Temporally Dynamic Neural Networks for Speaker Independent Phoneme Recognition

by K. Etemad
Advisor: S. Shamma

M.S. 93-17

The Institute for Systems Research is supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

Abstract

Title of Thesis: Temporally Dynamic Neural Networks for Speaker Independent Phoneme Recognition

Name of degree candidate: Kamran Etemad

Degree and year: Master of Science, 1993

Thesis directed by: Associate Professor S. Shamma
Department of Electrical Engineering and
Institute for Systems Research

An alternative view of neural network based phoneme recognition based on multiresolution signal processing which incorporates noncausal context is suggested. Also some suggestions are made regarding target and error weight functions to improve performance and simplify training. Based on these observations a temporally dynamic neural network with self recurrent links of different delays is suggested and tested on speaker independent unvoiced plosives, (p,t,k), recognition task with input feature vectors derived from an auditory model.

Temporally Dynamic Neural Networks for Speaker Independent Phoneme Recognition

by

Kamran Etemad

Thesis submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1993

Advisory Committee:

Associate Professor S. Shamma, Chairman/Advisor
Professor R. Chellappa
Professor P.S. Krishnaprasad

Acknowledgements

First of all, I thank God for everything, including the ability to learn, the pleasure of sharing it, and the wisdom of appreciating it.

My parents deserve the most thanks for their understanding and moral support and encouraging me to pursue all my goals.

I would like to thank Dr. Raymond Watrous and William Byrne for their very helpful comments and suggestions and helping me in both theoretical and practical problems.

I would like to expess my sincere gratitude to my advisor Dr. Shihab Shamma who provided me with guidance, encouragement and support. Also, I wish to thank Dr. R. Chellappa and Dr. P.S. Krishnaprasad (members of my advisory committee) for their helpful comments.

Contents

List of Figures

Chapter 1

Introduction

Automatic speech recognition (ASR) is one of the most challenging topics of research and has fascinated many speech scientists and engineers for over three decades. But despite the extensive efforts and progress in this field, the best systems developed so far have recognition capabilities far below those of a child.

The reason lies in both the complexity and variability of the acoustic speech signals and the various non-acoustic cues involved in human speech recognition. Variability and stochastic nature of speech signals are due to stochastic nature of their sources, namely motions and sizes of our vocal tract articulators and their constraints as well as variations in characteristics of the acoustic media. In other words these variabilities can be classified in the following groups:

1) Acoustic media: noise, interference and changes in environment, position and characteristics of transducer.

2) Across-speakers variability: speech signals contain talker dependent features as well as phonetic and linguistic information, and it is not always easy to separate them. In fact these talker dependent features are the basis for speaker recognition systems.

3) Within speaker variability: even for a single speaker the articulatory mo-

tions in vocal tract do not follow exactly the same path every time they are performed and therefore the spectro-temporal characteristics of produced sound, even for the same phoneme, are different. Variations caused by carelessness, stress and other psychological states and co-articulation effects (i.e. effects of context) are in this group.

4) Temporal variations: e.g. rate, intonation.

5) Ambiguity: generally there is not a distinct one-to-one map from acoustic features to phonemic variables.

Despite all these variabilities, speech signals are highly structured and are subject to phonetic and linguistic rules. Our knowledge of these rules and incorporating them as sets of constraints help us to remove most of the variabilities and ambiguities and enables us as humans to recognize speech sounds in very noisy conditions. Needless to say, in many cases some non-linguistic cues like visual information (e.g. gestures and lip reading) and our knowledge of speaker and subject of speech also contribute to our recognition performance.

Before considering the general schematic of an ASR system, let us look at a typical partitioning of them based on speaker variety and speaking rate. With respect to speaker variety, these systems can be divided into three categories, each involving different training paradigms.

Speaker dependent systems are trained with one speaker and perform satisfactorily for only that speaker while speaker independent ASR systems are capable of correct recognition regardless of the speaker. Of course to achieve speaker independence we need more complex systems with training data which represents a large population of speakers. So we train the system for a group of speakers and test them on a separate group of them. There are also multi-speaker systems that perform well for a certain group of speakers for which the system has been trained.

One can also classify ASR systems based on the speaking rate. Discrete utterance or isolated word recognition involves the recognition of single words. The talker may pronounce these words in isolation or in an utterance consisting of several words separated by distinct pauses. In connected speech recognition words are clearly articulated but there are no pauses between words. Because of possible co-articulation between words and unclear word boundaries these tasks are more involved.

More difficulty arises when we deal with continuous speech; where there are no pauses between words, they are not necessarily articulated clearly, also considerable extent of co-articulation effects are involved. These ASR systems are designed for relatively small (less than 100 word) to large (over 5000 word) vocabularies.

In many cases the fundamental part of an ASR system is the phoneme classification, this is specially the case for large vocabulary systems. Phoneme recognition in a sense can be considered as a difficult continuous speech recognition problem because at phoneme level not only there are co-articulation effects and unclear boundary problems but also there is not as much structure (to be used in recognition) as there is in word and sentence levels. In fact experiments show that despite excellent performance of human speech recognition people have many errors in recognizing isolated and segmented phonemes.

Despite all these difficulties, there is a considerable amount of literature on ASR systems, many of which have appeared in the commercial market place and perform well in constrained speech recognition tasks. The most popular recent techniques in speech recognition are Hidden Markov Model (HMM) based and Neural Network(NN) based schemes. Neural network models seem to be a suitable approach for phoneme recognition whereas HMM are more effective at word level. Some hybrid systems based on both HMM and neural networks are also the subject of recent research.

In this study we are interested in neural network based speaker independent phoneme recognition. Several major neural network models have been suggested and tested for this task. Some of these models are "temporally static" and perform classification based on a single [1,2] or a fixed number of temporally aligned frames[3,4] for each phoneme, while more recent models are "temporally dynamic and time shift invariant" systems that do not require pre-segmentation and time alignment and have a kind of "memory" associated with them. Temporally dynamic models seem to be a natural and effective way for phoneme recognition. These models are one of the major topics of our study. After introducing several major classes of these models, namely Time Delay Neural Network[5,6], Temporal Flow Model [7,8] and Recurrent Networks[9,10], a modified and relatively small sized network will be suggested and tested on the task of discriminating unvoiced stops, /p,t,k/. The modifications are inspired by some observations based on multiresolution and multirate signal processing ideas, incorporating non-causal context in recognition scheme and combined target and error weight function selection in training paradigm. In experiments, data has been extracted from TIMIT database and input feature vectors are outputs of an auditory model[11,12], and the results confirm the suggested approach and that the auditory model used preserves the perceptually important features of acoustic speech signals.

As a start we first look at a general schematic of an ASR system with more emphasis on neural network based schemes then temporally dynamic versus temporally static models will be introduced. The suggested observations will be discussed next, an alternative approach to network analysis and pruning will be given and subsequently the detail of experiment and results based on suggested ideas will be provided.

Chapter 2

A general neural network based ASR system

2.1 Introduction

Typically an ASR system can be divided into several modules (Figure(1)).

2.1.1 Signal processing and feature vector computation

The sampled speech signal are processed to produce a representation which conveys all linguistic information and suppresses all extra-linguistic information such as amplitude variations, talker stress, noise and other acoustic environmental interferences.

How we perceive and how machines can be made to perceive these auditory signals means in part finding appropriate representation for them and ways to compute them. This problem is in a sense harder than visual analysis where it is clear that the 2D image is a natural starting point and the most important primitives are edges and lines.

As we mentioned there are different kinds of cues (linguistic and nonlinguis-

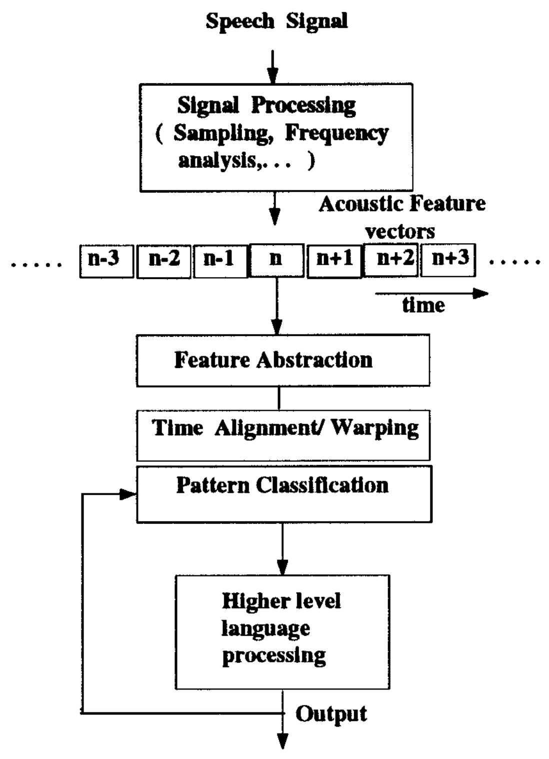

Speech Signal
Signal Processing (Sampling, Frequency analysis,. . .)
Acoustic Feature vectors
. . . . . n-3 n-2 n-1 n n+1 n+2 n+3 . . . . .
time
Feature Abstraction
Time Alignment/ Warping
Pattern Classification
Higher level language processing
Output

Figure 2.1: General schematic of an Automatic Speech Recognition (ASR) system

tic, acoustic and nonacoustic) involved in human speech recognition, but it is not possible to incorporate all of them in an ASR system. In fact in most cases the acoustic signal in the only tangible information. Chief among the acoustic cues is the frequency content of the speech waveform and its variations in time. In most simplistic form this frequency content can be viewed as a stochastic process involving two principle dimensions, time and frequency. This stochastic 2D signal is our major source for speech understanding.

In auditory systems it's harder to make precise which 2D representation are to be used and what are the appropriate primitives. Especially when we note that the combined time and frequency resolutions that can be achieved is bounded by uncertainty principle[13].

When we speak of frequency content, we mostly mean the local spectral energy concentrations that vary in center frequency as functions of time. These peaks are due in part to resonances in the vocal tract and are called "Formants". The formants locations (labeled F1, F2, ... in increasing order) specify the general vowel quality, recoloring and roundness while formant transitions between consonants and vowels play an important role in consonant identification. For voiced speech the first formant, F1, fall in the range 250-900 Hz, F2 has a wider range 600-3600 Hz. Formants F3, F4 and F5 may also be present in voiced speech. However, the lowest two(and some times F3) are usually sufficient to identify specific phonemes, while the location of higher formants are generally speaker dependent (Morgan[14]). In fact (A. Libermann 1967) claims that "The second formant transition is probably the single most carrier of linguistic information in speech signal"[14]. So most ASR systems focus on the first three formants and their trajectories in time.

Previous studies had indicated that the choice of preprocessing significantly influences the performance of an ASR system(Bengio and De Mori 89 [15])

The initial stages in speech processing are commonly performed using a Short Time Fourier Transform (STFT) of the digitally sampled acoustic time series. Several representations of the STFT have been employed for ASR systems, including Linear, Logarithmic scale, Logarithmic mel-scale, Cepstral and differenced Cepstral coefficient. Recently some feature representations based on mammalian auditory system models have been suggested, and some neural network based experiments on speech recognition have been performed (Cosi, Bengio and De Mori 90,[16]) which show that the performance of the ASR system using the auditory model is better than those based on STFT representations.

All feature vectors used in ASR systems, e.g. Filter banks, FFT or LPC coefficients, Cepstral coefficient, auditory model based features. are computed in such a way that , they reveal these formant trajectories.

In continuous speech, different speech sounds have very different average durations. The human auditory system adjusts attention easily and quickly to recognize not only relatively steady-state sounds whose identity is determined primarily by the location of spectral peaks of formants(e.g. vowel like /ae/ and /a/ and fricatives like /s/ and /z/) but also very brief impulse-like sounds (for example stops like /p/ or /d/) and much longer periodic sounds whose phonetic identity is determined over a large extent by their spectral dynamics (e.g. diphthongs).

Therefore in an ASR system the choice of input window duration is important, because selecting an input window which is too long may make detection of short term events difficult and lead to prohibitive training times. On the other hand a very short window may lead to poor generalization as a result of encoding acoustic events which are not of sufficient duration to be relevant for discrimination.

In this study we use an auditory model based time-frequency representation a brief description of which will be given later.

2.1.2 Feature Abstraction

After feature vectors have been generated they can be directly used for pattern matching, but often it is more convenient to first perform kind of feature abstraction, by which we mean removing most redundant variabilities and come up with compact feature vectors of smaller dimension. These secondary internal representations are typically abstract in a sense that, they may not have any specific acoustic or phonemic meaning, but they are more robust to non-linguistic variations than initial features and their combination contains almost all information we need for our discrimination task. The acoustic feature vectors and their corresponding internal abstract versions are computed in short time and typically equally spaced intervals such that the sequence of these vectors reveals all formant transitions.

2.1.3 Time-warping and Pattern Matching

Variabilities in duration and rate of speech utterances makes their acoustic (time-frequency) representations different. In other words when the same speech pattern is spoken with different rates , duration of the stationary segments (like vowels) may vary but the non-stationary segments (e.g. stops) remain of almost the same length. In comparing the two acoustic features X and Y it is desirable to absorb this kind of unessential distance caused by rate difference. Dynamic time warping is a matching method for time-frequency patterns to absorb this

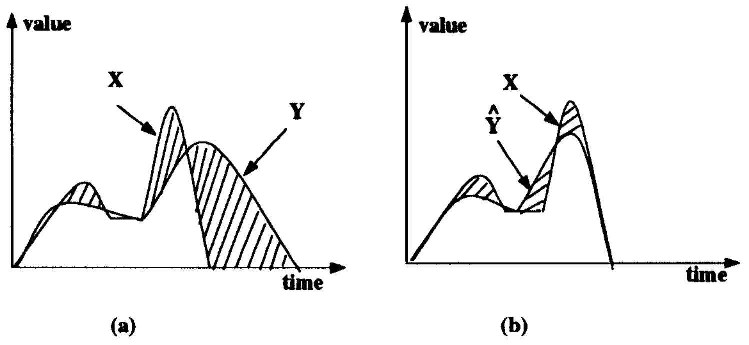


Figure 2.2: Nonlinear Time Warping, (a) original patterns to be matched, (b) after time warping

distance by non-linear time alignment, i.e. non-linear stretching or compressing in time, for speech patterns,[14,17], figure(2).

Statistical methods (which can remove acoustic variabilities) can be integrated with DTW approaches (that absorb the time variabilities) to achieve robust recognition and this leads us to the idea of Hidden Markov Model (HMM) based methods.

The HMM uses a Markov chain to model the changing statistical characteristics observed in speech signals. It is a parametric modeling technique in contrast to the non-parametric DTW schemes.

If the Viterbi algorithm is used for decoding in HMM based speech recognition, decision will be made based on the probability $\mathrm{pr}(x_i \mid s_j)$. This is the probability that the input frame x_i is produced from the template state s_j which is the information theoretic extension of distance to probability.

The power of HMM lies in the fact that the parameters used to model the speech signal can be trained to be optimized based on our prior knowledge of some speech waveforms. This results in lower computational complexity and improved recognition accuracy[17].

The other major class of pattern classification algorithms are neural network based systems. Neural networks are particularly interesting for speech recognition, which requires massive constraint satisfaction, i.e. the parallel evaluation of many clues and facts and their interpretation in the light of numerous interrelated constraints. The most distinctive feature of neural network based classifiers is that they compute matching scores in parallel and have parallel input and outputs where internal parameters (connection weights) are typically trained adaptively using training data.

2.1.4 Higher level language processing

The outputs of the decision units of the primary classifiers will be given, to higher level language processing modules where the knowledge of the specific language (at subword, word and may be phrase levels) is used to improve the recognition performance, through removing impossible and "invalid" combination of phonemes. The results of these decisions, in the form of last labels and their confidence measures could be fed back to earlier stages, as kind of context inputs, to reduce the number of similar possibilities among which the classifier should make its decision.

2.2 The Cochlear Model

As we mentioned before, the spectro-temporal characteristic of speech waveform and especially the first three formants trajectories on time form the most informative tangible feature set for speech recognition. In order to obtain these formants i.e. ridges and peaks in the spectrum, we can start with a fine short time frequency analysis of speech and then perform kind of edge detection on this 2D (time-frequency) plane.

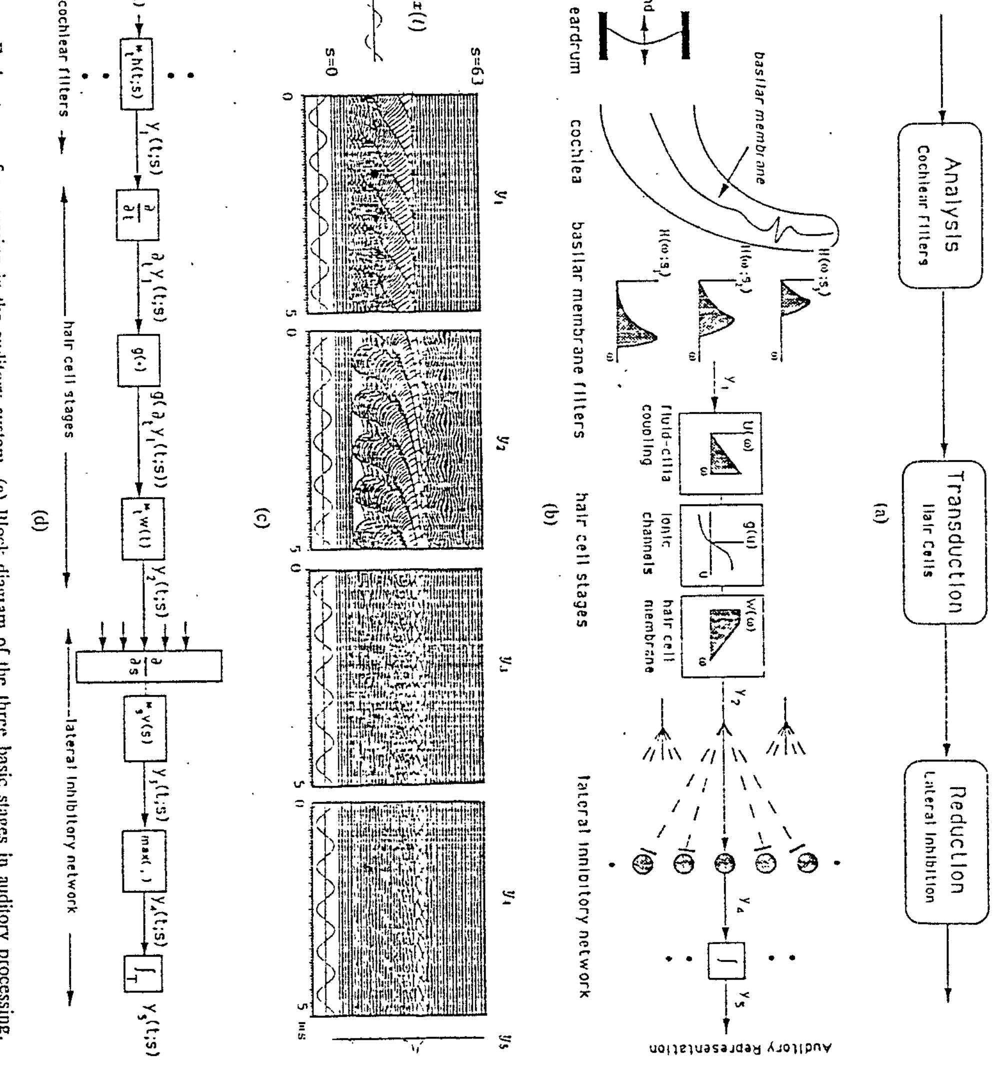
The input feature vectors in our experiments have been computed following the same idea and based on an auditory model, figure(3)

The model is based on the spatio-temporal frequency decomposition effects in cochlear and some nonlinear processings (including lateral inhibition) in human auditory system. Details of this model can be found in Shamma[5][6]. What we explain here is a very brief, simplified and functional description of the model as far as it relates to this study.

Sound pressure waves after guided by organs of outer, middle, and inner ear, cause mechanical displacement of the so called Basilar membrane in the cochlea of our inner ear. Frequency and amplitude of these displacements is directly related to the frequency and power of the acoustic stimuli. Because of the spatial selectivity of the membrane to frequency of the mechanical stimuli, it maps different frequency components of incoming stimuli onto different spatial locations in a tonotopically ordered manner along its length which results in a spatio-temporal patterns of input sound.

These patterns of displacements are transfered to hair cells along the cochlea and after three complex stages , namely the fluid-cilia coupling, the ionic channels and the membrane potentials, they produce signals of electrical nature in neurons. These pulse modulated signals are sent to the auditory cortex where higher level processing is performed.

From our functional point of view, the cochlear can be considered as a parallel bank of bandpass filters i.e. 128 channels with frequency responses inspired by the observed patterns of frequency selectivity along the cochlea. These frequency responses are shifted versions of each other on a logarithmic frequency scale, maintaining a constant Q-factor. At this stage outputs of these channels roughly



Early stages of processing in the auditory system. (a) Block diagram of the three basic stages in auditory processing. (b) Quasi-anatomical sketches of the auditory stages. (c) Examples of the response patterns at various stages of processing. (d) Mathematical models of the different auditory stages. See Section II in the text for details of derivations.

Figure 2.3: Cochlear Model, Shamma[11,12]

give the fine spectra of speech sound. The three mentioned complex stages can be modeled by a time derivative, a simple nonlinearity and a lowpass filter respectively. Inspired by the observations in human visual and auditory system, and to enhance peak and ridges in the fine spectra, lateral inhibition (LIN I and II) has been included in the model[11,12]. Therefore the spectral profile of the stimuli are found by rapidly detecting discontinuities along the spectral (spatial) axis of the auditory nerve patterns and integrating its outputs over a few mili-seconds. This final auditory representation ,although is different from SFT, contains most of spectral information and may even high light feature that from recognition point of view are more important.

This auditory representation forms the input feature vectors through out this set of experiments. Of course computation of input feature vectors, the correct choice of model parameters, especially window size and step size in the short time frequency analysis and a preemphasis parameter includes in the model is of great importance.

Chapter 3

Neural Networks in Phoneme Recognition

There are several major classes of neural networks that have been used for phoneme and word recognition.

3.1 Temporally static networks with time averaged inputs:

Assuming that speech signal has been already segmented and phoneme boundaries are, at least approximately, known and that the signal is almost stationary in each segment; one can perform the classification based on the averaged feature vector obtained from the whole duration of each phoneme segment or based on a pre-selected typical frame from the middle of the utterance.

For this classification one may use any static pattern classifier, including single and multilayer backpropagation networks or Kohonen feature map, Learning Vector Quantization(LVQ) or Radial Basis Function networks(RBF)[14].

An example of this approach, is Huang and Lippmann's experiment[1] where they used a simple 3 layer feed forward network to form the nonlinear classification boundaries of the first two formants of 10 vowels. Their network had two inputs representing F1 and F2, 50 hidden units and 10 outputs corresponding to each vowel. Another example of this approach which is in a sense related to our study is K. Wang[2]'s experiment on phoneme recognition, where he starts from an auditory model based representation (consisting of 128 channels of cochlear model) and then computes the average feature vectors over hand segmented phoneme boundaries. In this case again a simple two layer back propagation network is used for classification.

As we mentioned before these methods lie on the restricting assumptions that the input signal is segmented correctly and that for each segment the averaged vector over the whole sequence is a "good" representative of the that segment. Considering that speech patterns are nonstationary in most cases the above method cannot be used in general.

3.2 Temporally static networks with explicit representation of time:

This class consists of methods in which the network does not have a dynamic structure (or memory as we discuss later on) but the time variation of input feature vectors are given to the network by incorporating time as a second dimension. The result is a static 2D input feature space, so its resembles an image pattern recognition. It does not require any kind of stationarity assumptions as previous class, because here the time variations of feature vectors are included. However in these networks because of the "static presentation of time" to the network precise segmentation and time alignment of input patterns are required for pattern matching. In other words although the time-frequency representation is a 2D signal ,it must be noted that one of the dimensions is "time", and in order to have correct recognition the system should take a shift invariant look at input frames. Otherwise sequences like X=000101000 and Y=001010000) that are two instances of the same basic pattern observed at different times will be considered as completely different patterns because they are "spatially" quite dissimilar and distant. Therefore in speech and generally in time sequence recognition it is desirable to keep track of relative temporal position of events rather than absolute temporal positions.

An examples of these schemes is Elman and Zipser's experiment[3] on phoneme recognition for the voiced stop consonants / b,d,g / followed by the vowels / a,i,u /. Their input feature space consisted of twenty frames of 16 DFT coefficients computing at overlapping 3.2 msec time intervals that are given to a 3 layer backpropagation network.

Of course the same classification task can be performed using Kohonen feature map and LVQ ideas. All these networks, because of their shift variance with respect to incoming sequence of feature vectors, rely on pre-segmentation and time warping.

3.3 Temporally dynamic networks:

The principal goal of an effective phoneme recognizer is to capture the dynamic nature of the acoustic-phonetic trajectory of the speech signal. The temporal aspect of this task is particularly challenging, some speech recognition systems attempt to parse or segment speech patterns into discrete units roughly corresponding to phonemes. However, the best segmentation schemes are highly susceptible to errors; these errors in turn, result in higher errors rates further along in the recognition process. As a result a robust speech recognition should simply scan the speech signal for useful cues without relying on pre-segmentation, basing its overall decision on the sequence and co-occurrence of a sufficient set of these cues. This, in turn, suggests a system that is temporally dynamic (i.e. a system whose recognition performance is unaffected by temporal shifts of the input speech train). The experiments mentioned above used utterances that were precisely parsed from the speech signal, obviating the need for shift invariance of the system. The following series of experiments all employ techniques aimed at yielding shift invariant phoneme recognition.

Temporally dynamic networks, because of their special delay structure, take a shift invariant look at input sequences and so they are more suitable for time sequence recognition tasks. The delayed links in feed forward and feed back paths give the network kind of a "memory", which enables the network to inspect temporal variations in the history of the input sequence and therefore to capture the dynamic as well as static discriminative features needed in classification. Because of the shift invariance we do not need strict time alignment and segmentation, but we have to enlarge the training set such that for each phoneme, it includes examples of different durations.

Several neural network models of this characteristic have been developed so far. Among them are the Time Delay Neural Network(TDNN), (Waibel[5,18,19]), Temporal Flow Model(TFM) (Watrous[7,8]) ,and Elman's recurrent network (Elman [9]) (figure(4)).

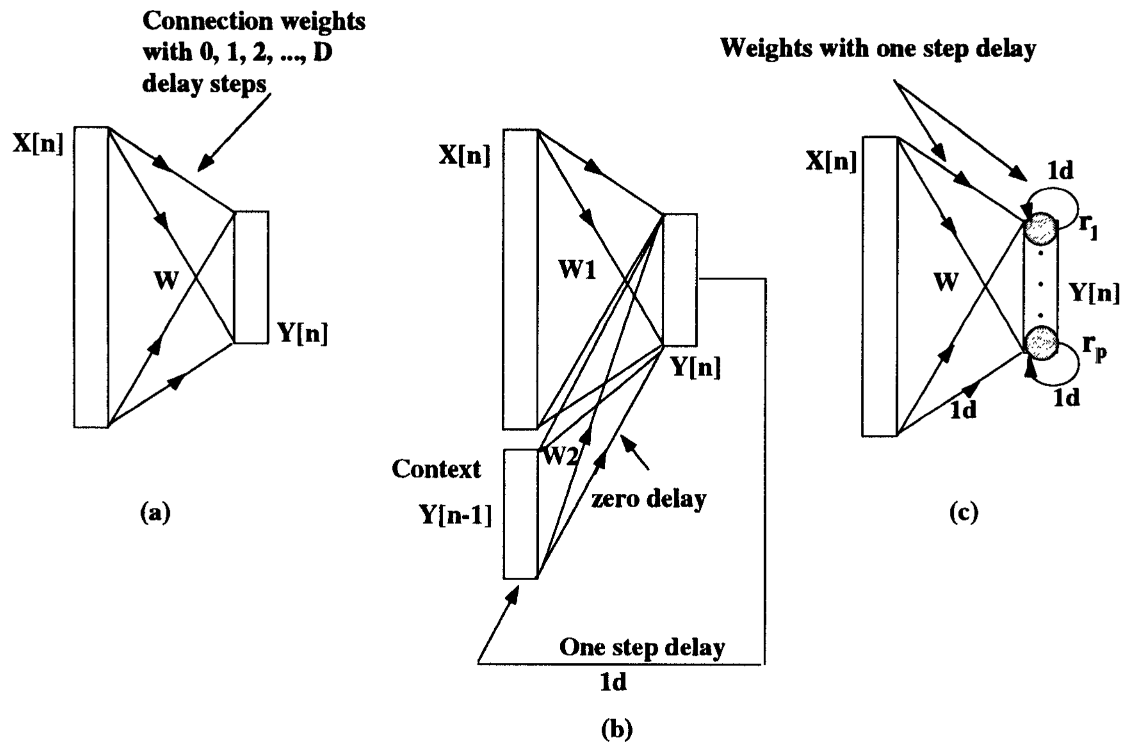


Figure 3.1: Examples of basic blocks of temporally dynamic networks, (a) Time Delay Neural Network (TDNN), (b) Elman's Recurrent Network, (c) Temporal Flow Model

3.3.1 Time Delay Neural Networks

The TDNN is a multilayer feed forward neural network which can be trained to recognize specific structures within consecutive frames of speech. Temporal structure in the TDNN is represented as increasing levels of abstraction and duration in progression from the input layer to the output layer. Time delayed input frames allow the weights in the initial layers to account for variations in the spectral representation of speech. The cells of TDNN integrate activity from adjacent time delayed vectors, in such a way that these vectors are treated as additional inputs occuring at the same time.

The TDNN is trained using Back Propagation(BP) algorithm[20], however cells within a TDNN compute activities at different time intervals, with cells at the input layer active at input frame rate and cells at the output layer activated only after a complete speech segment has been processed.

According to Waibel[5,19], the TDNN is a coherent architecture for integrating temporal and spectral models within a single neural network. The network is apparently able to discriminate acoustically similar patterns with high accuracy and robustness given widely varying contexts. In essence, the network is forced to uncover the underlying spectro-temporal structure which discriminates one class from another.

Perhaps the most important characteristic of the TDNN is the temporal compression which results from this architecture. Note that in TDNN the number of delays at each layer is more than the number of delays in previous layer i.e. $d1 < d2 < d3$. This property of the TDNN allows the length of the input "window" on the feature representation to be short in comparison to the full duration of the utterance. In applications discussed above, the input window corresponded to three adjacent vectors (30 msec) of spectral features. The du-

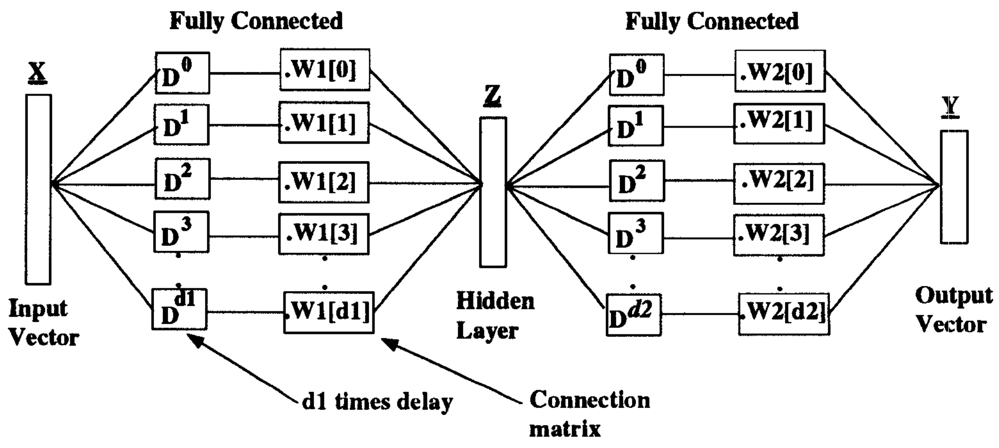


Figure 3.2: Delay structure in a two layer TDNN

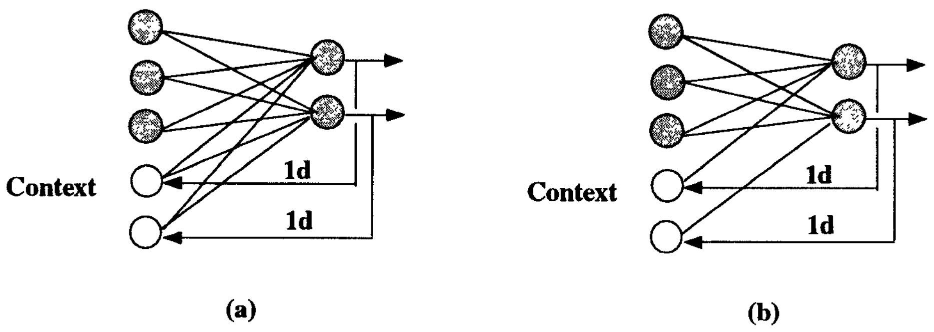


Figure 3.3: self-recurrent weights are special cases of context weights

ration of this window was selected to match the expected duration of acoustic events sufficient for the discrimination of the stop-consonant segments in the training set[5].

The TDNN architecture substitutes the problem of matching the utterance duration with similar problem of matching low level acoustic events. the selection of a TDNN input duration to match these acoustic events will depend on task-specific knowledge. For example, because the expect duration of a vowel is on the order of 100 msec, 10 frames of length 10 msec are needed to observe the context variation due to the preceding and following phonemes.

In addition to minimizing the problem of time alignment, the temporal compression architecture can force cells within hidden layers to fire for "sequences" of acoustic events. Cells within the first hidden layer compensate for short-term acoustic events, while cells in the second hidden layer detect events of longer duration.

According to Waibel[5] analysis of the internal representation of the first and second hidden layers for phoneme discrimination demonstrated that the cells of the first layer detect second formant slope transitions and vowel boundaries. However, the cells of the second layer did not seem to be sensitive to sequences of these events. It may be possible to generalize the type of representations learned by the TDNN to more complex problems which occur in word and language perception. However, extension to word-lenght duration may require significant modification to the TDNN to permit realistic training times. This is due to the fact that the size of the TDNN is proportional to the duration of the acoustic events to be recognized. In this respect, recurrent networks are more efficient because their size is independent of the duration of the acoustic event.

3.3.2 Elman's Recurrent network and Temporal Flow Model

Recurrent neural networks are characterized by both feedforward and feedback paths. The feedback paths enable the output at any layer to be used either as input to a previous layer, or to be returned to that layer after one or more time steps. One of the motivations for using recurrent networks for temporal modeling is that they allow time to be represented by the effect it has on processing a sequence (Elman[9]).

in most recurrent networks, the output of the node in layer q at time t is stored in a context node so that it may be used as an input to some set of nodes (generally within the same layer) at time $t + 1$. The weights between the context nodes and the nodes they feed into are called the context weights. In the simplest example, a node has a self-recurrent weight when the output at time t is delayed and then used as an input to determine the output at time $t + 1$. As figure (6) shows the self-recurrent link configuration is a special case of context weight connections. That is, the self-recurrent link connects the node from which the activation originated, while the context weights are fully connected to the nodes in layer q.

Although in this study we are more interested in temporal flow model introduced by watrous and Shastri, it must be noted that other recurrent ANN paradigms which achieve similar results have also been developed[15,16]. Recurrent networks are appealing because they can integrate activity levels over time and thus "remember" activations from previous inputs. The temporal flow model designed by Watrous and Shastri uses a three layer network in which each node in the hidden and output layer contains self-recurrent weights with a single time step delay. It also has a unit time step delay between the output of successive layers. Thus, for the three layer network two time steps are required for the output to progress from input to output layer.

The temporal flow network permits the context weights to vary and uses the standard Back Propagation training algorithm to determine the values of the weights (including the context weights). Thus in the BP equations, the context weight are treated as additional weights from layer $q-1$. The activation at the context node is updated after all the weights have been adjusted.

In order to train this network, the desired output must be expressed as a function of time. This is a significant departure from the training paradigms of static networks in which the desired network output is fixed for the duration of a specific pattern.

The selection of an appropriate time dependent "target function" is a somewhat empirical process. However the specification of the network behavior over time is very important because the target function determines how network weights are updated and how intermediate and final performances are evaluated.

One of the most interesting results of this experiment is that the network performed the classification task without making a direct comparison of acoustic segments across the CV syllable.

The basic operation of these temporally dynamic networks can be expressed in terms of the following equations for single layer networks for each model.

TDNN:

$$Y[n] = f(\sum_{d}(X^{T}[n-d] \cdot W[d]) + \theta) \tag{3.3.1}$$

Elman Recurrent Network:

$$Y[n] = f(X^T[n] \cdot W1 + Y[n-1] \cdot W2 + \theta) \tag{3.3.2}$$

Temporal Flow Model:

$$y_i[n] = f(X^T \cdot W_i + r_i y_i[n-1] + \theta_i) \tag{3.3.3}$$

where $X[n]$'s $(X[n] = (x_1, x_2, ..., x_p)[n])$ and $Y[n]$'s $(Y = (y_1, y_2, ...y_m)[n])$ are input and output vector sequences respectively. $\theta = (\theta_1, ..., \theta_N)$ is the vector of thresholds for all cells in the network. r_i is the weight of the recurrent link of i^{th} cell, W, W1 and W2 are connection weight matrices between the two layers, and function $f(.)$ is the nonlinearity, typically sigmoid, associated with each unit, Figure(4).

Chapter 4

Some Observations and Modifications

4.1 Introduction

We are seeking in our research effort the advantages of combining the flexibility and learning abilities of the neural networks with as much knowledge from speech science as possible in order to advance the construction of a speaker independent phoneme recognizer. Prior knowledge can be used in many steps of this construction, e.g. preprocessing, choice of input feature vectors, output supervision, and design of network architecture.

In particular, hypothesis about the nature of the processing to be performed by the network based on prior knowledge of speech production and recognition processes enables to put constraints on the architecture. These constraints result in a network that generalizes better than a fully connected network (Baum and Haussler [21]). This fact leads also to the idea of modularization and scaling in design of neural network based ASR systems, (Waibel[18]).

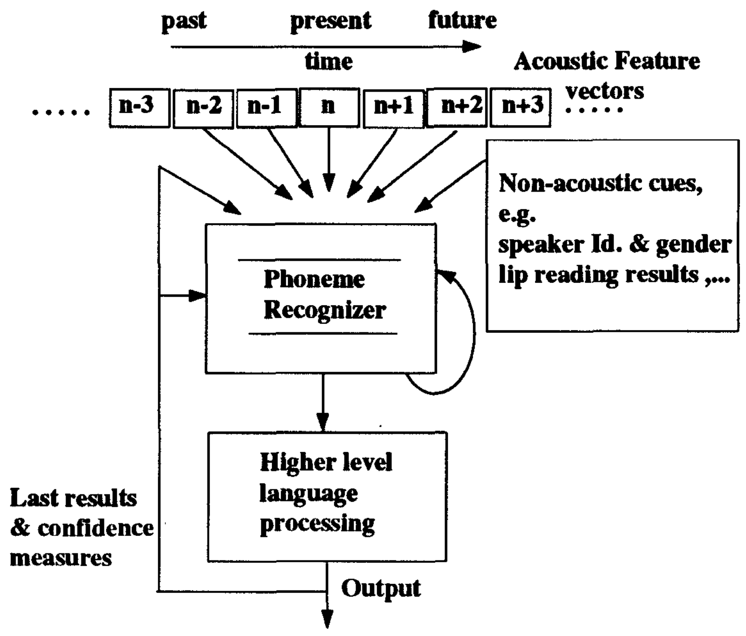

past
present
future
time
Acoustic Feature vectors
n-3
n-2
n-1
n
n+1
n+2
n+3
Non-acoustic cues, e.g. speaker Id. & gender lip reading results ,...
Phoneme Recognizer
Higher level language processing
Last results & confidence measures
Output

Figure 4.1: Incorporating contextual information in speech recognition

4.2 Incorporating causal and non-causal context

Human perception of speech highly relies on contextual information. In many cases our knowledge of the speaker, language and subject of speech help us to recognize ambiguous or incomplete speech utterances even in very noisy condition. Based on this fact most modern automatic speech recognition systems incorporate, in one way or another, the contextual(mostly acoustic and some times visual [22]) clues in their schemes[23]. In neural network based automatic speech recognition systems this ideal can be approached using delayed links in feed forward and feedback paths and also feedback from higher level language processing modules.

The study of the speech production mechanism shows that the patterns of articulatory movements in our vocal tract for any phoneme depends on both previous and next phoneme i.e. co-articulation affects the acoustic pattern of speech in an "noncausal" way[24]. So in this case and for all signals for which the present state of signal depends on both past and future, it would make sense to expect that incorporating contextual information from both previous and next frames improve the recognition performance. As always we should accept some delay from input to output to realize this noncausality. We will discuss this later.

4.3 Multi-resolution in time

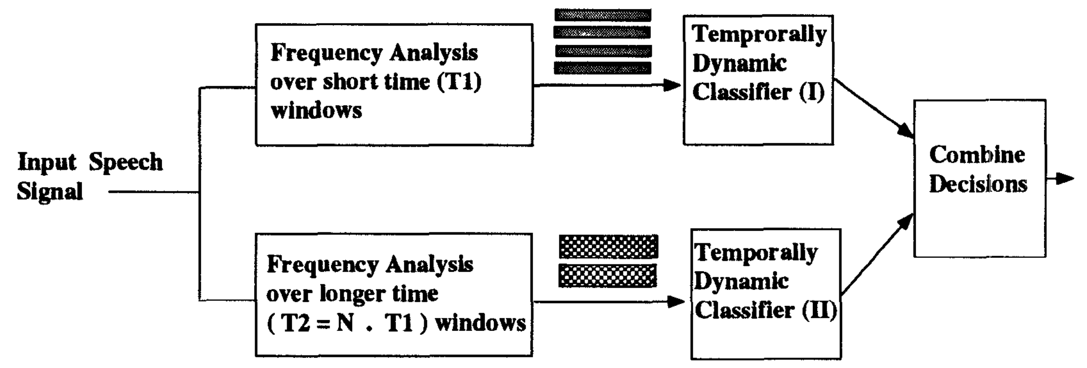
An important observation is that we can use, for recognition purposes, the multiresolution signal processing techniques and related ideas, that have been studied extensively in signal compression applications. Multi-resolution signal processing techniques have been shown to be a valuable tool in signal understanding, specially in image understanding and partly in speech. Neural networks using a single temporal window to provide successive fixed length "glimpses" of a speech signal should be capable of classifying steady state sounds of any duration. However, such a network would seem to be sub-optimal for classifying phonemes that are distinguished by dynamic characteristics that extend over a longer duration than the networks input span, and also may be unreliable at detecting events that are significantly shorter in duration than the networks input span. Consequently, the choice of input span may be critical to recognition performance for some phoneme classes. Based on this observation Kamm and Singal[25] have tested several feed-forward neural networks of different input spans (e.g. 35-ms, 65-ms and 245-ms) to classify sub-word speech segments. Their experiment showed that brief sounds can be reliably detected by networks with longer input spans, but results for a subset of longer duration phonemes(diphthongs) indicate that a network needs a wide enough view of the input to capture the salient features of the output class[26]. An alternative approach is to use a network architecture that has both short time fine resolution and long time coarse resolution spans simultaneously.

This leads us to the idea of multi-resolution inputs which implies primarily multi-resolution in time. The implementation of multiresolution methods for speech requires frequency analysis of short time speech segments using several window sizes. The final or even intermediate results of pattern classification based on different resolutions have to be combined later in a suitable way to come up with better and consistent final decision. This adds to the complexity of the system but improves the resulting performance. To avoid complexity, but based on the same idea, one can suggest the following methods to be used in ANN design of a phoneme recognition system:

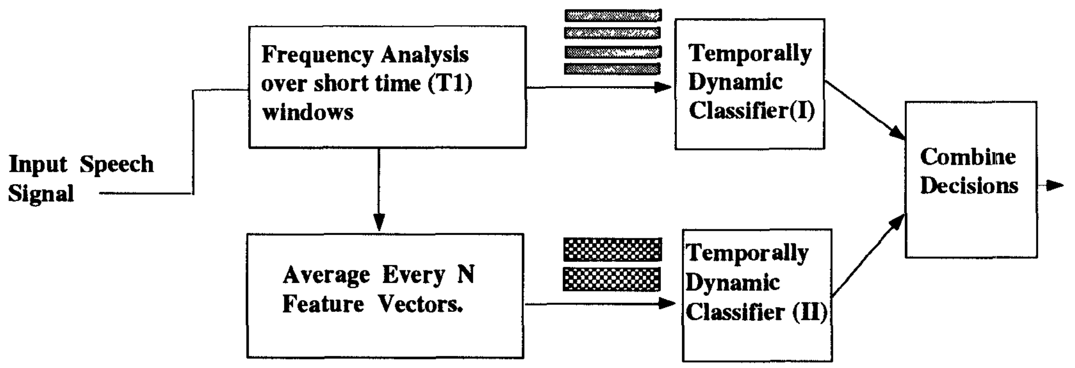
1) Fine scale feature vectors are averaged to estimate feature vectors at coarser scales, which are then used in the network as additional information corresponding to lower resolution "view" of the incoming signal.

2) For further simplification, the delayed structure of the network is chosen such that one class of units ("scale class") observe all successive frames, while units in other scale classes observe every other frame, or every other two frames.

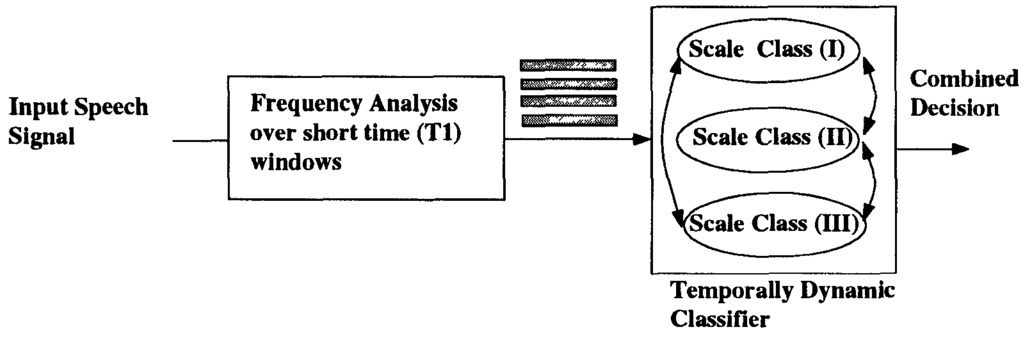
Figure(8) shows this idea, where scale classes are subunits(subnetworks) that operate with different frame rates. Using the results obtained from all scales the network can produce the final decision about the phoneme class. There are many strategies to combine informations derived from scales. For simplicity we may just combine the outputs of units corresponding to different scale classes as a combined hidden layer, which is fully connected to next (e.g. output) layer.


Input Speech Signal
Frequency Analysis over short time (T1) windows
Frequency Analysis over longer time (T2 = N . T1) windows
Temprorally Dynamic Classifier (I)
Temporally Dynamic Classifier (II)
Combine Decisions

(a)


Input Speech Signal
Frequency Analysis over short time (T1) windows
Average Every N Feature Vectors.
Temporally Dynamic Classifier(I)
Temporally Dynamic Classifier (II)
Combine Decisions

(b)



(c)

Figure 4.2: The idea of incorporating multiresolution in time, (a) original form , (b) first simplified version, (c) second simplified version

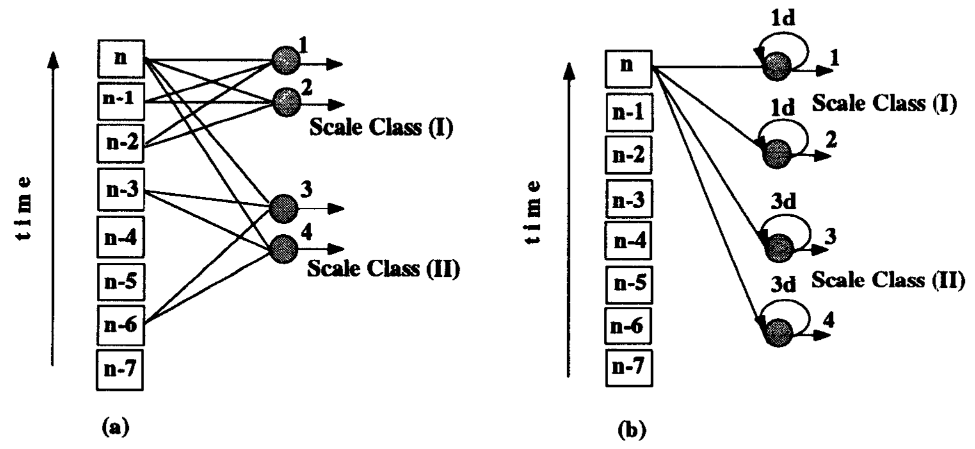


Figure 4.3: The simplified idea of multiresolution input for; (a) a Time Delay Neural Network, (b) a recurrent network

The idea above can easily be implemented in neural networks with time delay structure. Figure (9) illustrates this for a Time Delay Neural Network (TDNN) and a recurrent ("Temporal Flow") network. These classes of dynamic networks have been studied (Waibel[5,19,27] and Watrous[5,10]) mostly for phoneme classification.

After training we expect to see units sensitive to short time features in the fine scale class and others sensitive to long time features in coarser scale classes. For example in Figure (9, (a) and (b)) the first and second units in this single layer network contribute to short time feature detection and third and forth units observe farther in past with less detail and therefore contribute to long term "memory" of the network. For a TDNN it is easy to tell quantitatively how much memory it has i.e. how many frames are in the input span of the network. But for a recurrent network it is not as straight forward, because of the nonlinearities and dynamic feedback, and all one can do is to find a rough estimate of this memory.

Roughly speaking, linearizing the nonlinearity function locally, taking the Z-transform, expanding the fraction using geometric series and finding the inverse transform, the output can be expanded in terms of delayed inputs as follows:

$$z_i[n] = X^T \cdot W_i \tag{4.3.1}$$

$$y_i[n] = f(z_i[n] + r_i \cdot y_i[n-d]) \tag{4.3.2}$$

$$Y_i[z] = \frac{Z_i[z]}{1 - r_i z^{-d}} \tag{4.3.3}$$

$$y_i[n] \approx z_i[n] + r_i \cdot z_i[n-1d] + r_i^2 \cdot z_i[n-2d] + r_i^3 \cdot z_i[n-3d] + \dots \tag{4.3.4}$$

$z_i[n]$ is the total activation level at the input of i^{th} unit at time/frame n; W_i is the connection weight vector from input "X" to output y_i ; r_i is the weight of the self-recurrent link to i^{th} unit and " d" is number of delays associated to this link. The function f(.) is typically a Sigmoid.

If r_i is not too small we expect to have the effect of past few frames, roughly $4 \cdot d$ or $5 \cdot d$ frames. The recurrent links may be fixed or left as free parameters during the training.

It may be argued that for the TDNN architecture the fully connected network with identical delay structure contains all connections in figure(9.a) above and should be more general. But the goal is to find a simple network with small number of free parameters to make training, analysis and implementation easier and to improve generalization. Note that the recurrent network is much smaller in terms of free weights, and that the self-recurrent links can provide a very simple and useful memory structure.

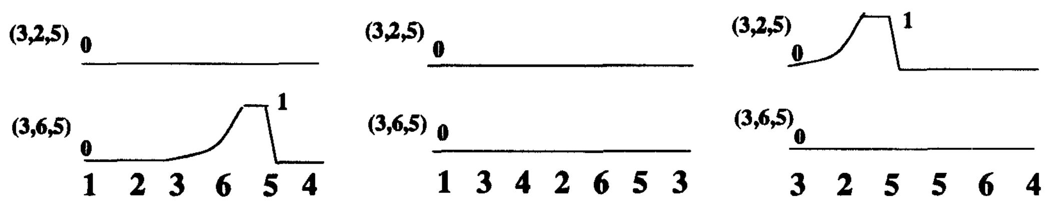


Figure 4.4: Choosing suitable target function

4.4 Target and Error Weight Functions

In the supervised learning mode of a temporally dynamic network used for time sequence recognition, in most cases we need a target function. The target function is a sequence of desired outputs defined for any input sequence from training set. For example, for applications in which recognition is performed within context, the target function should be zero everywhere except for the duration of desired process at the input. Sometimes the role of target function and the importance of an appropriate choice of its shape and timing is overlooked. A good choice of this function simplifies training and improves the generalization of the network. One can easily give examples of sets of time sequences (to be recognized) for which a flat "0" and "1" target value for the whole interval is not a good choice; even though the average error could be made small with long exhaustive training of large number of free parameters. For example, consider a time sequence of labeled events, and suppose we want to discriminate two possible sequences $\{3, 2, 5\}$ and $\{3, 6, 5\}$ so for this task and assuming that all possible sequences may occur, we should choose the target function as in figure(10).

To choose a suitable target function, we should first estimate the point in time where a perfect recognizer can identify a time sequence from other possibilities. For speech signals an accurate estimate of such points is a separate difficult

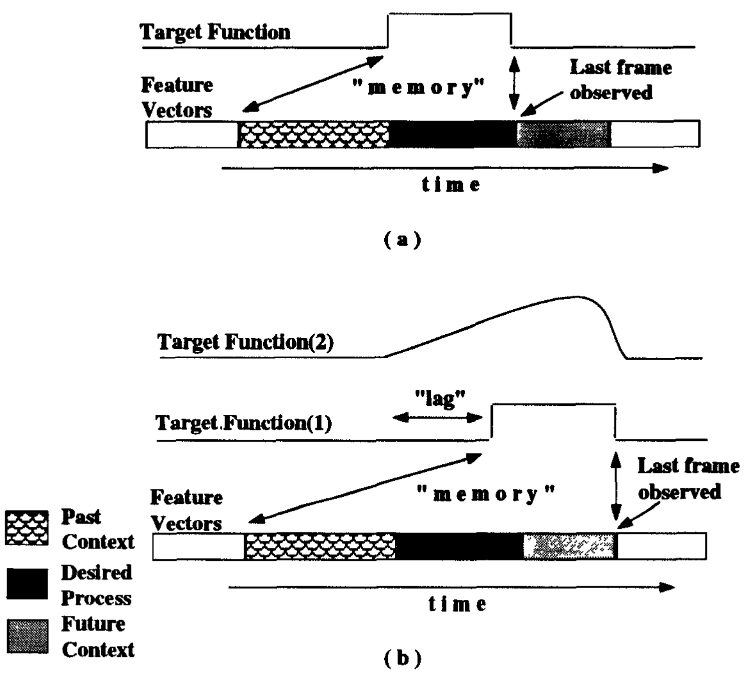


Figure 4.5: Target function and noncausal frame level context

problem, but it is possible to find rough estimates for our purpose.

In phoneme recognition a continuous target function (e.g. Gaussian, Trapezoid or raised cosine) with suitable transition slope and peak timing would be suitable.

Another important observation is that by considering a target function which lags the input sequence by a few frames the noncausal context can be simulated at the frame basis level. Here the network asserts a phoneme class after it observes a few frames following the "present" frame.

Having introduced the target function, it is important to mention the weighted mean square error as opposed to conventional uniform mean square error function, otherwise called the error weight function. In many cases errors in different parts of the output sequence are not of equal importance, so instead of mean square error, our objective function could be a weighted sum of the frame level square error. Proper choice of this error weight function can reduce training time.

4.5 Training, Validation and Scoring

Training a neural network means finding a good set of connections weights that provides the required input-output mapping, for all input patterns in the training set. A successfully trained network, with appropriate choice of architecture and suitable output supervision, should be able to generalize the desired input-output map to examples which have not been presented to it during the training time.

Most training algorithms, are based on gradient descent idea, among which the so called Back Propagation(BP)[20] is the most popular algorithm in training multilayer feed forward networks. There are also Quasi-Newton nonlinear optimization methods that are used in training in connectionist models[28]. The BFGS algorithm[29] is one of these methods which will be introduced later.

The whole data set is first separated into two disjoint groups of training and test sets. Only training data is used for training the network. During the training process the training set itself is divided dynamically into two subsets, namely training subset and "Validation" subset. The training is performed in several steps:

1) Start with a small training subset.

2) initialize all connection weights to random small values (typically in [-1,1]).

3) present the input training and validation data and their corresponding desired output to the network. Evaluate the average objective error function for

each subset separately. Save the two computed errors.

4) Update the connection weights based on the calculated errors. In this step one may use BP or BFGS or other optimization methods.

5) perform this updating few times, (5-10 iterations).

6) reevaluate the objective error function on validation set; If it is smaller than what has been saved in step 3 continue from steps 4 until desired small error is obtained. Otherwise enlarge the training set by including some of examples in validation set and then continue from step 3.

This approach is taken to avoid over-training and achieve better generalization. The update rule for connection weights based on an "Accelerated BP" algorithm is given as follows:

$$\Delta w_{i,j}(n+1) = \mu \Delta w_{i,j}(n) - \eta \frac{\partial E}{\partial w_{i,j}(n)} \tag{4.5.5}$$

where $w_{i,j}(n)$ is the connection weight from unit (i) to unit (j) evaluated at iteration (n). E is the error function value, η is called the learning rate, and μ is the momentum constant. The effect of the momentum term is to magnify the learning rate for regions of weight space where the gradient is essentially constant. In simple BP algorithm momentum is zero and learning rate is constant, whereas in faster versions of that both η and μ are nonzero and change adaptively during the training process. Back propagation update formula has a simple form for squared error objective function with neural units of sigmoidal non-linearity[20].

For neural units with sigmoid nonlinearity we have,

$$y_j[n] = (1 + e^{-z_j[n]+\theta_j})^{-1} \tag{4.5.6}$$

$$z_j[n] = \sum_{i,d} w_{ijd} y_i[n-d] \tag{4.5.7}$$

also for mean square error (MSE):

$$E \propto \sum_{x,j,n} ((y_j^x[n] - trg_j^x[n])^2 \cdot wgt^x[n]) \tag{4.5.8}$$

we can write,

$$\frac{\partial E}{\partial w_{i,j,d}(n)} = \sum_{x,j,n} [y_j^x(n) - trg_j^x(n)] \frac{\partial y_j^x(n)}{\partial w_{ijd}} \tag{4.5.9}$$

$$\frac{\partial E}{\partial w_{i,j,d}(n)} = \sum_{ijd} [y_i^x(n-d) \delta_j^x(n)] \tag{4.5.10}$$

where error signals $\delta_j^x(n)$ are computed recursively in terms of the error signals of units which they are connected to, and finally in terms of differences between output and target function values. So there is a backward recursion as follows.

for output units,

$$\delta_j^x = [y_j^x(n) - trg_j^x(n)] \frac{\partial y_j}{\partial z_j}(n) \tag{4.5.11}$$

while, for other units,

$$\delta_j^x = (\sum_{i,d} w_{jid} \delta_i^x(n-d)) \frac{\partial y_j}{\partial z_j}(n) \tag{4.5.12}$$

The BFGS algorithm also gives an update method given in the following sequence:

For $x_0 \in R^n, S_0 \in R^{n \times n}$, Positive definite (e.g. $S_0 = I$)

```
while $\nabla f(x_i) \neq 0$ do
$h_i = -S_i \nabla f(x_i)$
pick $\lambda_i \in \arg\min_\lambda (f(x_i + \lambda h_i) \mid \lambda \geq 0)$
$x_{i+1} = x_i + \lambda_i h_i$
Compute $S_{i+1}$, positive definite using some update formula.
stop.
```

and update formula for BFGS is the following.

$$S_{i+1} = S_i + \frac{\gamma\gamma^T}{\delta^T\gamma} - \frac{S_i\delta\delta^T S_i}{\delta^T S_i \delta} \tag{4.5.13}$$

where

$$\delta = x_{i+1} - x_i \tag{4.5.14}$$

$$\gamma = \nabla f(x_{i+1}) - \nabla f(x_i) \tag{4.5.15}$$

Where $f(.)$ is the objective function and $x \in R^n$ is the parameter with respect to which the minimization has to be done. After some iterations the error decreases to some value which is typically a local (an hopefully global) minimum of the objective function. Perturbation, sometimes helps to escape from a local minimum. In other words it is recommended to perturb all network parameters by small amounts and use it as initial condition of post-training phase to see whether the overall error decreases further.

Scoring and evaluation of network performance is done in the following way.

In training step, as mentioned before, the objective function is the average frame based weighted mean square error over all tokens. Similarly for testing the network the mean square error between network output values and their corresponding desired target values is computed over each example and over all examples in the test set. Scoring can be done based on both these errors, namely the overall frame based error or the classification error rate. In other words assuming that each output unit represents a separate hypothesis regarding the class of the input observation, error corresponding to each hypothesis is evaluated using the appropriate target function or parameters based on which the corresponding objective value is computed. The hypothesis with the lowest mean square error, is considered as the decision made by the network under the condition of prechosen class. For context feedbacks and also for higher level language processing modules it is desired to have a soft decision, therefore the two most probable classes with a confidence measure based on the ratio of the lowest error and second lowest error can also be provided at the output of the decision module.

Chapter 5

Experiment; Network Pruning and Analysis

5.1 Experiment

Results are presented for an experiment on speech data; specifically the (p,t,k) task in the context of front, middle and back vowels, taken from TIMIT database. Each example consists of the closure, burst and the following vowel. No further segmentation, normalization or time warping is performed. The data has been randomly selected from about 100 different speakers from the different dialect regions included in this data base.

INPUT: the output of 128 channels of a Cochlear model, Shamma[11,12] are used as reference input feature vectors, and only those channels in the frequency range of speech data, e.g. 60 channels are included. This representation is chosen because one of our other objectives in this experiment is to test whether this model preserves the short time features of speech spectra.

NETWORK: the recurrent network is used for simplicity, the network must have 60 inputs, which results in many connections at the first layer of the net-

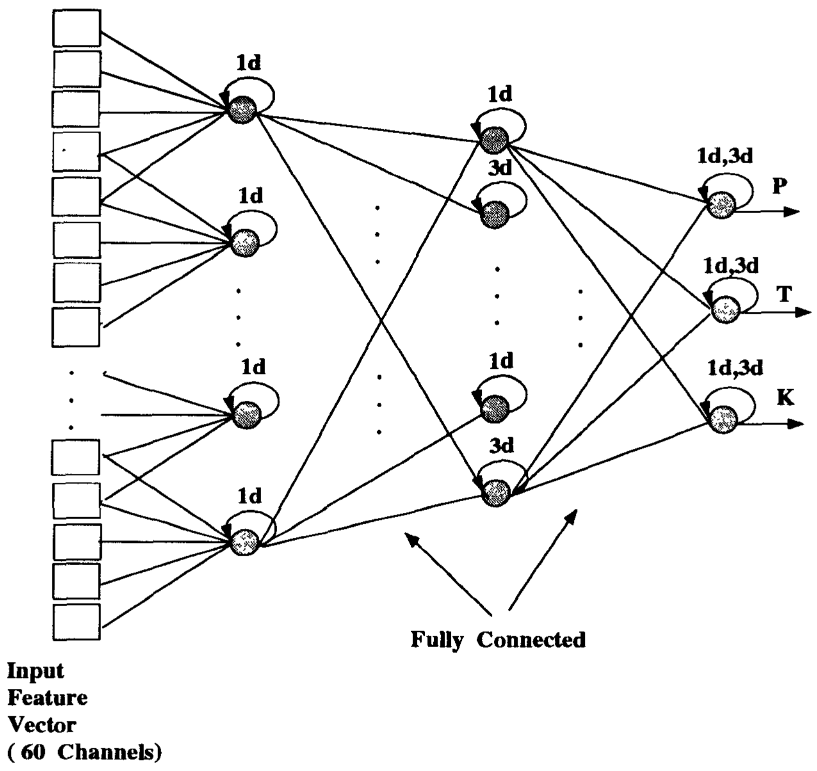


Figure 5.1: The recurrent neural network used in experiment

work. In order to reduce the size of effective inputs a kind of averaging is performed as in figure(12), such that the receptive field of each unit in the first hidden layer consists of 5 input channels with two channels overlap. It may be suggested that instead of this averaging, it is easier to consider only every other 3 input channels, but it is not a good idea because the Cochlear filters have small bandwidths with sharp transition bands, and by choosing just some channels important frequency ranges may be missed. Appropriate connection weights in this layer can be found via training on a small training set. These weights are then fixed for the rest of the training phase. Note that with this structure we can find a secondary set of input feature vectors with small dimension from original channels for each specific class of phonemes. The self-recurrency at succeeding layers provide the required temporally dynamic structure (i.e memory). Note that the recurrent links at the first hidden layer have delays of either one or three time steps to provide two scale classes following the mentioned multi-resolution ideas.

WINDOW and STEP SIZES and NETWORK MEMORY: these parameters are related to each other, in that the delay structure in the network and size of input windows and their overlap determine the overall network input span or memory.

Neither large nor small window sizes gives both time and frequency resolution. Without suitable time and frequency resolution transition parts of the CV segments will not appear clearly at the output of the cochlear model. The maximum number of frames that the network can cover in its "memory" is finite and depends on the delay structure of the network. It is therefore necessary to check that the maximum extent of the desired feature, is less than the maximum number of frames that the network can span in time. As mentioned before;

for TDNN this memory can be determined explicitly but for recurrent it is not clearly determined but can be roughly estimated. After inspecting input patterns it was determined that the typical extent of unvoiced plosives in time is about 40 to 60 msec (including transition times) so 10ms window size and 5 ms step size were chosen.

TARGET and ERROR WEIGHT FUNCTION: for each example in the training set a Gaussian target function with suitable width and timing is arbitrarily chosen i.e. for a Gaussian function:

$$trg(t) = 0.5 \pm 0.5\left(\frac{1}{(2\pi)^{1/2}\sigma} e^{\frac{-(t-m)^2}{2\sigma^2}}\right) \qquad (5.1.1)$$

For each input example file with the total duration normalized to one the parameters σ and m are chosen such that the function value is close to zero during the closure and vowel parts of the CV segment and is high during the burst and transition parts. So in equation above $+$ and $-$ is used for outputs corresponding to correct and incorrect hypothesis respectively. Experiments showed that the best performance is achieved when the peak of this target function is placed roughly after the end of burst and begining of the transition part. In particular the performance obtained in this case is better than the one obtained using a target function centered and concentrated at the middle of the burst. This confirms the fact that formant transitions in a CV segment are very important, even more than burst, for the discrimination of the stops.

A weight function as depicted in figure(13) is also used so that in training output values are not forced to rise and fall exactly like a Gaussian. In fact, what is really known about the optimal target function for classification is, it has low values during the context (e.g. preceding ang following phonemes) and a high value "somewhere" during the desired phoneme. Using this function

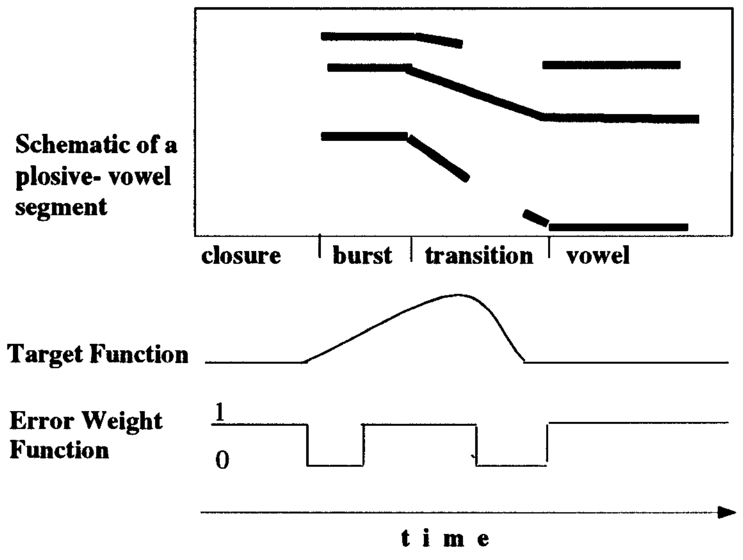


Figure 5.2: A schematic of a typical CV segment for unvoiced plosive (i.e. closure, burst, transition and vowel), and the corresponding target and error weight function

seems to improve the final performance in terms of average error and speed of convergence.

TRAINING: is performed using both BP and BFGS algorithms which we introduced before. The BFGS algorithm provided faster error convergence rate as expected. Even accelerated BP is slower that BFGS. The network simulator GRADSIM[28] was used in most of the experiments. Training set consisting of 120 examples from TIMIT database, is divided dynamically into validation and training subsets, and in order have a fast initial convergence rate small initial training subset(e.g 12 examples) is chosen, which gradually, by including more and more examples in the validation set, increases to the whole set of 120 examples.

After required average weighted mean square error is achieved the final performance is tested on a separate set of examples from other speakers. This performance either as mean square error or recognition rate, evaluates the generalization capability of the speaker independent phoneme recognizer.

For training BFGS algorithm was used which as mentioned is a nonlinear optimization method and is faster that usual back propagation. The network simulator GRADSIM[28] is used in most of these experiments. Data was divided into two sets of training and test sets. The training set also consisted of a subset for validation. The final performance is evaluated on the test set which was not used for training, and shows the average mean square error per frame and also the recognition rate.

5.2 Network Analysis

The next objective is to analyze what the network has learned and possibly simplify the network. In other words the question is first, what features from input data have been extracted by the network in its abstract internal representations, and second which input components, network units or connection links are redundant, i.e. are not actively involved in classification. This in turn leads us to the idea of network pruning. As one can see network analysis is related to pruning in the sense that in analysis we are looking for most important features and in order to find them or at least simplify our search, we have to get rid of redundancies in the system. On the other hand removing all redundancies, and using a small size network with fewer number of parameters, enables us to achieve better generalizations. Figure(14) shows qualitatively that small networks are more effective in generalizing from training examples and are less susceptible to over-training than networks with large number of free parameters.

Although in this pruning stage our main concern is to retain the overall performance, we may loose some of that. So, It is recommended to post-train the simplified network, i.e. use the pruned network parameters as an initial condition and redo training to obtain better results.

The analysis and pruning is easy when we deal with small sets of static patterns and temporally static networks, but it is not very straight forward for temporally dynamic networks with large number of time varying inputs.

Assuming that the training has been done correctly, one can approach the problem of network pruning in several ways:

TRIAL AND ERROR: remove links one at a time and each time evaluate the recognition performance measures, discard that link if the performance is

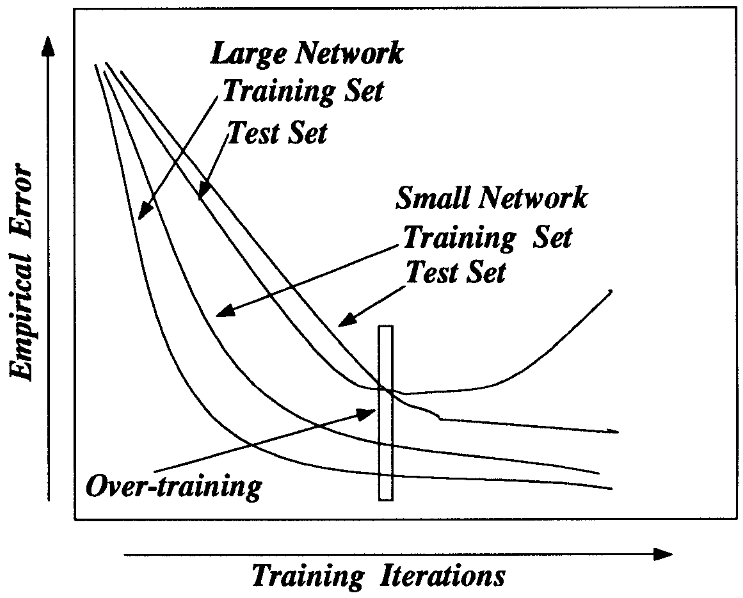


Figure 5.3: Small size networks have better generalization and are less susceptible to over-training than large networks

not affected significantly otherwise retain it.

SENSITIVITY ANALYSIS: write output vector as function of input and connection weights find the partial derivative of outputs with respect to each connection weight and do the well known sensitivity analysis. Although sensitivity analysis has been studied mostly for static feedforward networks, the idea seems to be extendable to dynamic networks as well.

Also recently some other algorithms for network pruning have been suggested that are based on information from all second order derivatives of the error function[30].

VISUAL INSPECTION: of weighted activation at the input of each neural unit. Figure(15) below shows the basic idea and figure(16) gives one example of such observation on actual data and network in the experiment above.

As we see in figure(15), w_{46} can be removed because its corresponding weighted activation $w_{46} \cdot y_4$ is small. Also w_{26} can be removed because its corresponding weighted activation as a function of time is identical to that of w_{16}. For each unit, the same set of inspection has to be made for all input patterns. Thus all idle connections and units as well as all but one of connections with the same effect on their destination can be removed [6].

5.2.1 Variance-Covariance Analysis

This method relies on the same idea of visual inspection. When we have a relatively large network with large training and test sets the visual inspection of all examples to give a general statement about important and redundant connections and input features would be very difficult. So it is better to give some structure to this analysis so that we can perform it automatically with a computer. Basically we want to discard all units that are not responsive to

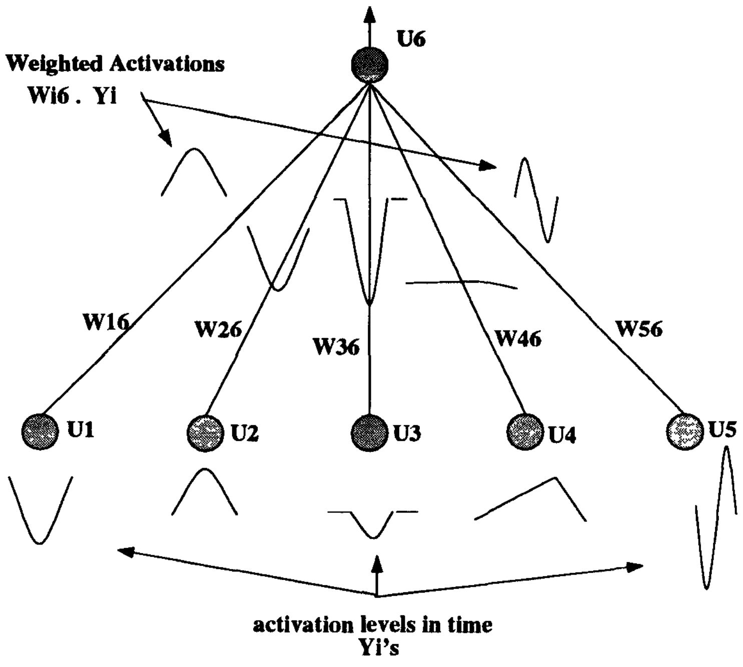


Figure 5.4: Visual inspection of weighted activations

our desired part of the input file, i.e. those units(or weighted activations that are almost constant or have very small deviation from their mean value during that time interval). It is possible to write a code to do this analysis and give a quantitative measure of importance of units and weights as well as the simplified (pruned) network. The basic idea is the following:

a) concatenate all input files in the training and/or test set into a single big file arranged in time so that for each unit in the network there is a long time waveform; figure(16) shows the idea for a small number of input files. Then mark those time intervals during which the desired input process (for instance the utterance of plosive) is located.

For a successfully trained network these time interval are where outputs depart from their rest(mean) value and only one of them goes "high" and others go to "low". These intervals should be expanded few frames to the past and future to include the dynamic effects of propagation of activations through the network delay structure. After this expansion we have a set of time intervals $[t_{fr}(i), t_{to}(i)]$ where i goes from zero to number of examples in the set.

b) Now that we know all "interesting" parts of input files happen during these time intervals; we can focus on them and discard all units/connections that do not change their status during these intervals. In order to make it computable, for each neuron's waveform we find the mean and variance in those time intervals and save them for the next step. Note that in computation of variance we just consider those mentioned time intervals and we are not interested in variations in activities far from our interested process. i.e. in our case we do not look at variations that happen far into closure or vowel part of input waveform.

c) Based on these variances we can define the following arbitrary quantitative "Importance Measure" for both links and units. Of course one may define other measures of importance.

For connection weight w_{ij}, the link from unit i to unit j:

$$Imp(w_{ij}) \equiv w_{ij}^2 \times Var(y_i) \tag{5.2.2}$$

For the i^{th} neural unit;

$$Imp(y_i) \equiv \sum_j (Imp(w_{ij}) \times Imp(y_j)) \tag{5.2.3}$$

measures can be compared within each layer, so they can be normalized for each network layer separately. So importance factors for units must be calculated recursively from the output layer toward input. In order to do that normalized importance factors are assigned to each output, depending on the purpose of the analysis.

For example, in the mentioned experiment outputs correspond to (p, t, k); by assigning 1, 0, 0, for their importance we will be able to look at important units and features to detecting t's only. Normally we assign (1, 1, 1) to consider all phonemes of this class simultaneously. Thus all connections and units for the discrimination of each of these phonemes from the other two wil be included.

Based on these relative importance measure one can tell for each unit in the network how important it is compared to other units at that level, and which of the links at its input are redundant. Thus by setting an appropriate threshold we can discard all inactive units and links and hence prune the initially fully connected network.

Also for those weighted links that are discarded we can add all of their weighted means to the bias of their destination unit.

d) Analogous to what we did in visual inspection to remove "equivalent inputs" to each unit, here we can find the cross covariance of weight activation

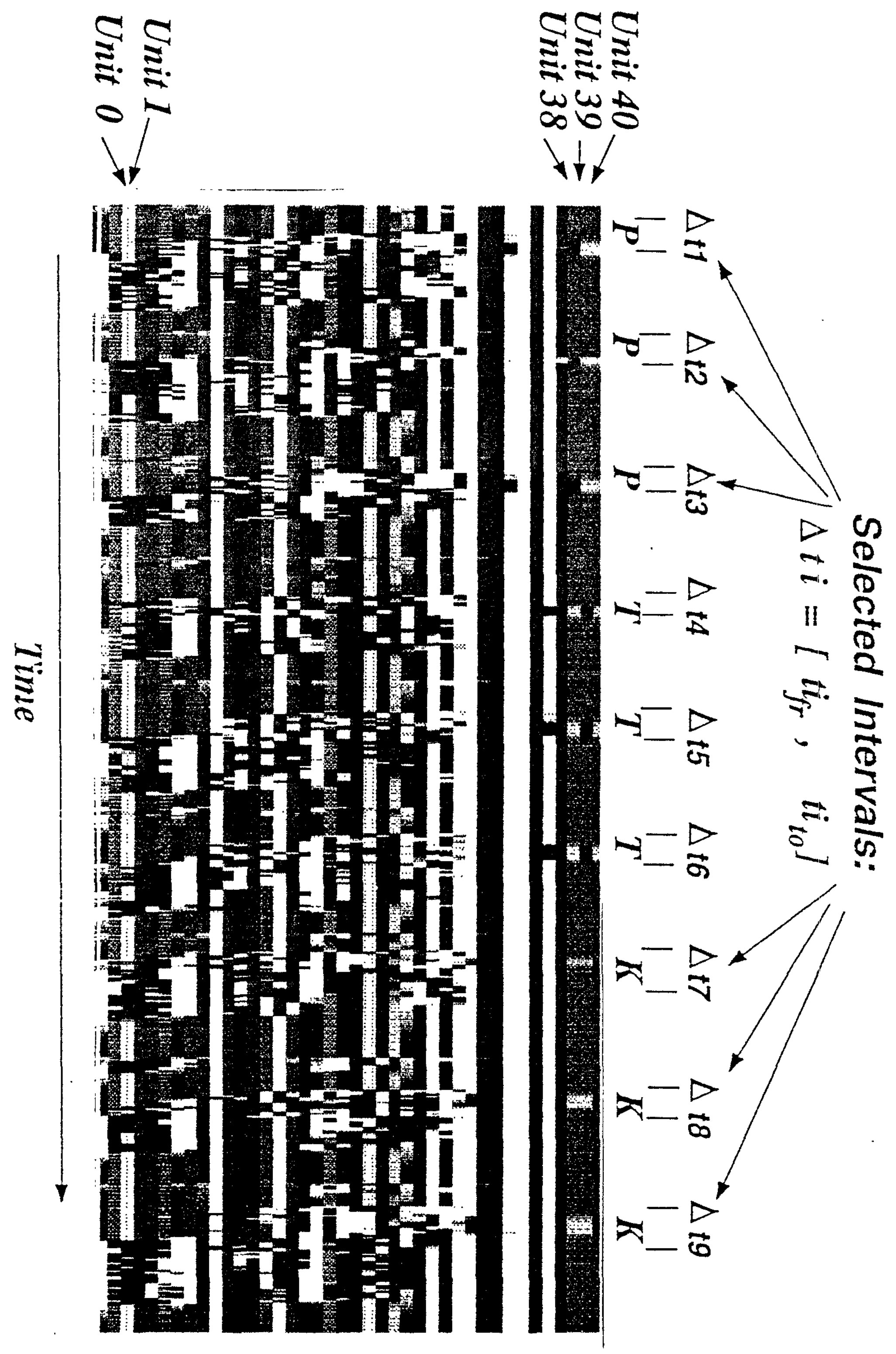


Figure 5.5: Spotting interesting intervals for variance/covariance analysis

sequences for all links with sources in the same layer. Thus large covariance function at time 0 (and may be a few frames to the past) means highly correlated inputs, i.e. one of them is redundant and can be removed, by adjusting the relative weights. The result of this computations is an ordered list of important inputs for each unit. Along with importance of units, and based on a preset threshold for pruning process network description for a simplified network can be generated.

As we mentioned before one expects to have slightly inferior performance for the resulting network. Performance can be improved by considering the pruned network as an initial network description and retrain it. The final net would typically have smaller number of links and by looking at the outputs of program one can tell which components of input feature vector have been considered and picked by network as a clue for its discrimination task.

5.2.2 Input Test Signals

In this last method of analysis there is some expectation about the features that might be extracted by the network and it is desired to check whether they are indeed picked by the network. In other words the question is whether there are some unit(s) just sensitive to that specific feature. In order to do this analysis one can generate a set of synthetic signals with structure that is an simulation of the target feature and use them as input test signals to the network and inspect their corresponding outputs in time.

In our case, we want to check whether some of the units in the network is "specifically" sensitive to formant transition slopes at the input. In order to check this several FM signals have been generated in which frequency has triangular and staircase variations in time with different rate and range of change.

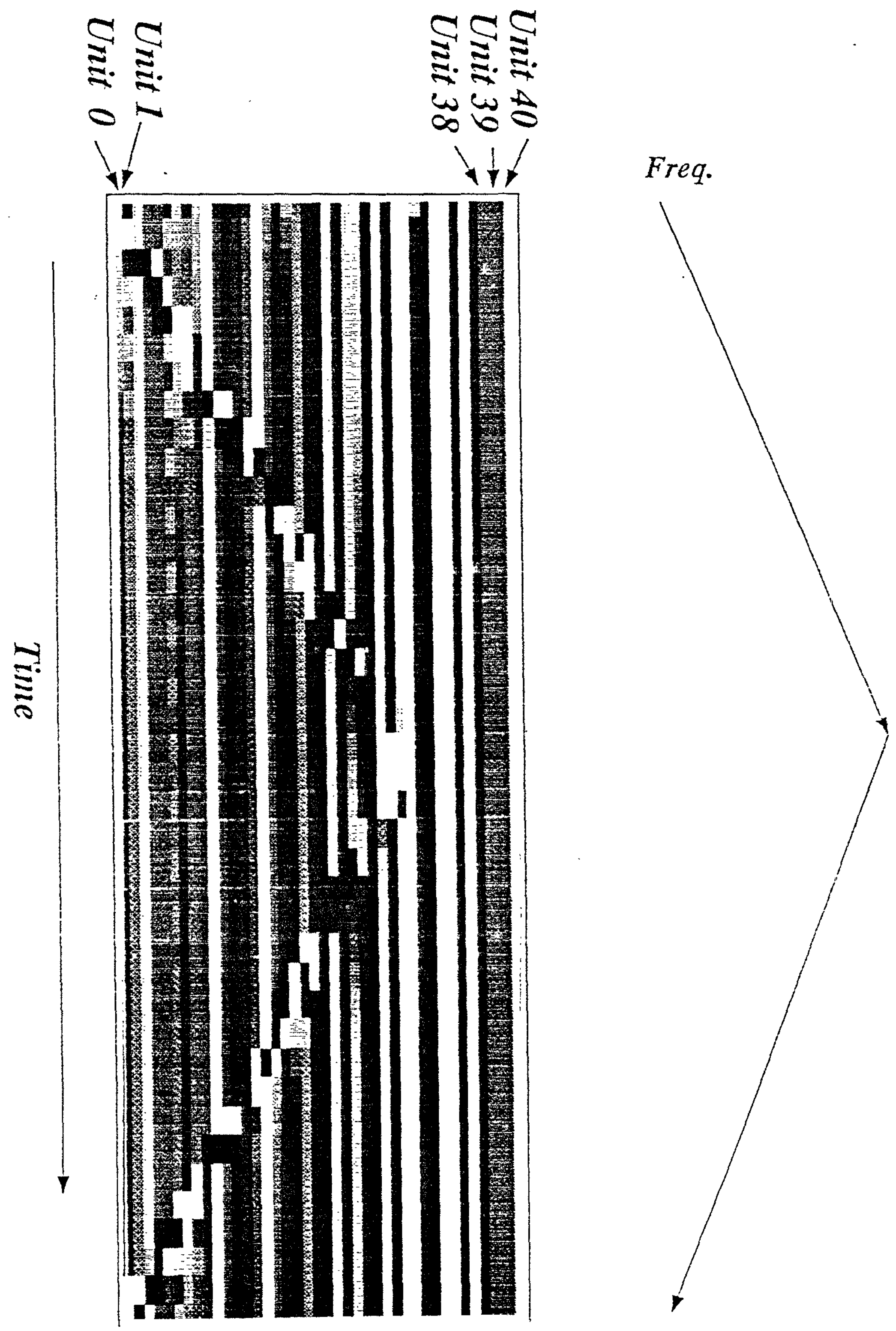


Figure 5.6: Output activation levels for an FM input signal with ramp frequency variation

We tried to make the frequency transition slopes close to what is observed in speech data. Then these synthetically generated inputs were passed through cochlear model and LIN(II). The derived signal was used as input to the trained network and figure (17) shows one of the results for a triangular FM test signal for which the slope of frequency change is in the same range as those observed in the transition parts of the tested plosive CV segments. As one can see none of the units is purely dedicated to pick up transition slopes because for such a unit activation waveforms should be antisymmetric, because of different response to upward and downward transitions. But generally the total activation patterns are not symmetric during the upward and downward transitions. This is expected because of the network's memory and dynamics of activation flow in the network. Note also that with the combination of staircase and triangular changes in input vector one can get some idea about which units are sensitive to which frequency range.

Observation: Although some results have been reported about neural networks phoneme classifiers that have hidden units specifically responsive to formant transition slopes[5], such units were not been observed in the experiments above using recurrent networks. This is because, the self-recurrent configuration of the temporal flow model is not suitable for detecting transitions (shifts) in input vector, whereas with TDNN these slopes can be detected easily. This can be shown using the following short experiment:

Try to train small networks with few binary inputs (e.g. 5-tuple input vectors) to pick up the transition slopes, i.e. direction of shifts, in the input vector. Figure(18) shows both a recurrent and a feedforward time delay single layer network to be trained, along with one of training sequences. The result of training shows that a satisfactory performance can not be achieved for the first network even after long training time, whereas for the second and third configurations convergence is quite fast.

Using the result of this experiment and also using input-output equations we see, a suitable network architecture for slope or shft detection is as in figure(18.e).

$$z[n] = X[n]^T \cdot W \tag{5.2.4}$$

$$y[n] = z[n] - z[n-1] \propto \frac{dx_t}{dt} \,|_{t=n} \tag{5.2.5}$$

These observations justify why in the suggested network none of units is specifically dedicated to pick up formant transition, since extracting just transition slopes requires co-laboration of a large number of hidden units, at least for the temporal flow model. In other words it would be very "expensive" for the network to pick this feature distinctively. Also we could expect that based on network architecture and training method a network might generate its internal representations in a very complicated way and although we know that the transition part of plosives are important for their discrimination and the previously mentioned experiment results with different target functions also confirmed that but there might be no special unit dedicated to detection of transition slopes.

One may think of including pre-trained specialized subnetworks like figure(18-e) as "fixed modules" within the hidden layers of the big network and train the remaining free weights to achieve desired performance. This kind of network design may result is efficient and yet small sized architectures and may also improve the generalization and final performance of the overall system. This approach is not a completely connectionist paradigm since the network is forced to pick some features.

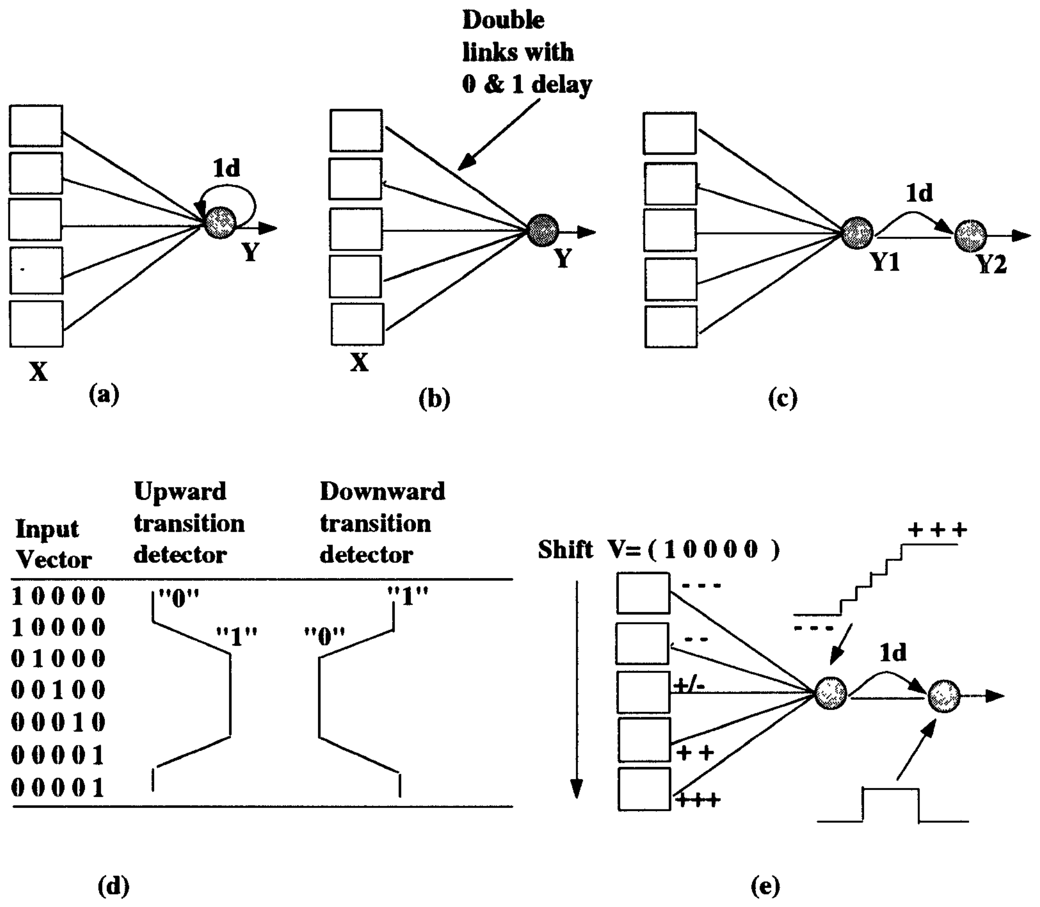


Figure 5.7: Transition slope detection (a) Self-recurrent, (b) TDNN1 ,(c) TDNN2 , (d) example of a training sequence, (e) a suitable configuration for up/downward slope detection

Chapter 6

Results and Discussions

6.1 Comments on a suitable data base for training a phoneme recognizer

Considering that isolated phonemes, without any information of the words they are contained in, are quite confusing even to humans, one can suggest the following features for a suitable database for training phoneme recognition systems.

1) Having a set of carefully uttered phonemes in different contexts for all speakers.

2) Having many examples of the same phoneme for each speaker.

3) Having an exact labeling for all phoneme segments.

4) Including some more realistic examples extracted from continuous speech.

Considering the way we, as humans, learn to recognize speech sounds, in training a speaker independent phoneme recognizer it makes sense to start with very clearly uttered examples from all speakers, and perform training on that, and when satisfactory results obtained on this set, we can add few more realistic segments extracted from continuous speech as additional inputs to check and improve the generalization of the system. This approach becomes more important specially when we want to analyze the internal abstract representation of the network.

TIMIT database is one of the standard data bases that is used for speech recognition, and mostly speaker independent word recognition. It consists of a large number of examples of continuous speech files from many speakers. But based on above observation, TIMIT database, like some others, is not an ideal database for some phoneme recognition tasks It also has some labeling problems. For example in most cases it includes transition parts of the plosive-vowel segments, under vowel label so for choosing a suitable target function for input files one has to look at them one by one. In many cases phonemes are not uttered carefully so although they might be understandable at word level they are quite unclear at subword and phoneme levels. Again as an example in many cases K-waveforms when played, sound exactly like T or P and vice versa. Also the transitions parts are not consistently observed in randomly selected examples from this data set. So for the sake of analysis it is much better to deal with smaller group of speakers but with many examples of carefully uttered phonemes for each of them. With large variety of speech quality, analysis of extracted acoustic feature would be more difficult and also less accurate. In fact most analysis results that have been reported are based on small databases from single speakers.

6.2 Results and Discussion

Having explained the suggested approach and its plausibility, the results of experiment on speech data are presented. The convergence of averaged weighted error using BFGS algorithm was fairly fast for the simple recurrent network

phonemes	/p/	/t/	/k/
/p/	0.98	0.02	0.00
/t/	0.00	1.00	0.00
/k/	0.00	0.00	1.00
MSE error	0.001		

Table 6.1: The confusion matrix for the training set.

phonemes	/p/	/t/	/k/
/p/	0.97	0.01	0.02
/t/	0.10	0.85	0.05
/k/	0.02	0.02	0.96
MSE error	0.003		

Table 6.2: The confusion matrix for the test set.

whereas for the same task the TDNN mentioned above, convergence takes much more time and the performance is almost the same. For a TFM network of the same size mean square error converges to 0.0035 on training set and to 0.006 on the test set. For the suggested modified TFM the final normalized error per frame was about 0.001, and the average recognition score was about 98% on the training set and about 88% on the test set (Tables 1,2). When we presented the examples, which had errors, to human listeners, we realized that in 80% of cases the confusion made by the network was the same as human subjects. Thus, further improvement of recognition would be possible by using higher level (e.g. word, sentence) language processing. Examples of scoring results on the test set are given in figure(18) and more examples are provided in appendix. Note that in this list hypothesis 0, 1 and 2 refer to /p/ , /t/ and /k/ respectively. The CORRECT or ERROR decisions are based on pre-defined labels in the database. Also note that for incorrect decision cases where score ratios are greater than one these ratios are actually very close to the threshold "one".

In general it is difficult to compare recognition results obtained by researchers across laboratories, because of different databases, recognition conditions, signal representations, speaking rates and speaker dependencies. There are many factors involved, for example in phoneme recognition case, very high recognition rates (more than 90 percent) can be achieved if one uses isolated utterances rather than segments obtained from continuous speech. Also speaker normalization, noise removal and lots of other pre/post processing could affect the results significantly. In addition to all of these the complexity of the system and speed are other major factors. In other words the comparison of different recognition schemes is possible if the same input feature vectors are used and similarly in order to decide between different speech representations, one should test them

```
GRADSIM 3.0x Connectionist Simulator

        Optimizer ../../score compiled for NOT COMPILED

Optimizing network: net1.60.all

        Experiment: exp.tst.60



Example /vp/ti.7.IIf.bin  Hypothesis 0    Score 0.009660

Example /vp/ti.7.IIf.bin  Hypothesis 1    Score 0.009640

Example /vp/ti.7.IIf.bin  Hypothesis 2    Score 0.009426

RATIO 1.024912  ERROR


Example /vp/pa.2.IIf.bin  Hypothesis 0    Score 0.012657

Example /vp/pa.2.IIf.bin  Hypothesis 1    Score 0.000925

Example /vp/pa.2.IIf.bin  Hypothesis 2    Score 0.012569

RATIO 0.073601  CORRECT


Example /vp/ku.0.IIf.bin  Hypothesis 0    Score 0.007440

Example /vp/ku.0.IIf.bin  Hypothesis 1    Score 0.007219

Example /vp/ku.0.IIf.bin  Hypothesis 2    Score 0.006938

RATIO 0.961073  CORRECT
```

Figure 6.1: Some of scoring results on test set, Hypothesis 0,1 and 2 refer to phonemes T, P and K respectively

using the same shceme. This comparison has not been given so far because of different practical difficulties. Just for the sake of a rough comparison, experiments performed by Zue et. al. [31] Waibel [19] and Robinson and Fallside [32] on TIMIT database, can be mentioned, where they have achieved recognition rates between 50 and 75 percent on different tasks, using parts or all of examples in the database. Considering the fact that we have extracted our data from continuous speech of many speakers without any normalization and time warping the results are acceptable and comparable to previous result reported with larger networks. And this confirms the validity of our arguments on one hand and on the other hand it shows that the auditory model used in this experiment preserves important features of short time frequency spectra of speech.

Chapter 7

Appendix

7.1 Some of scoring results for training set

Hypothesis 0 , 1 , 2 correspond to phonemes T, P, K respectively.

```
GRADSIM 3.0x Connectionist Simulator
        Optimizer ../../score compiled for NOT COMPILED
Optimizing network: net1.60.all
        Experiment: exp.trn.60

Example /vp/ta.1.IIf.bin          Hypothesis 0     Score 0.000075
Example /vp/ta.1.IIf.bin          Hypothesis 1     Score 0.022253
Example /vp/ta.1.IIf.bin          Hypothesis 2     Score 0.022302
RATIO 0.003386  CORRECT
Example /vp/ta.4.IIf.bin          Hypothesis 0     Score 0.000473
Example /vp/ta.4.IIf.bin          Hypothesis 1     Score 0.023471

Example /vp/ta.4.IIf.bin          Hypothesis 2     Score 0.023526

RATIO 0.020167  CORRECT

Example /vp/ta.6.IIf.bin          Hypothesis 0     Score 0.001773

Example /vp/ta.6.IIf.bin          Hypothesis 1     Score 0.015120

Example /vp/ta.6.IIf.bin          Hypothesis 2     Score 0.015027

RATIO 0.117964  CORRECT

Example /vp/ti.2.IIf.bin          Hypothesis 0     Score 0.000382

Example /vp/ti.2.IIf.bin          Hypothesis 1     Score 0.019217

Example /vp/ti.2.IIf.bin          Hypothesis 2     Score 0.019037

RATIO 0.020055  CORRECT

Example /vp/ti.4.IIf.bin          Hypothesis 0     Score 0.000212

Example /vp/ti.4.IIf.bin          Hypothesis 1     Score 0.018453

Example /vp/ti.4.IIf.bin          Hypothesis 2     Score 0.018673

RATIO 0.011486  CORRECT

Example /vp/ti.6.IIf.bin          Hypothesis 0     Score 0.000516

Example /vp/ti.6.IIf.bin          Hypothesis 1     Score 0.021508

Example /vp/ti.6.IIf.bin          Hypothesis 2     Score 0.021555

RATIO 0.023969  CORRECT

Example /vp/tu.1.IIf.bin          Hypothesis 0     Score 0.001636

Example /vp/tu.1.IIf.bin          Hypothesis 1     Score 0.019157

Example /vp/tu.1.IIf.bin          Hypothesis 2     Score 0.019173

RATIO 0.085402  CORRECT

Example /vp/tu.3.IIf.bin          Hypothesis 0     Score 0.000189

Example /vp/tu.3.IIf.bin          Hypothesis 1     Score 0.016470

Example /vp/tu.3.IIf.bin          Hypothesis 2     Score 0.016510
```

```
RATIO 0.011461  CORRECT
Example /vp/tw.0.IIf.bin        Hypothesis 0    Score 0.000104
Example /vp/tw.0.IIf.bin        Hypothesis 1    Score 0.015386
Example /vp/tw.0.IIf.bin        Hypothesis 2    Score 0.015794
RATIO 0.006773  CORRECT
Example /vp/tw.7.IIf.bin        Hypothesis 0    Score 0.000086
Example /vp/tw.7.IIf.bin        Hypothesis 1    Score 0.024384
Example /vp/tw.7.IIf.bin        Hypothesis 2    Score 0.024433
RATIO 0.003526  CORRECT
Example /vp/pa.1.IIf.bin        Hypothesis 0    Score 0.013572
Example /vp/pa.1.IIf.bin        Hypothesis 1    Score 0.001944
Example /vp/pa.1.IIf.bin        Hypothesis 2    Score 0.013480
RATIO 0.144201  CORRECT
Example /vp/pa.3.IIf.bin        Hypothesis 0    Score 0.026333
Example /vp/pa.3.IIf.bin        Hypothesis 1    Score 0.000441
Example /vp/pa.3.IIf.bin        Hypothesis 2    Score 0.026271
RATIO 0.016796  CORRECT
Example /vp/pa.8.IIf.bin        Hypothesis 0    Score 0.020409
Example /vp/pa.8.IIf.bin        Hypothesis 1    Score 0.000779
Example /vp/pa.8.IIf.bin        Hypothesis 2    Score 0.020587
RATIO 0.038175  CORRECT
Example /vp/pa.12.IIf.bin       Hypothesis 0    Score 0.016174
Example /vp/pa.12.IIf.bin       Hypothesis 1    Score 0.000712
Example /vp/pa.12.IIf.bin       Hypothesis 2    Score 0.016117
RATIO 0.044199  CORRECT
Example /vp/pi.7.IIf.bin        Hypothesis 0    Score 0.020528
Example /vp/pi.7.IIf.bin          Hypothesis 1     Score 0.000491
Example /vp/pi.7.IIf.bin          Hypothesis 2     Score 0.020596
RATIO 0.023910  CORRECT
Example /vp/pi.10.IIf.bin         Hypothesis 0     Score 0.013898
Example /vp/pi.10.IIf.bin         Hypothesis 1     Score 0.000380
Example /vp/pi.10.IIf.bin         Hypothesis 2     Score 0.013812
RATIO 0.027526  CORRECT
Example /vp/pi.18.IIf.bin         Hypothesis 0     Score 0.005043
Example /vp/pi.18.IIf.bin         Hypothesis 1     Score 0.004977
Example /vp/pi.18.IIf.bin         Hypothesis 2     Score 0.004823
RATIO 1.031918  ERROR
Example /vp/pu.2.IIf.bin          Hypothesis 0     Score 0.020440
Example /vp/pu.2.IIf.bin          Hypothesis 1     Score 0.000464
Example /vp/pu.2.IIf.bin          Hypothesis 2     Score 0.020346
RATIO 0.022806  CORRECT
Example /vp/pw.1.IIf.bin          Hypothesis 0     Score 0.019155
Example /vp/pw.1.IIf.bin          Hypothesis 1     Score 0.000688
Example /vp/pw.1.IIf.bin          Hypothesis 2     Score 0.019050
RATIO 0.036108  CORRECT
Example /vp/ka.2.IIf.bin          Hypothesis 0     Score 0.012801
Example /vp/ka.2.IIf.bin          Hypothesis 1     Score 0.012651
Example /vp/ka.2.IIf.bin          Hypothesis 2     Score 0.008003
RATIO 0.632643  CORRECT
Example /vp/ka.22.IIf.bin         Hypothesis 0     Score 0.013877
Example /vp/ka.22.IIf.bin         Hypothesis 1     Score 0.013761
Example /vp/ka.22.IIf.bin         Hypothesis 2     Score 0.006389
```

```
RATIO 0.464331  CORRECT
Example /vp/ka.24.IIf.bin        Hypothesis 0    Score 0.021523
Example /vp/ka.24.IIf.bin        Hypothesis 1    Score 0.021473
Example /vp/ka.24.IIf.bin        Hypothesis 2    Score 0.000340
RATIO 0.015851  CORRECT
Example /vp/ka.26.IIf.bin        Hypothesis 0    Score 0.015607
Example /vp/ka.26.IIf.bin        Hypothesis 1    Score 0.015620
Example /vp/ka.26.IIf.bin        Hypothesis 2    Score 0.000137
RATIO 0.008770  CORRECT
Example /vp/ki.0.IIf.bin         Hypothesis 0    Score 0.007525
Example /vp/ki.0.IIf.bin         Hypothesis 1    Score 0.007436
Example /vp/ki.0.IIf.bin         Hypothesis 2    Score 0.007263
RATIO 0.976691  CORRECT
Example /vp/ki.8.IIf.bin         Hypothesis 0    Score 0.006996
Example /vp/ki.8.IIf.bin         Hypothesis 1    Score 0.007002
Example /vp/ki.8.IIf.bin         Hypothesis 2    Score 0.006840
RATIO 0.977698  CORRECT
Example /vp/ku.2.IIf.bin         Hypothesis 0    Score 0.021404
Example /vp/ku.2.IIf.bin         Hypothesis 1    Score 0.021706
Example /vp/ku.2.IIf.bin         Hypothesis 2    Score 0.000222
RATIO 0.010383  CORRECT
Example /vp/ku.4.IIf.bin         Hypothesis 0    Score 0.018570
Example /vp/ku.4.IIf.bin         Hypothesis 1    Score 0.018833
Example /vp/ku.4.IIf.bin         Hypothesis 2    Score 0.000970
RATIO 0.052231  CORRECT
Example /vp/ku.7.IIf.bin         Hypothesis 0    Score 0.017232
Example /vp/ku.7.IIf.bin         Hypothesis 1     Score 0.017034
Example /vp/ku.7.IIf.bin         Hypothesis 2     Score 0.000161
RATIO 0.009462  CORRECT
Example /vp/kw.1.IIf.bin         Hypothesis 0     Score 0.020990
Example /vp/kw.1.IIf.bin         Hypothesis 1     Score 0.021051
Example /vp/kw.1.IIf.bin         Hypothesis 2     Score 0.000166
RATIO 0.007913  CORRECT
```

7.2 Some of scoring results for test set

Hypothesis 0 , 1 , 2 correspond to phonemes T, P, K respectively.

```
GRADSIM 3.0x Connectionist Simulator
        Optimizer ../../score compiled for NOT COMPILED
Optimizing network: net1.60.all
        Experiment: exp.tst.60


Example /vp/ta.2.IIf.bin  Hypothesis 0    Score 0.000416
Example /vp/ta.2.IIf.bin  Hypothesis 1    Score 0.016294
Example /vp/ta.2.IIf.bin  Hypothesis 2    Score 0.017646
```

```
RATIO 0.025514  CORRECT

Example /vp/ta.5.IIf.bin  Hypothesis 0    Score 0.000586

Example /vp/ta.5.IIf.bin  Hypothesis 1    Score 0.018462

Example /vp/ta.5.IIf.bin  Hypothesis 2    Score 0.018400

RATIO 0.031859  CORRECT

Example /vp/ta.8.IIf.bin  Hypothesis 0    Score 0.000954

Example /vp/ta.8.IIf.bin  Hypothesis 1    Score 0.018598

Example /vp/ta.8.IIf.bin  Hypothesis 2    Score 0.019285

RATIO 0.051319  CORRECT

Example /vp/ta.10.IIf.bin Hypothesis 0    Score 0.000863

Example /vp/ta.10.IIf.bin Hypothesis 1    Score 0.009833

Example /vp/ta.10.IIf.bin Hypothesis 2    Score 0.010151

RATIO 0.087750  CORRECT

Example /vp/ti.5.IIf.bin  Hypothesis 0    Score 0.000275

Example /vp/ti.5.IIf.bin  Hypothesis 1    Score 0.015205

Example /vp/ti.5.IIf.bin  Hypothesis 2    Score 0.015783

RATIO 0.018056  CORRECT

Example /vp/ti.7.IIf.bin  Hypothesis 0    Score 0.009660

Example /vp/ti.7.IIf.bin  Hypothesis 1    Score 0.009640

Example /vp/ti.7.IIf.bin  Hypothesis 2    Score 0.009426

RATIO 1.024912  ERROR

Example /vp/ti.15.IIf.bin Hypothesis 0    Score 0.000258

Example /vp/ti.15.IIf.bin Hypothesis 1    Score 0.011066

Example /vp/ti.15.IIf.bin Hypothesis 2    Score 0.011450

RATIO 0.023348  CORRECT

Example /vp/tu.0.IIf.bin  Hypothesis 0    Score 0.000755
Example /vp/tu.0.IIf.bin  Hypothesis 1    Score 0.015135

Example /vp/tu.0.IIf.bin  Hypothesis 2    Score 0.015241

RATIO 0.049884  CORRECT

Example /vp/tu.2.IIf.bin  Hypothesis 0    Score 0.002097

Example /vp/tu.2.IIf.bin  Hypothesis 1    Score 0.025454

Example /vp/tu.2.IIf.bin  Hypothesis 2    Score 0.023734

RATIO 0.088363  CORRECT

Example /vp/tu.6.IIf.bin  Hypothesis 0    Score 0.006705

Example /vp/tu.6.IIf.bin  Hypothesis 1    Score 0.006559

Example /vp/tu.6.IIf.bin  Hypothesis 2    Score 0.006194

RATIO 1.082371  ERROR

Example /vp/tw.2.IIf.bin  Hypothesis 0    Score 0.001367

Example /vp/tw.2.IIf.bin  Hypothesis 1    Score 0.019993

Example /vp/tw.2.IIf.bin  Hypothesis 2    Score 0.019921

RATIO 0.068642  CORRECT

Example /vp/tw.6.IIf.bin  Hypothesis 0    Score 0.007669

Example /vp/tw.6.IIf.bin  Hypothesis 1    Score 0.007497

Example /vp/tw.6.IIf.bin  Hypothesis 2    Score 0.006928

RATIO 1.106883  ERROR

Example /vp/tw.17.IIf.bin Hypothesis 0    Score 0.001247

Example /vp/tw.17.IIf.bin Hypothesis 1    Score 0.025173

Example /vp/tw.17.IIf.bin Hypothesis 2    Score 0.025933

RATIO 0.049553  CORRECT

Example /vp/tw.21.IIf.bin Hypothesis 0    Score 0.002061

Example /vp/tw.21.IIf.bin Hypothesis 1    Score 0.017677

Example /vp/tw.21.IIf.bin Hypothesis 2    Score 0.018101
```

```
RATIO 0.116603  CORRECT

Example /vp/pa.5.IIf.bin  Hypothesis 0    Score 0.005812

Example /vp/pa.5.IIf.bin  Hypothesis 1    Score 0.005687

Example /vp/pa.5.IIf.bin  Hypothesis 2    Score 0.005447

RATIO 1.044033  ERROR

Example /vp/pa.10.IIf.bin Hypothesis 0    Score 0.008071

Example /vp/pa.10.IIf.bin Hypothesis 1    Score 0.004111

Example /vp/pa.10.IIf.bin Hypothesis 2    Score 0.007822

RATIO 0.525504  CORRECT

Example /vp/pa.14.IIf.bin Hypothesis 0    Score 0.020438

Example /vp/pa.14.IIf.bin Hypothesis 1    Score 0.003240

Example /vp/pa.14.IIf.bin Hypothesis 2    Score 0.018694

RATIO 0.173306  CORRECT

Example /vp/pa.19.IIf.bin Hypothesis 0    Score 0.017248

Example /vp/pa.19.IIf.bin Hypothesis 1    Score 0.000479

Example /vp/pa.19.IIf.bin Hypothesis 2    Score 0.017282

RATIO 0.027770  CORRECT

Example /vp/pa.22.IIf.bin Hypothesis 0    Score 0.009539

Example /vp/pa.22.IIf.bin Hypothesis 1    Score 0.004038

Example /vp/pa.22.IIf.bin Hypothesis 2    Score 0.009073

RATIO 0.445069  CORRECT

Example /vp/pa.25.IIf.bin Hypothesis 0    Score 0.021402

Example /vp/pa.25.IIf.bin Hypothesis 1    Score 0.005897

Example /vp/pa.25.IIf.bin Hypothesis 2    Score 0.018861

RATIO 0.312674  CORRECT

Example /vp/pi.6.IIf.bin  Hypothesis 0    Score 0.008013
Example /vp/pi.6.IIf.bin  Hypothesis 1    Score 0.007912

Example /vp/pi.6.IIf.bin  Hypothesis 2    Score 0.007543

RATIO 1.048896  ERROR

Example /vp/pi.8.IIf.bin  Hypothesis 0    Score 0.008025

Example /vp/pi.8.IIf.bin  Hypothesis 1    Score 0.002403

Example /vp/pi.8.IIf.bin  Hypothesis 2    Score 0.007793

RATIO 0.308380  CORRECT

Example /vp/pi.17.IIf.bin Hypothesis 0    Score 0.010675

Example /vp/pi.17.IIf.bin Hypothesis 1    Score 0.003538

Example /vp/pi.17.IIf.bin Hypothesis 2    Score 0.010541

RATIO 0.335662  CORRECT

Example /vp/pi.19.IIf.bin Hypothesis 0    Score 0.010994

Example /vp/pi.19.IIf.bin Hypothesis 1    Score 0.005418

Example /vp/pi.19.IIf.bin Hypothesis 2    Score 0.010682

RATIO 0.507179  CORRECT

Example /vp/pu.3.IIf.bin  Hypothesis 0    Score 0.012029

Example /vp/pu.3.IIf.bin  Hypothesis 1    Score 0.002549

Example /vp/pu.3.IIf.bin  Hypothesis 2    Score 0.011889

RATIO 0.214439  CORRECT

Example /vp/pu.5.IIf.bin  Hypothesis 0    Score 0.004381

Example /vp/pu.5.IIf.bin  Hypothesis 1    Score 0.004303

Example /vp/pu.5.IIf.bin  Hypothesis 2    Score 0.004123

RATIO 1.043729  ERROR

Example /vp/pw.0.IIf.bin  Hypothesis 0    Score 0.009867

Example /vp/pw.0.IIf.bin  Hypothesis 1    Score 0.012994

Example /vp/pw.0.IIf.bin  Hypothesis 2    Score 0.004696
```

```
RATIO 2.766700  ERROR
Example /vp/ka.1.IIf.bin  Hypothesis 0    Score 0.010001
Example /vp/ka.1.IIf.bin  Hypothesis 1    Score 0.010196
Example /vp/ka.1.IIf.bin  Hypothesis 2    Score 0.008006
RATIO 0.800507  CORRECT
Example /vp/ka.4.IIf.bin  Hypothesis 0    Score 0.024100
Example /vp/ka.4.IIf.bin  Hypothesis 1    Score 0.024417
Example /vp/ka.4.IIf.bin  Hypothesis 2    Score 0.004462
RATIO 0.185126  CORRECT
Example /vp/ka.9.IIf.bin  Hypothesis 0    Score 0.018690
Example /vp/ka.9.IIf.bin  Hypothesis 1    Score 0.018261
Example /vp/ka.9.IIf.bin  Hypothesis 2    Score 0.000935
RATIO 0.051192  CORRECT
Example /vp/ka.11.IIf.bin Hypothesis 0    Score 0.006037
Example /vp/ka.11.IIf.bin Hypothesis 1    Score 0.005927
Example /vp/ka.11.IIf.bin Hypothesis 2    Score 0.005677
RATIO 0.957839  CORRECT
Example /vp/ka.25.IIf.bin Hypothesis 0    Score 0.020515
Example /vp/ka.25.IIf.bin Hypothesis 1    Score 0.019587
Example /vp/ka.25.IIf.bin Hypothesis 2    Score 0.000437
RATIO 0.022334  CORRECT
Example /vp/ka.27.IIf.bin Hypothesis 0    Score 0.015253
Example /vp/ka.27.IIf.bin Hypothesis 1    Score 0.023140
Example /vp/ka.27.IIf.bin Hypothesis 2    Score 0.010754
RATIO 0.705051  CORRECT
Example /vp/ka.29.IIf.bin Hypothesis 0    Score 0.016071
Example /vp/ka.29.IIf.bin Hypothesis 1     Score 0.015168

Example /vp/ka.29.IIf.bin Hypothesis 2     Score 0.004787

RATIO 0.315629  CORRECT

Example /vp/ka.35.IIf.bin Hypothesis 0     Score 0.015242

Example /vp/ka.35.IIf.bin Hypothesis 1     Score 0.014413

Example /vp/ka.35.IIf.bin Hypothesis 2     Score 0.003234

RATIO 0.224379  CORRECT

Example /vp/ki.5.IIf.bin  Hypothesis 0     Score 0.006838

Example /vp/ki.5.IIf.bin  Hypothesis 1     Score 0.006659

Example /vp/ki.5.IIf.bin  Hypothesis 2     Score 0.006370

RATIO 0.956526  CORRECT

Example /vp/ku.5.IIf.bin  Hypothesis 0     Score 0.015640

Example /vp/ku.5.IIf.bin  Hypothesis 1     Score 0.015249

Example /vp/ku.5.IIf.bin  Hypothesis 2     Score 0.007355

RATIO 0.482327  CORRECT

Example /vp/ku.8.IIf.bin  Hypothesis 0     Score 0.009926

Example /vp/ku.8.IIf.bin  Hypothesis 1     Score 0.009669

Example /vp/ku.8.IIf.bin  Hypothesis 2     Score 0.009248

RATIO 0.956431  CORRECT

Example /vp/kw.1.IIf.bin  Hypothesis 0     Score 0.020892

Example /vp/kw.1.IIf.bin  Hypothesis 1     Score 0.020008

Example /vp/kw.1.IIf.bin  Hypothesis 2     Score 0.000303

RATIO 0.015132  CORRECT
```

Chapter 8

Bibliography

[1] W. Huang and R. Lippmann, " Neural nets and traditional Classifiers", Neural Information Processing Systems, D. Anderson ed., New York, Amc. Inst. of Phys., 1988 pp. 387-396.

[2] K. Wang,"Neural networks that recognize phonemes by their acoustic features " Masters Thesis Report,MS91-1 SRC, University of Maryland at College Park, 1989.

[3] J.L. Elman and D. Zipser," Learning the hidden structure of speech", Journal of Acoustical Society of America, 1988, 83, pp. 1615-1626.

[4] M. Niranjan and F. Fallside," Neural networks and Radial Basis Functions in classifying static speech patterns", Cambridge Univ. Eng. Dept. Tech. Rep. CUED/F-INFENG/TR22 1988.

[5] Waibel, A., Hanazawa, T., Hinton, G., Shikano K., and Lang, K.,"phoneme Recognition Using Time Delay Neural Networks." IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. S, pp.107-110, April, 1988.

[6] Watrous, R.L.,"Phoneme discrimination using connectionist networks." Journ. of Acoust. Soc. of Am. 87(40) pp. 1753-1772, April 1990.

[7] R.L. Watrous and l. Shastri," Learning phonemic features using connec-

tionist networks: an experiment in speech recognition", Proc. ICNN 1987 San Diego.

[8] R.L. Watrous, B. Ladendorf, G. Kuhn,"Complete gradient optimization of a recurrent network applied to /b/,/d/,/g/ discrimination" Journ. of Acoust. Soc. of Am. 87(3) pp. 1301-1309, March 1990. [9] J. L. Elman,"Finding Structure in time", Cognitive Science, Vol. 14, pp. 179-211, 1990.

[10] Y. Bengio and R. De Mori,"Recurrent networks with Radial Basis Functions for speech recognition", Neural Networks for Computing, Snowbird, Utah, 1990.

[11] Shamma, A. S.,"Speech processing in the auditory system II: lateral inhibition and the central processing of speech evoked activity in the auditory nerve",Journ. of Acoust. Soc. of Am. 78(5) pp. 1622-1632, November 1985.

[12] S. Shamma,"Spatial and temporal processing in central auditory networks",Methods in Neural Modeling, Kock and Segev (editors), MIT Press, 1990. pp. 247-289.

[13] M.D. Riley,"Speech time frequency representation", Kluwer Academic Publishers 1991.

[14] D.P. Morgan and C.L. Scofield," Neural networks for speech processing", Kluwer Academic Publishers 1991.

[15] Y. Bengio, P. Cosi, R. Cardin and R. De Mori,"Use of multi-layered networks for coding speech with phonetic features",in Adv. in Neural Info. Proc. Sys. 1, D.S. Touretzky ed.,1989, Morgan Kaufmann Publishers, pp. 224-231.

[16] Y. Bengio, R. Cardin and R. De Mori,"Speaker independent speech recognition with neural networks and speech knowledge",in Adv. in Neural Info. Proc. Sys. 2, D.S. Touretzky ed.,1990, Morgan Kaufmann Publishers.

[17] X.D. Huang, Y. Akiri, M.A. Jack,"Hidden Markov Models for speech recognition", Edinburgh University Press 1990.

[18] A. Waibel, H. Sawai and K. Shikano. "Modularity and scaling in large phonemic neural networks", IEEE Trans. on Acoustics,Speech and Signal Processing, Vol. 37, No. 12 , Dec. 1989, pp. 1888-1898.

[19] N. Hataoka and A. Waibel,"Speaker independent phoneme recognition on TIMIT database using integrated time delay neural networks(TDNN's)," Proc. ICNN 1990, Vol.1 , pp. 57-62.

[20] D. Rumelhart and J. McCelland,"Parallel Distributed Processing", Vol. 1, Cambridge MA, MIT press 1988, pp. 322-328.

[21] E.B. Baum and D. Haussler,"What size net gives valid generalization?", Neural Computation 1,1989, pp. 151-160.

[22] Stork, D.G., and Levine, E.,"Neural network lip reading system for improved speech recognition" Proc. IJCNN 1992,Baltimore, Vol. 2, pp. 285-295.

[23] Y. Konog and N. Morgan,"GDNN a gender dependent neural network for continuous speech recognition", Proc. IJCNN 1992, Baltimore, Vol. 2, pp. 332-337.

[24] R.D. Janssen, M. Fanty and R. Cole, "Speaker-Independent phonetic classification in continuous English letters", IJCNN 91, Vol.2 pp. 801-808.

[25] C.A. Kamm and S. Singal,"Effect of neural network input span on phoneme classification", IJCNN 90, Vol. 1,pp. 195-200.

[26] L.Y. Pratt and C.A. Kamm,"Improving a phoneme classification neural network through problem decomposition", IJCNN 91, pp. 821-826.

[27] A. Hirai and A. Waibel,"Phoneme-based word recognition by neural network, a step toward large vocabulary recognition", IJCNN 90, Vol. 3, pp.

671-676.

[28] R.L. Watrous, "GRADSIM: A connectionist network simulator using gradient optimization techniques. Technical Report MS-CIS-88-16, University of Pennsylvania, March 1988.

[29] Fletcher, R.,"Practical methods of Optimization", Vol. 1 , John Wiley and Sons, 2nd ed. 1987.

[30] B. Hassibi, D. G. Stork and G.J. Wolff,"Optimal Brain Surgeon and general network pruning", ICNN 93, Vol. 1, pp. 293-299.

[31] V. Zue, J. Glass, M. Philips and S. Seneff,"acoustic segmentation and phonetic classification in SUMMIT system", Proc. ICASSP 1989, pp. 389-392.

[32] T. Robinson and F. Fallside," Phoneme recognition from the TIMIT database using recurrent error propagation networks", CUED/F-INFENG/TR.42.