker characteristics) to be stored and adapted. In this chapter the design and implementation of such a common layer is investigated and its impact on both intra- and inter-speaker adaptation tested.

6.2 Speaker Adaptation Layer

The purpose of a common layer within the neural network is to allow each vowel phoneme to share a common series of weights. To allow their equal utilisation by each class and to ensure that the essence of the OCON's topology remains constant, the series of weighted neurons are best situated at the front of the network as shown in Figure 6.1 [2]. Forming only a single layer to avoid over-complex decision boundaries (Figure 5.1), the new layer would consist of identical neurons to those found in the rest of the network. This would prevent disproportionate weight changes within the network whilst training. The full training of such an architecture, to create a base classifier, leads to a hypothesis regarding the distribution of information within it. It is envisaged that information, unique to each class, will be stored in

each of the relevant OCON subnets. This would leave any common inter-class information stored in the front-end common layer. A successful system such as this would dramatically improve both intra- and inter-class adaptation, fulfilling the three research objectives.

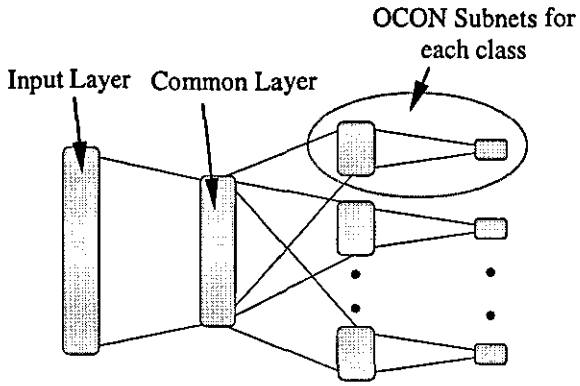


Figure 6.1 Graphical Representation of Modified OCON Network with Adaptation Layer.

A new utterance from a new speaker is unlikely to produce an ideal output from the neural network classifier. The difference between the actual and ideal outputs is said to be the network error (see Section 4.3.3) and is used to adapt the network towards the new utterance. For the adaptation cycle, this network error would then be fed back through the architecture using the back-propagation algorithm. Each of the subnets would act as a filter for the fed-back network error, extracting class-unique information. This, in theory, would leave only the network error relating to common inter-class information such as the new speaker's characteristics. By concentrating the adaptation only on the front end common layer, the fully trained network should then only adapt to the new speaker's characteristics aiding in the recognition of every class after a single utterance. This would then fulfill the second and third research objectives.

6.3 Effectiveness of Common Adaptation Layer

To prove the 'distribution of information' hypothesis, a test was conducted analysing three topologically identical pairs of base-classifiers. Each pair of base-classifiers, representing one of the three vowel tongue-hump groups (see Section 3.1), was subjected to one of two different adaptation procedures. The first procedure allowed unrestricted weight adaptation as the error was back-propagated through the correctly classified OCON subnet and common

front-end 'adaptation layer,' Figure 6.2a. The second procedure allowed only the weights within the adaptation layer to be altered; the area where the speaker characteristics are believed to be stored, Figure 6.2b. After full training by many speakers, each of the subnets should contain normalised raw phoneme data, each with their own unique acoustic space and containing no speaker information. Most neural network adaptation procedures allow the network error to be back-propagated through every weight within the network. This is so that an utterance not only adapts towards the correct class but also away from the remaining incorrect classes within the network. However, since each of the classes are already distinctly separated in their own acoustic space and a speaker's characteristics has equal effect on all the classes, no two classes should overlap after adaptation towards a new speaker. Therefore the network error need only pass through the correctly classified OCON subnet as adaptation away from the remaining unclassified phonemes is not necessary.

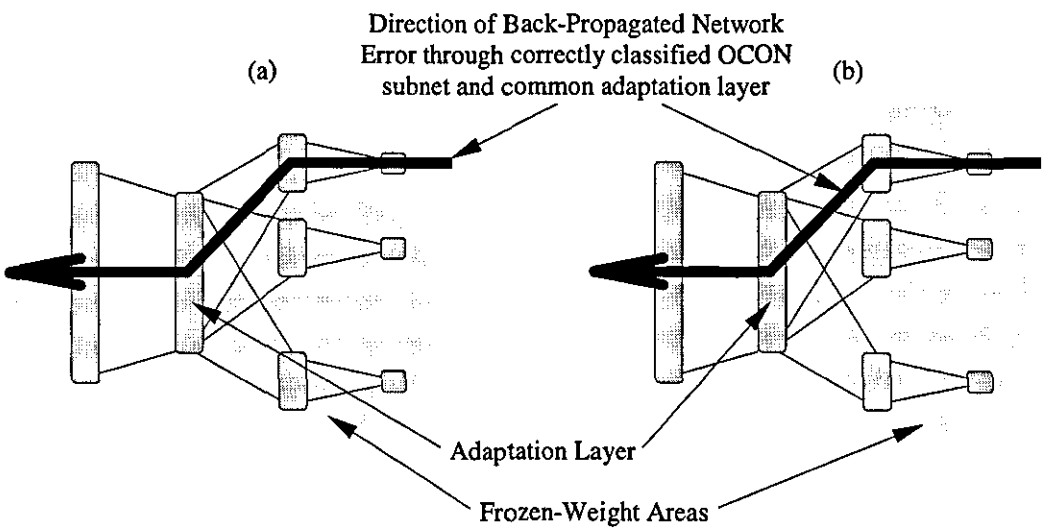


Figure 6.2 Graphical representation of the two adaptation procedures where in network (a) both the subnet and the adaptation layer adapt and in network (b) the subnet is frozen allowing only the adaptation layer to adapt.

The two adaptation procedures were chosen specifically to highlight any distribution of speaker characteristics within the adaptation layer. If the weights within a subnet are not frozen they, along with the weights in the adaptation layer, will adapt to a new utterance containing both speaker and phoneme information. Therefore the subnet will additionally adapt to the speaker's characteristics and the adaptation layer to the raw phoneme information. Although an improvement in the recognition of the adapted phoneme may be seen, the tainting of the adaptation layer with the new utterance's phoneme information may worsen inter-class adaptation.

6.31 Modelling of OCON Network with Common Adaptation Layer.

The three pairs of topologically identical networks were modelled using the Stuttgart Neural Network Simulator V4.1 (SNNS) [3]. Each of the networks had 88 input neurons, the training data format was identical to that in chapter 5, and the number of output neurons was dependent on the phoneme group they represented, 'front'=5, 'middle'=3 and 'back'=5. The general layout of the architecture had been decided (See section 6.2), with a fully connected subnet for each class connected to the input layer via a fully connected single adaptation layer, figure 6.1. Although the layout had been determined, a common problem encountered in the design of neural networks is deciding on the number of neurons in each layer. Finding the optimum number of neurons is typically found through trial and error. Some generalised rules do exist and have had some success in the past. Mirchandani and Cao [4] showed analytically that the maximum number of linearly separable decision boundaries M is a function of the number of hidden neurons H and the number of input neurons D, as follows:

$$M(H,D) = \sum_{k=0}^{D} \binom{H}{K} , \qquad (6.1)$$

where

$$\binom{H}{K} = 0 , \qquad H < k.$$

Equation (6.1) was derived by Mirchandani and Cao for a two-layered fully connected network. However the new network architectures had two hidden layers, making them three-layered networks, and more importantly, due to the nature of subnets, not all adjoining layers were fully connected. These differences made equation (6.1) unreliable and so a method of trial and error using various numbers of hidden neurons was employed. During the trial and error procedure, three-layered networks with 88 input neurons, 3 output neurons and an increasing numbers of hidden neurons were trained using the 'training set' and the back-propagation algorithm. The networks, each using the sigmoidal activation function and a topology of that shown in Figure 6.2, were then tested with the 'test set' until a saturation point in the recognition performance was found. The first network to obtain the recognition saturation had 15 neurons in the common hidden layer and three neurons in each of the

subnet hidden layers. Therefore trained networks with this number of hidden neurons were selected as the base-classifiers for the experiment, Figure 6.3.

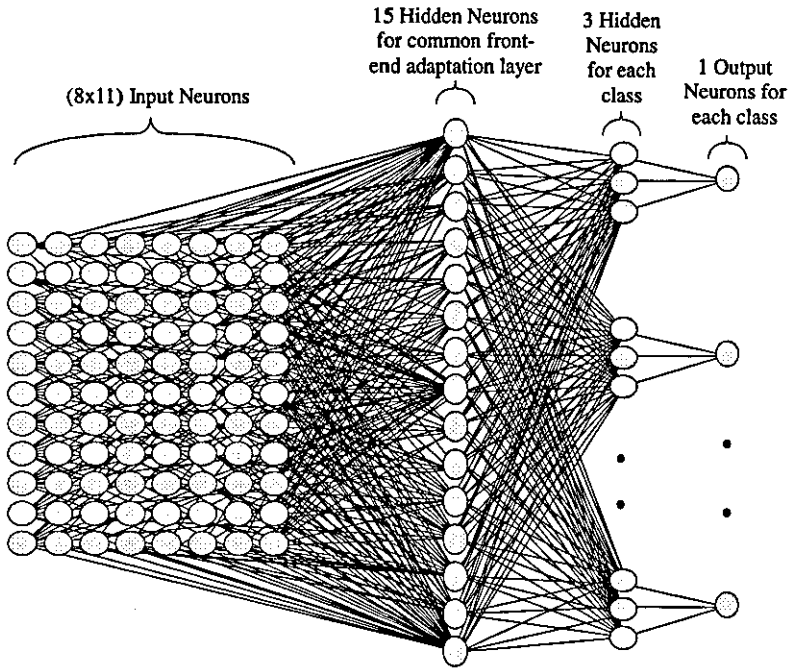


Figure 6.3 OCON network with front common adaptation layer.

6.3.2 Training Procedure

The experiment was a comparative study between the two adaptation procedures. This required the networks in each of the three phoneme group pairs to have identical architectures and training. Therefore to produce the six base classifiers needed for testing, one network for each of the three phoneme groups was trained and then duplicated to produce the second of each pair of identical networks. Using SNNS, the three networks, 'front', 'middle' and 'back' were initially trained. They were each trained with their relevant phoneme data from the 'training set' using the back-propagation algorithm set to a learning rate of 0.5. For experimentation, training and adaptation, the learning rate was set to the same as the previous experiment, 0.5, in chapter 5 for consistency.

6.3.3 Adaptation Procedures

All the speakers from the dialect region 7 TIMIT test set were used to monitor the effects of intra- and inter-class adaptation on the six base-classifiers. However, since only intra- and

inter-speaker effects for single speakers were needed at this stage the speech data from a single speaker at a time was used for the adaptation procedures. The experimentation was split into two sections, frozen and unfrozen adaptation, to prove the distribution of information hypothesis.

Unfrozen Adaptation

One from each pair of identical base-classifiers was adapted using the unfrozen approach. The unfrozen approach simply allowed adaptation to occur with all the weights in the network free to adapt. When the network was subjected to a new utterance from a single speaker a network error equal to the difference in the network's actual outputs and ideal outputs was created. However, only the network error for the correctly recognised class was noted with the network error for the unclassified outputs ignored. This was because phoneme information within each subnet was deemed sufficient and distinctly separated enough from the remaining classes to only require adaptation towards the recognised class not adaptation away from the unrecognised classes. The network error was fed back through the subnet of the recognised class and the adaptation layer using the back-propagation algorithm, Figure 6.2a. As the error passed through each weighted axis, the weights were altered to minimise the error adapting the network to the new utterance. Each phoneme, one at a time, was offered to the network for adaptation and the effect on all the phoneme classes in the network tested. Due to the size of the dialect region 7 test set for each male speaker, each phoneme was represented by only a relatively small number of utterances. This meant that phonemes used for adapting were also used for testing, possibly tainting the results. To minimise this risk, during each adaptation cycle, a single utterance was used for adapting and testing was with the remaining utterances in the phoneme set. For each successive adaptation cycle with the same phoneme, the next phoneme utterance was used for adaptation and as before the remaining phonemes for testing. When every phoneme utterance had been used for adaptation the process recommenced with the first utterance. After adaptation towards one phoneme the network's weights were reset to their initial base-classifier values ready for adaptation towards the next phoneme. This adaptation procedure continued for each of the 15 male speakers within the test set, obtaining recognition results for each of the speakers one at a time. After results from each of the 15 speakers had been collected, they were averaged

to obtain a clear picture of the adaptation procedures effects on intra- and inter-class adaptation.

Much of this process was automated with the use of the pattern and batch files used by SNNS. The pattern files, for supervised adaptation, contained input-output data pairs of LPC coefficients representing speech data and the desired outputs for each utterance, Figure 6.4.

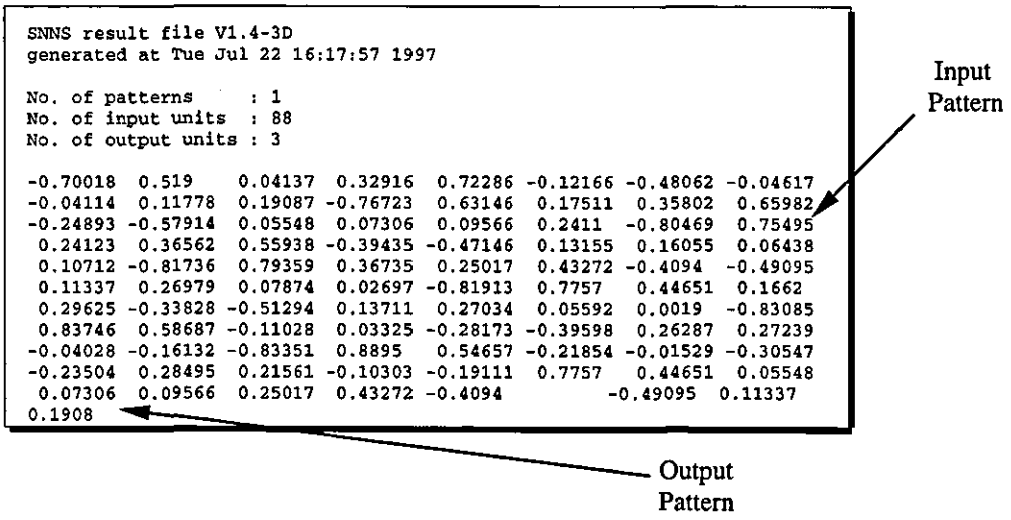


Figure 6.4 Example of a SNNS pattern file

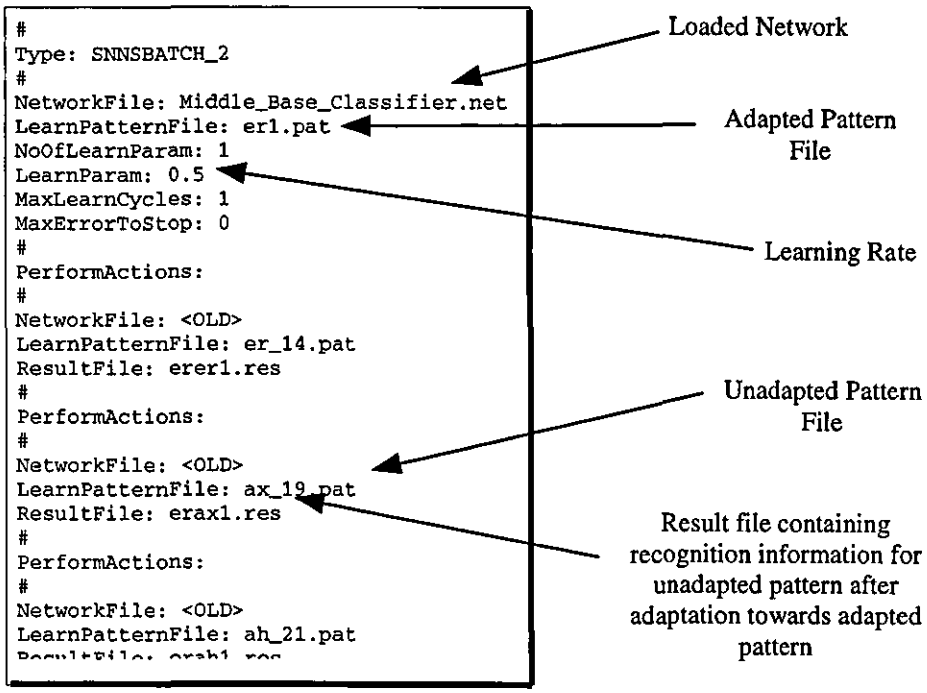


Figure 6.5 Example of a SNNS Batch File for One Adaptation Cycle

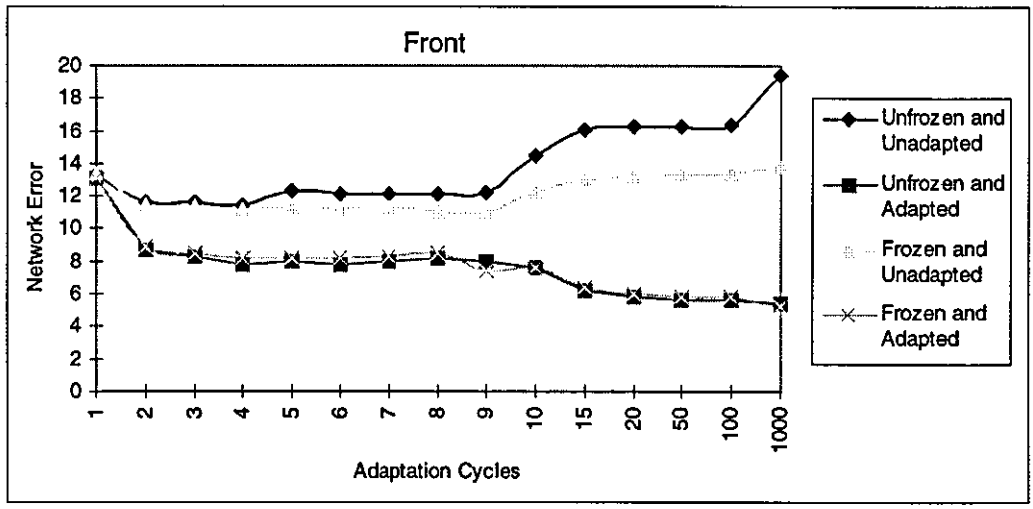
The batch files contained instructions on how to execute the adaptation procedure with information regarding which network and pattern files to use, learning algorithms and learning rates, and the generation of result files, Figure 6.5. Each phoneme was adapted for a total of 100 cycles, during which 15 result snapshots were taken at 1-10, 15, 20, 50 and 100 cycles.

Frozen Adaptation

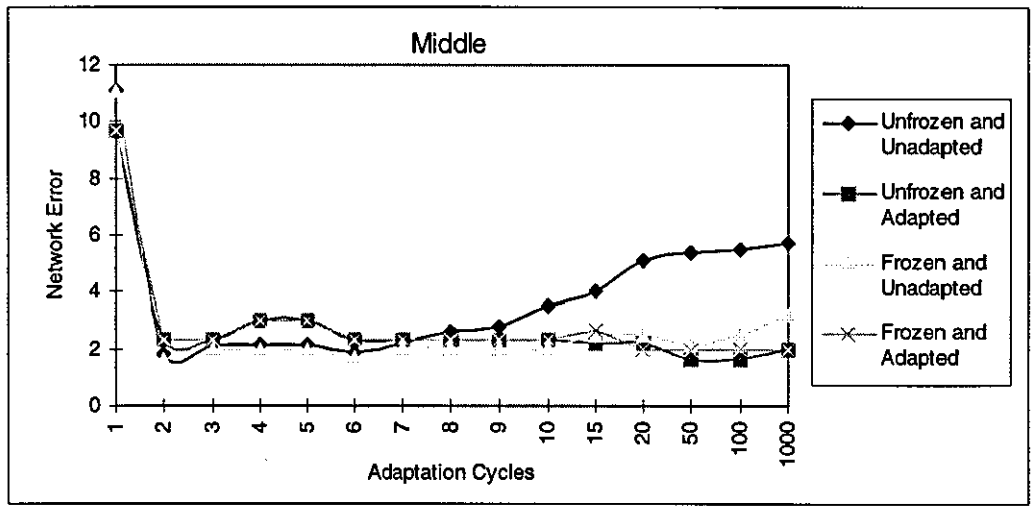
The remaining unadapted networks from each pair of base-classifiers were used for the frozen adaptation approach. The method used was very similar to that of the unfrozen procedure except that selected weights within each network were frozen to concentrate the adaptation on the common front-end 'adaptation' layer. All the networks were modelled, trained and tested using the Stuttgart Neural Network Simulator 'SNNS' (see Section 4.4.3). This software allowed complex networks to be modelled, using a variety of neurons, activation functions and learning algorithms. However, the software was unable to freeze selected weights during the training, testing and adaptation procedures. Because the freezing of select weights during adaptation was an integral part of the experimentation, the 'C++' source code had to be rewritten and the software recompiled, (see Appendix B). The source code was rewritten so that two buttons would appear on the information panel allowing, when depressed, all weights linked to a highlighted neuron in the display panel to be either frozen or unfrozen. Initially the goal was to be able to freeze and unfreeze individual weights. However, due to the resolution of the display panel, selecting one of many weights was not practical. It was also unnecessary since freezing all the weights adjoining a selected neuron was all that was needed. To achieve this two main areas of the source code had to be re-written. The first was the creation of two buttons in the graphical-interface source code which, when depressed, toggled a new flag 'FREEZE' for a selected neuron between '0' and '1'. The second alteration concerned the learning-algorithm's source code to restrict learning of the 'frozen' neurons. A conditional statement within the back-pass of each learning algorithm, allowed the updating of a weight to be skipped if the variable 'FREEZE' for that adjoining neuron was equal to '1'. After the software had been recompiled, all the weights in the subnets were frozen ready for the adaptation procedure. The adaptation procedure was, from then on, identical to the unfrozen procedure using the same pattern and batch files on the 'front', 'middle' and 'back' base-classifiers.

6.4 Results

After both adaptation procedures, the result files generated by the configuration files were ready for analysing. This was accomplished using another piece of software called 'Analyze' which came with SNNS. As its name suggests, it analysed each result file giving the network error along with the percentage of unknown, correctly and incorrectly recognised utterances. When dealing with a speech recognition system we are primarily concerned with the percentage of correctly classified utterances. However, due to the limited number of utterances of each phoneme from each individual speaker, subtle changes within the networks were often lost. Despite a clearer indication being given when the results for each speaker were averaged it was still felt that it was best to show the network's behaviour using the network error. Twelve graphs were generated, one for each vowel phoneme, showing their effect on the network error after 100 adaptation cycles and their effect on the remaining classes in the same vowel phoneme group, (see Appendix C). To get a more general view on the effect of the two adaptation procedures on the adapted and unadapted phonemes, four further graphs were generated. Three graphs showed the average effects on each phoneme group, Figure 6.6(a) - (c), and the fourth graph showed the total average effect on all the vowel phonemes, Figure 6.7. Since the percentage of recognised classes was also of importance, a fifth graph showing the combined average recognition rates of all the adapted and unadapted phonemes for both adaptation procedures was generated, Figure 6.8. Although for individual phonemes showing the recognition rates was not detailed enough, the average for all the phoneme utterances gave a clearer picture of the networks' behaviour. Using the total averaged recognition rates the convergence rate for each adaptation procedure was also calculated using equation (5.1), Figure 6.9.



Figures 6.6(a) Graph showing the average network errors for adapted and unadapted vowel phonemes from the front tongue-hump groups using both the frozen and unfrozen adaptation procedures.



Figures 6.6(b) Graph showing the average network errors for adapted and unadapted vowel phonemes from the middle tongue-hump groups using both the frozen and unfrozen adaptation procedures.

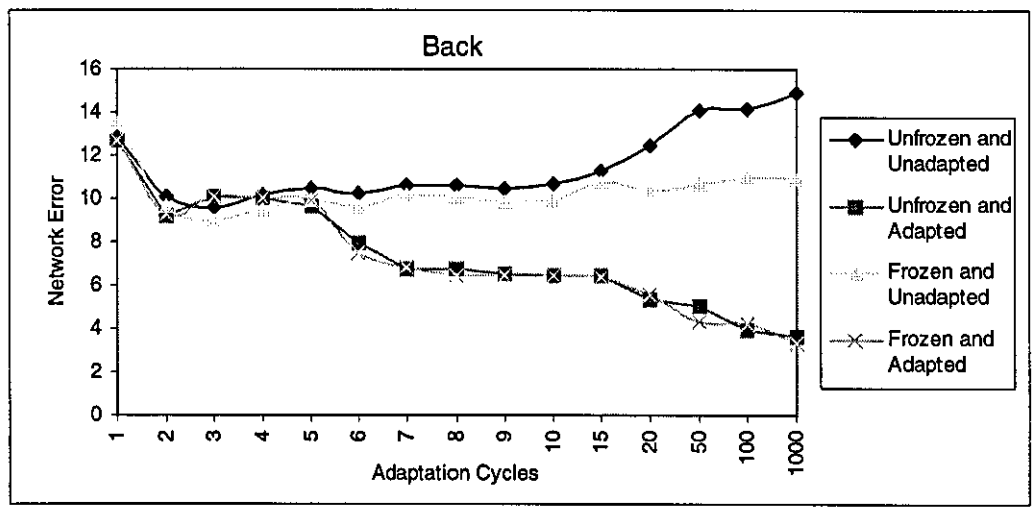


Figure 6.6(c) Graph showing the average network errors for adapted and unadapted vowel phonemes from the back tongue-hump group using both the frozen and unfrozen adaptation procedures.

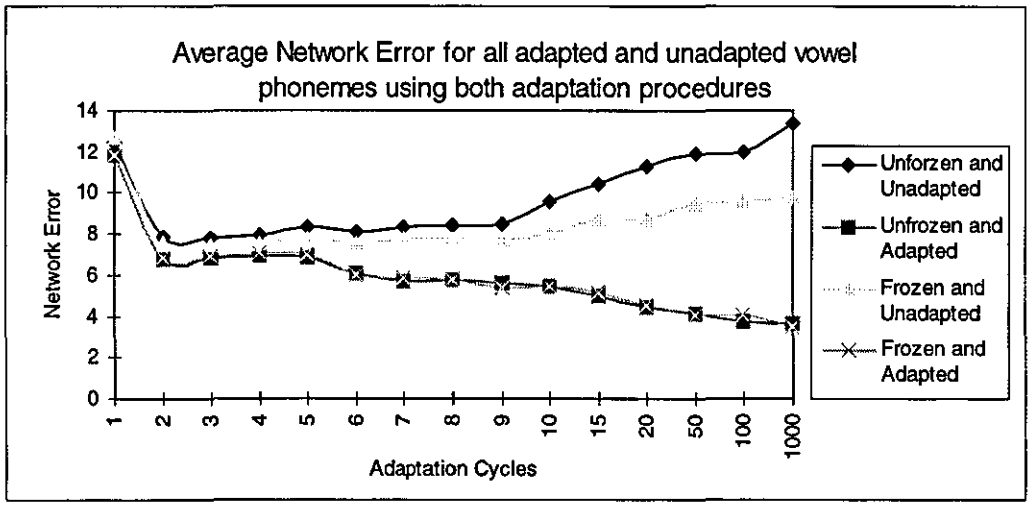


Figure 6.7 Graph showing the average network errors for all the adapted and unadapted vowel phonemes using both the frozen and unfrozen adaptation procedures.

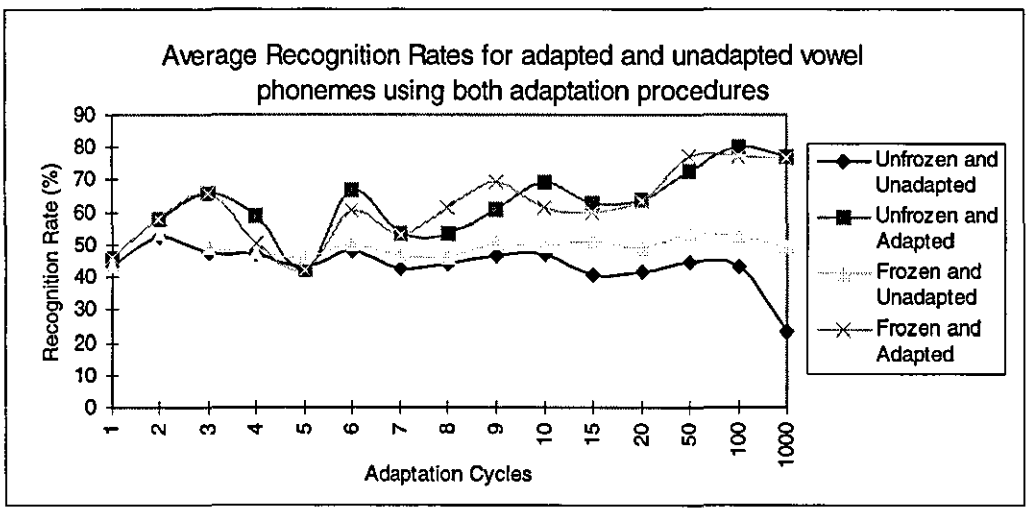


Figure 6.8 Graph showing the average recognition rates for all the adapted and unadapted vowel phonemes using both the frozen and unfrozen adaptation procedures.

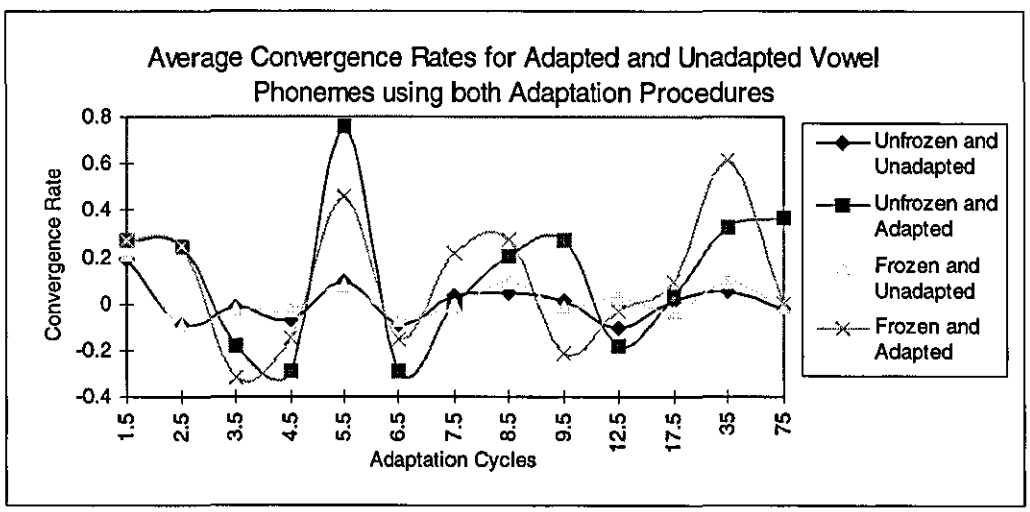


Figure 6.9 Graph showing the average convergence rate for all the adapted and unadapted vowel phonemes using both the frozen and unfrozen adaptation procedures.

Figures 6.6(a) - (c) show a close comparison between the two adaptation procedures for the adapted phonemes. The recognition results from Figure 6.8 show that the frozen adaptation procedures offers a slight 0.65% average drop across the 100 adaptation cycles. However for the effects on the unadapted phonemes using the frozen procedure, Figures 6.6(a) - (c) show a definite improvement over the unfrozen procedure. Despite the recognition rates for the two unadapted results from Figure 6.8 falling, the frozen procedure offers a 5.5% average increase in performance over the 100 adaptation cycles than the unfrozen procedure. Figure 6.9 shows a rapid change in the convergence rate for adapted vowel phonemes for both adaptation procedures and a lower more controlled convergence rate for the recognition rates of unadapted vowel phonemes. Convergence rates for the adapted and unadapted recognition results were very similar for both adaptation procedures although the frozen procedures had very slightly higher convergence rates than the unfrozen procedures.

6.5 Conclusion

All the graphs displayed similar trends for each of the vowel phoneme groups during both adaptation procedures. For the adapted phonemes in each vowel group, the network errors almost mirrored each other identically for both the frozen and unfrozen adaptation procedures. This was expected since, for adapted classes, the only difference between the two procedures was the number of neurons the networks could utilise during the adaptation process. Therefore concentrating the adaptation within just the adaptation layer had the same effect as spreading the adaptation over the adaptation layer and relevant subnet.

For the unadapted phonemes, the frozen and unfrozen adaptation procedures, during the first 9-10 adaptation cycles, had closely matched network errors. However after the first 9-10 adaptation cycles, each vowel group began to show a difference between the two adaptation procedures. Both adaptation procedures showed an increase in network error and a slight decrease in recognition rates displaying a 'negative' adaptation. However the unfrozen procedure for both the network error and recognition rates displayed a larger 'negative' adaptation. This 'negative' adaptation persisted until, between 50-100 adaptation cycles, the total average recognition rate (figures 6.8) was worse than if no adaptation had occurred. The frozen adaptation procedure offered a more flat response after 9-10 adaptation cycles but did worsen slightly.

Although the frozen procedure yielded better results than the unfrozen procedure the failure of the unadapted vowel phonemes (during the frozen adaptation procedure) to positively adapt indicates that the adaptation layer did not contain purely speaker information. However, because the frozen adaptation procedure did behave noticeably better than the unfrozen adaptation procedure, this suggests that either the adaptation layer utilises 'some' speaker information, the frozen subnet does contain primarily class specific information or both. Had the adapting test set been larger and a mixture of phoneme classes been used during the same test, the frozen procedure could have given far more promising results. Despite these problems the results showed that the frozen procedure compared well against the more conventional adaptation method. In addition, more detailed results from chapter 7 using various speakers should substantiate further the already promising results.

6.6 Summary

To fulfill the second and third research objectives a front-end adaptation layer was introduced. To highlight its effectiveness two adaptation procedures, namely frozen and unfrozen, were used. The results indicated the presence of some common speaker information being utilised within the adaptation layer. However, these were only intra-speaker results and so inter-speaker experimentation is now required to give a clearer indication of the distribution of information within the modified OCON ANN. Chapter 7 describes the full intra- and inter-speaker testing using the frozen adaptation procedure.

6.7 References

[1] R. R. Verbrugge, W. Strange, D. P. Shankweiler, and T. R. Edman, "What information enables a listener to map to a talker's vowel space?," J. Acoustical Society of America, Vol. 60, pp. 198-212, 1976.

[2] S.J.Haskey and S. Datta, "Selective Adaptation of Speaker Characteristics within a Subcluster Neural Network," Proc of the Seoul International Conference on Phonetic Sciences, Seoul, Korea, October 1996, pp.464-467.

[3] Stuttgart Neural Network Simulator, User Manual, Version 4.1, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart, 1995. Web Address: //www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html/

[4] G. Mirchandani, W. Cao, "On Hidden Nodes for Neural Nets," IEEE Trans on Circuits and Systems, Vol. 36, No 5, May 1989

CHAPTER 7

New Architecture

7.1 Introduction

With the proven effectiveness of the frozen training procedure on adapted phonemes for a single speaker, multi-speaker tests were arranged to explore the newly modified ANN's intra- and inter-speaker adaptation effects. Intra-speaker effects relate to the behaviour of unadapted phonemes after adaptation towards phonemes from the same speaker. For experimentation, intra-speaker adaptation was restricted to the effects on vowel phonemes from the same tongue-hump group due to the categorisation of vowels within each of the base classifiers. Intra-speaker adaptation is high beneficial since it allows adaptation towards a single phoneme class to aid in the recognition of the remaining phoneme classes uttered by the same speaker. Inter-speaker effects are also highly beneficial, improving recognition results between speakers with acoustically similar utterances. Such a response is ideal for a system used in a homogeneous geographical area, taking advantage of dialectal similarities between speakers

For experimentation, speakers from the TIMIT test set were specifically selected to highlight inter-speaker variations. This required an experimental test set containing a variety of speakers who were both acoustically similar and dissimilar from one another.

7.2 Test Set Selection

Due to the labour intensive nature of experimentation, a minimal number of speakers were selected for the multi-speaker tests. Three were sufficient to indicate any intra- and inter-adaptation effects, by selecting two acoustically similar speakers and a third acoustically dissimilar. As with all the experimentation, the three speakers were selected from the 15

male speakers contained within the same dialect region 7 TIMIT test set (See section 3.2). Although the idea behind the inter-speaker test data was to incorporate dialect differences as well as similarities, it was felt that if two speakers were too dissimilar with different dialects, then subtle adaptation effects would be difficult to monitor. Consequently all the speakers, although some with large acoustic differences, were from the same dialect region to indicate clearly any inter-speaker variations. Three analysis procedures were employed to find the three speakers needed for the test set. The first was a listening test. This involved 10 listeners to audible select two speakers which they found to be the most similar and a third which was the most dissimilar from the first two they selected. This was achieved by playing, via headphones, a standard 'SA' sentence from each of the 15 male test-set speakers to the listener. A TIMIT 'SA' sentence was used specifically since it was common to all speakers and was designed to expose dialectal variants. Using the speaker coding format (see Section 3.3) eight of the ten listeners selected speakers MKJL0 and MTWH0 as the most similar. Of those eight listeners five selected speaker MDVC0 as the furthest away the two selected speaker MPAB0. The second analysis procedure involved the creation of a small two-layered ANN, trained with phoneme data from the 'SA1' sentences. Since the training data was in the 'training set' data format, (see Section 3.3), the ANN consisted of 88 input nodes and an output node was created for each of the 15 male test-set speakers. Using Mirchandani and Cao's equation, equation (6.1), 4 nodes were selected for the hidden layer to offer at least 15 linearly separable decision boundaries, one for each output class. The 'SA2' sentences from each speaker were used for testing, noting the output scores for each speaker input. Ignoring the output scores relating to the present speaker, the highest scores represented the speakers with the most similar utterance. The results showed that speakers MKJL0 and MTWH0 were the most similar and that speaker MPAB0 was the furthest away, followed closely by speaker MDVC0 (Appendix D). The first two analysis procedures showed a common selection for the two closest speakers, MKJL0 and MTWH0. However, when selecting the most distant speaker from the two closest speakers there were slightly conflicting results. This was expected since for an audible test it is difficult to select the two most distant utterances when there are so many parameters involved such as the speakers pitch, volume, pronunciation and intonation. The third and final analysis procedure was a verification of the two previous analysis procedures using spectrograms. Selecting speakers MKJL0, MTWH0 and MPAB0, the formant information from each of their two 'SA' utterances was studied. Analysis was concentrated on vowel phonemes which give clear formant definition. Speakers MKJL0 and MTWH0 revealed similar vowel phonemes up to

the second and third formant and obvious variations when both compared with the same vowels from speaker MPAB0. As a result the three male speakers MKJL0, MTWH0 and MPAB0 from dialect region 7 were selected for multi-speaker experimentation, offering the necessary inter-speaker similarities and differences.

7.3 Intra-Speaker Adaptation

The three base classifiers, 'front', 'middle' and 'back' (see Section 6.3.2), were adapted towards each of the relevant phoneme utterances, one phoneme class from each speaker at a time, using the frozen adaptation procedure. After adaptation towards a single phoneme class, with the back-propagation learning rate set to 0.5, the effects on recognition for all the phonemes from the same tongue-hump group and the same speaker were recorded. Each phoneme was adapted for a total of 100 adaptation cycles, during which 15 result snapshots were taken at 1-10, 15, 20, 50 and 100 cycles. After adaptation towards each phoneme class the ANN weight and bias values were reset to the original base-classifier settings, ready for adaptation towards the next phoneme class.

7.4 Intra-Speaker Results

Due to the limited number of phoneme classes uttered by each speaker, the network error was used as an indication of the networks adaptation. This ensured that any network trend from one adaptation cycle to the next could be clearly seen. Initially results were represented using network error versus adaptation cycle plots. The network errors are not an absolute indication of recognition rates and the range of network errors for a phoneme class, across all the adaptation cycles, were small relative to the variety of network error values from one class to the next. Consequently plots often showed an array of what appeared to be horizontal lines for each phoneme class. To rectify this problem the network errors, each phoneme class, were normalised with respect to their maximum and minimum values using :

$$y_n = \left(\frac{\chi_n - \chi_{\min}}{\chi_{\max} - \chi_{\min}} \right) \tag{7.1}$$

where χ_{min} is the minimum network error value, χ_{max} is the maximum network error value, χ_n is the network error after the n^{th} adaptation cycle and y_n is the normalised output of the nth network error value. Although this showed either an increasing of decreasing trend for each network error, this did not include any indication of the trends magnitude. Therefore a more refined network error normalisation was used using :

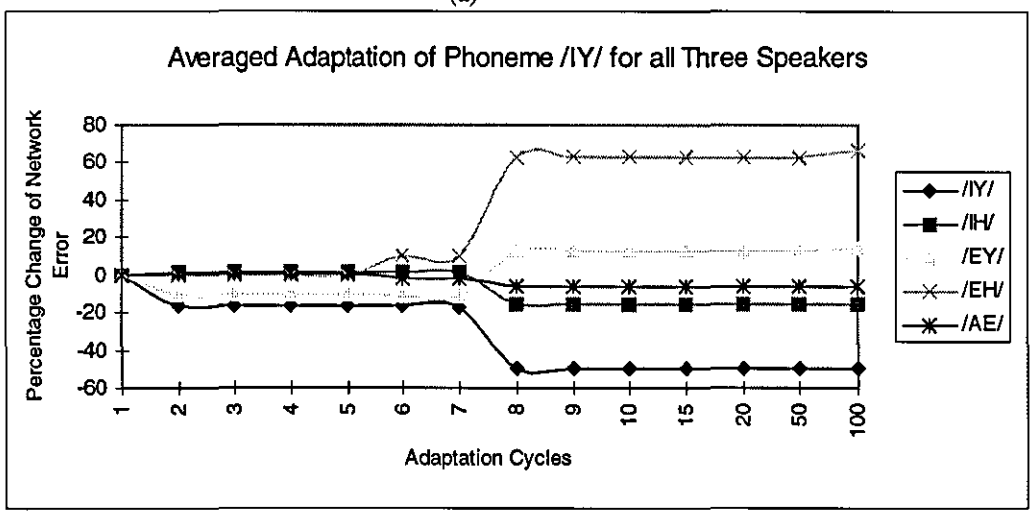
$$y_n = \left(\frac{\chi_n}{\chi_1} - 1 \right) \times 100 \tag{7.2}$$

where χ_1 is the network error after the first adaptation cycle.

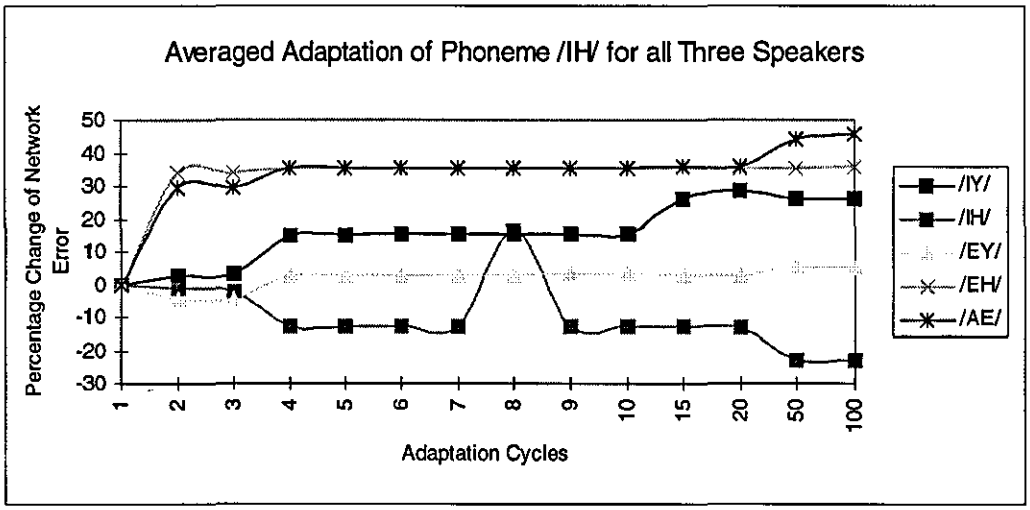
This normalised the network errors for each phoneme class with respect to their first value. This ensured that all normalised network errors began at zero and both the trends and their magnitudes were represented. Also by multiplying the normalised network errors by 100, it allowed the magnitudes of the network errors to be represented as percentage changes. However, this normalisation assumed that a set percentage drop in the network error was of equal significance, regardless of initial network error values. This is not always the case since network learning is often non-linear. Despite this, the normalisation offered a good indication of the networks intra-speaker behaviour during adaptation.

Using the normalised network error data, graphs for each of the twelve adapted vowel phonemes were created, Figures 7.1(a) - 7.1(l). Each graph represented the average network error of the three test speakers, MTWH0, MKJL0 and MPAB0. These graphs were used to highlight any intra-speaker trends and their relationship with respect to the tongue-hump constriction (see Figure 3.1).

(a)



(b)



(c)

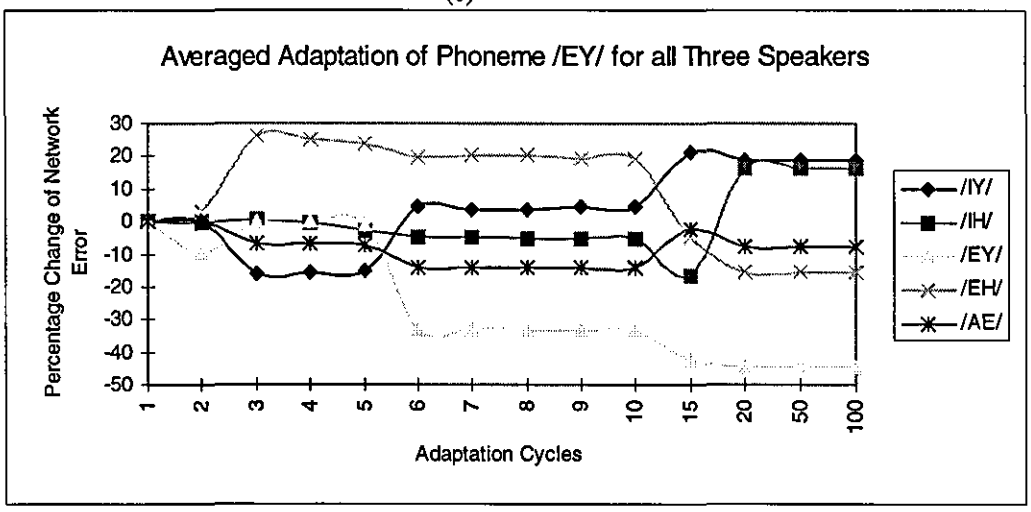
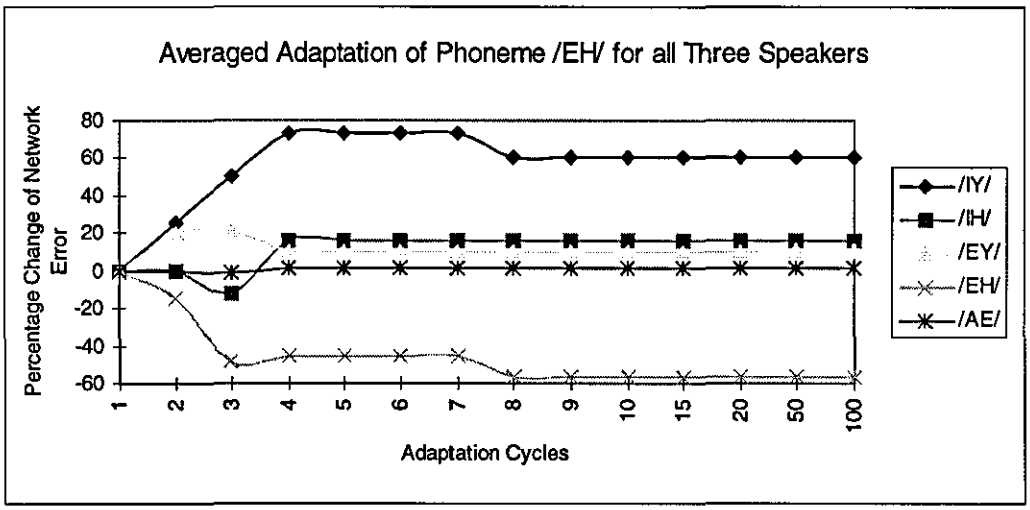
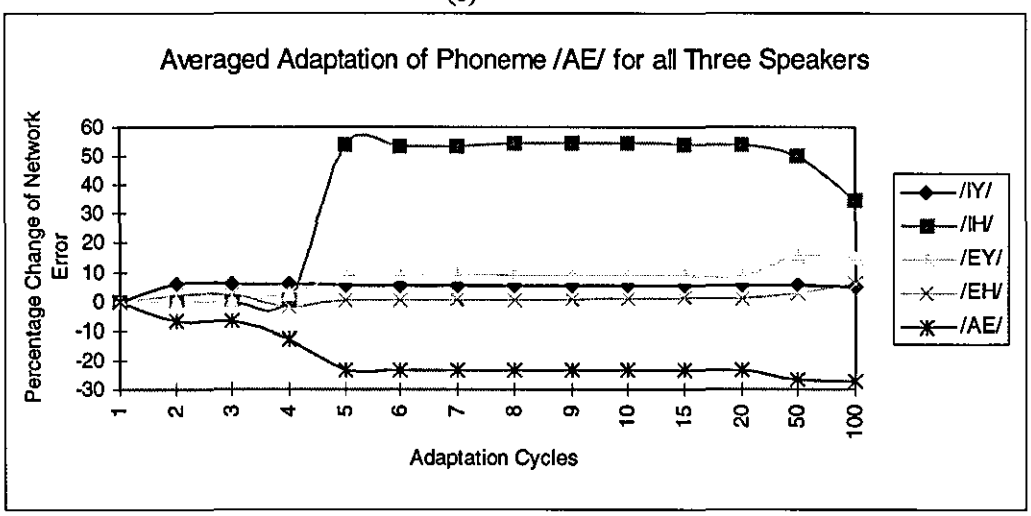


Figure 7.1(a), (b) and (c). Graphs showing the averaged adaptation results for all speakers MTWH0, MKJL0 and MPAB0 applied to vowel phonemes /IY/, /IH/ and /EY/.

(d)



(e)



(f)

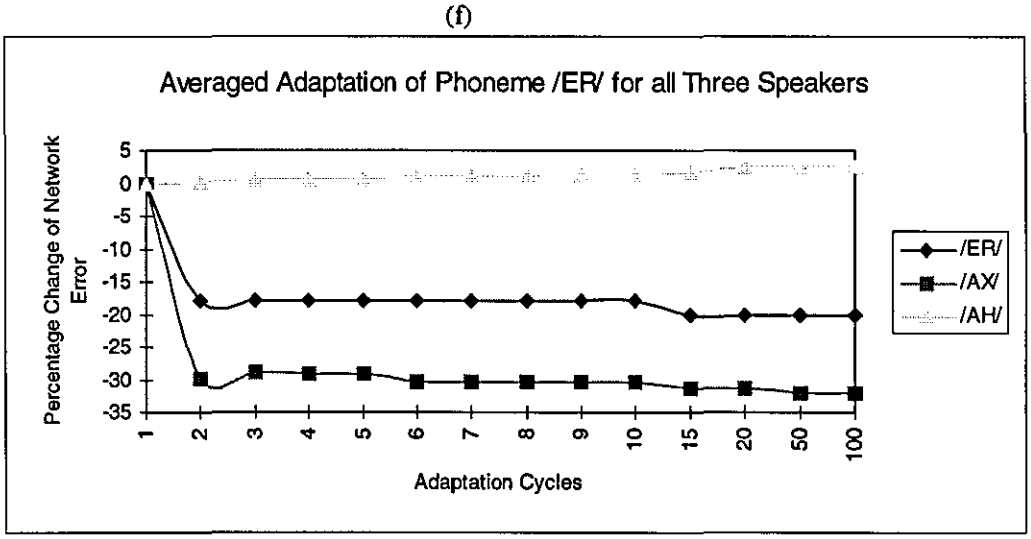
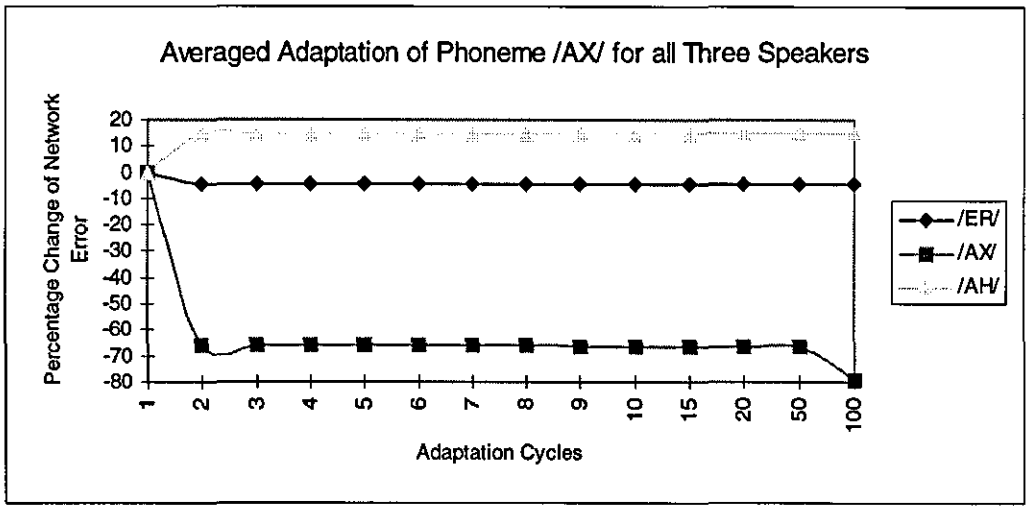
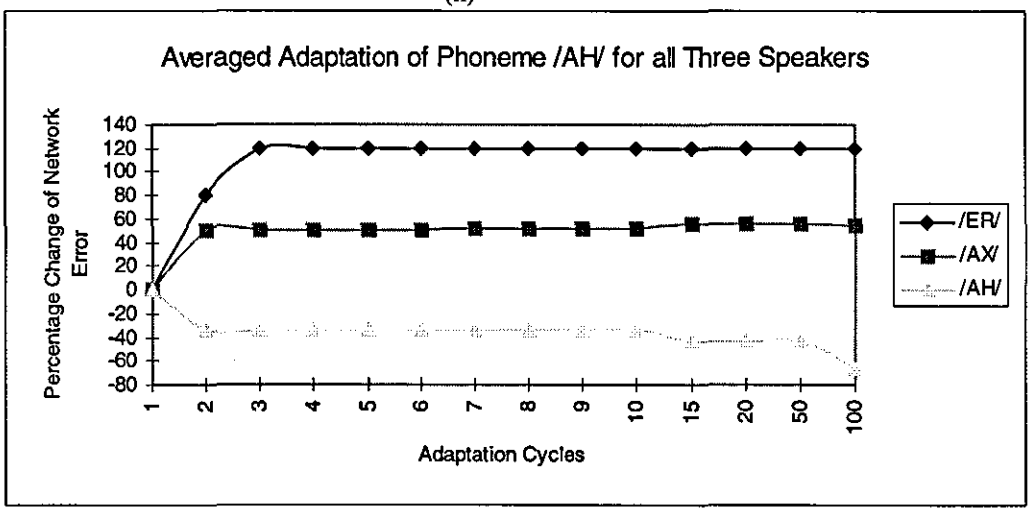


Figure 7.1(d), (e) and (f) Graphs showing the averaged adaptation results for all speakers MTWH0, MKJL0 and MPAB0 applied to vowel phonemes /EH/, /AE/ and /ER/.

(g)



(h)



(i)

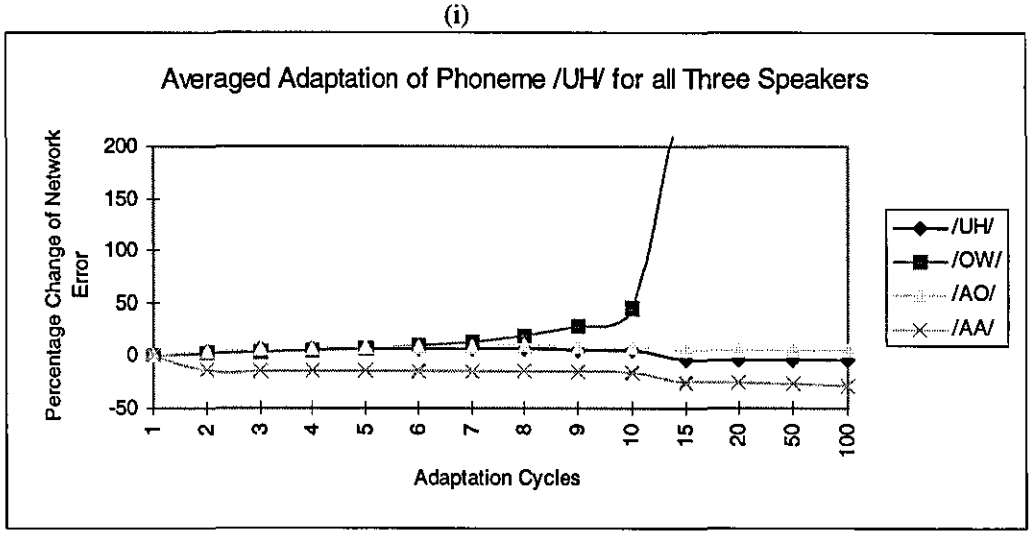
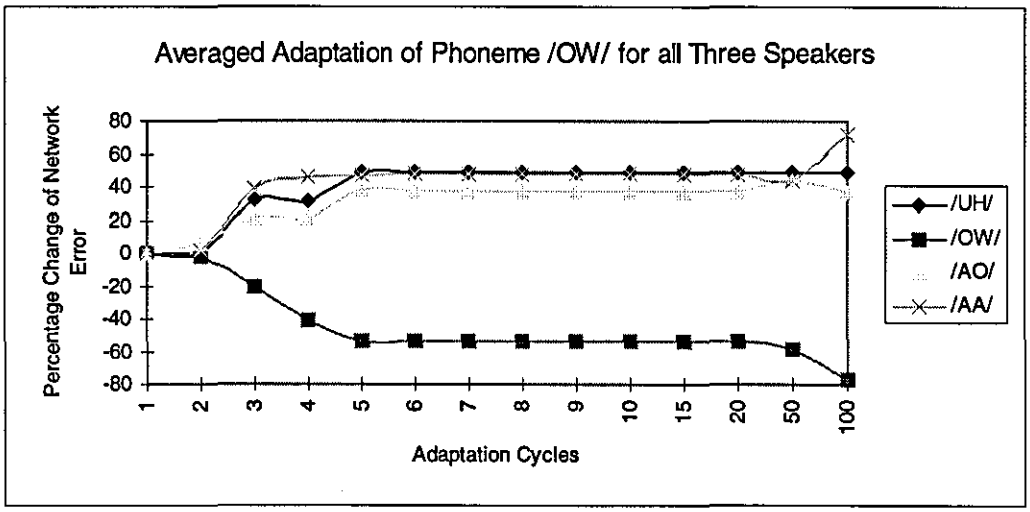
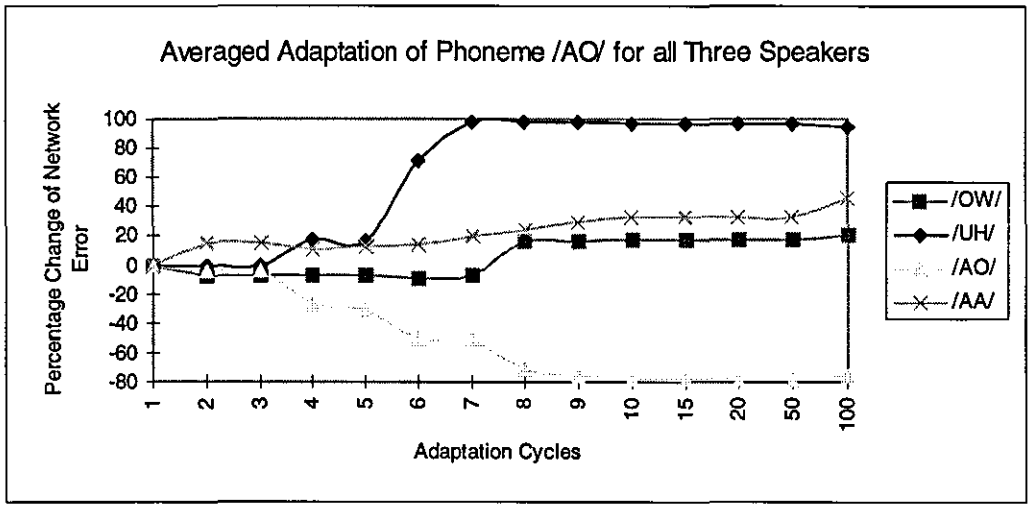


Figure 7.1(g), (h) and (i) Graphs showing the averaged adaptation results for all speakers MTWH0, MKJL0 and MPAB0 applied to vowel phonemes /AX/, /AH/ and /UH/.

(j)



(k)



(l)

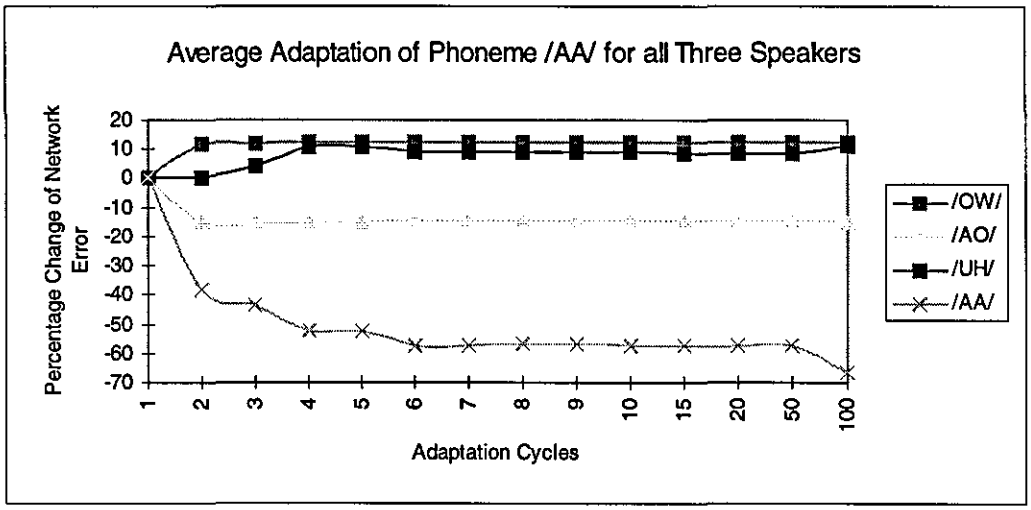


Figure 7.1(j), (k) and (l) Graphs showing the averaged adaptation results for all speakers MTWH0, MKJL0 and MPAB0 applied to vowel phonemes /OW/, /AO/ and /AA/.

All twelve phonemes from the three tongue-hump groups exhibited some adaptation trends due to tongue-hump constriction. The closer a phoneme to the adapted phoneme, with respect to tongue-hump position, the greater the rate of adaptation. However, there were some fluctuations due to experimental errors and some possible misleading results due to the way in which the magnitude of the normalised network errors was calculated. The normalised network errors for the adapted phoneme fell dramatically after adaptation, as expected, although in some cases the network errors for the remaining unadapted phonemes increased. As explained in Section 6.4, the rise in network error on unadapted phonemes could be due to the common front-end adaptation layer of the modified ANN not containing pure speaker information. Although we would like to see all the network errors fall after adaptation towards a single class, results show from Section 6.3 that these increases in network error are still relatively smaller when compared with results from a conventional MLP.

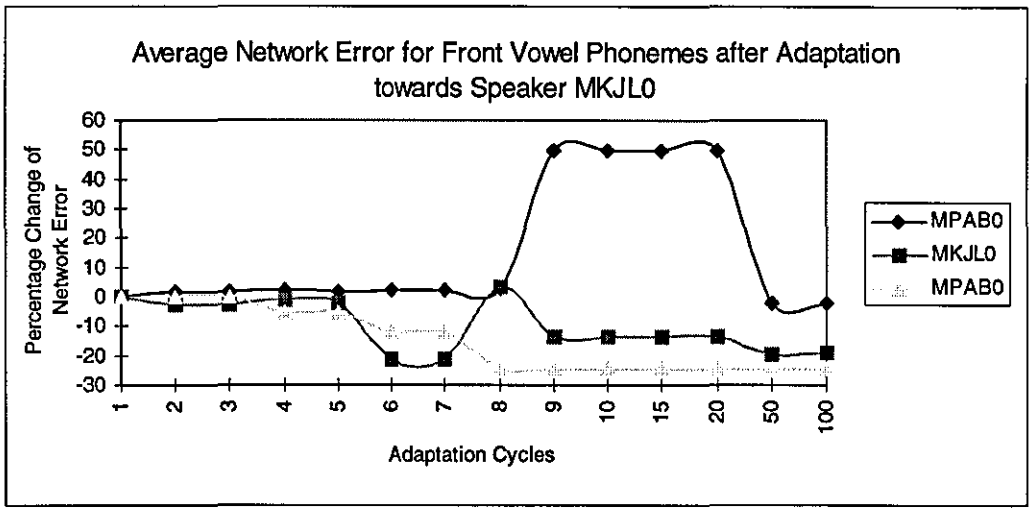
7.5 Inter-Speaker Adaptation

Using the same experimentation procedure to obtain the intra-speaker results, the three base classifiers, 'front', 'middle' and 'back', were again adapted towards each of the relevant phoneme utterances, one phoneme class from each speaker at a time. After adaptation towards a single phoneme class, the effects on the same phoneme classes from the remaining two speakers was recorded. As before, each phoneme was adapted for a total of 100 adaptation cycles, during which 15 result snapshots were taken at 1-10, 15, 20, 50 and 100 cycles, and then the ANN weight and bias values were reset.

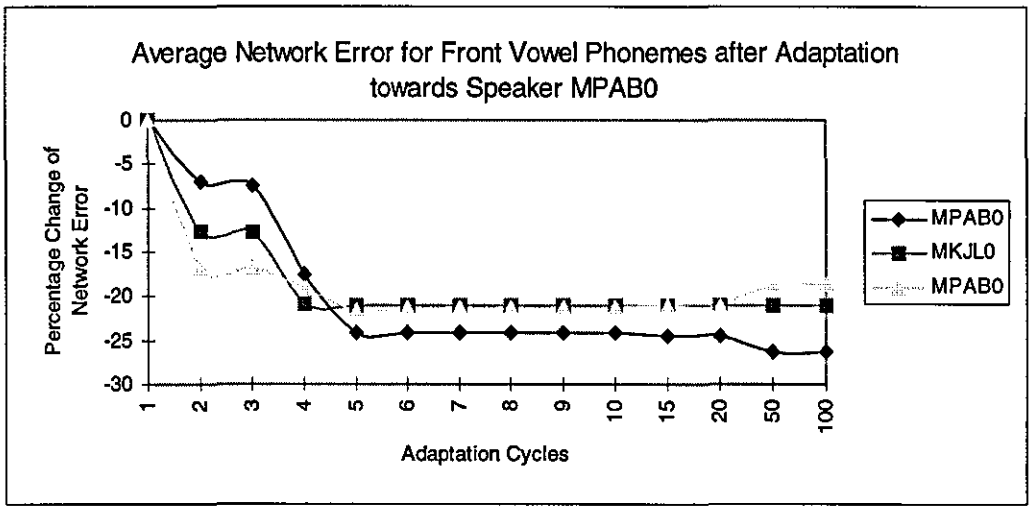
7.6 Inter-Speaker Results

The same normalisation technique, equation (7.2), was used to obtain the network error results with respect to their percentage change. The average results for adaptation towards the three speakers for the three tongue-hump groups was displayed using 9 graphs, Figure 7.2(a)-(i). A further 3 graphs showing the total average over all the vowel phonemes for each speaker offered a more general view of the networks behaviour, Figures 7.3(a) - (c).

(a)



(b)



(c)

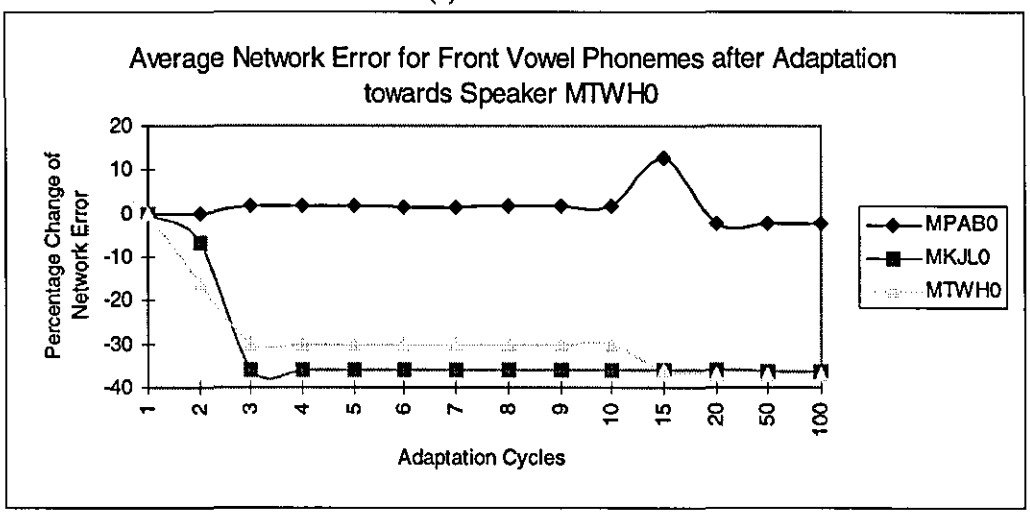
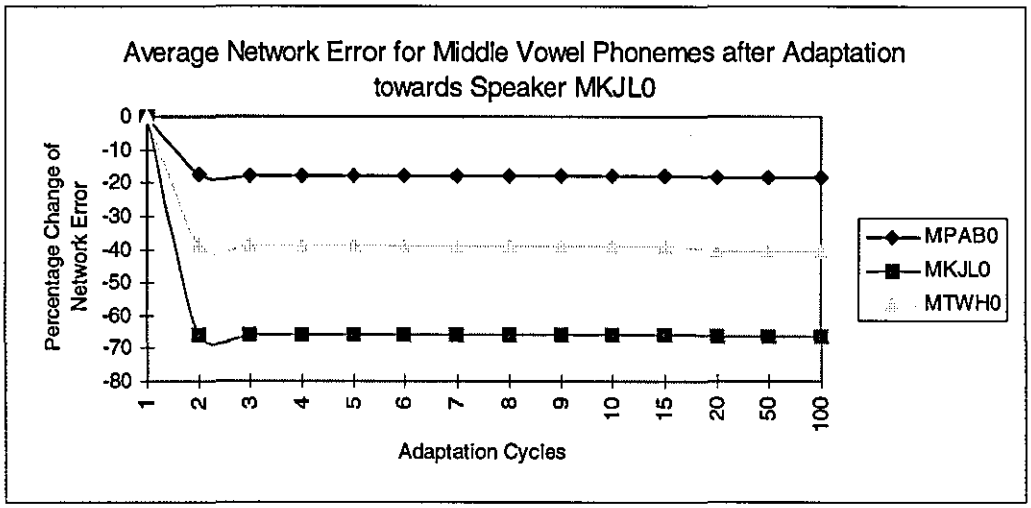
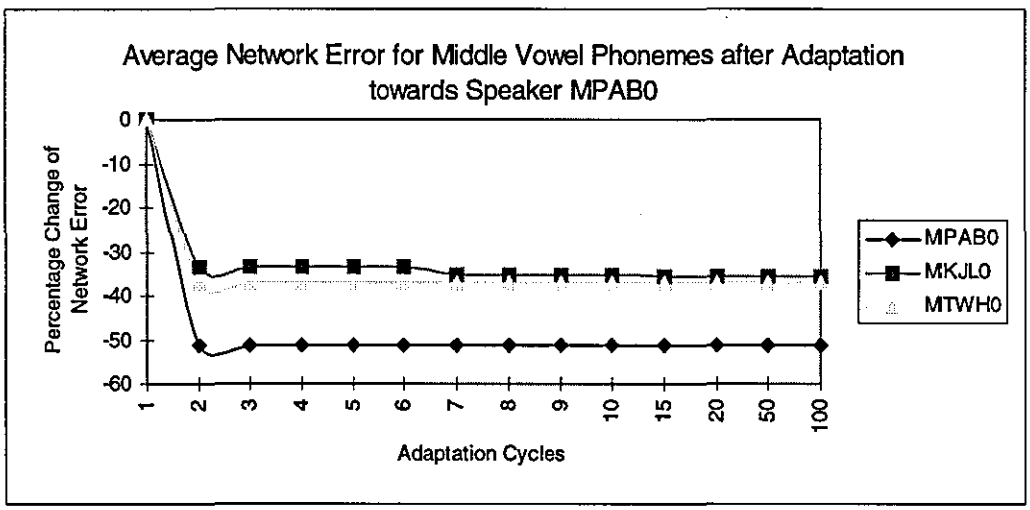


Figure 7.2(a), (b) and (c) Graphs showing the averaged inter-speaker adaptation results for the front vowel phonemes.

(d)



(e)



(e)

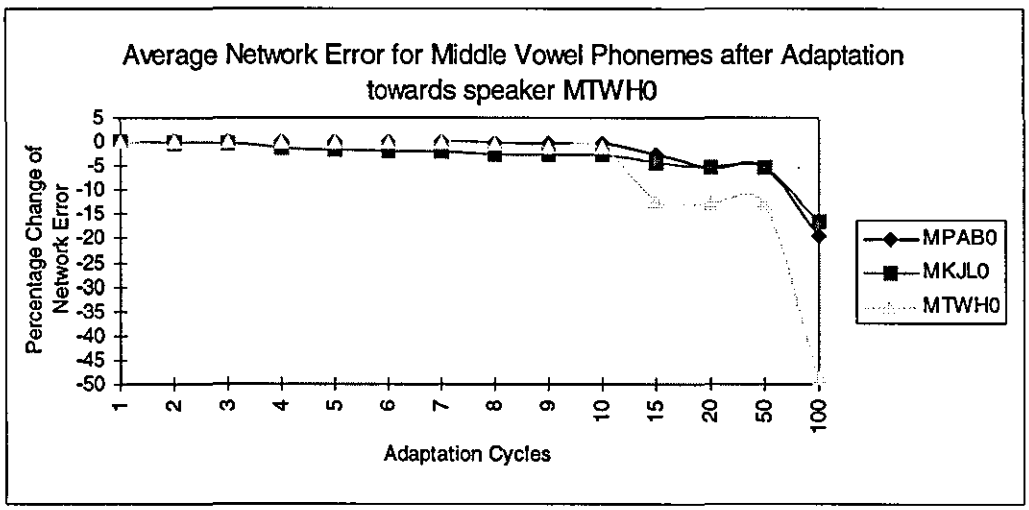
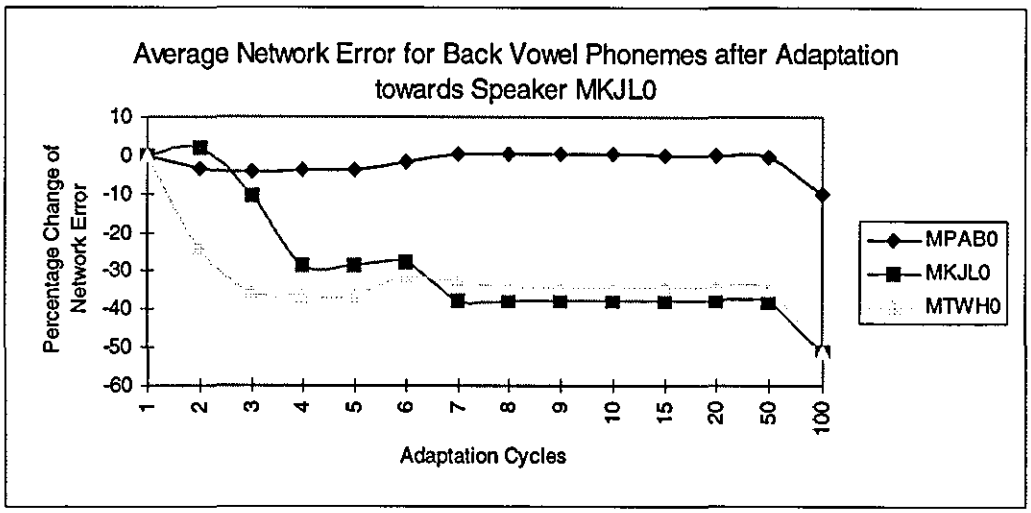
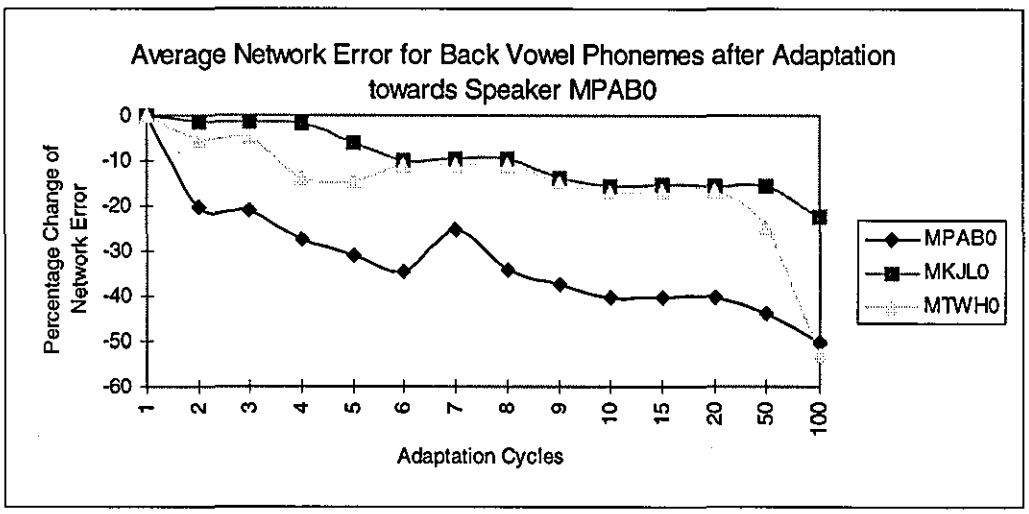


Figure 7.2(d), (e) and (f) Graphs showing the averaged inter-speaker adaptation results for the middle vowel phonemes.

(f)



(g)



(h)

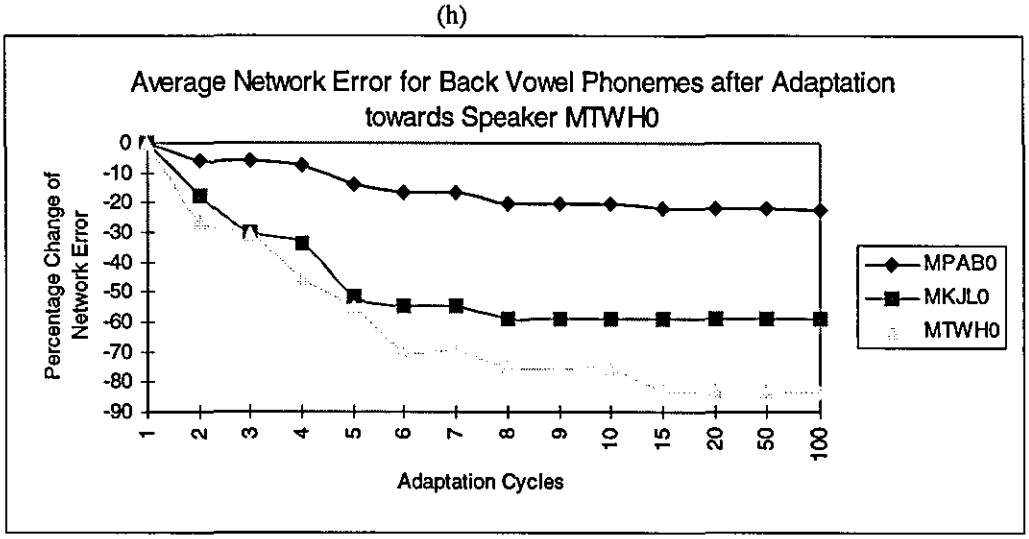
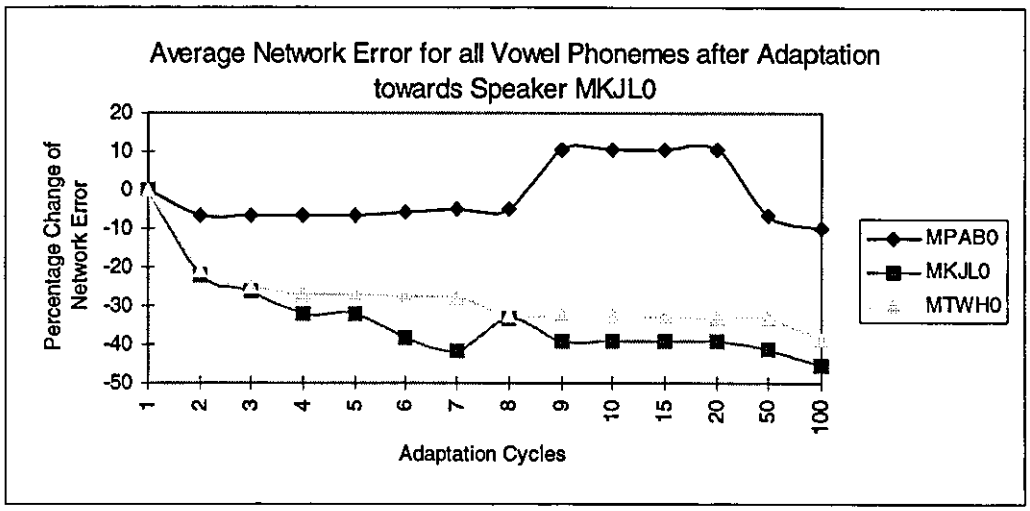
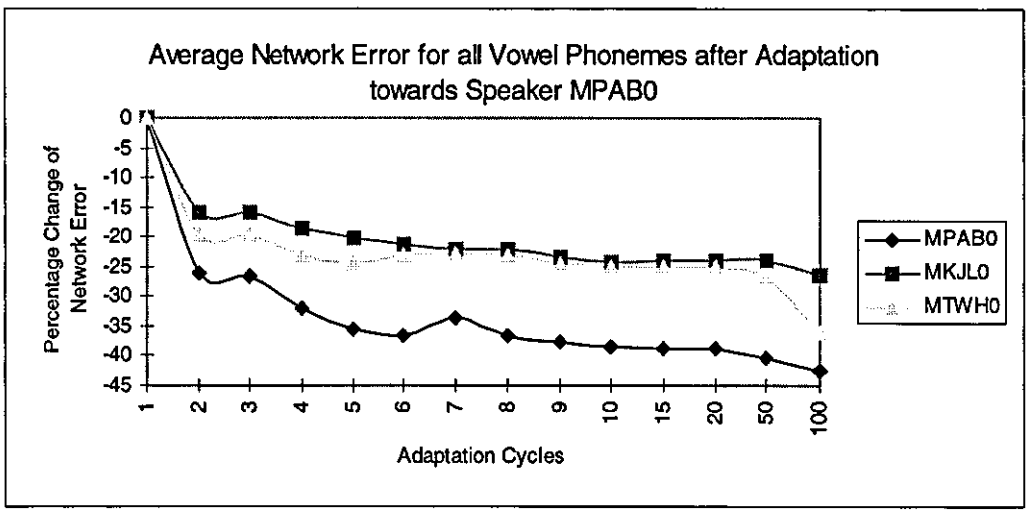


Figure 7.2(g), (h) and (i) Graphs showing the averaged inter-speaker adaptation results for the back vowel phonemes.

(a)



(b)



(c)

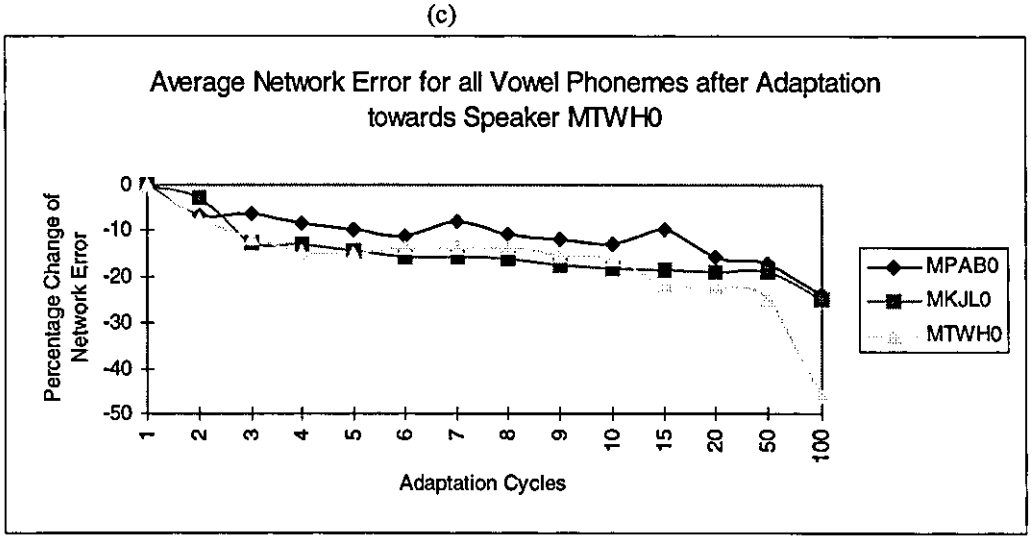
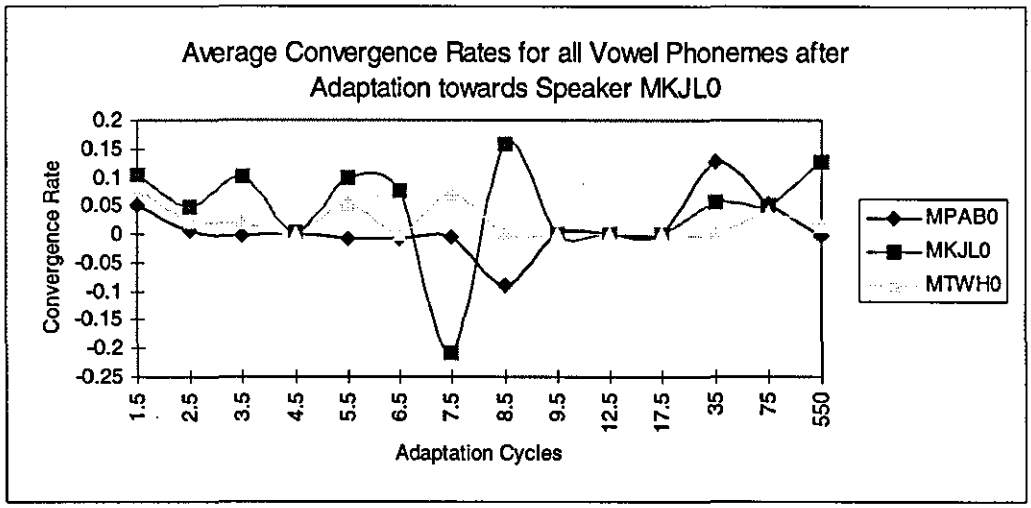
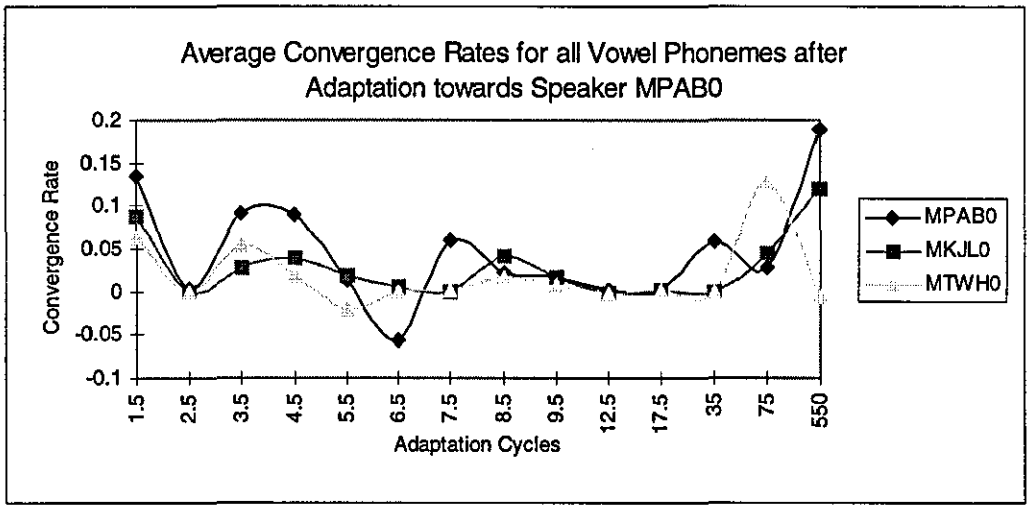
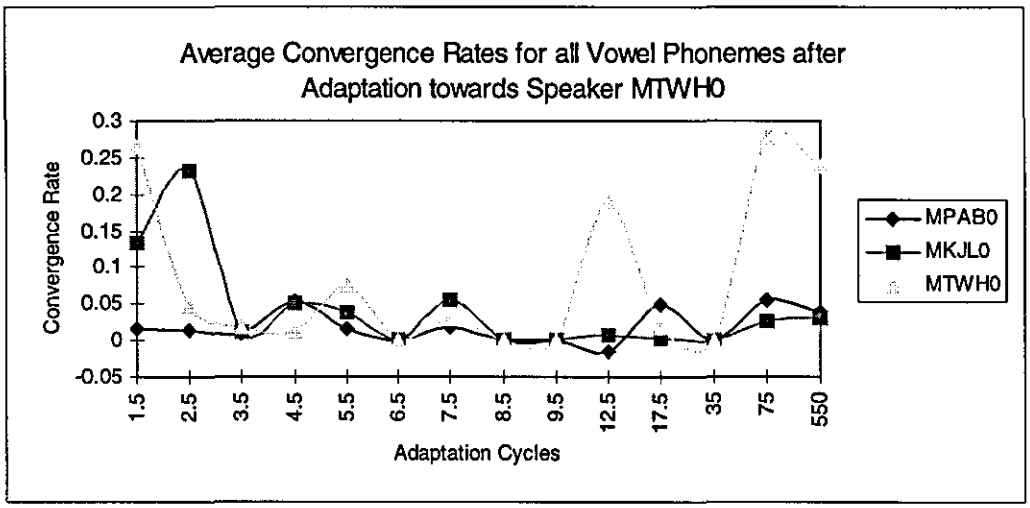


Figure 7.3(a), (b) and (c) Graphs showing the averaged inter-speaker adaptation results for all the vowel phonemes.


Average Convergence Rates for all Vowel Phonemes after Adaptation towards Speaker MKJL0
Convergence Rate
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
1.5
2.5
3.5
4.5
5.5
6.5
7.5
8.5
9.5
12.5
17.5
35
75
550
Adaptation Cycles
MPAB0
MKJL0
MTWH0


Average Convergence Rates for all Vowel Phonemes after Adaptation towards Speaker MPAB0
Convergence Rate
0.2
0.15
0.1
0.05
0
-0.05
-0.1
1.5
2.5
3.5
4.5
5.5
6.5
7.5
8.5
9.5
12.5
17.5
35
75
550
Adaptation Cycles
MPAB0
MKJL0
MTWH0


Average Convergence Rates for all Vowel Phonemes after Adaptation towards Speaker MTWH0
Convergence Rate
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
1.5
2.5
3.5
4.5
5.5
6.5
7.5
8.5
9.5
12.5
17.5
35
75
550
Adaptation Cycles
MPAB0
MKJL0
MTWH0

The inter-speaker results show conclusively the similarity between the speakers MKJL0 and MTWH0 and the differences they both have from the utterances of speaker MPAB0. Except for two accounts during adaptation towards front vowel phonemes, inter-speaker adaptation results in a fall in network error for all speakers. Even when adapting towards the distant speaker MPAB0, the similar speakers MKJL0 and MTWH0 adapt positively. This demonstrates the effectiveness of the front-end adaptation layer, storing common speaker information in the front-end adaptation layer. The more common the speaker information between speakers the better the inter-speaker adaptation, improving total speaker recognition.

7.7 Conclusion

The results indicate an improved performance for the newly modified ANN over a conventional MLP with respect to both intra- and inter-speaker adaptation. For intra-speaker adaptation, the network error for the adapted phoneme fell as expected. However, some of the network errors for the remaining unadapted phonemes increased. Although as explained earlier, this still offers better results than a conventional MLP, this rise could also be an indication that there is limited acoustic information between vowel phonemes from the same tongue-hump position. Although vowel phoneme from the same tongue-hump position contain comparable formant F_1 and F_2 information, Figure 3.2, this may not be enough to incorporate a reasonable proportion of speaker information. The inter-speaker results showed a network error fall for the adapted speaker phonemes, as well as for phonemes from the acoustically similar speaker. During the inter-speaker adaptation procedure, it was seen that the network error for the acoustically different speaker fell as well. This was something that was initially unexpected since the adaptation layer already containing dialect region 7 speaker data from the training set. Therefore adaptation towards a speaker's characteristics contained within a subset of that dialect acoustic space, should alienate a speaker's characteristics contained with a subset at the other end of the dialect acoustic space. However, when training an ANN, training doesn't continue until the network error for the entire training set is zero. This is because over-training of a network can lead to impaired recognition of new classes. Consequently this adaptation towards one speaker, could be adapting the adaptation layer towards new speaker information that is contained within both the similar and dissimilar speakers with respect to the existing information in the adaptation layer.

CHAPTER 8

Conclusion and Future Work

8.1 Experimental Conclusions

In this thesis a modified one-class-one-network ANN architecture was created to fulfill the three main research objectives:

(i) Develop an adaptation procedure that increases the convergence and reduces the processing time for each adaptation cycle.

(ii) Concentrate adaptation only on intra- and inter-speaker variables, i.e. speaker characteristics.

(iii) Allow adaptation towards a single speech class to improve recognition of remaining speech classes within the same vocabulary.

The first of the three objectives was fulfilled with the implementation of the One-Class-One-Network (OCON), details of which can be seen in Chapter 5. An OCON inherently contains less weighted connections than a conventional MLP resulting in a reduction in the processing time for each training and adaptation cycle. There is also an increase in the convergence rate since each OCON is dedicated to only a single class. As expected, Chapter 5 showed an increase in convergence rate and an improvement of adapted recognition rates against the MLP of over 12%. However, the OCON failed to fulfill the second and third research objectives. This was due to the adaptation procedure allowing global weight changes within each OCON and to the isolated nature of each OCON, preventing the use of inter-class information. This failure to utilise inter-class information was echoed in the results of Chapter 5 which saw a reduction of over 6% for unadapted phonemes, compared to the MLP results.

Although these results highlighted some benefits of OCON networks for phoneme recognition, some modification to the architecture was necessary to allow the isolation and utilisation of common class information, i.e. speaker characteristics. A modification to achieve this goal was introduced in Chapter 6. It consisted of a common front-end layer joining all the OCONs from the same tongue-hump group. Vowel phonemes containing the same tongue-hump position were grouped since they exhibited similar F_1-F_2 plane positions within the vowel triangle, indicating some acoustic similarities between them. The hypothesis for the modification was that such a layer should, after training, contain information common to all classes with class specific information stored in each of the relevant OCONs. By concentrating the adaptation only on this front-end layer such a network would in theory allow only common information, i.e. speaker characteristics, to be adapted. This would result in a more efficient adaptation procedure and should aid inter-class adaptation. To test this hypothesis two adaptation procedures were used, one allowing global weight changes throughout each OCON 'the unfrozen procedure' and the other only allowing weights within the front-end 'adaptation' layer to change 'the frozen procedure.' Results from Chapter 6 on the effects of adaptation on adapted phonemes showed a minimal difference between the two adaptation procedures. This was expected since the only differences between the two adaptation procedures was the number of neurons within each network that were updated. However, after adapting towards a single class, the two adaptation procedures showed clear differences in the recognition rates for the remaining classes within the same network. Despite both recognition rates for the unadapted phonemes falling during both adaptation procedures, the frozen adaptation procedures displayed an average 5.5% improvement over the unfrozen adaptation procedure. This improvement in the frozen adaptation procedure indicates that some common information is in fact being utilised by all the classes within the same network to help inter-class adaptation. Whether or not the common information contained speaker characteristics was not entirely clear at this stage, so inter-speaker experimentation was used to offer a better indication (Chapter 7).

Chapter 7 applied the frozen adaptation procedure to three speakers selected from the dialect region 7 TIMIT test set; two of the speakers were acoustically similar and the third was considered to be the most acoustically dissimilar. Adaptation towards a single class from one of the similar speakers improved the recognition rates of the same class for both the similar

and dissimilar speakers. However, the results for the similar speakers were considerable better than for the dissimilar speaker, strengthening the hypothesis that the front-end adaptation layer contained speaker characteristics. Also, adaptation towards a class from the dissimilar speaker equally improved the recognition rates of the same class for both the similar speakers. This additionally indicated that, although within the same dialect region they were considered to be different, the modified OCON network was utilising some common dialect information from the two similar and the dissimilar speakers. The intra- and inter-speaker results also showed that each network begins to converge immediately after the first adaptation cycle and that convergence is significant within 3-5 adaptation cycles. This is ideal, since a spoken work can quite often contain more than 3-5 vowel phonemes ensuring that speaker adaptation can begin after only a single utterance. This single word adaptation towards a new speaker would further improve recognition with the implementation of networks using fricative and plosive phonemes.

A data flow diagram of the completed system used during experimentation (see Figure 8.1) shows the pre-processing stage and the modified ANN classifier with its back-propagation feedback loop used during adaptation. The back-propagated error is fed back through the relevant fixed-weight OCON and used to adapt the weights within the adaptation layer.

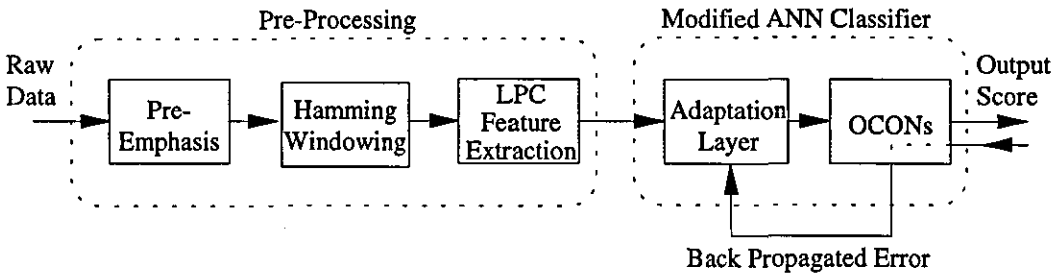


Figure 8.1 Data Flow Diagram of Modified ANN Recognition System

8.2 Concluding Remarks

The modified OCON ANN architecture fulfills all three of the research objectives offering improved and more efficient dynamic intra- and inter-speaker adaptation for vowel phonemes. However, although the behaviour of the network is decisive, the causes are not.

The ANN does indeed improve intra- and inter-speaker adaptation, but is this because the adaptation layer contains speaker characteristics? The nature of neural networks makes it very difficult to look at the output of a single layer or node and note what information is being held by its connecting weights. Consequently it is only possible to predict what is going on within an ANN by offering specific input data and noting its effect. This highlights the importance of good training and testing data and its application to the problem at hand. The speech data used was of high quality and obtained from a well recognised speech database [1]. However, the quantity of data was limited and due to the labour intensive nature of experimentation, only a few tests were applied. Despite this, the results obtained show promise for the modified OCON ANN architecture and emphasise the need for further experimentation with multiple speakers from multiple dialect regions.

8.3 Future Work

The results obtained from the research pinpoint two main areas of question. Firstly, is the adaptation layer within the modified ANN holding any speaker characteristics and secondly, what common speaker characteristics are actually contained within the vowel phonemes? Investigating the first question can be achieved by further experimentation with multiple speakers from multiple dialect regions. This would give a clearer indication of the information held within the adaptation layer and the limitations of its use for speaker from differing dialect regions. With regards to the second question, the issue of vowel grouping with respect to their tongue-hump position also needs to be addressed. Vowel phonemes from the same tongue-hump group do have similar F_1-F_2 plane positions but are these similarities used by the modified ANN? Intra-speaker results showed no obvious indication that tongue position influenced the rate of adaptation between phoneme classes, although using all the vowel phonemes in a single ANN may have degraded the results. However, it would be of interest to see the behaviour of the modified ANN using all the vowel phonemes within the same network or to train and test other phoneme classes such as fricatives and plosives. Finally it would be of interest to apply the modified ANN to other applications, such as optical character recognition, to monitor the segregation of any data into common class and class unique information.

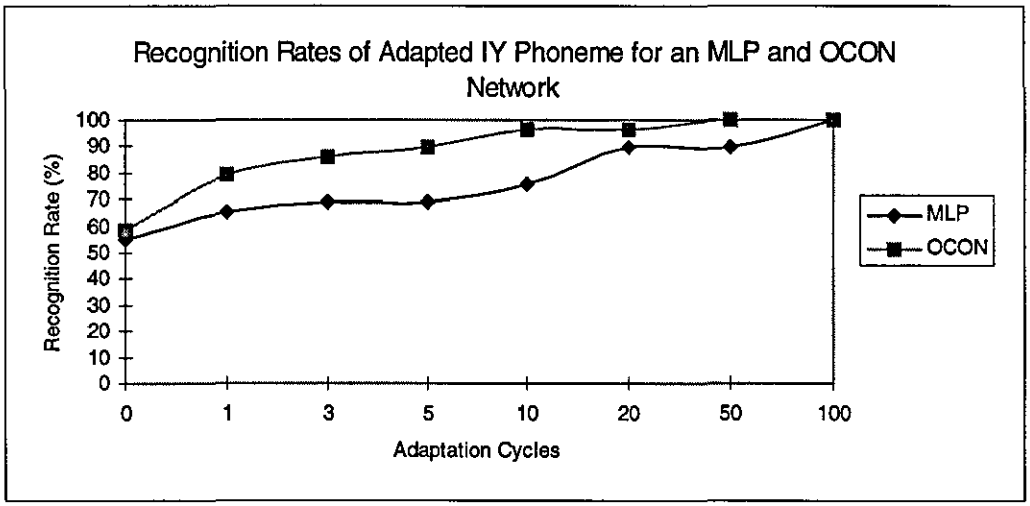
8.4 References

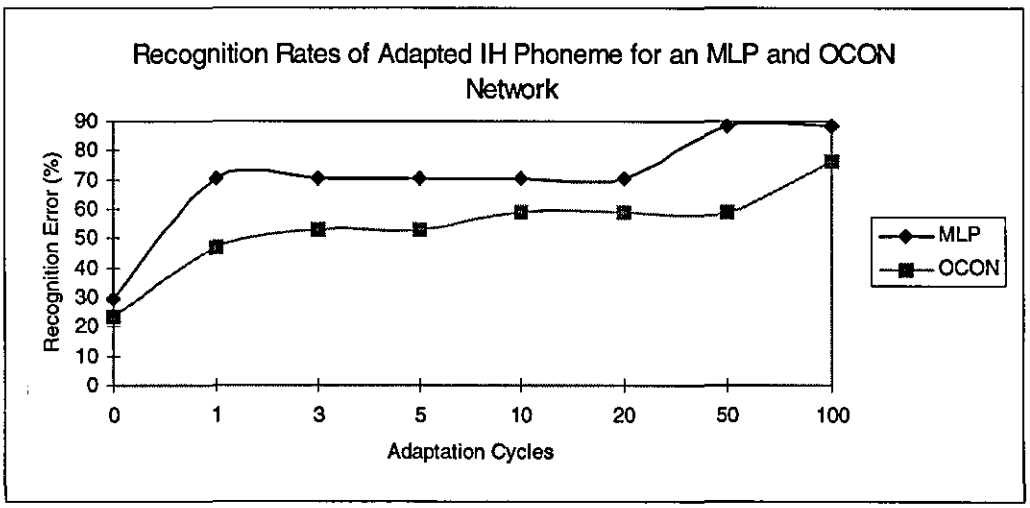
[1] DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM, Oct 1990. Web Address: //www.ldc.upenn.edu/

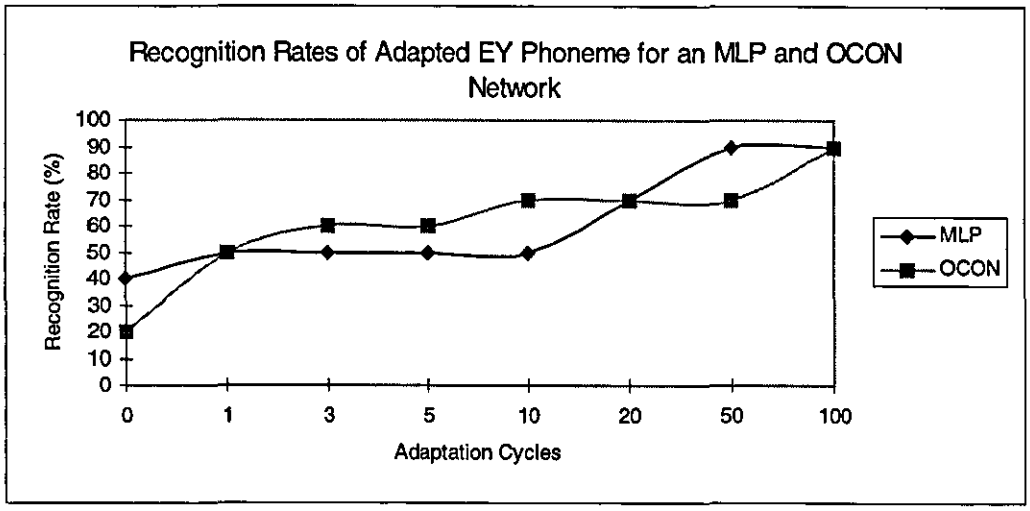
APPENDIX A

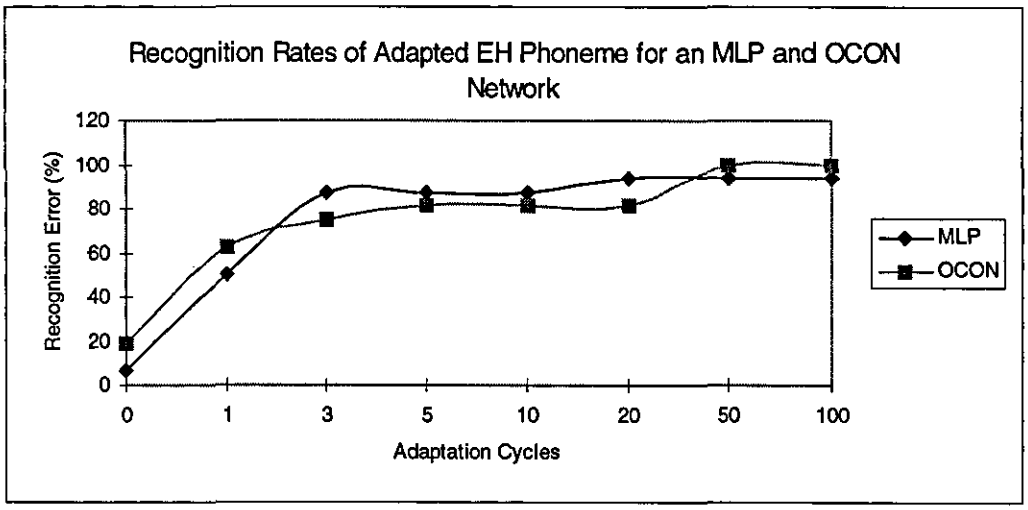
Chapter 5 Results

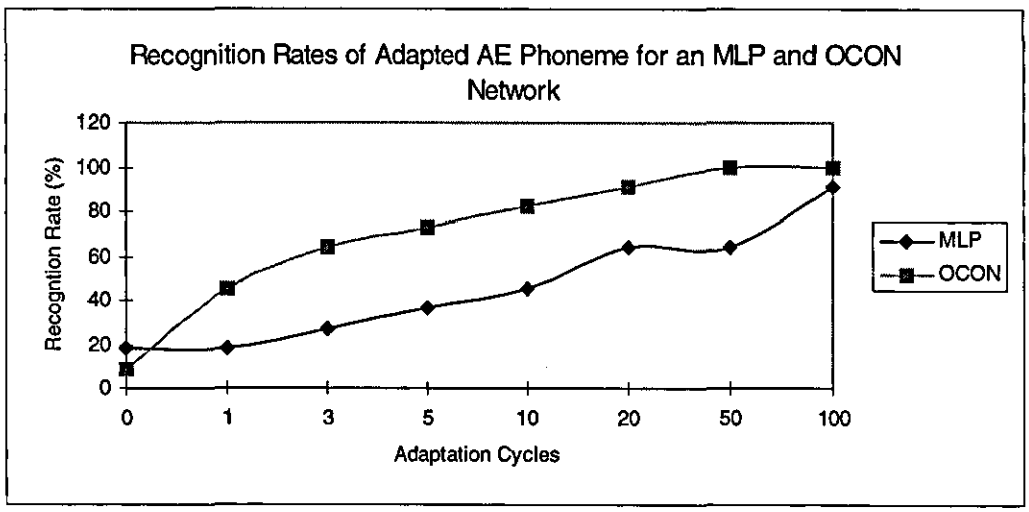
Appendix A contains 24 graphs showing comparative recognition rate results between MLP and OCON networks for each of the vowel phonemes and their average effect on the remaining unadapted vowel phonemes from the same tongue-hump group. Appendix A contains a further 6 graphs showing the convergence rates for the adapted and unadapted vowel phonemes from the three tongue-hump groups.

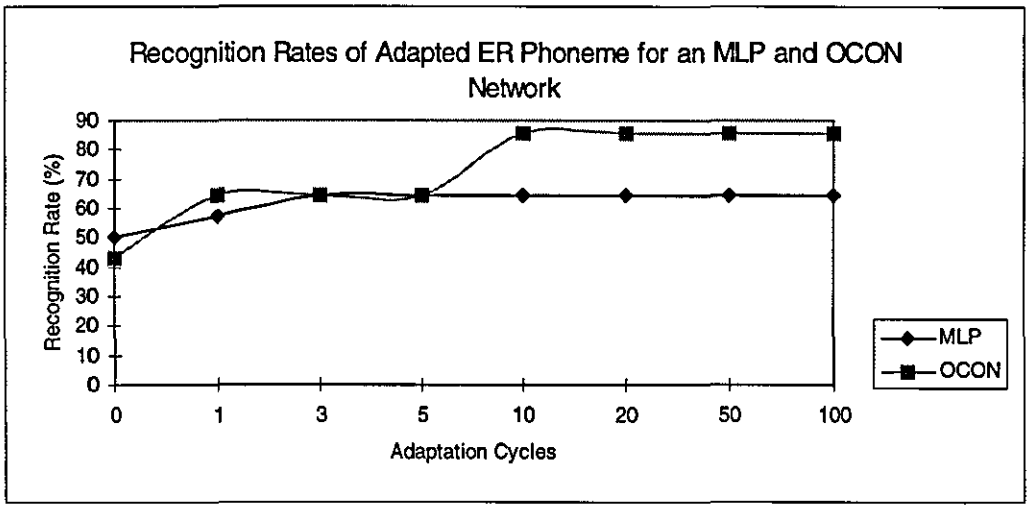


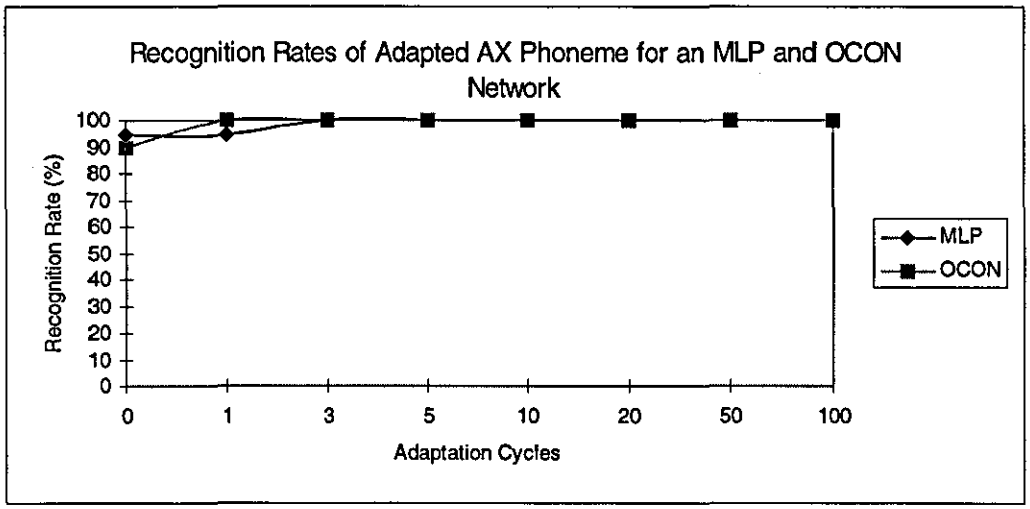


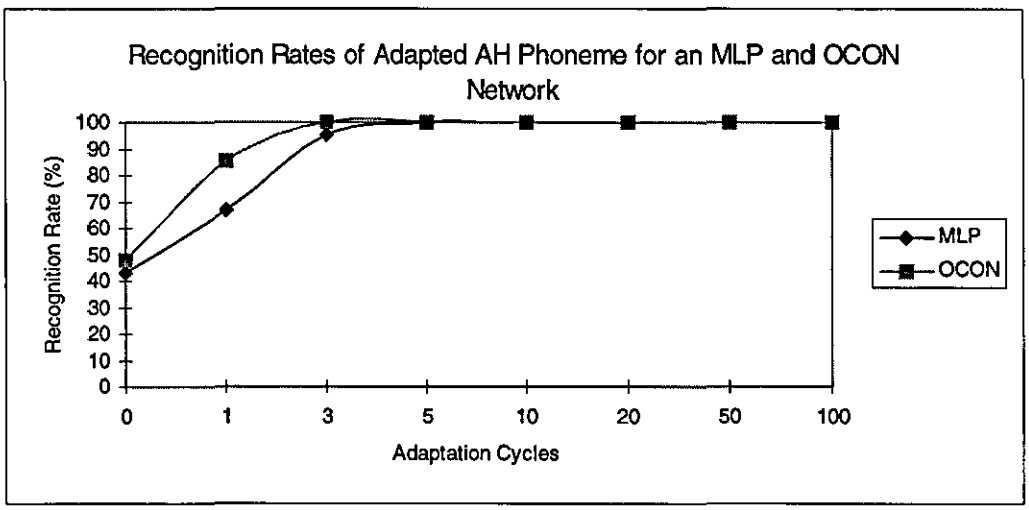

Recognition Rates of Adapted EY Phoneme for an MLP and OCON Network
Recognition Rate (%)
100
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

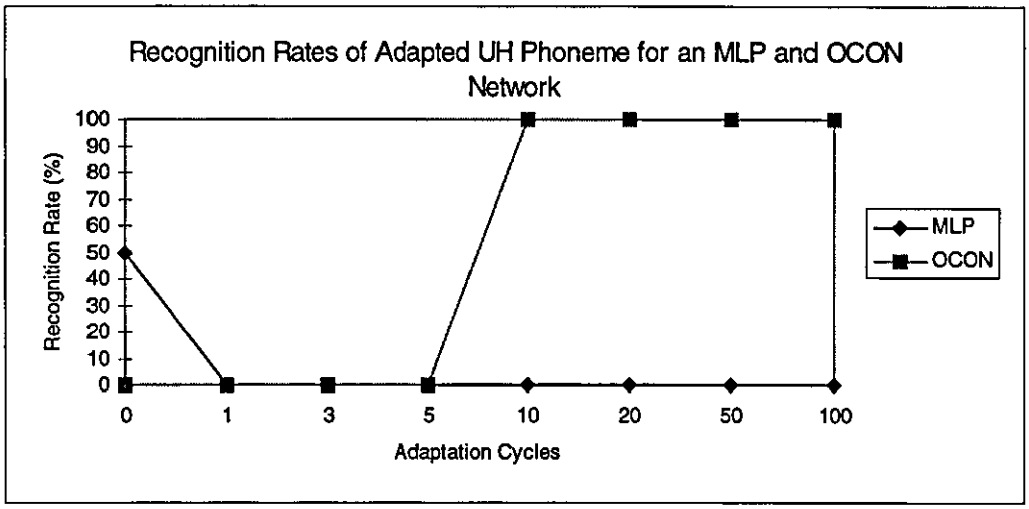

Recognition Rates of Adapted EH Phoneme for an MLP and OCON Network
Recognition Error (%)
120
100
80
60
40
20
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

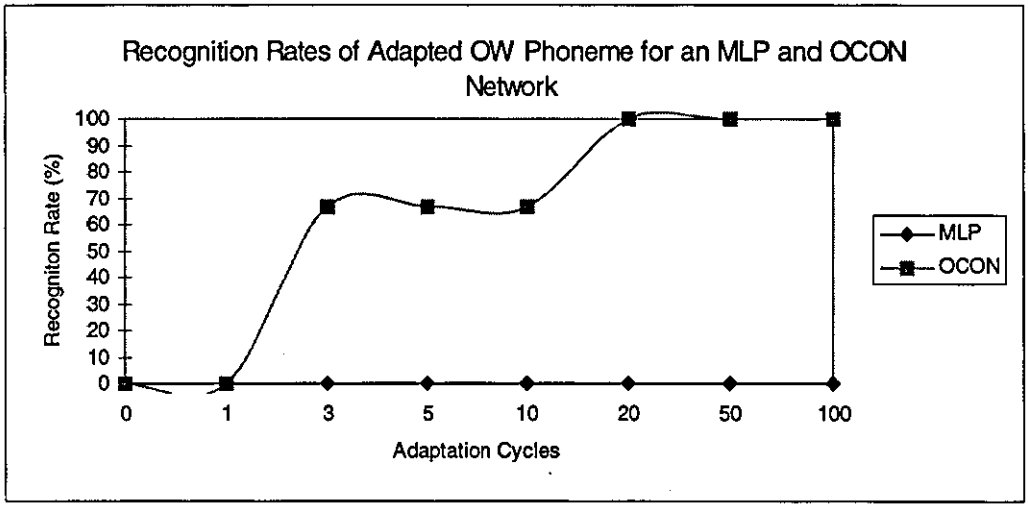

Recognition Rates of Adapted AE Phoneme for an MLP and OCON Network
Recognition Rate (%)
120
100
80
60
40
20
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

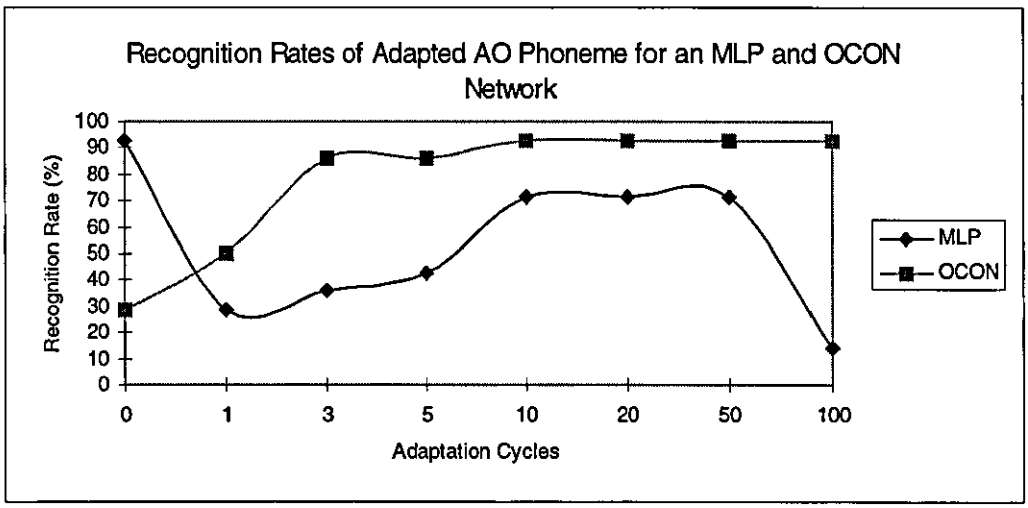

Recognition Rates of Adapted ER Phoneme for an MLP and OCON Network
Recognition Rate (%)
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

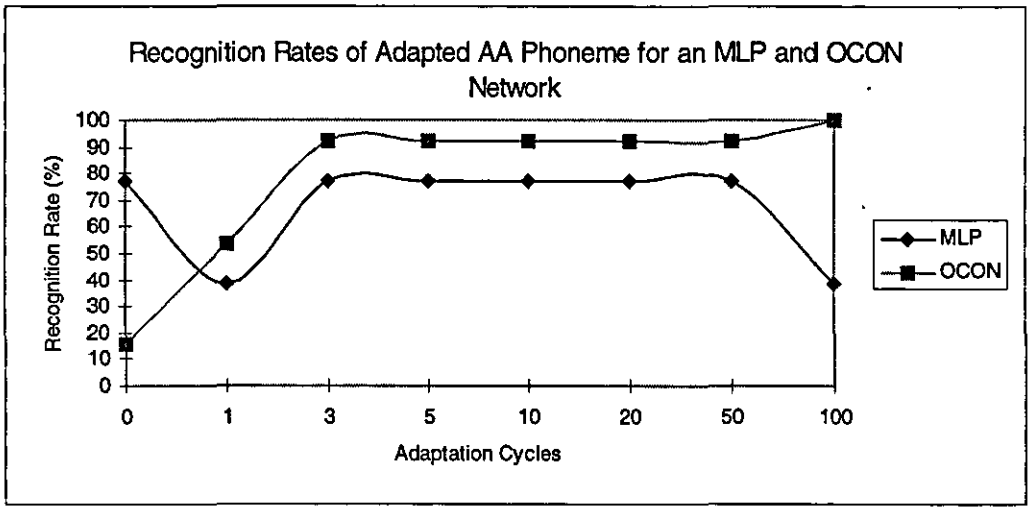

Recognition Rates of Adapted AX Phoneme for an MLP and OCON Network
Recognition Rate (%)
100
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

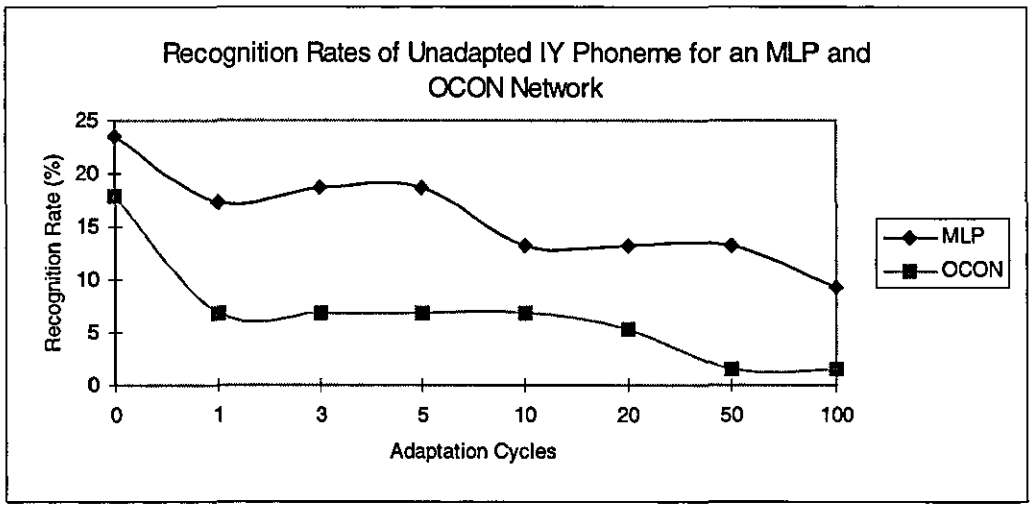

Recognition Rates of Adapted AH Phoneme for an MLP and OCON Network
Recognition Rate (%)
100
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

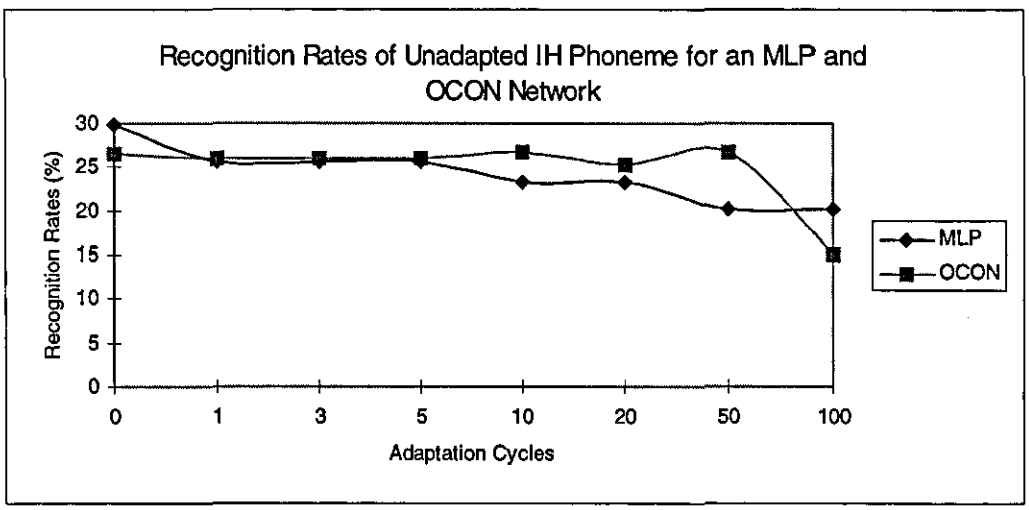

Recognition Rates of Adapted UH Phoneme for an MLP and OCON Network
Recognition Rate (%)
100
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

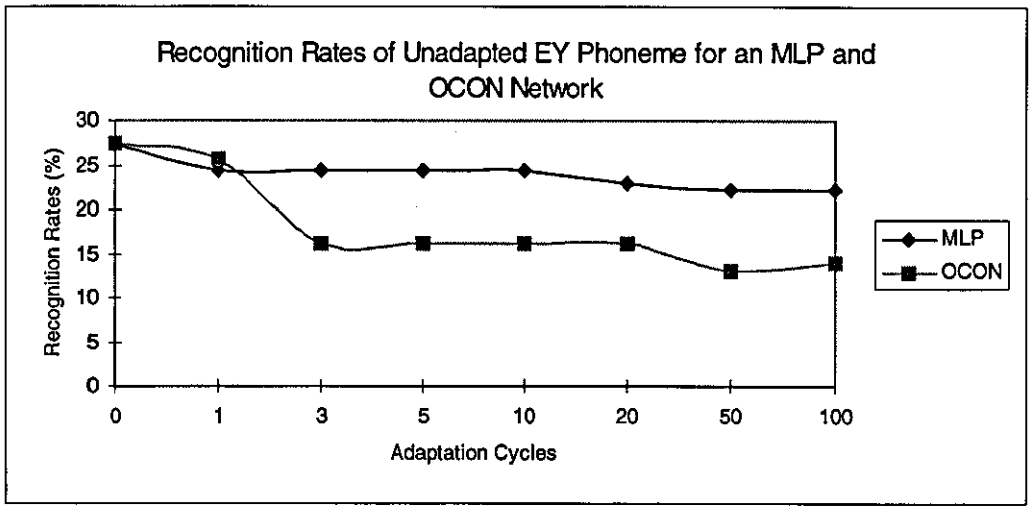

Recognition Rates of Adapted OW Phoneme for an MLP and OCON Network
Recogniton Rate (%)
100
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

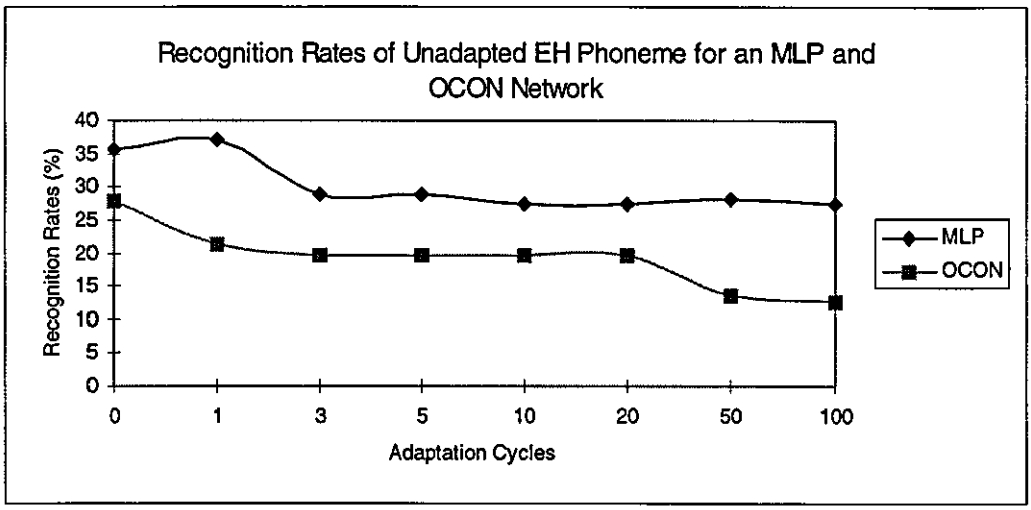

Recognition Rates of Adapted AO Phoneme for an MLP and OCON Network
Recognition Rate (%)
100
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

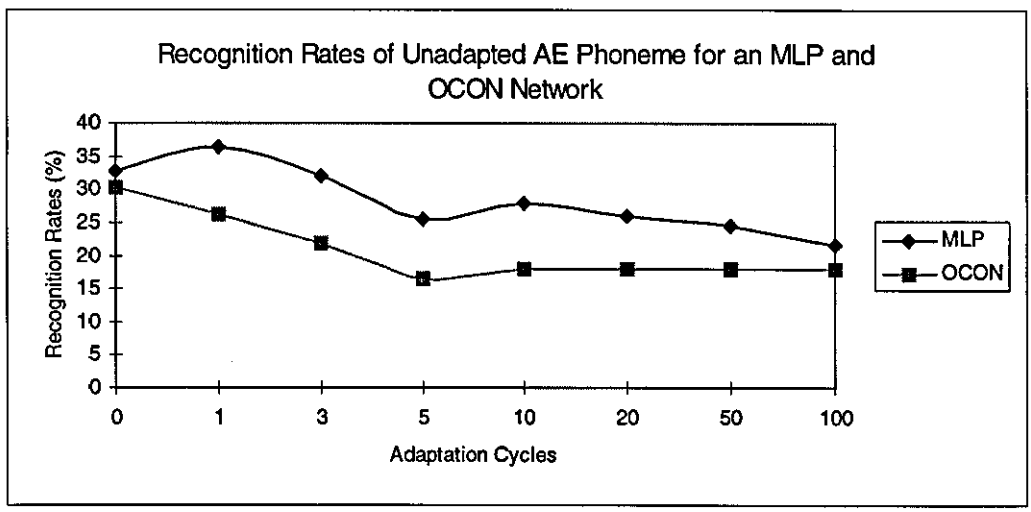

Recognition Rates of Adapted AA Phoneme for an MLP and OCON Network
Recognition Rate (%)
100
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

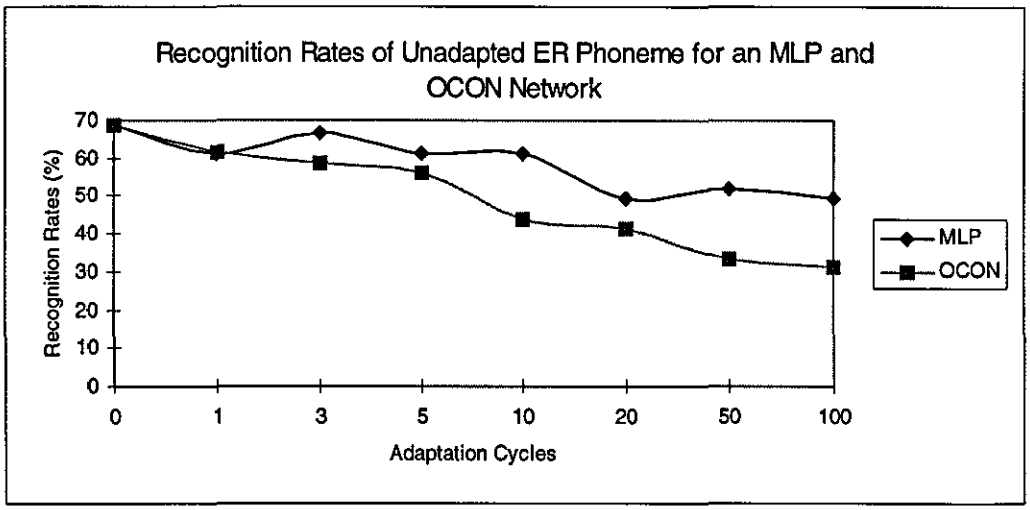

Recognition Rates of Unadapted IY Phoneme for an MLP and OCON Network
Recognition Rate (%)
25
20
15
10
5
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

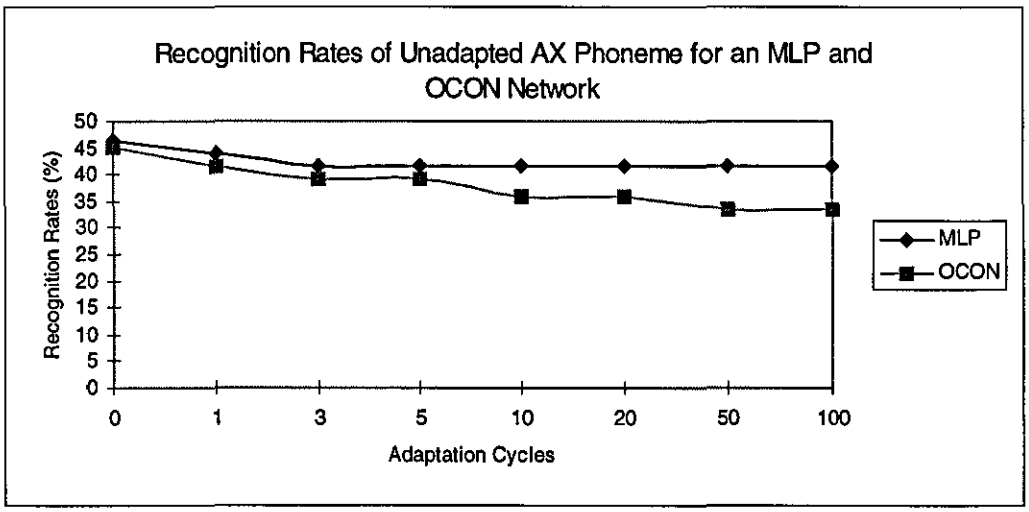

Recognition Rates of Unadapted IH Phoneme for an MLP and OCON Network
Recognition Rates (%)
30
25
20
15
10
5
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

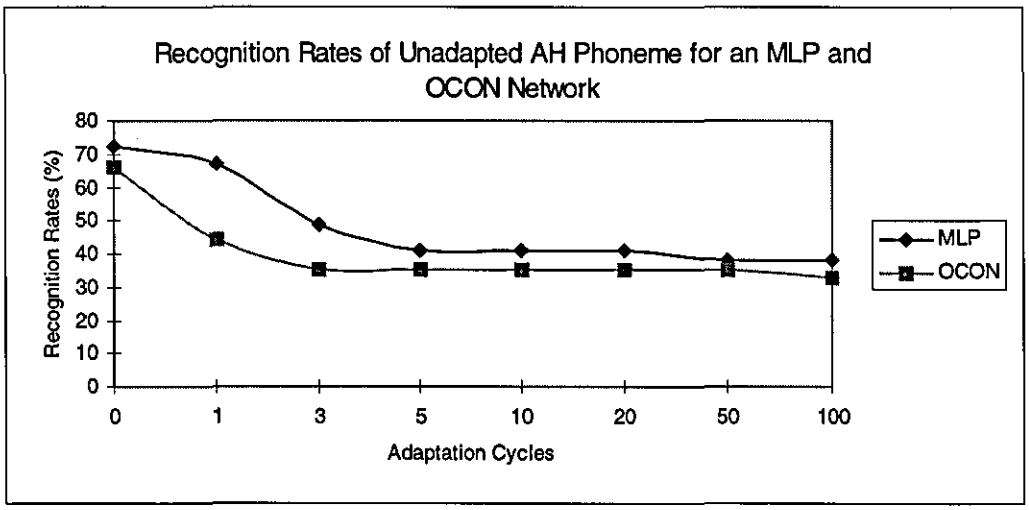

Recognition Rates of Unadapted EY Phoneme for an MLP and OCON Network
Recognition Rates (%)
30
25
20
15
10
5
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

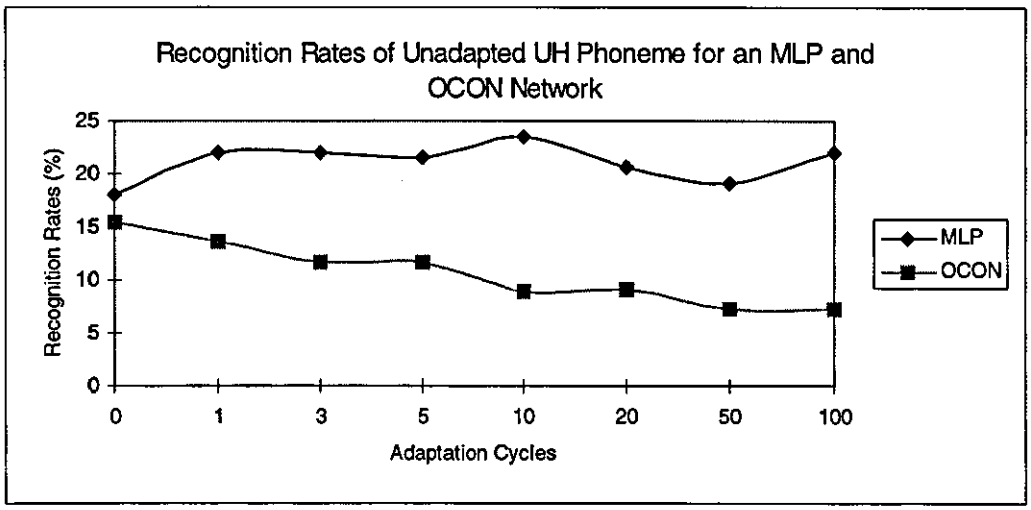

Recognition Rates of Unadapted EH Phoneme for an MLP and OCON Network
Recognition Rates (%)
40
35
30
25
20
15
10
5
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

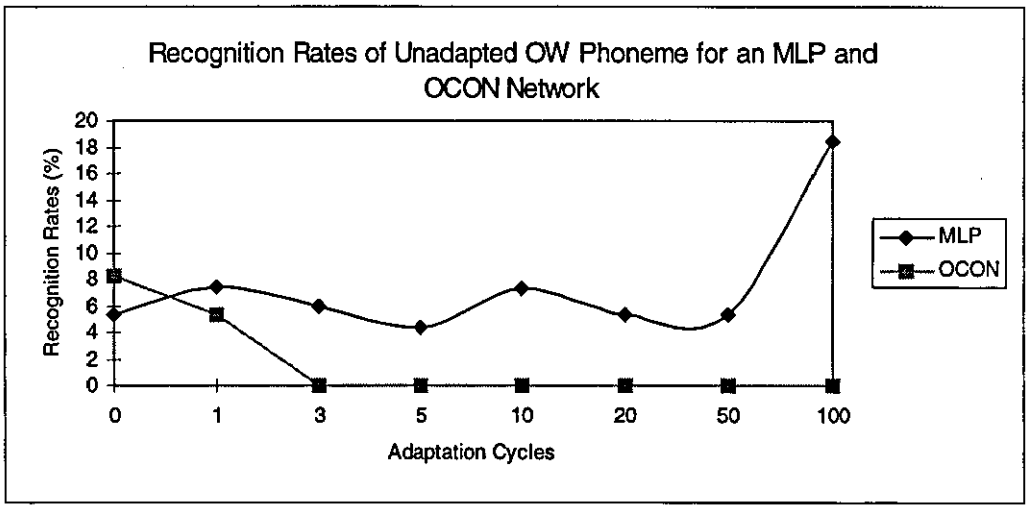

Recognition Rates of Unadapted AE Phoneme for an MLP and OCON Network
Recognition Rates (%)
40
35
30
25
20
15
10
5
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

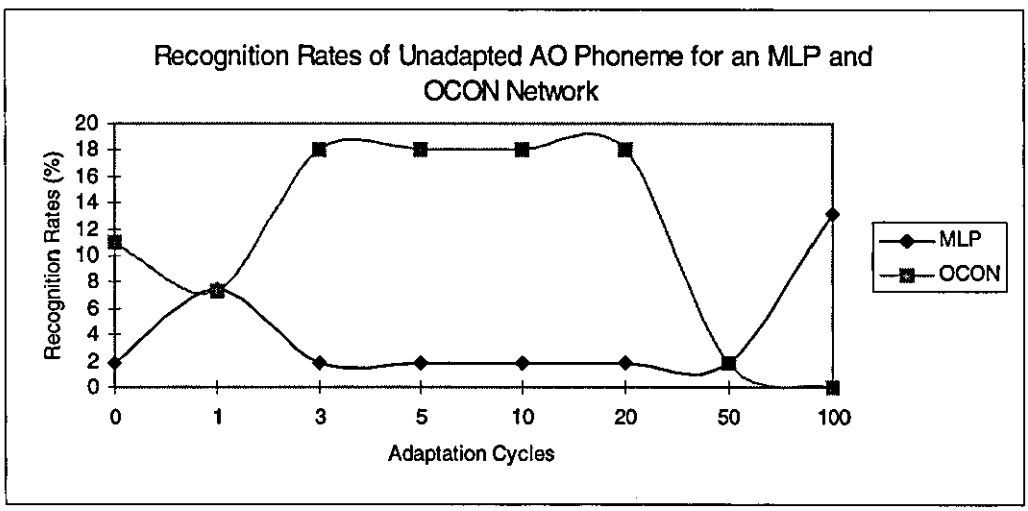

Recognition Rates of Unadapted ER Phoneme for an MLP and OCON Network
Recognition Rates (%)
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

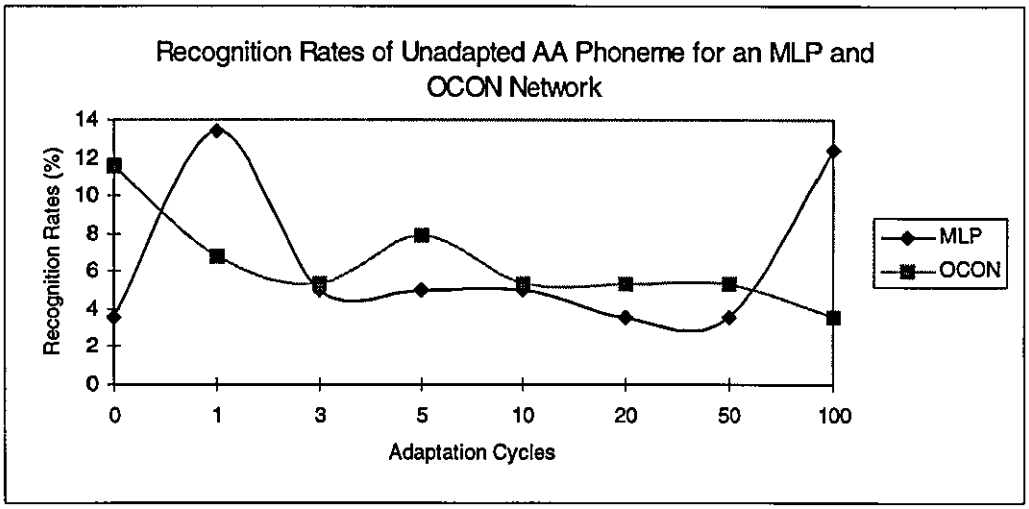

Recognition Rates of Unadapted AX Phoneme for an MLP and OCON Network
Recognition Rates (%)
50
45
40
35
30
25
20
15
10
5
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

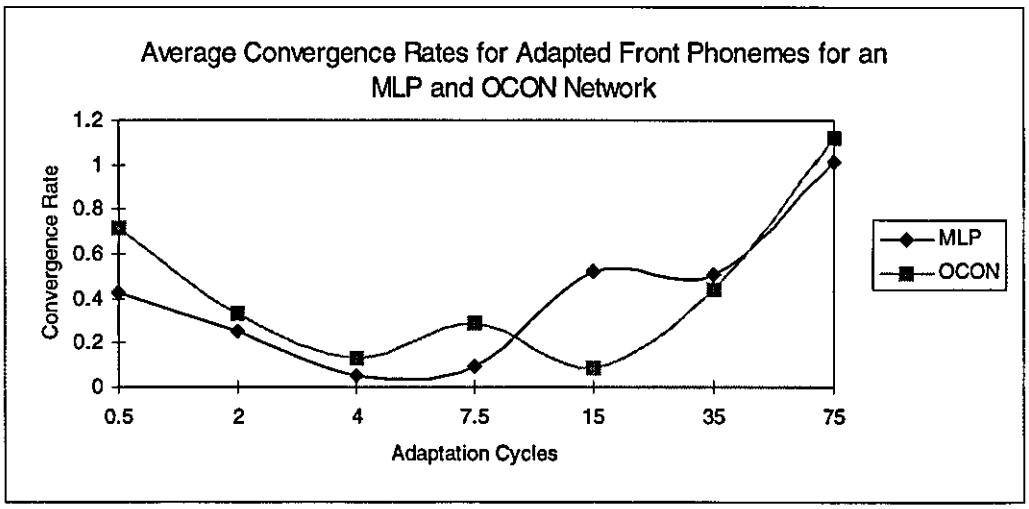

Recognition Rates of Unadapted AH Phoneme for an MLP and OCON Network
Recognition Rates (%)
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

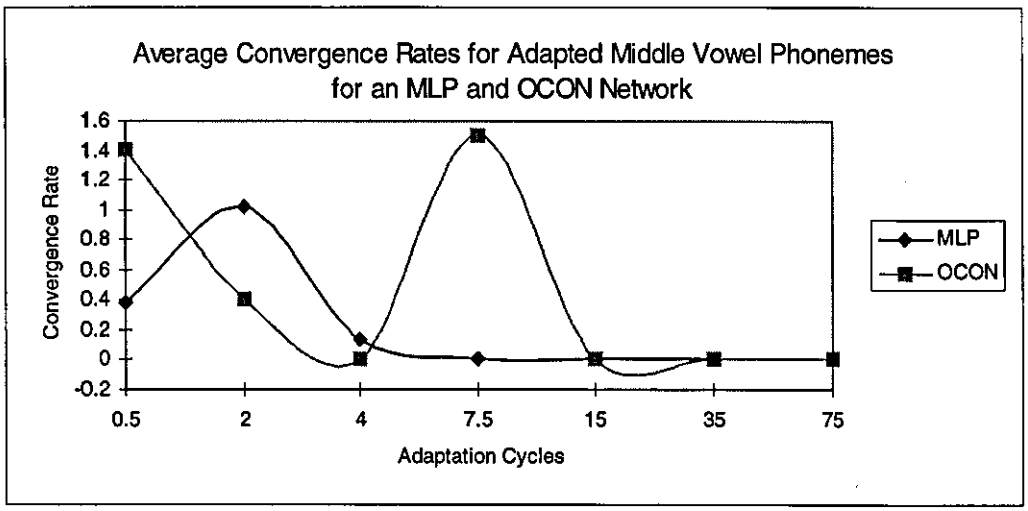

Recognition Rates of Unadapted UH Phoneme for an MLP and OCON Network
Recognition Rates (%)
25
20
15
10
5
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

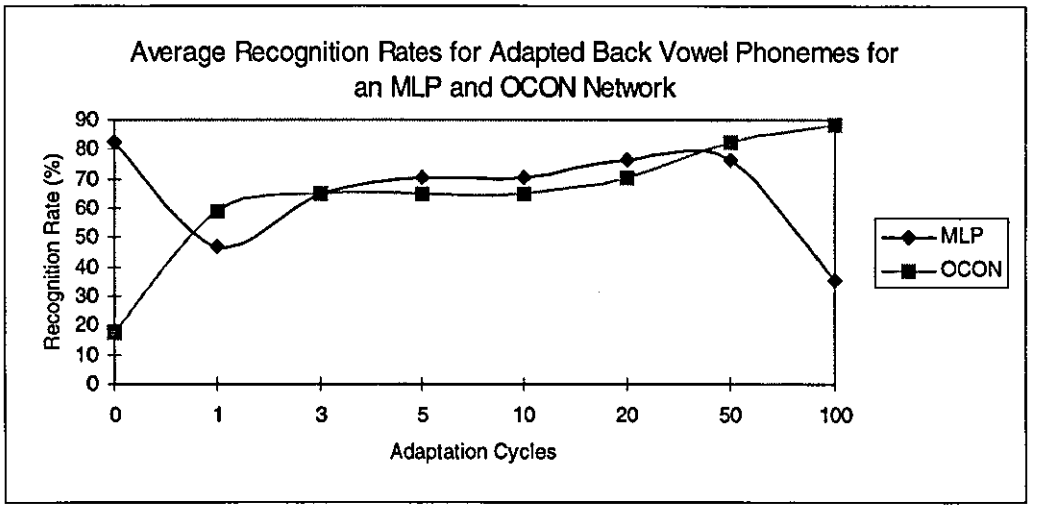

Recognition Rates of Unadapted OW Phoneme for an MLP and OCON Network
Recognition Rates (%)
20
18
16
14
12
10
8
6
4
2
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

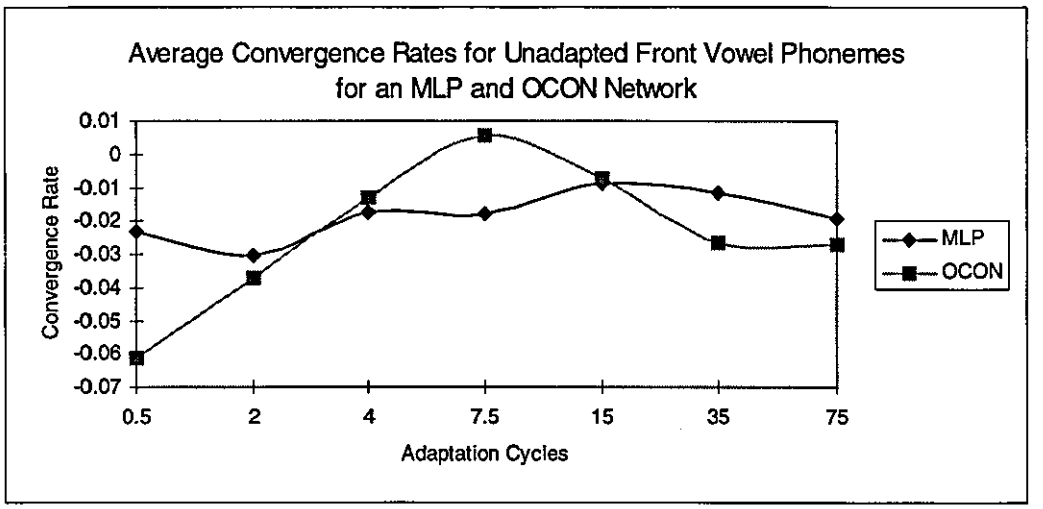

Recognition Rates of Unadapted AO Phoneme for an MLP and OCON Network
Recognition Rates (%)
20
18
16
14
12
10
8
6
4
2
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

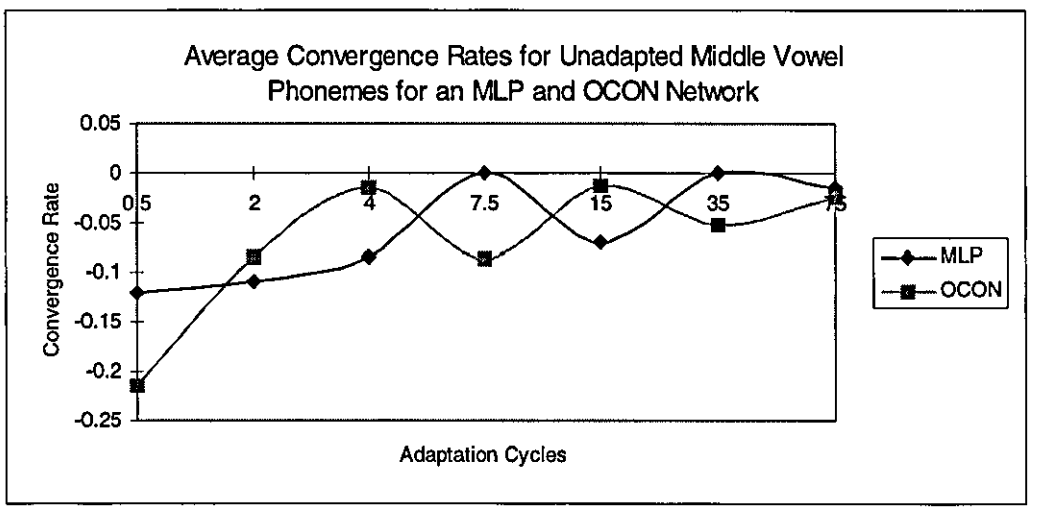

Recognition Rates of Unadapted AA Phoneme for an MLP and OCON Network
Recognition Rates (%)
14
12
10
8
6
4
2
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON

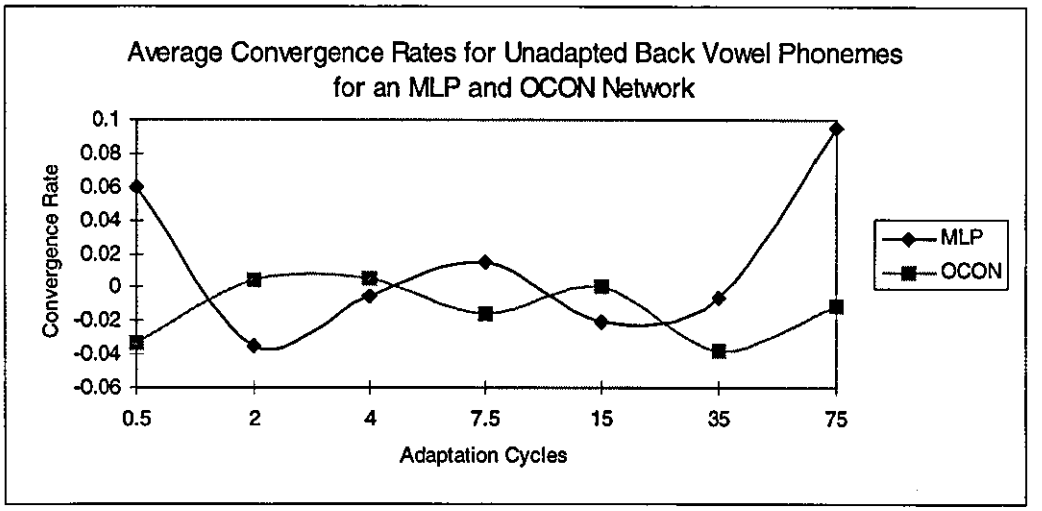

Average Convergence Rates for Adapted Front Phonemes for an MLP and OCON Network
Convergence Rate
1.2
1
0.8
0.6
0.4
0.2
0
0.5
2
4
7.5
15
35
75
Adaptation Cycles
MLP
OCON


Average Convergence Rates for Adapted Middle Vowel Phonemes for an MLP and OCON Network
Convergence Rate
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.5
2
4
7.5
15
35
75
Adaptation Cycles
MLP
OCON


Average Recognition Rates for Adapted Back Vowel Phonemes for an MLP and OCON Network
Recognition Rate (%)
90
80
70
60
50
40
30
20
10
0
0
1
3
5
10
20
50
100
Adaptation Cycles
MLP
OCON


Average Convergence Rates for Unadapted Front Vowel Phonemes for an MLP and OCON Network
Convergence Rate
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
0.5
2
4
7.5
15
35
75
Adaptation Cycles
MLP
OCON


Average Convergence Rates for Unadapted Middle Vowel Phonemes for an MLP and OCON Network
Convergence Rate
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0.5
2
4
7.5
15
35
75
Adaptation Cycles
MLP
OCON


Average Convergence Rates for Unadapted Back Vowel Phonemes
for an MLP and OCON Network
Convergence Rate
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
0.5
2
4
7.5
15
35
75
Adaptation Cycles
MLP
OCON

APPENDIX B

Alterations to SNNS Software

The software alterations were made in two main areas, the source code's 'kernel' and the graphical interface 'XGUI.'

XGUI Alterations

The first procedure was to create two buttons on the 'Info Panel' that when depressed toggled a new flag 'frozen' between 0 and 1. This required alterations being made to four files, 'Ui_info.c' to create the button, 'Ui_infop.c' to create a procedure to change the 'frozen' flag and two header files, 'Kr_typ.h' and 'Ui_infop.h,' to initialise the new flag and procedures. The altered code of 'Ui_info.c,' Figure B.1, created two buttons called 'WEIGHT FREEZE' and 'WEIGHT UNFREEZE' which were linked to the two procedures 'freeze_links' and 'unfreeze_links' respectively.

```
/**************  BUTTONS & NAME TARGET   **********/

button =
     ui_xCreateButtonItem("WEIGHT FREEZE",
     ui_infoPanel,button,tarFuncLabel);

     XtAddCallback(button, XtNcallback,(XtCallbackProc)
     freeze_links, (caddr_t) UI_TARGET);

button =
     ui_xCreateButtonItem("WEIGHT UNFREEZE",
     ui_infoPanel,button,tarFuncLabel);

     XtAddCallback(button, XtNcallback,(XtCallbackProc)
     unfreeze_links, (caddr_t) UI_TARGET);
```

Figure B.1 Changes made to 'Ui_info.c' code

The procedures 'freeze_links' and 'unfreeze_links,' called by the two new buttons, were initialised using changes made to 'Ui_infop.h,' figure B.2.

```
extern void freeze_links (Widget, int, caddr_t);
extern void unfreeze_links (Widget, int, caddr_t);
```

Figure B.2 Changes made to 'Ui_infop.h' code

Kernel Alterations

The first kernel alteration initialised the flag 'frozen' in the header file 'Kr_typ.h,' (Figure B.3). Then the two procedures 'freeze_links' and 'unfreeze_links' were created in the file 'Ui_infop.c,' (Figure B.4(a) and B.4(b)). In these two procedures the flag 'frozen' for the desired network node 'unit.no' was assigned either a value of 0 to unfreeze or 1 to freeze all links to that node.

```
/*  Link structure  */

struct   Link
     {
     struct Unit *to;  /* points to the source unit */
     FlintType   weight;           /*  link weight  */
     FlintType   value_a,       /* general elements */
                 value_b,    /* learning functions  */
                 value_c;

     struct Link *next;              /*  next link  */
     FlintType   frozen;          /*  freeze weight */
     };
```

Figure B.3 Changes made to file 'Kr_typ.h'

```
/*******************************************************
 FUNCTION : freeze_links

 PURPOSE  : To freeze weights connected to selected node
 RETURNS  :
 NOTES    :

 UPDATES  :  25/11/96
*******************************************************/

void freeze_links (Widget w, int structure, caddr_t call_data)
{
      char buf[MAX_NAME_LENGTH];
      register struct Link *link_ptr;
      register struct Unit *unit_ptr;
      register struct UnitAttributeType unit;

      if (NOT ui_info_anyUnitSelected(UI_SOURCE))
      {
            ui_printMessage("Select a Unit!");
            return;
      }

      if (NOT ui_info_anyUnitSelected(UI_TARGET))
      {
            ui_printMessage("Select a Unit!");
            return;
      }

      if (structure == UI_SOURCE)
            unit = ui_sourceUnit;
      else
            unit = ui_targetUnit;

      ui_info_storeAttributes(unit.no, unit);

      sprintf(buf,"Unit %d is now frozen.",unit.no);
      ui_printMessage(buf);

      unit_ptr = kr_getUnitPtr(unit.no);

      unit_ptr->frozen = 1;

      FOR_ALL_LINKS (unit_ptr, link_ptr)
            link_ptr->frozen = 1;
}
```

Figure B.4(a) Freeze_links procedure created in the file 'Ui_infop.c.'

```
/*************************************************************
 FUNCTION : unfreeze_links

 PURPOSE  : To unfreeze weights connected to selected nodes
 RETURNS  :
 NOTES    :

 UPDATES  :  25/11/96
*************************************************************/

void unfreeze_links (Widget w, int structure, caddr_t call_data)
{

      char buf[MAX_NAME_LENGTH];
      register struct Link *link_ptr;
      register struct Unit *unit_ptr;
      register struct UnitAttributeType unit;

      if (NOT ui_info_anyUnitSelected(UI_SOURCE))
      {
            ui_printMessage("Select a Unit!");
            return;
      }

      if (NOT ui_info_anyUnitSelected(UI_TARGET))
      {
            ui_printMessage("Select a Unit!");
            return;
      }

      if (structure == UI_SOURCE)
            unit = ui_sourceUnit;
      else
            unit = ui_targetUnit;

      ui_info_storeAttributes(unit.no, unit);

      sprintf(buf,"Unit %d is now unfrozen.",unit.no);
      ui_printMessage(buf);

      unit_ptr = kr_getUnitPtr(unit.no);

      unit_ptr->frozen = 0;

      FOR_ALL_LINKS (unit_ptr, link_ptr)
            link_ptr->frozen = 0;
}
```

Figure B.4(b) Unfreeze_links procedure created in the file 'Ui_infop.c.'

The final alterations to the kernel source code were to the file 'learn_f.c' which contained all the learning algorithms (see Figure B.5). Selecting the learning algorithms that would be used during experimentation, primarily back-propagation, conditional statements were added so that weight changes would not take place when the flag 'freeze' for a selected node were equal to 1.

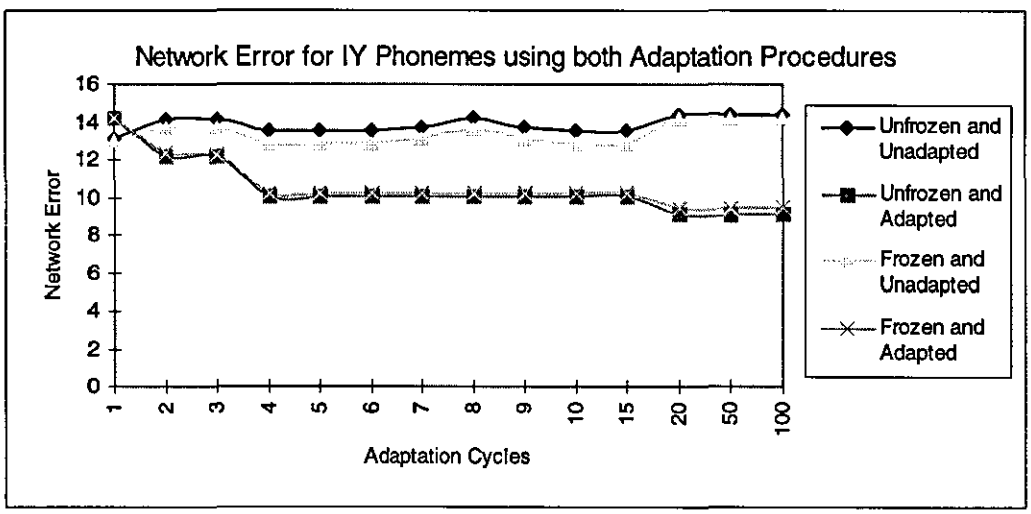
```
if ( (flags & UFLAG_IN_USE) == UFLAG_IN_USE)
 {
      if(!unit_ptr->frozen)
            unit_ptr->bias += unit_ptr->value_a * eta;

      if (flags & UFLAG_SITES)
            {
            FOR_ALL_SITES_AND_LINKS( unit_ptr, site_ptr,
            link_ptr )

            if(!link_ptr->frozen)
                  link_ptr->weight += link_ptr->value_a * eta;
            }
      else
      {
            if (flags & UFLAG_DLINKS)
            {
                  FOR_ALL_LINKS( unit_ptr, link_ptr )

                  if(!link_ptr->frozen)
                  link_ptr->weight += link_ptr->value_a * eta;
            }
      }
}
```

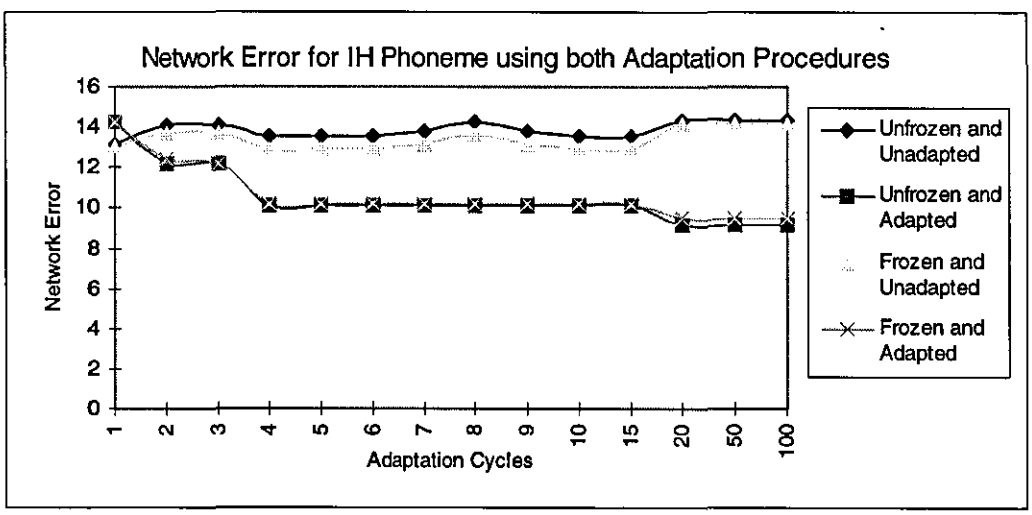
Figure B.5 Changes to the file 'Learn_f.c'

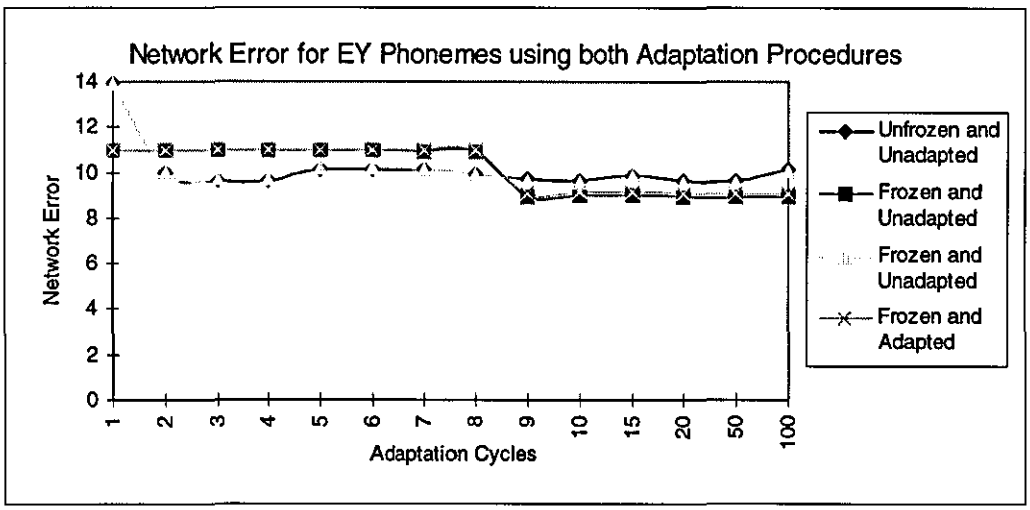
APPENDIX C

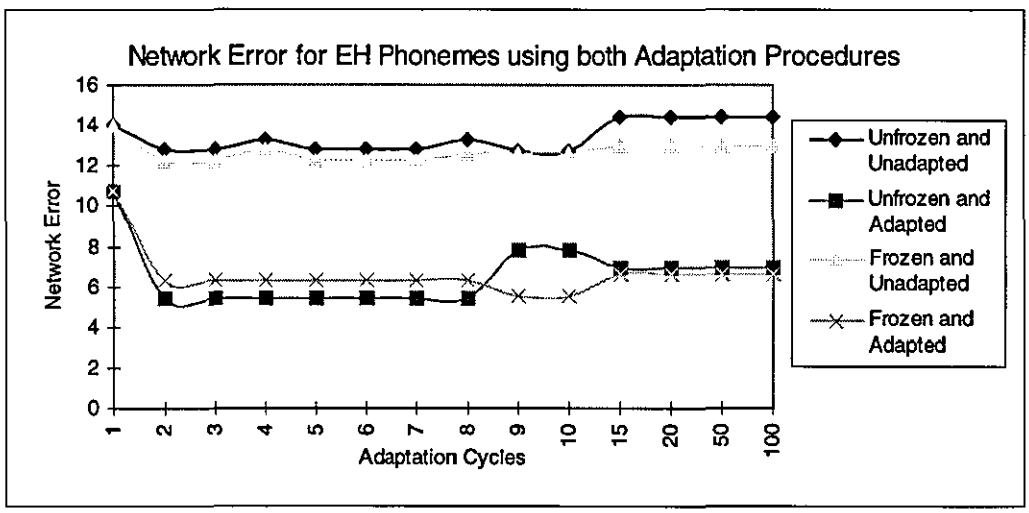
Chapter 6 Results

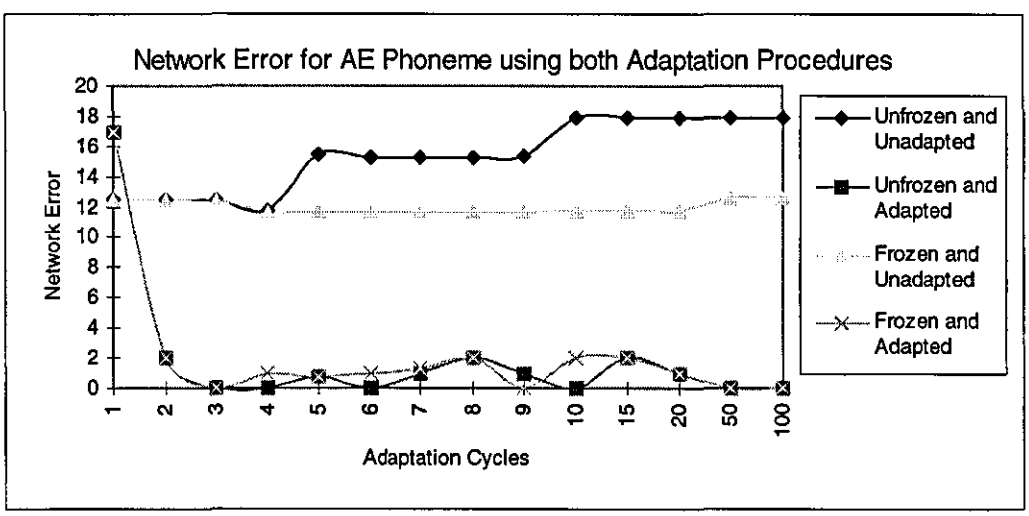
Appendix C contains 12 graphs showing the effects of the two adaptation procedures, frozen and unfrozen, on each of the vowel phonemes and their average effect on the remaining unadapted vowel phonemes from the same tongue-hump group.

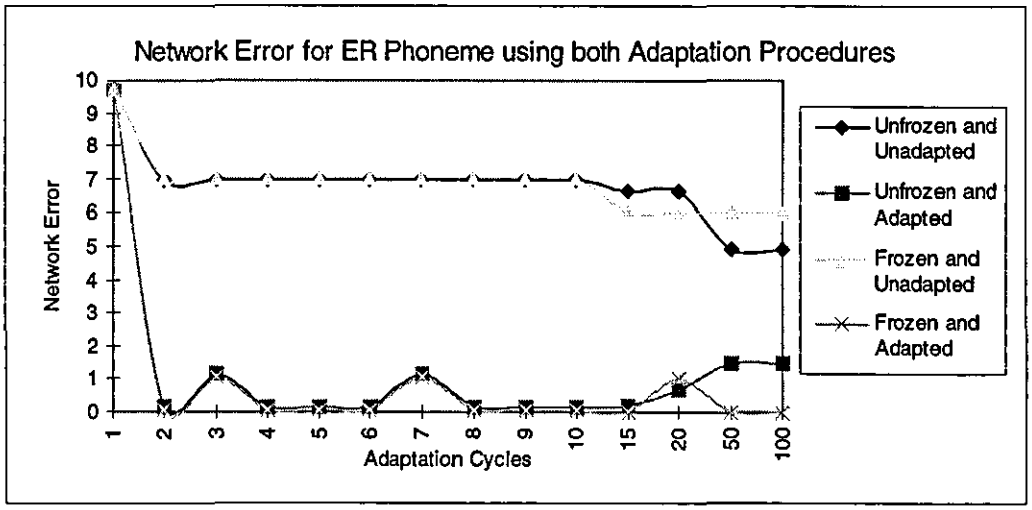


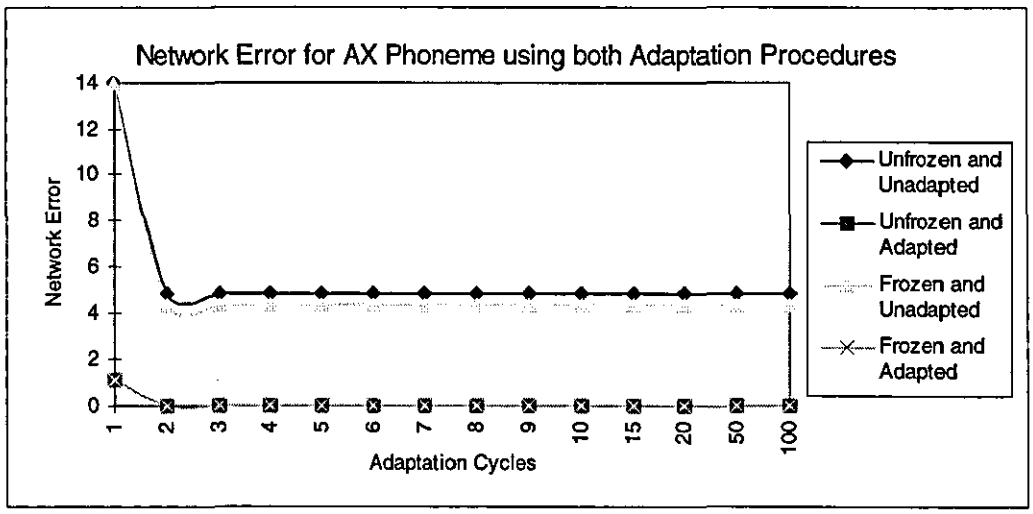


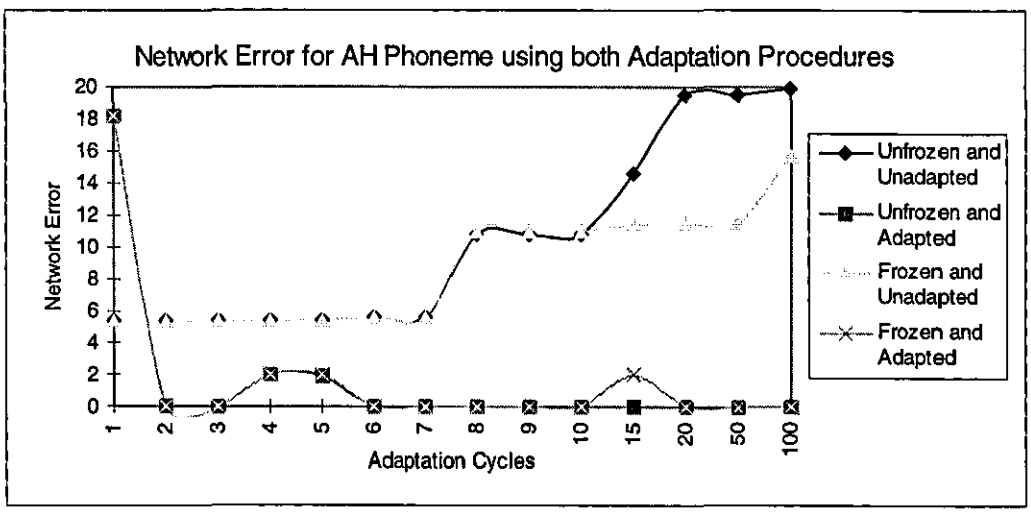

Network Error for EY Phonemes using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Frozen and Unadapted
Frozen and Unadapted
Frozen and Adapted

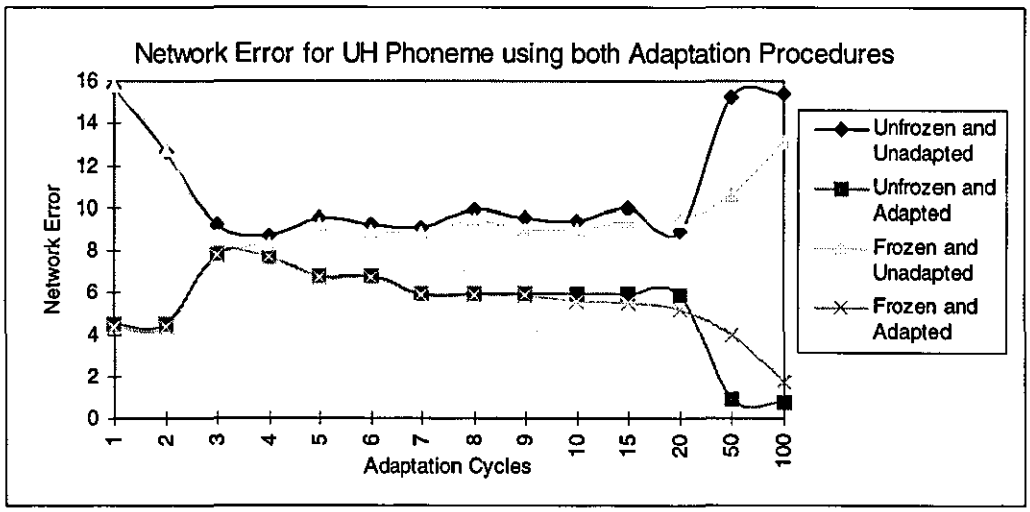

Network Error for EH Phonemes using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted

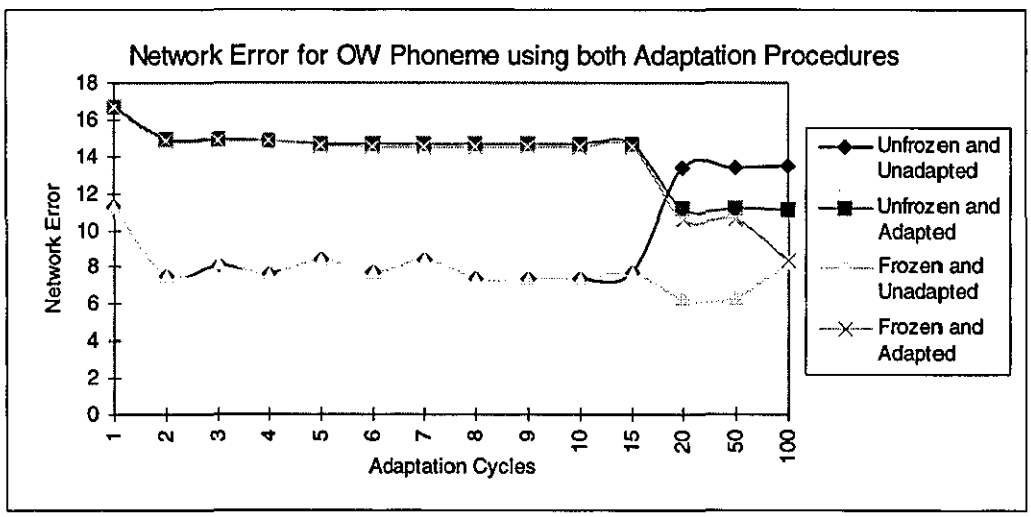

Network Error for AE Phoneme using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted

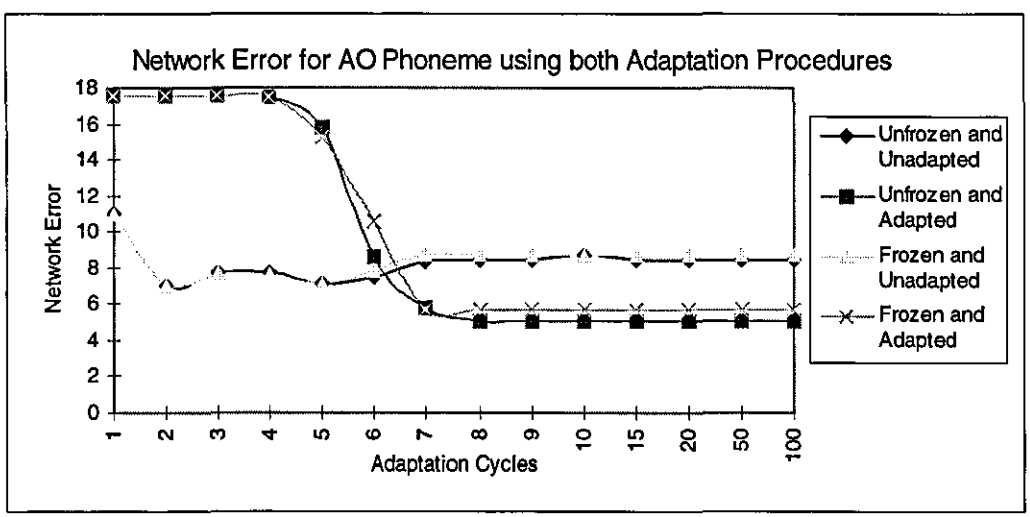

Network Error for ER Phoneme using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted

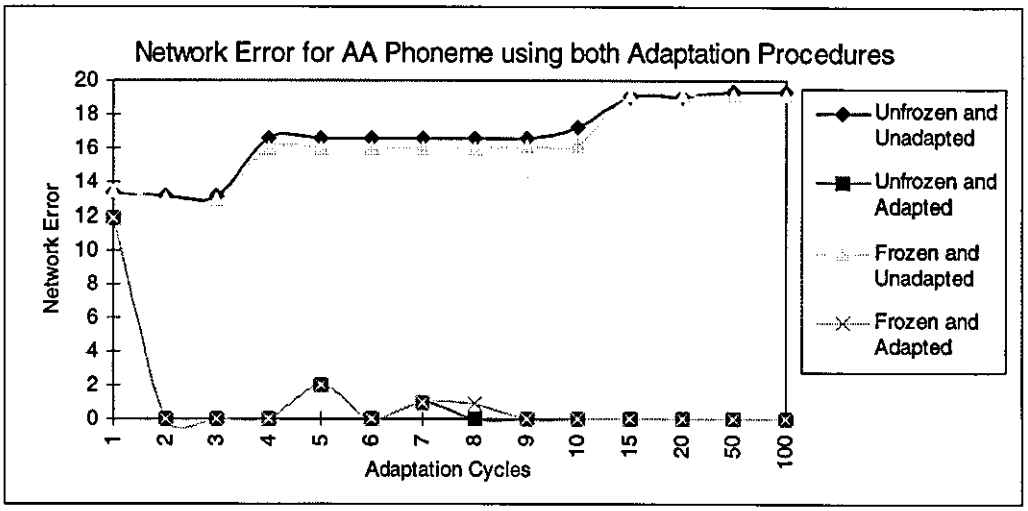

Network Error for AX Phoneme using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted


Network Error for AH Phoneme using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted


Network Error for UH Phoneme using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted


Network Error for OW Phoneme using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted


Network Error for AO Phoneme using both Adaptation Procedures
Network Error
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted


Network Error for AA Phoneme using both Adaptation Procedures
Network Error
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
15
20
50
100
Adaptation Cycles
Unfrozen and Unadapted
Unfrozen and Adapted
Frozen and Unadapted
Frozen and Adapted

APPENDIX D

Chapter 7 Test Set Selection

Trained Speakers

Tested Speakers	MCHH0	MDLF0	MDVC0	MERS0	MGRT0	MKDR0	MKJL0	MNJM0	MNLS0	MPAB0	MRCS0	MRJM4	MRMS1	MRPC0	MTWH0
MCHH0	98.8	41.9	22.1	24.4	29.0	36.0	31.4	32.5	31.4	20.9	29.0	13.9	49.3	41.9	50.0
MDLF0	43.5	100	32.2	20.9	21.2	37.1	17.7	37.1	21.2	29.0	29.0	33.9	46.8	43.5	48.4
MDVC0	41.1	30.3	98.2	25.0	33.9	23.2	8.3	19.6	41.1	39.3	32.1	23.2	14.3	12.5	18.5
MERS0	35.1	48.1	28.2	92.6	11.1	18.5	14.8	25.9	33.3	22.2	35.1	18.5	35.1	51.8	28.6
MGRT0	29.5	33.3	19.2	6.4	92.1	41.0	64.1	16.6	23.1	47.2	41.0	37.2	46.1	44.9	50.6
MKDR0	13.9	23.7	21.3	21.1	11.1	100	19.4	39.4	39.5	11.1	41.6	34.2	23.7	36.8	21.1
MKJL0	36.1	33.3	11.1	22.2	13.8	30.5	95.8	50.0	50.0	3.8	44.4	33.3	22.2	13.8	54.2
MNJM0	8.3	12.5	24.0	14.6	17.7	46.8	19.8	98.9	43.7	27.1	36.4	34.4	34.4	45.9	22.9
MNLS0	25.0	8.3	37.5	37.5	37.2	29.2	16.6	12.5	100	12.5	20.8	16.6	29.2	45.8	25
MPAB0	16.6	7.6	11.1	38.5	16.6	26.9	5.2	20.5	25.6	100	25.6	11.5	31.0	41.0	6.7
MRCS0	18.5	29.6	46.3	27.8	24.1	9.2	42.6	24.1	48.1	24.1	94.4	18.5	11.1	22.2	16.6
MRJM4	16.6	19.4	36.1	26.4	38.8	34.7	25.8	19.4	23.6	25.0	50.0	94.4	16.6	30.5	18.1
MRMS1	20.7	31.0	43.1	27.5	39.6	22.4	16.6	22.4	29.3	33.7	37.9	10.3	100	44.8	30.8
MRPC0	17.5	20.6	39.7	20.6	27.0	36.5	11.1	22.2	36.5	36.5	20.6	25.3	12.7	98.4	47.6
MTWH0	22.5	12.3	7.8	15.7	20.5	19.1	53.6	19.1	43.8	16.7	34.8	8.9	44.9	22.5	100

Figure D.1 Test Set Selection Results showing inter-speaker Recognition Rates for the 15 Speakers from the TIMIT Dialect Region 7 Test Set.

APPENDIX E

Published Papers

Appendix E contains the four following publications :

S.J.Haskey and S. Datta, "Selective Adaptation of Speaker Characteristics within a Subcluster Neural Network," Proc of the Seoul International Conference on Phonetic Sciences, Seoul, Korea, October 1996, pp464-467.

S.J.Haskey and S. Datta, "Using Tongue-Hump-Position Information for Vowel Adaptation within a Subcluster Neural Network," Proc of IEE Colloquium on Pattern Recognition, London, 26 Feb 1997, pp 9/1 – 9/6.

S.J.Haskey and S. Datta, "Dynamic Speaker Adaptation for Acoustically Similar Vowel Sounds using Sub-Cluster Neural Network," Convergence and Workshop on New Ideas in Computing, Part 2, Coventry University, May 1997, pp41-44.

S.J.Haskey and S.Datta, "A Comparative Study of OCON and MLP Architectures for Phoneme Recognition," ICSLP'98.

Selective Adaptation of Speaker Characteristics within a Subcluster Neural Network

S.J.Haskey and S.Datta

Electronic and Electrical Engineering Department
Loughborough University, Loughborough, LE11 3TU, U.K.
S.Datta@lboro.ac.uk

ABSTRACT

This paper aims to exploit inter/intra-speaker phoneme sub-class variations as criteria for adaptation in a phoneme recognition system based on a novel neural network architecture.
Using a subcluster neural network design based on the One-Class-in-One-Network (OCON) feed forward subnets, similar to those proposed by Kung [2]and Jou[1], joined by a common front-end layer, the idea is to adapt only the neurons within the common front-end layer of the network. Consequently resulting in an adaptation which can be concentrated primarily on the speakers vocal characteristics. Since the adaptation occurs in an area common to all classes, convergence on a single class will improve the recognition of the remaining classes in the network.
Results show that adaptation towards a phoneme, in the vowel sub-class, for speakers *MDAB0* and *MWBT0* improve the recognition of remaining vowel sub-class phonemes from the same speaker.

INTRODUCTION

Inter/intra speaker variations can cause significant problems with speaker independent recognition systems. Variations such as vocal tract length and dialect differences from speaker to speaker or the intonation, rhythm or stress variations from the same speaker. To over come this problem it is necessary to have a recognition system, that has been trained with utterances from a representative subset of speakers, to dynamically adapt after an initial correct utterance, latching onto the new speakers vocal characteristics.
Adaptation of conventional connectionist architectures generally involves network-wide weight changes. This is undesirable for the purposes of phoneme recognition due in part to computational inefficiency, but mainly due to the fact that the network will be susceptible to cross-class interference.
The main objectives of the new neural network architecture were to avoid cross-class interference during adaptation towards a phoneme class and to separate the phoneme information from the speaker information within the network. Separation of speaker and phoneme information would allow adaptation to be concentrated purely on speaker variations, reducing the need for network-wide adaptation. An assumption however is made that for similar dialects, inter speaker phoneme sub-class variations are roughly constant, i.e. vowel sound differences from speaker to speaker are consistent for all vowel sounds.

NETWORK ARCHITECTURE

Unlike conventional subnet structures (OCONs)[1][2] , fig 1, this new neural network architecture consists of OCONs, one for each phoneme class, joined by a common front-end adaptation layer, fig 2. Each OCON structure consists of a fully connected two layered network with a single output neuron. The adaptation layer fully connects to each of the OCON structures and in turn fully connects to the input layer. All the neurons within the network use the sigmoidal activation function and the weights of each connection are trained using the back-propagation algorithm [5].
After the network is initially trained with speech data it is assumed that all class specific information unique to that phoneme is stored in the relevant OCON subnet and that information common to all the classes, such as speaker information, is stored within the weights of the common front-end adaptation layer. When the network is introduced with speech data from a new speaker the score at the output is computed in much the same way as a conventional network. All the OCON outputs connect to a MAXNET[1] which finds the highest score, as long as it exceeds a minimum threshold level, which is assumed to be the correct utterance. Using back-propagation, the error is then fed back through the OCON structure to the front-end adaptation layer where the weights are adapted to minimise the error. As each new speaker uses the system the updated adaptation weights are reset to their initial values ready for adaptation towards the next speaker.
By concentrating the adaptation only on this front layer it is expected that only information unique to the speaker will change, resulting in a more efficiently controlled application-driven (speech recognition) connectionist regime. (Since the adaptation occurs in

By concentrating the adaptation only on this front layer it is expected that only information unique to the speaker will change, resulting in a more efficiently controlled application-driven (speech recognition) connectionist regime. (Since the adaptation occurs in an area common to all classes, it is envisaged that convergence on a single class will improve the recognition of the remaining classes in the network, for the same speaker, by eliminating the need to update each class for full adaptation to take place.)

ADAPTATION PROCEDURE

Forward Pass

When confronted with an utterance from a new speaker the output score is computed in much the same way as any conventional neural network.

Define:

I : Network Input.

A_j: Output of the j-th adaptation neuron.

ω_{ij}: the weights from the i-th input neuron to the j-th adaptation neuron.

$\bar{\omega}_{jk}$: the weights from the j-th adaptation neuron to the k-th hidden neuron in the subnet m.

$\hat{\omega}_k^{[m]}$: the weights from the k-th hidden neuron to the output neuron in the subnet m.

θ_j: the bias of the j-th adaptation neuron.

$\bar{\theta}_k^{[m]}$: the bias of the k-th hidden neuron in the subnet m.

$\hat{\theta}^{[m]}$: the bias of the output neuron in the subnet m.

$H_k^{[m]}$: Output of the k-th hidden neuron in the subnet m.

$O^{[m]}$: Output from the subnet m.

η : Learning rate of the adaptation layer.

Each neuron uses the sigmoidal activation function. Therefore the output of the j-th adaptation neuron is :

$$A_j = f\left(\sum_i \omega_{ij} \cdot I_i\right)$$

$$= 1/\left[1+\exp-\left(\sum_i \omega_{ij} \cdot I_i + \theta_j\right)\right]$$

Using the values of A_j the output of the k-th hidden neuron of the m-th subnet is calculated according to :

$$H_k^{[m]} = f\left(\sum_j \bar{\omega}_{jk}^{[m]} \cdot A_j\right)$$

$$= 1/\left[1+\exp-\left(\sum_j \bar{\omega}_{jk}^{[m]} \cdot A_j + \bar{\theta}_k^{[m]}\right)\right]$$

Finally, using the output of the hidden layer, $H_k^{[m]}$, from each corresponding subnet the output of the m-th subnet is :

$$O^{[m]}(I) = f\left(\sum_k \hat{\omega}_k^{[m]} \cdot H_k^{[m]}\right)$$

$$= 1/\left[1+\exp-\left(\sum_k \hat{\omega}_k^{[m]} \cdot H_k^{[m]} + \hat{\theta}^{[m]}\right)\right]$$

The outputs from each OCON subnet are fed through a MAXNET to find the winner, assuming the highest score achieves a minimum threshold score.

Backward Pass

Now we begin the back pass of the back-propagation algorithm to adapt the weights and bias values of the adaptation layer. Firstly we need the error E of each of the m subnets to feed-back. The error is :

$$\mathrm{E}^{[m]}(I) = \left(\mathrm{T} - O^{[m]}(I)\right)$$

where T is the target values.

If the input pattern I belongs to the m-th subnet then the target T is 1. Otherwise T is 0.

For the sigmoidal activation function, the error signal $\hat{\delta}^{[m]}$, for the output of the hidden layer is given by :

$$\hat{}^{[m]} = \mathrm{E}^{[m]}(I) \cdot O^{[m]}(I) \cdot \left(1 - O^{[m]}(I)\right)$$

Feed-back this error through now to the hidden neuron:

$$\bar{}_k^{[m]} = H_k^{[m]}\left(1 - H_k^{[m]}\right) \cdot \sum_k \hat{\delta}^{[m]} \hat{\omega}_k^{[m]}$$

Feed-back this error through to the adaptation layer, adding the errors from all the OCON subnets.

$$\delta_j = \sum_m \left\{ A_j\left(1 - A_j\right) \sum_j \bar{\delta}_k^{[m]} \bar{\omega}_{jk}^{[m]} \right\}$$

Now we have fed back the errors through the whole network we can modify the adaptation weights and bias values using the following:

$$\Delta\omega_{ij} = \eta \cdot \delta_j \cdot I$$

$$\Delta\theta_{ij} = \eta \cdot \delta_j$$

As the adaptation weights and bias values are modified, the old weight and bias values are stored so that the adaptation layer can be reset for each new user.

RESULTS

The main objective was to monitor the improved recognition rates of every phoneme class within the neural network after adaptation towards a single phoneme class from the same speaker.

Since we made the assumption concerning inter speaker phoneme sub-class variations remaining roughly constant, all the training data was from one phoneme sub-class, the vowel sub-class, of the DARPA TIMIT database. From this sub-class, 8 phonemes */ix, iy, eh, ah, ax, ih, ey, aa/* from 24 male speakers from dialect region one were used to train the network. The back-propagation algorithm was used for training, with all the weights within the network initially randomised, along with the order of the speech training data, to maximise convergence

The network consisted of 8 OCON subnet structures, one for each of the phoneme classes, all having a single output and containing a 15 neuron fully connected hidden layer. The 8 hidden layers from each of the OCONs were fully connected to the 15 neuron adaptation layer which in turn was fully connected to the 75 neuron input layer. The input data comprised of the sampled phonemes being split into 15 overlapping hamming windows, each of which was represented by 5 linear predictive coefficients [6].

The test set for the experiment contained utterances of the 8 selected vowel sub-class phonemes spoken by 2 male speakers (*DAB0, WBT0*) from the same dialect region as the training set. Initial recognition rates were noted for all the 8 phonemes from both speakers before adaptation began.

The adaptation procedure involved adapting the network towards a phoneme by feeding back any errors through the network and using these to modify the weights and bias values within the adaptation layer. After adaptation, recognition rates of the phonemes uttered by the same speaker were recorded and any variation calculated. The Recognition rates for speaker DAB0 incresed by a average 16.5% for adaptatio towards the same phonemes and an increase of 8.3% in the recognition of unadapted phonemes.saw a rise of 16.5% and 20.2% for DAB0 and WBTO respectiverly with a rise See Table 1 and 2 for speakers *MDAB0* and *MWBT0* respectively.

After each test, the adaptation weights and bias values were reset.

CONCLUSION

It can be seen from Table 1 and Table 2 that adaptation towards a phoneme, in the vowel sub-class, for speakers *MDAB0* and *MWBT0* can indeed improve the recognition of the remaining phonemes from the same speaker. After adaptation towards a phoneme the average recognition rate of that phoneme increases by 18.35% and the recognition rate of the remaining phonemes increases by 8.9%. This highlights the idea of speaker information being stored in the common front end adaptation layer, resulting in a more efficient adaptation system. Further tests need to be applied to other phoneme sub-classes such as stops and fricatives and at present, adaptation itself is still slow. This is because only simple back propagation is being used, although faster existing adaptation techniques can be applied to the same architecture

REFERENCES

(1) I. C. Jou, Y. J. Tsay, S. C. Tsay, Q. Z. Wu, and S.S. Yu. **Parallel distributed processing with multiple one-output back-propagation neural networks.** Proceedings, International Symposium on Circuits and Systems, Singapore, pp 1408-11, 1991.

(2) S. Y. Kung, J. S. Taur **Decision-Based Neural Networks with Signal/Image Classification Applications.** IEEE Transactions on Neural Networks, Vol 6, No 1, pp170-81, 1995.

(3) R. P. Lippmann **An Introduction to Computing with Neural Nets.** IEEE ASSP Magazine, April 1987, pp 4-22.

(4) R.P.Lippmann **Review of Neural Networks for Speech Recognition.** Neural Computation 1, pp 1-38, 1989.

(5) D.Rumelhart, J.McClelland **Parallel Distributed Processing.** Cambridge, MIT Press, 1986.

(6) J.Makhoul **Linear Prediction: a tutorial review**, Proc IEEE, Vol 63, No 4, pp 561-580, April, 1975.

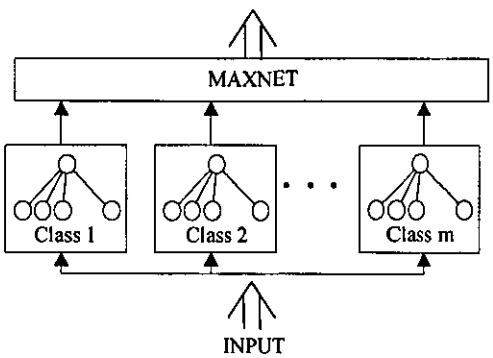


Fig. 1: Conventional OCON Architecture

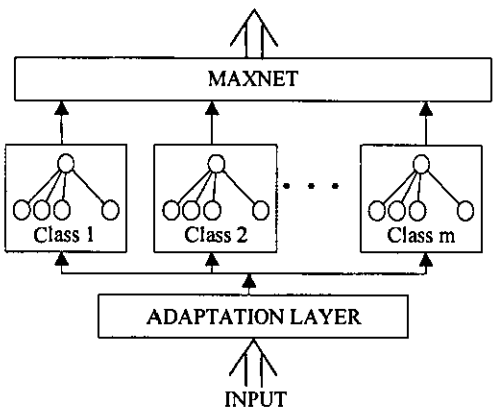


Fig 2: OCON Architecture with Common Front-End Adaptation Layer.

Adapted Phonemes

Tested Phonemes	ix	iy	eh	ah	ax	ih	ey	aa
ix	+13.3%	+13.3%	+13.3%	0	0	+13.3%	+6.6%	0
iy	+7.1%	+21.4%	+7.1%	+7.1%	0	+21.4%	+7.1%	+7.1%
eh	0	+11.1%	+11.1%	0	0	+11.1%	+11.1%	0
ah	+14.3%	+14.3%	0	+28.6%	+14.3%	+28.3%	+14.3%	0
ax	+10%	+10%	0	+10%	+20%	+20%	+10%	+10%
ih	+12.5%	+12.5%	0	0	+12.5%	+12.5%	+12.5%	0
ey	+25%	0	+25%	+25%	+25%	+25%	+25%	0
aa	0	0	0	0	0	0	0	0

Table 1: Recognition Results Using Speaker *DAB0*

Adapted Phonemes

Tested Phonemes	ix	iy	eh	ah	ax	ih	ey	aa
ix	+18.2%	+18.2%	+18.2%	+9.1%	0	+18.2%	+9.1%	0
iy	+12.5%	+37.5%	+12.5%	+12.5%	0	+37.5%	+12.5%	+12.5%
eh	0	+10%	+10%	0	0	+10%	+10%	0
ah	+9.1%	+9.1%	0	+18.2%	+9.1%	+18.2%	+9.1%	0
ax	+12.5%	+12.5%	0	+12.5%	+25%	+25%	+12.5%	+12.5%
ih	+11.1%	+11.1%	0	0	+11.1%	+11.1%	+11.1%	0
ey	+16.7%	0	+16.7%	+16.7%	+16.7%	+16.7%	+16.7%	0
aa	+25%	+25%	+25%	0	0	0	0	+25%

Table 2: Recognition Results Using Speaker *WBT0*

DYNAMIC SPEAKER ADAPTATION FOR ACOUSTICALLY SIMILAR VOWEL SOUNDS USING SUB-CLUSTER NEURAL NETWORKS

S.J.Haskey and S.Datta
Loughborough University

Abstract

In this paper we present an adaptation technique which exploits the inter/intra speaker phoneme variations of acoustically similar vowel sounds.
The 13 vowels of American English speech can be classified into three acoustically similar areas according to the relevant tongue-hump-position. The vowels, taken from the DARPA TIMIT phonetic database [1], in each of these areas are classified using One-Class-in-One-Network (OCON) feed forward subnets, similar to those proposed by Kung[3] and Jou[2], joined by a common front-end adaptation layer [4][7]. This allows adaptation to be concentrated primarily on speaker characteristics, since speaker information is comparable within these areas, allowing adaptation towards a single phoneme to improve recognition of other vowel phonemes within the same network. This reduces the need for total vowel recital for complete vowel phoneme adaptation towards a new speaker.
Results show increases of over 12% in the recognition rates of vowel phonemes after adaptation towards other phonemes in the same tongue-hump-position area. However, vowels that are well separated in the same group have little, even negative, effect on recognition after adaptation.

Introduction

If a speaker could consistently and precisely produce the English American phonemes, speech would amount to a flow of discrete sounds. However, due to inter/intra speaker variations such as vocal tract length, dialect differences, rhythm, intonation, stress and most importantly co-articulation effects, a given 'phoneme' will have a variety of acoustic manifestations in the course of continuous speech. This can cause significant problems with speaker independent recognition systems and so some form of dynamic adaptation is necessary to achieve a speaker transparent recognition system. This paper will concentrate on acoustically similar vowel sounds and in particular the inter/intra speaker similarities corresponding to tongue-hump-position [5]. The tongue-hump-position and the degree of which the tongue causes a constriction in the oral cavity creates variations in cross-sectional area along the vocal tract which determines the formants of the vowel. The position of the hump of the tongue (front, central, back) divides the vowel phonemes into three main groups. An assumption is made that the vowel phonemes within each group, due to the constant tongue-hump-position, all contain comparable speaker information. Using OCON subnet structures [2][3] with a front-end adaptation layer [4][7] for each group, speaker and phoneme information can be separated. Therefore, common speaker information from the vowel phonemes can be isolated in the front-end layer. This then allows adaptation towards a phoneme class to improve the recognition rate of other uttered phonemes sharing the same adaptation layer from the same speaker. The resultant system not only abolishes the need for total network adaptation but also reduces the need for every vowel to be recited for total vowel phoneme adaptation.

Acoustically Similar Vowel Sounds

The 13 vowel sounds of American English, although produced solely by vocal cord movement, vary dramatically with cross-sectional area along the vocal tract. This cross-sectional area, particularly in the oral cavity can be altered by movement of the articulators, mainly the tongue. Consequently the tongue position plays a fundamental part in the production of the resonant frequencies (formants) in the vocal tract that make up the vowel sounds. The tongue varies the formants in the vocal tract in two ways. By the tongue-hump-position and by the degree-of-constriction the tongue hump causes,
Fig 1.

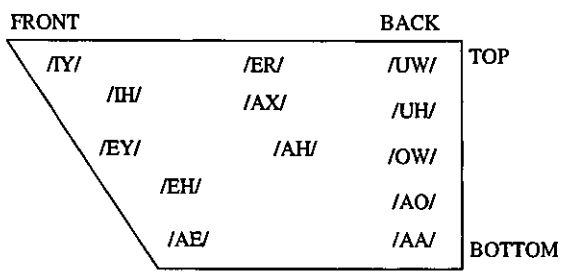


Fig 1: The Position of the Hump of the Tongue in the Oral Cavity during the Production of the American English Vowels.

As is shown in Fig 2 the three vowels /IY, AA, UW/ represent the extreme frequency locations for F_1 and F_2. It can be seen from Fig 2 that moving from /IY/ to /AE/, /ER/ to /AH/ and from /UW/ to /AA/ the first formant, F_1, increases as the tongue constriction

F_2. It can be seen from Fig 2 that moving from /IY/ to /AE/, /ER/ to /AH/ and from /UW/ to /AA/ the first formant, F_1, increases as the tongue constriction increases whereas moving from /UW/ to /IY/ and /AA/ to /AE/ the second formant, F_2, alters with the tongue-hump-position. As the tongue moves towards the front of the oral cavity so F_2 increases [6].

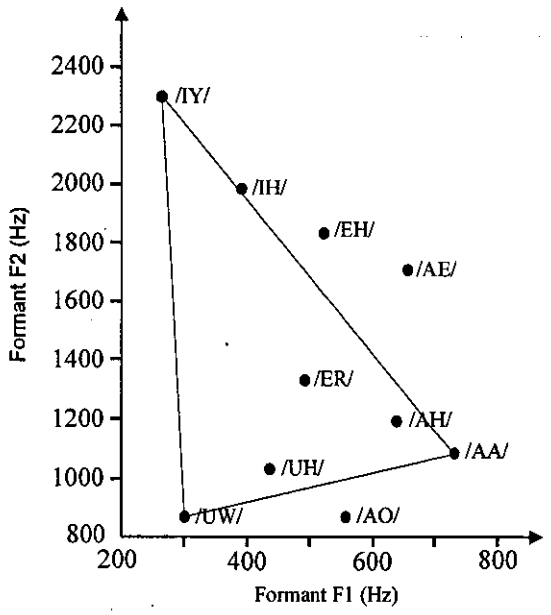


Fig 2: The Vowel Triangle. A plot of Average Formants, F_1 and F_2, for American English Vowels.

These fundemental frequency F_0 and the first formants F_1 are responsible for creating the raw phoneme sound. This allows the phoneme to be recognised but contains little speaker information. The majority of the speaker information within a phoneme comes from the second and third formants F_2 and F_3.

To take advantage of the network architecture we need to cluster vowel phonemes into groups that contain comparable speaker information so that this information can be stored in the common front-end adaptation layer. Since F_2 does contain some speaker characteristics and varies with respect to the tongue-hump-position in the oral cavity the vowels can be segregated into three groups:- front, middle and back.

Network Architecture

Unlike conventional subnet structures, this neural network architecture consists of (OCONs)[2][3]. Each vowel phoneme class has it's own OCON, with the OCONs representing phonemes from the same tongue-hump-position group, joined using a common front-end adaptation layer, Fig 3. Each OCON structure consists of a hidden layer fully connected to a single output neuron. The adaptation layer fully connects to each of the OCON structures and in turn fully connects to the input layer. All the neurons within the feed forward network use the sigmoidal activation function and the weights of each connection are trained using the back-propagation algorithm.

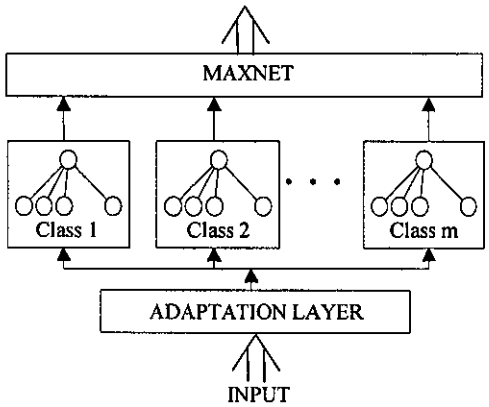


Fig 3: OCON Architecture with Common Front-End Adaptation Layer.

After the network is initially trained with speech data it is assumed that all class specific information unique to that phoneme is stored in the relevant OCON subnet and that information common to all the classes, such as speaker information (F_2 & F_3), is stored within the weights of the common front-end adaptation layer. When the network is introduced with speech data from a new speaker the error is fed back through the OCON structure, using back-propagation, to the front-end adaptation layer where only the weights in this layer are altered to minimise the error. As each new speaker uses the system the updated adaptation weights are reset to their initial post-trained values ready for adaptation towards the next speaker.

Since the adaptation occurs in an area common to all classes within the network, it is envisaged that convergence on a single class will improve the recognition of the remaining classes, for the same speaker, eliminating the need to update each class for full adaptation to take place. Therefore by concentrating the adaptation only on this front layer, only information unique to the speaker within the same tongue-hump-position group will change, resulting in a more efficiently speech recognition system.

Results

The main objectives were to monitor the improved recognition rates of vowel phonemes after adaptation towards a single vowel phoneme within the same tongue-hump-position group uttered by the same speaker.

Since we made the earlier assumption that vowel phonemes within each group all contain comparable speaker information, training and test data from the DARPA TIMIT database [1] was

split into the three tongue-hump-position groups. Front /IY, IH, EY, EH, AE/, middle /ER, AX, AH/ and back /UW, UH, OW, AO, AA/. Each of the three networks consisted of one OCON subnet structures for each of the phoneme classes, all having a single output and containing a 3 neuron fully connected hidden layer. The hidden layers from each of the OCONs were fully connected to the 10 neuron adaptation layer which in turn was fully connected to the 56 neuron input layer.

Training data was concentrated on one dialect region only, the Western dialect region. This was so that testing and adapting with another dialect region would accentuate any speaker differences, primarily dialect differences. Therefore highlighting the effect, if any, of adaptation towards a vowel phoneme influencing the recognition of other phonemes within the same network. All SX and SI sentences from the 79 male speakers of the Western dialect region were used as training data. The relevant vowel phonemes from each sentence were pre-emphasised and then split into eight windowed segments, with each window represented by 7^{th} order linear prediction coefficients. The back-propagation algorithm was used for training, with all the weights within the network initially randomised, along with the order of the speech training data, to maximise convergence.

The test data contained utterances from all the male speakers from the dialect region three, Northern Midland. All the data was pre-processed identically to the training data and the recognition rates for each vowel phoneme from each speaker noted. Then, one speaker at a time, the network was adapted towards a vowel phoneme and the changes in recognition performance of the other remaining vowel phonemes in the same network monitored. Table 1, 2 and 3 show the average change in recognition performance, from 23 male speakers, after adaptation towards other vowel phonemes in the same group. Table 1, corresponding to the front of the oral cavity, shows an average increase of 4.4%, table 2, the middle, shows an average increase of 2.8% and for table 3, the back, we have an average increase of 4.6%.

Conclusion

The exhibited improvements in recognition seem to correlate to the distance measure between the tongue positions of the relevant tested and adapted vowels. The closer the vowels in the oral cavity the larger the recognition improvement. Although these results look promising there are some negative changes. This is probably caused by large speaker variations between well separated vowels. To eliminate this problem the existing groups may have to be further split to reduce the maximum distance between adapted and tested vowels. However, further research is required into F_2 and F_3 information and its distribution within the oral tract since both these formants contain the majority of speaker information. This additional formant data will undoubtedly influence the perimeters of further vowel grouping.

References

(1) DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM Oct, 1990

(2) I. C. Jou, Y. J. Tsay, S. C. Tsay, Q. Z. Wu, and S.S. Yu: "Parallel distributed processing with_multiple one-output back-propagation neural networks", Proceedings, International Symposium on Circuits and Systems, Singapore, pp 1408-11, 1991

(3) S. Y. Kung, J. S. Taur : "Decision-Based Neural Networks with Signal/Image Classification Applications" IEEE Transactions on Neural Networks, Vol 6, No 1, pp170-81, 1995

(4) S.J.Haskey, S.Datta: "Selective Adaptation of Speaker Characteristics within a Subcluster Neural Network", SICOPS, pp 464-467, 1996

(5) J.R.Deller et al: "Discrete-Time Processing of Speech Signals" Maxwell Macmillan International, 1993

(6) G.E.Peterson and H.L.Barney: "Control Methods Used in a Study of the Vowels", J Acoust Soc Am, Vol 24, pp 175-184, March 1952.

(7) S.J.Haskey, S.Datta: "Using Tongue-Hump-Position Information for Vowel Adaptation within a Subcluster Neural Network", IEE Colloquium on "Pattern recognition", Feb 1997.

Vowel Phonemes Tested

Vowel Phonemes Adapted Towards	/IY/	/IH/	/EY/	/EH/	/AE/
/IY/	+16.83	+9.16	+3.82	-2.53	-1.18
/IH/	+4.34	+18.54	+6.94	0	+1.89
/EY/	+4.58	+11.20	+24.61	+7.17	+3.04
/EH/	0	+5.09	+8.28	+22.96	+9.36
/AE/	-6.67	-0.24	+11.39	+12.24	+31.53

Table 1: Average Changes (%) of 23 Male Speakers after Adaptation to a Single Phoneme from the front of the oral cavity.

Vowel Phonemes Tested

Vowel Phonemes Adapted Towards	/ER/	/AX/	/AE/
/ER/	+24.67	+8.08	+0.67
/AX/	+5.18	+28	0
/AE/	+1.09	+2.14	+18.33

Table 2: Average Changes (%) of 23 Male Speakers after Adaptation to a Single Phoneme from the middle of the oral cavity.

Vowel Phonemes Tested

Vowel Phonemes Adapted Towards	/UW/	/UH/	/OW/	/AO/	/AA/
/UW/	+31.25	+4	+2.07	+10.67	-1.18
/UH/	0	+20	+10	+12	-2.35
/OW/	+6.25	+8	+22.78	+4	+5.88
/AO/	+6.25	+4	+1.67	+30	+6.47
/AA/	0	0	+8.34	+5.33	+23.53

Table 3: Average Change (%) of 23 Male Speakers after Adaptation to a Single Phoneme from the back of the oral cavity.

Using Tongue-Hump-Position Information for Vowel Adaptation within a Subcluster Neural Network

S.J.Haskey and S.Datta

Electronic and Electrical Engineering Department
Loughborough University, Loughborough, LE11 3TU, U.K.
S.Datta@lboro.ac.uk

ABSTRACT

In this paper we present an adaptation technique which exploits the inter/intra speaker vowel phoneme variations with respect to the tongue-hump-position within the oral cavity.
The 13 vowels of American English speech can be classified into three areas according to the tongue-hump-position. The vowels, taken from the DARPA TIMIT phonetic database [1], in each of these areas are classified using One-Class-in-One-Network (OCON) feed forward subnets, similar to those proposed by Kung[3] and Jou[2], joined by a common front-end adaptation layer [4]. This allows adaptation to be concentrated primarily on speaker characteristics, since speaker information is comparable within these areas, allowing adaptation towards a single phoneme to improve recognition of other vowel phonemes within the same network. This reduces the need for total vowel recital for complete vowel phoneme adaptation towards a new speaker.
Results show increases of over 12% in the recognition rate of vowel phonemes after adaptation towards other phonemes in the same tongue-hump-position area. However, vowels that are well separated in the same group have little, even negative, effect on recognition after adaptation.

INTRODUCTION

If a speaker could consistently and precisely produce the English American phonemes, speech would amount to a flow of discrete sounds. Of course, due to inter/intra speaker variations such as vocal tract length, dialect differences, intonation, rhythm, stress and most importantly co-articulation effects, a given 'phoneme' will have a variety of acoustic manifestations in the course of continuous speech. Consequently this can cause significant problems with speaker independent recognition systems and so some form of dynamic adaptation is necessary to achieve a speaker transparent recognition system. This paper will concentrate on vowel phonemes and in particular the inter/intra speaker similarities corresponding to tongue-hump-position [5]. The tongue-hump-position and the degree of which the tongue causes a constriction in the oral cavity creates variations in cross-sectional area along the vocal tract which determines the formants of the vowel. The position of the hump of the tongue (front, central, back) divides the vowel phonemes into three main groups. An assumption is made that the vowel phonemes within each group, due to the constant tongue-hump-position, all contain comparable speaker information. Using OCON subnet structures [2][3] with a front-end adaptation layer [4] for each group, speaker and phoneme information can be separated. Therefore, common speaker information from the vowel phonemes can be isolated in the front-end layer. This then allows adaptation towards a phoneme class to improve the recognition rate of other phoneme sharing the same adaptation layer. The resultant system not only abolishes the need for total network adaptation but also reduces the need for every vowel to be recited for total vowel phoneme adaptation.

VOWEL FORMANT VARIATIONS

The 13 vowel sounds of American English, although produced solely by vocal cord

movement, vary dramatically with cross-sectional area along the vocal tract. This cross-sectional area, particularly in the oral cavity can be altered by movement of the articulators, mainly the tongue. Consequently the tongue position plays a fundamental part in the production of the resonant frequencies (formants) in the vocal tract that make up the vowel sounds. The tongue varies the formants in the vocal tract in two ways. By the tongue-hump-position and by the degree-of-constriction the tongue hump causes, fig 1. As is shown in fig 2 the three vowels /IY, AA, UW/ represent the extreme frequency locations for F_1 and F_2. It can be seen from Fig 2 that moving from /IY/ to /AE/, /ER/ to /AH/ and from /UW/ to /AA/ the first formant, F_1, increases as the tongue constriction increases whereas moving from /UW/ to /IY/ and /AA/ to /AE/ the second formant, F_2, alters with the tongue-hump-position. As the tongue moves towards the front of the oral cavity so F_2 increases [6]. These first two formants F_1 and F_2 are responsible for creating the raw phoneme sound. This allows the phoneme to be recognised but contains little speaker information. The majority of the speaker information within a phoneme comes from the third and fourth formants F_3 and F_4 however, some speaker information is retained in the second formant F_2.
To take advantage of the network architecture we need to cluster vowel phonemes into groups that contain comparable speaker information so that this information can be stored in the common front-end adaptation layer. Since F_2 does contain some speaker characteristics and varies with respect to the tongue-hump-position in the oral cavity the vowels can be segregated into three groups:- front, middle and back.

NETWORK ARCHITECTURE

Unlike conventional subnet structures, (OCONs)[2][3], this neural network architecture consists of one OCON for each vowel phoneme class, joined along with other classes of the same tongue-hump-position group using a common front-end adaptation layer, fig 3. Each OCON structure consists of a hidden layer fully connected to a single output neuron. The adaptation layer fully connects to each of the OCON structures and in turn fully connects to the input layer. All the neurons within the feed forward network use the sigmoidal activation function and the weights of each connection are trained using the back-propagation algorithm.
After the network is initially trained with speech data it is assumed that all class specific information unique to that phoneme is stored in the relevant OCON subnet and that information common to all the classes, such as speaker information (F_2, F_3 & F_4), is stored within the weights of the common front-end adaptation layer. When the network is introduced with speech data from a new speaker the error is fed back through the OCON structure, using back-propagation, to the front-end adaptation layer where only the weights in this layer are altered to minimise the error. As each new speaker uses the system the updated adaptation weights are reset to their initial values ready for adaptation towards the next speaker.
By concentrating the adaptation on this front layer only information unique to the speaker within the same tongue-hump-position group will change, resulting in a more efficiently controlled application driven (speech recognition) connectionist regime. Since the adaptation occurs in an area common to all classes within the network, it is envisaged that convergence on a single class will improve the recognition of the remaining classes, for the same speaker, eliminating the need to update each class for full adaptation to take place.

RESULTS

The main objectives were to monitor the improved recognition rates of vowel phonemes after adaptation towards a single vowel phoneme within the same tongue-hump-position group uttered by the same speaker.
Since we made the earlier assumption that vowel phonemes within each group all contain comparable speaker information, training and test data from the DARPA TIMIT database [1] was split into the three tongue-hump-position groups. Front /IY, IH, EY, EH, AE/, middle /ER, AX, AH/ and back /UW, UH, OW, AO, AA/. Each of the three networks consisted of one OCON subnet structures for each of the phoneme classes, all having a single output and containing a 3 neuron fully connected hidden layer. The hidden layers from each of the OCONs were fully connected to

the 10 neuron adaptation layer which in turn was fully connected to the 56 neuron input layer.
Training data was concentrated on one dialect region only, the Western dialect region. This was so that testing and adapting with another dialect region would accentuate any speaker differences, primarily dialect differences. Therefore highlighting the effect, if any, of adaptation towards a vowel phoneme influencing the recognition of other phonemes within the same network. All SX and SI sentences from the 79 male speakers of the Western dialect region were used as training data. The relevant vowel phonemes from each sentence were pre-emphasised and then split into eight windowed segments, with each window represented by 7^{th} order linear prediction coefficients. The back-propagation algorithm was used for training, with all the weights within the network initially randomised, along with the order of the speech training data, to maximise convergence.
The test data contained utterances from all the male speakers from the dialect region three, Northern Midland. All the data was pre-processed in a similar fashion to the training data and the recognition rates for each vowel phoneme from each speaker noted. Then, one speaker at a time, the network was adapted towards a vowel phoneme and the changes in recognition performance of the other remaining vowel phonemes in the same network monitored. Table 1, 2 and 3 show the average change in recognition performance, from 23 male speakers, after adaptation towards other vowel phonemes in the same group. Table 1, corresponding to the front of the oral cavity, shows an average increase of 4.4%, table 2, the middle, shows an average increase of 2.86% and for the table 3, the back, we have an average increase of 4.57%.

CONCLUSION

The exhibited improvements in recognition seem to correlate to the distance measure between the tongue positions of the relevant tested and adapted vowels. The closer the vowels in the oral cavity the larger the recognition improvements. Although these results look promising there are some negative changes. This is probably caused by large speaker variations between well separated vowels. To eliminate this problem the existing groups may have to be further split to reduce the maximum distance between adapted and tested vowels. However, further research is required into F3 and F$ information and its distribution within the oral tract since both there formants contain the majority of speaker information. This additional formant data will undoubtedly influence the perimeters of further vowel grouping.

REFERENCES

(1) DARPA TIMIT **Acoustic-Phonetic Continuous Speech Corpus CD-ROM** Oct, 1990.

(2) I. C. Jou, Y. J. Tsay, S. C. Tsay, Q. Z. Wu, and S.S. Yu. **Parallel distributed processing with multiple one-output back-propagation neural networks.** Proceedings, International Symposium on Circuits and Systems, Singapore, pp 1408-11, 1991.

(3) S. Y. Kung, J. S. Taur **Decision-Based Neural Networks with Signal/Image Classification Applications.** IEEE Transactions on Neural Networks, Vol 6, No 1, pp170-81, 1995.

(4) S.J.Haskey, S.Datta **Selective Adaptation of Speaker Characteristics within a Subcluster Neural Network,** SICOPS, pp 464-467, 1996

(5) J.R.Deller et al **Discrete-Time Processing of Speech Signals** Maxwell Macmillan International, 1993

(6) G.E.Peterson and H.L.Barney **Control Methods Used in a Study of the Vowels,** J Acoust Soc Am, Vol 24, pp 175-184, March 1952.

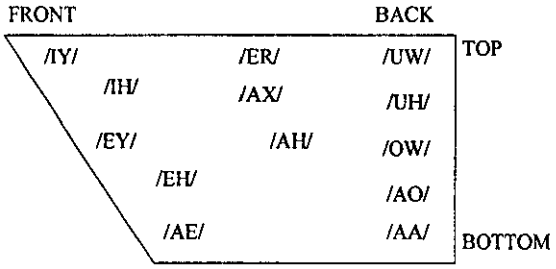


Fig 1: The Position of the Hump of the Tongue in the Oral Cavity during the Production of the American English Vowels.

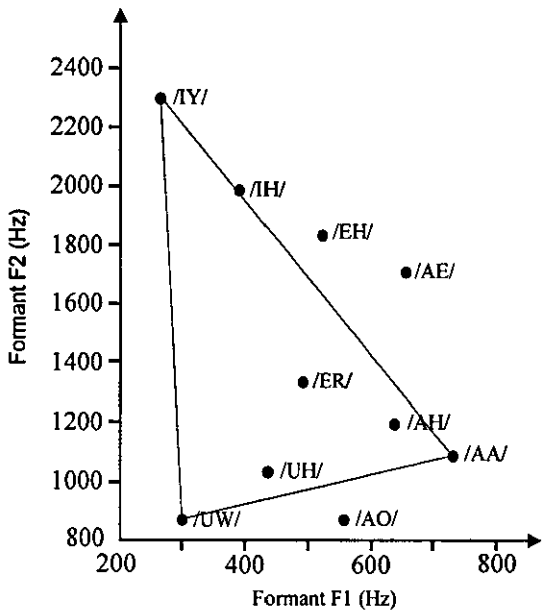


Fig 2: The Vowel Triangle. A plot of Average Formants, F_1 and F_2, for American English Vowels.

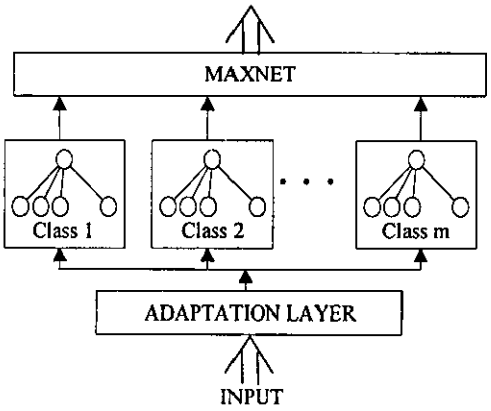


Fig 3: OCON Architecture with Common Font-End Adaptation Layer.

Vowel Phonemes Tested

Vowel Phonemes Adapted Towards	/IY/	/IH/	/EY/	/EH/	/AE/
/IY/	+16.83	+9.16	+3.82	-2.53	-1.18
/IH/	+4.34	+18.54	+6.94	0	+1.89
/EY/	+4.58	+11.20	+24.61	+7.17	+3.04
/EH/	0	+5.09	+8.28	+22.96	+9.39
/AE/	-6.67	-0.24	+11.39	+12.24	+31.53

Table 1: Average Scores of 23 Male Speakers after Adaptation to a Single Phoneme from the front of the oral cavity.

Vowel Phonemes Tested

Vowel Phonemes Adapted Towards	/ER/	/AX/	/AE/
/ER/	+24.67	+8.08	+0.67
/AX/	+5.18	+28	0
/AE/	+1.09	+2.14	+18.33

Table 2: Average Scores of 23 Male Speakers after Adaptation to a Single Phoneme from the middle of the oral cavity.

Vowel Phonemes Tested

Vowel Phonemes Adapted Towards	/UW/	/UH/	/OW/	/AO/	/AA/
/UW/	+31.25	+4	+2.07	+10.67	-1.18
/UH/	0	+20	+10	+12	-2.35
/OW/	+6.25	+8	+22.78	+4	+5.88
/AO/	+6.25	+4	+1.67	+30	+6.47
/AA/	0	0	+8.34	+5.33	+23.53

Table 3: Average Scores of 23 Male Speakers after Adaptation to a Single Phoneme from the back of the oral cavity.

A Comparative Study of OCON and MLP Architectures for Phoneme Recognition.

S.J.Haskey and S.Datta

Electronic and Electrical Engineering Department
Loughborough University, Loughborough, LE11 3TU, UK.

ABSTRACT

In this paper a comparative study between One-Class-One-Network (OCON) and Multi-Layered Perceptron (MLP) neural networks for vowel phoneme recognition is presented. The OCON architecture, first proposed by I.C.Jou et al 1991, is similar in design to a conventional feed-forward MLP, only each class had its own dedicated sub-network containing a single output node. Conventional MLPs usually consist of fully-connected nodes which not only result in a large number of weighted connections but also create the problem of cross-class interference. Using vowel phoneme data from the DARPA TIMIT corpus of read speech, MLP and OCON architectures were trained and the relative effects of recognition rates and convergence during both intra and inter-class adaptation tested. The OCON showed an increase in the convergence rate of 273% and an improvement of adapted recognition rates against the MLP of over 12%. However, due to the isolated nature of each OCON class, it was unable to utilise inter-class information. This resulted in a recognition rate reduction of over 6% for unadapted phonemes during adaptation, compared with the MLP results.

1. THE OCON

A large fully-connected network can potentially contain many hundreds of neurons, each connected via weights to many others. This can make the training and adapting of such a network a long and difficult task. In addition, fully connected networks are prone to cross-class interference. Cross-class interference occurs when adapting towards a single class in a multi-class network, inevitably altering shared weights. As the network gets larger the interference increases, drastically degrading the convergence rate of the shared weights due to the influence of conflicting signals. This can lead to, after adaptation towards a single class, the impaired classification for the remaining classes within the network. To eliminate these problems, I.C.Jou et al [2] proposed a new neural network architecture called the One-Net-One-Class. The same principle was later taken on by S.Y.Kung[3][4], who named the architecture the 'One-Class-One-Net' or the 'OCON' for short. The OCON is similar in design to that of a conventional MLP (see Figure 1a) only each class has its own dedicated subnet containing a single output neuron (see Figure 1b). Each OCON subnet is
specialised for distinguishing its own class from other patterns, resulting in fewer nodes being required in the hidden layers. I.C.Jou first used the OCON architecture in 1991 for optical character recognition (OCR). Later S.Y.Kung [4] also applied the OCON architecture to OCR, achieving a training accuracy of 99.5% compared with 94% from a conventional MLP. Such architectures have also been used for texture classification, Electrocardiograph (ECG) analysis and the classification of mandarin speech syllables and isolated English words with a hybrid Time Delay Neural Networks (TDNN) and OCON structure [5].

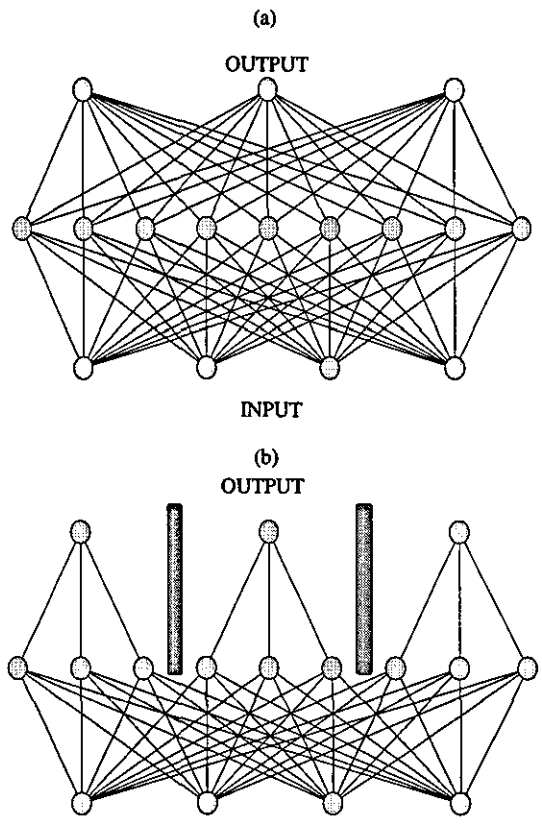


Figure 1: (a) A fully-connected MLP architecture.
(b) An OCON Neural Network Architecture

2. THE SPEECH DATA

All the speech data used during the comparative study was obtained from the DARPA TIMIT corpus of read speech [1]. 12 vowel phonemes spoken by male speakers from the TIMIT dialect region 7, the western geographical area of the U.S, were used for training and testing the ANN architectures. Vowel

phonemes were specifically chosen since they are the most spectrally well defined of all phonemes making them more easily and reliably recognised and ideal for a comparative study. For the study the phonemes were also from speakers with the same gender and dialect as large deviations between phonemes needed to be avoided during the comparative study. Male speakers from dialect region 7 were selected because of the availability and good representation of training and testing data available from this group. However, of the 13 vowel phonemes available, using the ARPABET representation [6], vowel /UW/ was not used due to the limited number of utterances leaving the 12 vowel phonemes, /IY/, /IH/, /EY/, /EH/, /AE/, /ER/, /AX/, /AH/, /UH/, /OW/, /AO/, /AA/. During the experimentation it was not only of interest to test the effect of recognition rates and convergence on the adapted vowels but also the effect the adaptation had on the remaining unadapted vowels. Unfortunately, testing the effects of inter-class adaptation on 12 vowel phonemes is a very labour intensive procedure and so the phoneme groups were reduced further. They were split into 3 distinct groups with respect to the tongue-hump position in the oral cavity during their production, 'front', 'middle', and 'back'. They were grouped in this way since phonemes from the same tongue-hump group show some acoustic similarities [7]. The front vowel phonemes were /IY/, /IH/, /EY/, /EH/ and /AE/, the middle vowel phonemes were /ER/, /AX/ and /AH/, and the back vowel phonemes were /UW/, /UH/, /OW/, /AO/ and /AA/. Using 'Speech Tools' [8] the relevant phoneme data was extracted from the recorded 16kHz speech files within the TIMIT corpus. Each phoneme file was pre-emphasised, to compensate for the -6db/octave roll-off of voiced speech and windowed using 8 over-lapping hamming windows, each representing 16ms of speech. The speech data in each window was used to generate 12 linear predictive coefficients (LPCs) which were normalised by dividing by the first. The first coefficient could therefore be eliminated since it was always equal to one. This left 11 LPCs for each window resulting in a total of 8x11=88 coefficients representing each vowel phoneme. Linear prediction with its simple coding and well documented behaviour was specifically chosen as the most appropriate form of speech pre-processing since all experimentation was primarily concerned with the performance of the ANN architectures.

3. ANN ARCHITECTURES

To test the performance of the OCON architecture on the vowel phoneme speech data, a comparative study with the more conventional MLP was set. The OCON and MLP architectures were represented by three networks each, corresponding to the 'front', 'middle' and 'back' tongue-hump groups of the speech data. For each phoneme group the MLP and OCON networks (see Figure 2) were modelled using the Stuttgart Neural Network Simulator (SNNS) [9]. All the networks contained the same number of input nodes, 88, dictated by the number of input coefficients representing each speech utterance. The total number of output nodes for each network was dependent on the phoneme group, five phoneme classes for the front and back and three phoneme classes for the middle. The six networks, with every node using the sigmoidal activation function, were modelled with fully connected adjoining layers, except for the hidden and output layers of the OCON architecture.

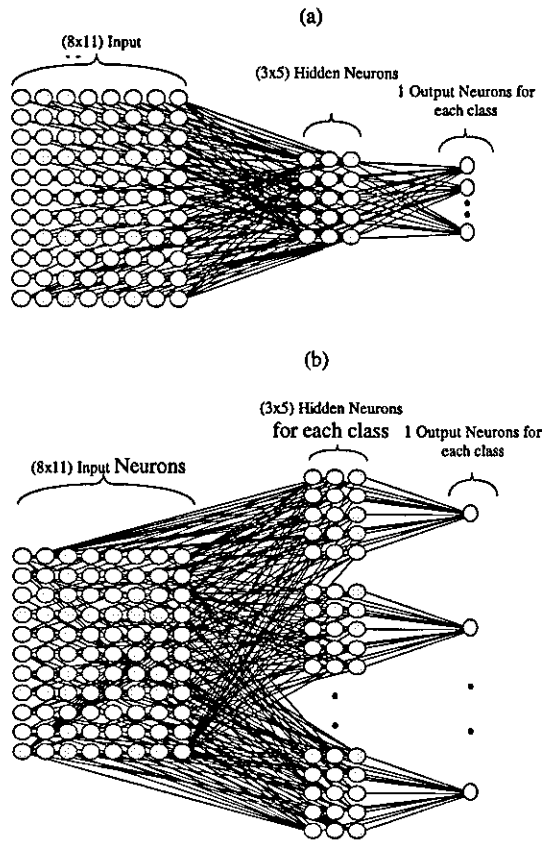


Figure 2: (a) Fully connected MLP architecture.
(b) Fully connected OCON architecture.

Each network was trained with male TIMIT training set from dialect region 7. The weight and bias values within the networks were initially randomised and the standard back propagation algorithm used to train the networks, producing the six 'base-classifiers' necessary for the experimentation. The male TIMIT 'test set' for dialect region 7 comprised of 15 male speakers. Since there was only interest in intra-speaker effects and not inter-speaker effects, all the speech data from every test speaker was amalgamated and categorised with respect to its phoneme content. The networks were then ready for adaptation and testing, but before that could occur, a single common back-propagation learning-rate for both the MLP and OCON networks had to be found. This was achieved by training one of the MLP and OCON networks with various learning rates. A learning rate of 0.5 was selected since it offered both networks fast convergence without any instabilities.

Each of the six base-classifiers was adapted and tested using the 'test set.' Each network was adapted towards one of its relevant phoneme classes for a total of 100 cycles, during which 7 result snapshots were taken at 1, 3, 5, 10, 20, 50 and 100 cycles. Due to the non-linearity of network adaptation, the number of cycles between each result snapshot increased to produce a graph that offered a clear picture of the network's behaviour. The results taken at each snapshot were the recognition rates of both the

adapted phonemes and the remaining unadapted phonemes within the same network. After adapting for 100 cycles towards each phoneme class, the weights and bias' within each network were reset to their initial base-classifier values ready for the next adaptation procedure involving another phoneme class.

4. RESULTS

Comparative results for the MLP and OCON architectures were obtained for adaptation towards each of vowel phoneme class and the effect on the remaining unadapted vowel phoneme classes within the same networks. 2 graphs were produced containing the averaged data from all the vowel phonemes for the adapted and unadapted phonemes recognition rates (see Figure 3(a)(b)). As well as recognition rates, another area of interest was each network's convergence rate. The convergence rate for each of the 2 averaged data graphs was calculated by differentiating the recognition-rate data (calculating the distance between adjacent rates). However calculating the convergence rate in this way was viewed as being unrealistic since the closer that recognition rates reach the perfect goal of 100%, the greater the significance of recognition improvement. To reflect this the convergence rate y was calculated using equation :

$$y = \left(\frac{100 - \chi_n}{100 - \chi_{n+1}} \right) - 1 \qquad (1)$$

where χ_n and χ_{n+1} are two adjacent recognition rates. The term -1 in equation 1 was used for normalise the graphs so that positive values indicated positive convergence and negative values negative convergence. The 2 convergence rates graphs were generated were for all the adapted vowel phonemes (see Figure 4(a)), and all the unadapted vowel phonemes (see Figure 4(b)). Figure 3(a) shows that the OCON networks show a clear improvement for the recognition rates of adapted vowel phonemes over the conventional MLP networks. On average, for all vowel phonemes, the experimentation shows a 12.3% increase in recognition rates for the OCON networks [10][11]. This result echoes the improvements shown in other data classification systems utilising OCON architectures [2][3][4][5]. Furthermore, the OCON architecture not only increases the adaptation rate but also reduces the processing time necessary for each adaptation cycle due to the reduction in network weights. This is shown in figure 4(a) with the increased rate of convergence for each OCON network, offering a 273% increase against the MLP for adapted phonemes. However, the OCON architectures as they stand, deal badly with inter-class adaptation. Although the rates of convergence for both networks are roughly the same, figure 4(b), figure 3 (b) shows that the OCON networks offer worse recognition rates for unadapted vowel phonemes over the conventional MLP networks. From figures 3 (b) we find that the average drop in recognition rates for the OCON networks, compared with the MLP networks, is 6.3%.

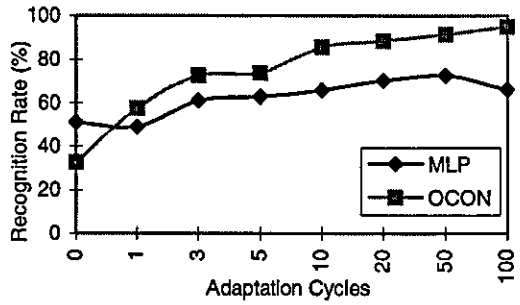


Figure 3(a): Average Recognition Rates for All Adapted Vowel Phonemes for an MLP and OCON Network

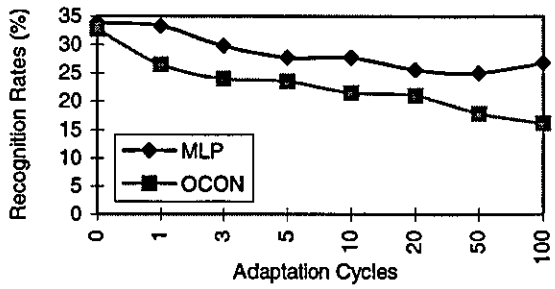


Figure 3(b): Average Recognition Rates for All Unadapted Vowel Phonemes for an MLP and OCON Network

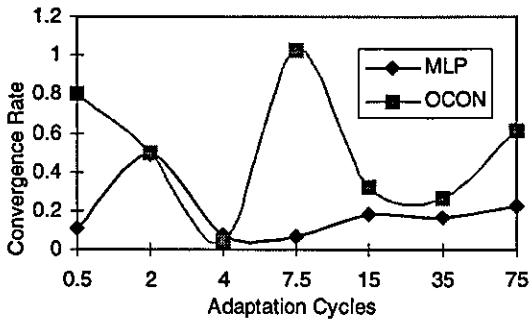


Figure 4(a): Average Convergence Rates for Adapted Vowel Phonemes for an MLP and OCON Network

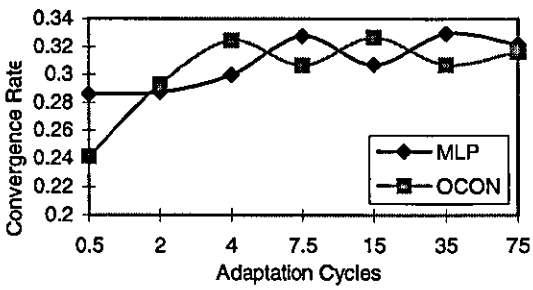


Figure 4(a): Average Convergence Rates for Unadapted Vowel Phonemes for an MLP and OCON Network

5. CONCLUSION

As expected the OCON behaves better than the MLP when adapting and testing the same phoneme. This is primarily due to the individual networks in each OCON network being dedicated to each class. Not only are there fewer connections and hence weighted axes to train, but each network only has to deal with information concerning a single class. As a result the OCON not only reduces the processing time for each adaptation cycle, but also rapidly increasing the convergence rate. However, the OCON architecture as it stands, deals badly with inter-class adaptation. When adapting to a class, the OCON shows a lower recognition rate for the remaining phonemes in the network compared to that of the MLP. This indicates that there must exist some common speaker information within all the classes in a network which isn't being exploited in the isolated networks of the OCON. Although in many applications cross-class interference can be a problem, MLPs compared to OCONs appear to use it to their advantage for inter-class adaptation. As a result an ideal network would be a hybrid OCON architecture containing isolated networks for improved single class adaptation but with some inter-class bonding to profit from any common speaker information. However it would be important that any hybrid OCON network should concentrate adaptation only on common speaker information as adaptation towards common class information could result in harmful cross-class interference.

6. REFERENCES

[1] DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM, Oct 1990.

[2] Jou, I., Tsay, Y., and Tsay, S. "Parallel Distributed Processing with Multiple One-Output Back-Propagation Neural Networks," *Proc, Int Symp on Circuits and Systems, Singapore,* pp.1408-1411, 1991.

[3] Kung, S.Y., and Taur, J.S. "Decision-Based Neural Networks with Signal/Image Classification Applications," *IEEE Transactions on Neural Networks, Vol.6, No.1,* pp. 170-181, January 1995.

[4] Kung, S.Y. "Digital Neural Networks," Prentice Hall, Englewood Cliffs, NJ.

[5] Hwang, J.N., and Hang Li. "Interactive query learning for isolated speech recognition." In Kung, S.Y., Fallside, F., Sorensen, J.A., and Kamm, C.A. "Neural Networks for Signal Processing," I, pp.513-522, *Proceeding of the 1991 IEEE Workshop,* Princeton, NJ, 1991.

[6] Shoup, J.E. "Phonological Aspects of Speech Recognition," 125-138, Ch6 in Trends in Speech Recognition, W. A. Lea, Ed., Prentice-Hall, Englewood Cliffs, NJ, 1980.

[7] Rabiner. L., and Juang, B. "Fundamentals of Speech Recognition," 26-27, Ch2, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[8] Speech Tools User Manual, Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology, August 1993.

[9] Stuttgart Neural Network Simulator, User Manual, Version 4.1, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart, 1995.

[10] Haskey, S.J., and Datta, S. "Dynamic Speaker Adaptation for Acoustically Similar Vowel Sounds using Sub-Cluster Neural Network," *Conference and Workshop on New Ideas in Computing,* Part 2, Coventry University, May 1997, pp41-44.

[11] Haskey, S.J., and Datta, S. "Using Tongue-Hump-Position Information for Vowel Adaptation within a Subcluster Neural Network," *IEE Colloquium on Pattern Recognition,*" Feb 1997.