Speech Recognition
5. SPEECH RECOGNITION

Automated continuous speech recognition (ACSR) in healthcare industry demands highly accurate and robust recognition software. Despite vendor claims, current implementations are suboptimal, leading to poor accuracy, and time and money wasted on proofreading. Thus, other methods must be considered for increasing the reliability and performance of ACSR before it becomes a viable alternative to human transcription. This chapter proposes two such alternatives that combine HMM and NN for continuous speech recognition.

5.1. INTRODUCTION

The goal of Automatic Speech Recognition is to develop techniques and systems that enable computers to accept speech input. The speech recognition problem is a speech-to-text conversion problem where users’ utterances are transcribed into text by a computer. The reason behind the failure of speaker independent continuous speech recognition is due to the variability and overlap of information in the acoustic signal, the need for high computation rates (a human-like system must match inputs to 50,000 words in real time), the multiplicity of analyses that must be performed (phonetic, phonemic, syntactic, semantic, and pragmatic), and the lack of any comprehensive theory of speech recognition. The dominant technological approaches for speech recognition systems are based on pattern matching of statistical representations of the acoustic speech signal, such as Hidden Markov Model (HMM) whole word and subword (e.g., phoneme) models. However, although significant progress has been made in the field of ASR these last years, the performance of the resulting systems is still not comparable to that achieved by human beings.
5.2. GENERAL SPEECH RECOGNITION SYSTEM

A block diagram of a general speech recognition system is shown in Figure 5.1.

![Block Diagram of a General Speech Recognition System](image)

**Figure 5.1. A general speech recognition system**

The recognition system accepts a speech signal as input and produces a classification decision as output. To perform continuous speech recognition, the system has four major operations:

1. Signal processing module: This module accepts the input speech signal
2. Feature extraction module: This module identifies key features of the input speech signal and removes redundant data
   - Time alignment and pattern matching module: This module performs phoneme and word detection and
   - Language processing module: This module selects the final recognized word string.

Language models

Speech Sentence Signal

Speech Processing

Feature Extraction

Pattern Classification

Acoustic models

Text of the sentence
5.2.1. Speech Processing and Feature Extraction

The purpose of signal processing is to derive a set of parameters to represent speech signals in a form which is convenient for subsequent processing and to process the sampled speech signal, produce a representation which is independent of amplitude variations, speaker stress and noise which is introduced from the transmission medium. Both time domain and frequency domain approaches can be used.

a) Recording, First Order Pre-emphasis, Windowing and Buffering

The task of encoding speech from sound involves a sequence of steps. They are:

1. Recording - The spoken word is sampled, digitized and filtered to remove the background noise. Digital filtering is used to emphasize frequencies that contain key speech energy and to compensate for nonlinearities in the recording process. The digitized speech signal is put through a first-order FIR filter. Pre-emphasis of speech signals can be performed to compensate the slopes in natural speech spectrum.

2. Buffering - In this step, the pre-emphasized speech signal is blocked into frame buffers of N samples with an adjacent frame separated by M samples. Speech is typically analyzed in overlapping short frames of about 30 msec long with a 10 msec frame shift.

3. Windowing - The next step in the process is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. A widely used window in speech recognition is the Hamming window.
b) **Spectral Analysis**

Mathematical techniques such as Fourier transforms and linear prediction on coefficients are used to quantify the power and the fundamental frequency of the samples, which are then concatenated to a single parameter vector for each frame. A speech recognition system can improve its speed and accuracy by restricting its analysis to those combinations of frequencies which are perceptually meaningful to the human auditory system (ear and brain) (Kinnunen, 2005). The following spectral analysis algorithms have been implemented and used in the proposed hybrid speech recognition systems.

(i) **Auditory Spectral Analysis**

The auditory filter bank is one of the most fundamental concepts in speech processing. An auditory filter bank can be regarded as a crude model of the initial stage of transduction in the human auditory system based upon the theory of the critical bandwidth and logarithmic scale in frequency. A perceptual measure, called the Bark scale or critical band rate, relates acoustical frequency to perceptual frequency resolution. A more popular approximation to this type of mapping in speech recognition is known as the Mel scale (O'Shaughnessy, 1987). The main application of auditory frequency scales is in the design of filter banks.

(ii) **Fourier Transform Filter Bank**

One of the easiest and more efficient ways to compute a non-uniformly spaced filter bank model of the signal is to simply perform a Fourier transform on the signal and sample the transform output at the desired frequencies. The FFT (Fast Fourier Transform), developed by Cooley and Turkey in 1965 is one such method and is computed as below. More details on FFT can be found in Deller *et al.* (2000), Smith (2002).
\[ S_l = 10 \log_{10} \left( \sum_{k=k_{\text{min}(l)}}^{k_{\text{max}(l)}} \phi_{XX}[k] \right) \]  

(5.1)

\[ K_{\text{min}(l)} = \frac{f_l + f_{t-1}}{2 \cdot F_s} \]  

(5.2)

\[ K_{\text{max}(l)} = \frac{f_l + f_{t+1}}{2 \cdot F_s} \]  

(5.3)

where \( S_l \) is the output spectrum for the \( l_{th} \) auditory spectral band, \( \phi_{XX}[k] \) is the FFT power spectrum of the \( k_{th} \) FFT frequency band, \( f_l \) is the frequency for the \( l_{th} \) auditory band and \( F_s \) is the sampling frequency.

(iii) LPC Analysis

Linear Predictive Coding (LPC) is one of the most powerful speech analysis techniques, and one of the most useful methods for encoding good quality speech at a low bit rate. It provides extremely accurate estimates of speech parameters, and is relatively efficient for computation. This section describes the basic ideas behind linear prediction, and discusses some of the issues involved in its use.

LPC analyzes the speech signal by estimating the formants, removing their effects from the speech signal, and estimating the intensity and frequency of the remaining signal. The process of removing the formants is called inverse filtering, and the remaining signal is called the residue.

The numbers which describe the formants and the residue can be stored or transmitted. LPC synthesizes the speech signal by reversing the process: use the residue to create a source signal, use the formants to create a filter (which represents the tube), and run the source through the filter, resulting in speech. Because speech signals vary with time, this process is done on short chunks of the
speech signal, which are called frames. Usually 30 to 50 frames per second give intelligible speech with good compression.

A great advantage of the LPC is the manipulation facilities and the narrow analogy to the human speech production. Since the main parameters of the speech production, namely the pitch and the articulation characteristics, expressed by the LPC coefficients, are directly accessible; the audible voice characteristics can be widely influenced. Also the number of filter coefficients can be varied to influence the sound characteristics, above all, the formant characteristics.

LPC analysis is carried out in an 8-step process.

Step 1 : Pre emphasis
Step 2 : Frame Blocking.
Step 3 : Windowing
Step 4 : Autocorrelation Analysis : Here each windowed frame is autocorrelated and the frames are prepared for LPC analysis.
Step 5 : LPC analysis : In this step, a formal method called Durbin’s method is used for converting autocorrelation coefficients to an LPC parameter.
Step 6 : LPC Parameter Conversion to Cepstral Coefficients : The cepstral coefficients, which are the resultant coefficients of the Fourier Transform representation of the log magnitude spectrum, have been shown to be more robust, reliable feature set for speech recognition than the LPC coefficients.
Step 7 : Parameter Weighting : Since low-order cepstral coefficients are sensitive to overall spectral slope and high-order coefficients are sensitive to noise, parameter weighting is considered as the standard weight of the cepstral coefficients.
Step 8 : Temporal Cepstral Derivative : Temporal Cepstral Derivative, an even better representation of the speech spectrum is a combination of the cepstral coefficients and the temporal cepstral derivative.
Once this is obtained, the result of the LPC analysis is a vector of \( Q \) weighted cepstral coefficients and an appended vector of \( Q \) cepstral time derivatives:

\[
O_t = (\hat{C}_1(t), \hat{C}_2(t), ..., \Delta C_1(t), \Delta C_2(t), ..., \Delta C_q(t))
\] (5.4)

(iv) Cepstral Analysis

Cepstral analysis is the most frequently used technique in speech recognizers because it can discriminate and represent the vocal tract and excitation separately as cepstral coefficients. There are two types of cepstral approaches: FFT cepstrum and LPC cepstrum. In the FFT cepstral analysis, the real cepstrum \( c(n) \) is defined as the inverse FFT transform of the logarithm of the speech magnitude spectrum.

\[
c(n) = \text{FFT}^{-1}(10 \log_{10} |\phi_{xx}[n]|) \quad 1 \leq n \leq N
\] (5.5)

where \( c(n) \) is the cepstral coefficient, \( \phi_{xx}[n] \) is the FFT power coefficient.

The procedure for cepstral analysis is to take the inverse Fourier Transform, thus converting the signal back into the time domain. This time domain signal is not a regular acoustic signal, since it was derived from the logarithmic spectrum and hence the name cepstrum. The important property of the cepstrum is that since the spectrum is the sum of two spectra, so the cepstrum is the sum of two components corresponding to the source and the filter. Usually the two components are separated: the lower end of the cepstrum corresponds to the filter while the higher end (or rather the middle of the reflected cepstrum) corresponds to the source.

A further operation on the cepstrum is to remove the central part of the reflected cepstrum, the part that corresponds to the source, and perform a Fourier Transform to again generate a frequency domain version of the signal. Since the effect of the source has been removed from this spectrum, it shows the
characteristics of the vocal tract filter. This is manifested as a much smoother spectrum than the original; the degree of smoothing depends upon the number of cepstral coefficients removed prior to the final Fourier Transform.

The technique can be used to generate smoothed spectra which show the characteristics of the filter as described above. The cepstrum can also be used as a means of determining the fundamental frequency of voiced speech since the part of the cepstrum corresponding to the source is often manifested as a single spike. The location of this spike gives a measure of the frequency of the source signal.

LPC (Linear Predictive Coding) is a method used to compress the spectral information for its efficient storage or transmission. The basic idea this method grounds upon is the source-filter model. According to this model, the source signal produced by the oscillation of the vocal folds is modified by the resonances determined by the morphology of the vocal tract and (oral and nasal) vocal cavities, acting as a filter on it.

The LPC analysis estimates the vocal tract resonances from a signal's waveform, removing their effects from the speech signal (inverse filtering) in order to get the source signal (or residue). In this way, besides getting the (estimated) source signal, information about the resonance features of the filtering vocal tract and cavities is also obtained. The information thus obtained is less redundant than the full description (in terms of data) of the original signal and therefore is more for speech encoding and transmitted along. A receiver is then able to reconstruct the signal by doing the inverse process (synthesis), namely filtering the residue using the resonance parameters.

c) Vector Quantization

Vector Quantization (VQ) is used to compare the representation obtained in the current analysis frame to a lookup table, called a codebook. The function of the
codebook is to determine the closest match in the codebook. The index to the codebook entry, rather than the initial presentation, is then used to simplify subsequent processing phases. A detailed description of vector quantization and its working is given in chapter 4.

5.2.2. Time Alignment and Pattern Matching

The procedures discussed in the previous section convert the speech samples into observation vectors representing events in a probability space. The next step is to perform a statistical analysis on the vectors to determine if they might be part of a spoken word or phrase or whether they are merely noise. Two popular techniques used are Dynamic Time Warping (DTW) and Hidden Markov Modeling (HMM). This research work is concentrating on developing techniques using HMM and hence is dealt in detail in the next section.

5.2.2.1. Hidden Markov Model

The dominant technique today for modeling the time course of a speech signal is Hidden Markov Modeling. The HMM was introduced in a landmark paper by Baum (1972), where the model was proposed as a statistical method for estimating the probabilistic function of a Markov chain. Essentially, HMMs are a method for modeling a system with discrete, time dependent behavior characterized by common, short-time processes and transitions between them.

A HMM can be considered as a finite state machine where the transitions between the states are dependent upon the occurrence of some symbols. Associated with each state transition is an output probability distribution which describes the probability with which a symbol will occur during the transition and a transition probability indicating the likelihood of this transition.

First-order, left to right HMMs are commonly used in ASR products. There are two assumptions behind them. The first is the Markov assumption, i.e., at each
observation time $t$, a new state is entered based on the transition probability, which depends only on the previous state. The transition may allow the process to remain in the previous state. The second assumption is the output-independence assumption, i.e., the output probability depends only on the state at that time regardless of when and how the state is entered. These two assumptions make calculation very efficient. A straightforward left-to-right HMM model is shown in Figure 5.2, where $Q_i \; (1 \leq i \leq 4)$ is the HMM state. This architecture is well suited for speech application because its inherent sequential structure models the temporal flow of the speech.

![Fig. 5.2: Left-to-right HMM](image)

The key parameters to be determined in an HMM-based ASR system are the number of states per unit, and the state transition and observation probabilities. The unit can be a word (the word model) or a phoneme (the phoneme model). Below, the HMM word model is examined. The results of the word model will apply directly to the phoneme model. Large amounts of training data are needed to obtain robust estimates of these probabilities so that the HMM algorithm is more powerful than the DTW algorithm, which uses a finite number of templates. A HMM-based ASR will generally have a number of HMMs. For instance 10 digit HMM word models correspond to 10 digits.
Training HMMs

The selection of the optimal number of states which properly describe the observed sequence of events for a given word is a somewhat empirical process. For discrete words, one might select a number of states which roughly correspond to the number of phonemes in the word to be modeled, with additional states corresponding to beginning and ending silences. An example structure of an HMM for the word “eight” is shown in Figure 5.3.

![Diagram of HMM for 'Eight']

Fig. 5.3 : HMM Word Model for ‘Eight’

HMM training can be implemented directly using an iterative procedure, known as the forward and backward algorithm or Baum-Welch algorithm (Baum, 1972), which is a computationally efficient method for determining the model parameters. This iterative procedure uses the forward probability and backward probability to update the observation probability and the transition probability. An alternative is to use the Viterbi algorithm, which offers a recursive optimal solution to estimate the state sequence. The Viterbi alignment is essentially a dynamic programming procedure where the probability between the test and reference model is computed. Training can accomplish speaker adaptation for existing models and cross-validation for discriminative training.

Testing HMMs

During recognition, the input symbols generate a particular sequence of states which are visited by the HMM in producing the observation sequence. The
state sequence essentially represents the segmentation of the word modeled by the HMM. However, to ensure that the optimal state sequence with the highest a posteriori probability is selected, the Viterbi algorithm is employed.

5.2.3. Natural Language Processing

The final stage of the recognition process consists of a Natural Language Processing (NLP) module which attempts to resolve the possible word selections using language specific constraints or knowledge. In ASR applications, the input to the NLP is often an N-Best list of potential words to be evaluated. Language processing is a crucial element in any text generating system with a large vocabulary.

5.3. SPEECH RECOGNITION TECHNIQUES

The present research work uses Multi Layer Perceptron (MLP) for continuous speech recognition. The working of the same is discussed in this section.

5.3.1. Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron architecture, which is one of the popular connectionist models, is a hierarchical design consisting of fully interconnected layers of computing units. It belongs to the class of mapping neural network architectures. In connectionist models, knowledge or constraints are not encoded in individual units, rules or procedures, but distributed across many simple computing units. Uncertainty is modeled not as likelihoods or probability density functions of a single unit, but by the pattern of activities in many units. Therefore, knowledge is not programmed into any individual unit’s function; rather, it lies in the connections and interactions between linked processing elements. A multi level network is realized by several units which are organized in a layered structure. The information flows through the layers connections, without any
connection inside the same layer. The layers other than input and output ones are known as hidden layers. A general structure is showed in Figure 5.4, where output layer represents the neural network’s output. Each layer is fully connected to the next one and such a structure is known as Multi Layer Perceptron (MLP).

Fig. 5.4 : MLP Structure

Considering its simple architecture, MLP offers a remarkably wide range of computational functions. Depending on the weights, the bias, and the input, a unit in a network can act as a simple linear boolean operator, or a non-linear analog processing element. The processing capacities of multi-layer perceptrons stem from the non-linearity used within units. If units were linear elements, then a single layer network with appropriately chosen weights could exactly duplicate these calculations performed by multi-layer networks. The MLP is a class of universal approximaters. Multi-Layer Perceptron classifiers have been applied to speech problems more often than any other neural network classifiers.
5.3.2. Back Propagation (BP)

The Back Propagation (BP) algorithm is the most powerful and widely applied MLP training algorithm. This popularity primarily revolves around the ability of back propagation networks to learn complicated multi-dimensional mapping.

The Back Propagation algorithm is a supervised learning procedure which involves the representation of a set of pairs of input and output patterns. The system first uses the input vector to produce its own output vector and then compares this with the desired output or the target vector. If there is no difference, then no learning takes place. Otherwise the weights are changed to reduce the difference. If the input units are directly connected to the output units, it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections, so as to progressively reduce the difference between the actual and desired output vector. The learning becomes more interesting but more difficult when hidden units whose actual or desired states are not specified by the task (input and output units can be) are introduced. The learning procedure must decide under what circumstances the hidden units should be active in order to help achieving the desired input-output behavior.

The Back Propagation algorithm is a remarkably simple extension of Widrow and Hoff's delta rule (Widrow and Hoff, 1960), and is also called the “generalized delta rule”. It uses a gradient search technique to minimize a cost function equal to the mean square difference between the desired and the actual network output. It requires a continuous differentiable non-linearity in the computational element. It consists of two passes of computation:

(i) the feed forward pass from bottom to top and
(ii) the feed backward pass from top to bottom.
In the forward pass, the synaptic weights remain unchanged throughout the network, and the function signals of the network are computed on a neuron-by-neuron basis. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying Equation (5.1) to Equation (5.3) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined. A block diagram of a unit is shown in Figure 5.5.

For the input pattern \( p \), the total input \( z_{pj} \) to unit \( j \) is a linear function of the input \( x_{pi} \) of the units that are connected to \( j \), the bias \( b_j \), and the weights \( w_{ij} \), i.e.,

\[
z_{pj} = \sum_i w_{ij} x_{pi} - b_j
\]  

(5.6)

The biases can be given by introducing an extra input to each unit which always has the value of 1. The weight on this extra input is equivalent to a threshold of the opposite sign. It can be treated just like the other weights, and the above formula becomes

\[
z_{pj} = \sum_i w_{ij} x_{pi}
\]  

(5.7)

A unit has a real-valued output \( y_{pj} \), which is a non-linear function of its total input

\[
y_{pj} = F(z_{pj})
\]  

(5.8)
where $F$ is a semi-linear function which is differentiable and non-decreasing. Here the activation function is the sigmoid function which is a continuous and nonlinear function as shown in Figure 5.6 and is given by the formula in Equation 5.9.

$$F(x) = \frac{1}{1 + e^{-x}}$$  \hspace{1cm} (5.9)

The derivation of $F(x)$ is,

$$F'(x) = F(z)(1-F(x))$$ \hspace{1cm} (5.10)

The use of a linear function for combining the input to a unit before applying the non-linearity greatly simplifies the learning procedure. The aim is to find a set of weights that ensures that for each input vector the output vector produced by the network is the same as (or sufficiently close to) the desired output vector. If there is a fixed, finite set of input/output cases, the total error in the performance of the network with a particular set of weights can be computed by comparing the actual and desired output vectors for every case. Let
be our measure of the error \( E_p \) on input/output pattern \( p \) where \( j \) is an index over the output units; \( y_{pj} \) and \( d_{pj} \) are the actual output value of an output unit and its desired output value respectively. Then

\[
E = \sum_p E_p
\]

(5.12)

where \( p \) is an index over all the cases (input-output pairs) and \( E \) is the overall measure of the error. To minimize \( E \) by gradient descent, it is necessary to compute the partial derivatives of \( E \) with respect to each weight in the network.

5.3.3. The Generalized Delta Rule and BP Algorithm

The Generalized Delta Rule implements learning by adjusting the weights \( w_{ij} \) according to the errors in the output layer. Errors in the output layer are “back propagated” to the hidden layer, and weights are adjusted based on these errors. In summary, the steps taken are:

- **Initialization of weights and offsets**: Start with a reasonable network configuration and set all synaptic weights and unit offsets to small random values that are uniformly distributed.

- **Input and output representation**: The input vector and the specified desired outputs are presented. In speech recognition, since MLP is used as a classifier, all desired outputs are typically set to zero except the one corresponding to the class the input comes from and thus the output is one. The inputs are samples from a training set which can be presented cyclically or randomly.

- **Forward pass**: During the forward pass, the input is presented and propagated forward through the network to compute the output value \( y_{pj} \) for each unit:
\[y_{pj} = F\left(\sum_i w_{ij}x_{pi}\right)\]  

(5.13)

where \(x_{pi}\) is the output of unit \(i\) in the current layer, \(y_{pj}\) is the output of unit \(j\) in the layer above the current layer, and \(w_{ij}\) is the weight from unit \(i\) to unit \(j\). \(F\) is the sigmoid function. This output is then compared with the target (the desired output \([0\ or\ 1]\)), resulting in an error signal \(\delta_{pj}\) for each output unit. This is simply the difference between the actual and desired output values.

\[\delta_{pj} = y_{pj}(1-y_{pj}) (d_{pj} - y_{pj})\]  

(5.14)

### Backward pass:
During the backward pass, the error signal is passed from the output layer to each unit of the input layer and the hidden layer:

\[\delta_{pj} = x_{pj}(1-x_{pj}) \sum_k \delta_{pk} w_{jk}\]  

(5.15)

where \(j\) is the unit in the current layer and \(k\) is the unit in the layer above the current layer. This propagates errors back one layer each time; the same process can be repeated for every internal layer.

### Adaptation of weights and offsets:
After \(\delta\) is calculated, weight changes can be computed for all the connections that feed into the final layer:

\[\Delta w_{ij} = \eta \delta_{pj} x_{pi}\]  

(5.16)

where \(\delta\) is the learning rate \(0 < \delta < 1\).

### Repetition by going to step 2:
The learning procedure is repeated again and again until a stop criterion is satisfied. The stop criterion can be a preset threshold of the average error in the output layer.

### 5.3.4. BP Implementation in Speech Recognition

The procedure of applying BP algorithm in speech recognition is explained in this section.
5.3.4.1. Network Initialization

In most of the situations, without any prior knowledge, the connection weights of the network are randomly initialized according to the minimum entropy criterion (Rumelhart et al., 1986; Rumelhart and McClelland, 1996; Thimm and Fiesler, 1997). Since the transition region of the sigmoid function is relatively narrow while the saturation regions are relatively wide, randomly initializing with very small values of random numbers and a uniform distribution between [-0.3, +0.3] will decrease the possibility that the basic units operate in the saturation regions of the sigmoid function.

5.3.4.2. Momentum Term and Learning Rate

The main requirement of the learning procedure in speech recognition is that the change in weight be proportional to the derivative $\delta E/\delta w_{ij}$. True gradient descent requires that infinitesimal steps be taken. The constant of proportionality $\eta$ is the learning rate in the proposed procedure. The larger this constant is, the larger the changes will be in the weights. This easily leads to oscillation but offers the most rapid learning. One way to increase the learning rate without leading to oscillation is to modify the generalized delta learning rule to include a “momentum” term. This can be accomplished by the following equation:

$$\Delta w_{ij}(n) = \eta \delta_{pi}x_i + \alpha w_{ij}(n-1)$$  \hspace{1cm} (5.17)

where the index $n$ shows the presentation number, $\eta$ is the learning rate, and $\alpha$ is a constant which determines the effect of past weight changes on the current direction of movement in weight space. This provides a kind of momentum in weight space that effectively filters out high-frequency variations of the error-surface in the weight space. This is useful in space containing long ravines that are characterized by sharp curvature across the ravine. The sharp curvature tends to cause divergent oscillations across the ravine. To prevent these, it is necessary to take very small steps. Taking small steps has the side effect of slow progress along...
the ravine. The momentum filters out the high curvature and thus allows the effective weight steps to be bigger.

5.3.4.3. Weight Adaptation

The weight can be changed in two ways. One is to change weights after each input-output case. This has the advantage that no separated memory is required for the derivatives, but it takes a long time to converge. It is suitable for a small training database and few patterns classified. The alternative scheme is to accumulate the partial derivative of the error $E$ with respect to the weight $w_{ij}$ over all the input-output cases and compute the average before changing the weights. Any conflicting and correlating patterns can be identified at this stage. However, if the number of training patterns is large, this updating scheme may not be feasible. To overcome this problem, in the proposed research work, the training data is divided into subsets such that the distribution of each subset is similar to the distribution of all training patterns. In this case the error energy space of each subset can be approximated to the global error energy space. Each subset is trained in the alternative cycle. The convergence speed is compromised. The main advantage of this strategy is that the created network has more flexibility and it may give better testing results.

5.3.4.4. Learning Time, Cross Validation and Stop Criterion

The essence of back-propagation learning is to encode an input-output relation, represented by synaptic weights. In practice, learning time is a big problem for the Back Propagation algorithm especially with large training databases as in speech recognition. Training the connectionist models can be very time consuming with improper network control coefficients such as initial weights, learning rate $\eta$ and momentum term $\alpha$. It needs several adjustments to have the network go to the converging direction.
Finding the optimum point of convergence is very difficult as the process becomes very slow after some iterations. Normally, the network is useable when the average error energy on each unit in the output layer is less than 0.1 empirically. However, this is not enough for speech recognition in which data for training and the data for testing are always quite different. The smaller the average error energy, the better the classification for the training data, however, the performance for test data may decrease. When the network is over-trained with the speech training data, the network could learn some database-dependent information from the training data. This kind of information degrades the network classification ability for test data. The network performance vs. the number of iterations is shown in Figure 5.7.

![Network performance vs. number of iterations](image)

**Fig. 5.7 MLP Performance and Number of Iterations**

A standard tool in statistics, known as cross validation, provides an appealing guiding principle (Soderstrom et al., 1988). The available training data set is randomly partitioned into a training set and a test set. The training set is further partitioned into two subsets: one subset used for MLP training and the other used for evaluation of the performance of the trained MLP model. When the performance goes down for the evaluation set, the training should be stopped. This
method is also used to decide upon the MLP structure: the number of units in each layer.

5.3.4.5. Modified Weight Adaptation and Weighted Training Data

In speech recognition, the training database is very large and might contain a lot of frames in which parameters have very little differences; for instance, speech parameters have little changes between two consecutive frames in the middle part of vowels (Macias-Guarasa et al., 2003; Bridle et al., 1982). Here the cumulating weight change method is used to update the weights. The reason is that the same input patterns will cause the same corresponding weight changes in the neural network and the summation can be done after all input patterns forward pass through the network. It is not necessary to pass the same pattern a lot of times. The equivalent results can be obtained by multiplying the weight changes of each input pattern by a factor. The average of cumulating weight changes is given by

\[
\Delta w_{ij}(n) = \frac{1}{N\sum_{p=1}^{N} \gamma_p} \sum_{p=1}^{N} \gamma_p \Delta w_{pij} + \alpha \Delta w_{ij}(n-1) \tag{5.18}
\]

where \(\gamma_p\) is the weighting factor for the pattern \(p\), \(N\) is the number of patterns. Using this procedure, a small subset of the training database can be selected to replace the large training database, and consequently, the training time will decrease.

Feature Emphasis

This method not only can speed up the training process but also emphasize the important input patterns. If the input pattern is important, a large weighting factor is assigned to it, otherwise a small factor is assigned. The network will be more robust to the important features. If the training patterns are contextually independent phoneme tokens, for instance, a large weighting factor \(p\) should be
applied to the frames in the middle part of the phoneme frame sequence and a small one to the boundary frames. The weighting function can simply be a Hamming window. Large weighting factors are assigned for nasal consonant n. When the speech data has more frames than the phoneme n, smaller values are applied to it. The noise is the least important, and the smallest values are used for those noise frames.

**Training Data Balance**

Moreover, the weighting method can also be used to balance the training database. In the training database, some patterns have much more samples than others. The network is often expected to have the same classifying capability with respect to the different sizes of the training patterns. This can be done by assigning small weighting factors to the patterns with the large number of samples or vice-versa, in order to avoid over-training the pattern with more input samples or under-training the pattern with fewer input samples caused by unbalanced training database.

**5.4. MLP FOR SPEECH RECOGNITION**

Speech features are frame based and acoustic speech signals can be phonetically segmented. In the present work, MLP is used for frame phoneme classification. The great advantage of using MLP in speech recognition is that it is able to gather all kinds of pattern classification information together to precisely present a pattern without any limitations. The structure of the MLP is shown in Figure 5.4. The system uses one hidden layer and has the advantage that it is able to gather all kinds of pattern classification information together to accurately represent a pattern without any limitations. The auditory Mel spectrum is used as input speech parameters' features. The inputs are one or multiple adjacent frames of those speech parameters. According to Rumelhart and McClelland (1996), five frames can be chosen for the calculation of first derivatives and second derivatives.
as an effective means for describing the speech dynamics. Following their advice, five frames are chosen, so that these context frames can increase the frame recognition scores substantially. Each unit corresponds to a phonetic label in the output layer. The target value on that unit corresponding to the current phoneme is set to one. The target values on the other units in the output layer are set to zero.

5.4.1. Training Data using Random Frame Selection Strategy

Consider the phonetic segments for the word ‘eight’, which is composed of two phoneme ‘ey’ and ‘t’. The total number of frames to represent this word was calculated as 16300 frames during experimentation. The phoneme ‘ey’ has more frames (1562 frames) than ‘t’ (103 frames). The noise or silent data is the rest which appears twice, before and after the word. These noise frames occupies more than one third of the training database. Training based on such a database takes an extremely long time and results in a phonetically biased network as well. The network is more sensitive to noise and to vowels than to consonants, which is not a desired property. In the proposed system, this problem is solved by using a random frame selection strategy. This technique is done in four steps.

1. Group all frames of each phoneme from the entire database.
2. Randomly select the same number of frames for each phoneme from the corresponding group.
3. Obtain the trained data by grouping together the selected frames.
4. Repeat steps 1-3 for each iteration of the MLP training.

The whole database is used and the result is a small balanced training database, which is a robust MLP produced in a short period of training time.

Boundary between two phonemes is difficult to determine and it is always observed that the boundaries in the middle part of the phoneme is always more stable than the boundaries. For better recognition, it is better to emphasize the
middle frames and de-emphasize the boundary frames. This can be achieved by using a window function that has large values in the middle and small values near the phonetic boundaries.

Table 5.1 shows an example of how weighted factors are assigned to phonetically labeled frames and the effect of applying random frame selection to it. Four frames per phoneme were randomly selected for applying random frame selection strategy. Combining the randomly selected frames with the training database weighting, the resulting set is a phonetically balanced, feature balanced smaller training database.

The random frame selection and the training database weighting are integrated. After being processed by those two methods, the received training data set is phoneme balanced, feature balanced and much smaller than the original training database.

The network is further trained by the above explained modified Back Propagation algorithm. The observations that were made while using the weighting method is that the output is a biased NN (the prior probability changes when using the trained MLP as a probability estimator) and the summation of weights are different from phoneme to phoneme. This can be avoided by using the window function as a threshold for frame selection to steer the weighting function to select the middle phonemes and then apply the unmodified BP algorithm.
TABLE 5.1 EXAMPLE OF APPLYING WEIGHTED FACTORS TO PHONETICALLY LABELED FRAMES AND TRAINING DATA GENERATED BY RANDOM FRAME SELECTION

<table>
<thead>
<tr>
<th>LABELS</th>
<th>WEIGHTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp</td>
<td>0.2</td>
</tr>
<tr>
<td>sp</td>
<td>0.4</td>
</tr>
<tr>
<td>sp</td>
<td>0.8</td>
</tr>
<tr>
<td>sp</td>
<td>0.5</td>
</tr>
<tr>
<td>sp</td>
<td>0.2</td>
</tr>
<tr>
<td>ey</td>
<td>0.2</td>
</tr>
<tr>
<td>ey</td>
<td>0.3</td>
</tr>
<tr>
<td>ey</td>
<td>0.5</td>
</tr>
<tr>
<td>ey</td>
<td>0.7</td>
</tr>
<tr>
<td>ey</td>
<td>0.8</td>
</tr>
<tr>
<td>ey</td>
<td>0.9</td>
</tr>
<tr>
<td>ey</td>
<td>0.8</td>
</tr>
<tr>
<td>ey</td>
<td>0.7</td>
</tr>
<tr>
<td>ey</td>
<td>0.5</td>
</tr>
<tr>
<td>ey</td>
<td>0.2</td>
</tr>
<tr>
<td>ey</td>
<td>0.2</td>
</tr>
<tr>
<td>t</td>
<td>0.2</td>
</tr>
<tr>
<td>t</td>
<td>0.7</td>
</tr>
<tr>
<td>t</td>
<td>0.9</td>
</tr>
<tr>
<td>t</td>
<td>0.7</td>
</tr>
<tr>
<td>t</td>
<td>0.2</td>
</tr>
<tr>
<td>sp</td>
<td>0.3</td>
</tr>
<tr>
<td>sp</td>
<td>0.4</td>
</tr>
<tr>
<td>sp</td>
<td>0.8</td>
</tr>
<tr>
<td>sp</td>
<td>0.4</td>
</tr>
<tr>
<td>sp</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LABELS</th>
<th>WEIGHTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp</td>
<td>0.2</td>
</tr>
<tr>
<td>sp</td>
<td>0.4</td>
</tr>
<tr>
<td>sp</td>
<td>0.6</td>
</tr>
<tr>
<td>sp</td>
<td>0.4</td>
</tr>
<tr>
<td>ey</td>
<td>0.8</td>
</tr>
<tr>
<td>ey</td>
<td>0.5</td>
</tr>
<tr>
<td>ey</td>
<td>0.8</td>
</tr>
<tr>
<td>ey</td>
<td>0.2</td>
</tr>
<tr>
<td>sp</td>
<td>0.2</td>
</tr>
<tr>
<td>sp</td>
<td>0.7</td>
</tr>
<tr>
<td>sp</td>
<td>0.9</td>
</tr>
<tr>
<td>sp</td>
<td>0.2</td>
</tr>
</tbody>
</table>

128
5.5. VECTOR QUANTIZATION AND LABELING

The basic concepts of vector quantization as applied to speech recognition and the modified K-means clustering technique are explained in detail in Chapter 4. After receiving the codebook by the modified K-means cluster, the input spectrum is quantized with the codebook. This process is called labeling. The classification procedure for arbitrary spectral vectors is basically a full search through the codebook to find the best match. Thus, for codebooks with large code vectors, the computation involved is very huge. To solve this, a VQ/HMM hybrid system is used.

5.6. HMM USED FOR TRAINING AND TESTING

The normal discrete HMM is usually defined as a 5-tuple \((Q, Z, \pi, A, B)\), where \(Q\) is a set of \(N\) states \(q_1, q_2, \ldots, q_N\), \(Z\) is the codebook, a set of \(L\) labels \(z_1, z_2, \ldots, z_L\) corresponding to prototypical spectra, \(\pi\) is a vector which specifies the initial distribution, \((\pi_1, \pi_2, \ldots, \pi_N)\), where \(\pi_i = \text{Prob}(q_t = i)\). The sequence of normal observations are denoted as \(O = (o_1, o_2, \ldots, o_T)\), where each \(o_t\) for \(1 \leq t \leq T\) is some \(z_i \in Z\). \(A\) is a matrix of state transition probabilities, \(A = [a_{ij}], 1 \leq i, j \leq L\), where \(a_{ij} = \text{Prob}(q_{t+1} = j \mid q_t = i)\). \(B\) is a matrix of observation probabilities, \(B = [b_j(k)], 1 \leq j \leq N, 1 \leq k \leq L\) where \(b_j(k) = \text{Prob}(o_t = z_k \mid q_t = j)\). The following HMM algorithms are used for HMM training and test.

5.6.1. Viterbi algorithm

The Viterbi algorithm is often used to save on computations and to obtain the state sequence at the same time. The implementation procedure is described as follows:

- Initialization
  \[ \delta_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N \quad (5.19) \]
\[ \phi_i(i) = 0 \] \hspace{1cm} (5.20)

- **Recursion**

\[ \delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i)a_{ij}]b_j(o_t) \quad 2 \leq t \leq T, \ 1 \leq j \leq N \] \hspace{1cm} (5.21)

\[ \phi_I(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i)a_{ij}] \quad 2 \leq t \leq T, \ 1 \leq j \leq N \] \hspace{1cm} (5.22)

- **Termination Star**

Star * indicates the optimized results

\[ \text{Prob}(O)^* = \max_{1 \leq i \leq N} \delta_T(i) \] \hspace{1cm} (5.23)

\[ q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i) \] \hspace{1cm} (5.24)

- **Path bracketing**

\[ q_t^* = \phi_{t+1}(q_{t+1}) \quad T-1 \geq t \geq 1 \] \hspace{1cm} (5.25)

where \( \delta_t(j) \) is the highest probability along a single path ending at state \( j \), at time \( t \). \( \phi_t(j) \) is the Viterbi path array.

Provided that each codeword of the VQ codebook \( Z \) is represented by a probability density function \( f(x \mid q_t) \), for a given state \( q_t \) of the HMM, the probability density function that produces a vector \( x \) can then be written as

\[ w_{q_t}(x) = \sum_{j=1}^{L} f(x \mid q_t) \] \hspace{1cm} (5.26)

\[ = \sum_{j=1}^{L} f(x \mid z_j, q_t) \text{Prob}(z_j \mid q_t) \] \hspace{1cm} (5.27)

where \( L \) denotes the VQ codebook level. For simplicity, the probability density function \( f(x \mid z_j, q_t) \) can be assumed to be independent of the Markov states \( q_t \).

Thus, for a given state \( i \), Equation 5.27, using codebook index \( j \) to represent \( z_j \), can be written as

\[ w_i(x) = \sum_{j=1}^{L} f(x \mid z_j) \text{Prob}(z_j \mid q_t = i) \] \hspace{1cm} (5.28)
In practice, Equation 5.29 can be simplified by using the M most significant values of $f(x \mid j)$ for each $x$ without affecting the performance. Experience has shown that values in the range of $2 - 8$ are adequate. This can be conveniently obtained during the VQ operations by sorting the VQ output and keeping the M most significant values. Let $\eta(x)$ denote the codeword entries $j$ of the set of VQ codewords, $z_j$, for those most significant values of $f(x \mid j)$ of $x$. Equation 5.29 can be rewritten as

$$w_1(x) = \sum_{j=1}^{L} f(x \mid j)b_1(j)$$  \hspace{1cm} (5.30)$$

Since the number of VQ codeword entries in $\eta(x)$ is of lower order than the VQ level $L$, Equation 5.30 can significantly reduce the amount of computational load for subsequent modeling compared with Equation 5.29. The computational complexity of the hybrid HMM mainly depends on the VQ level $L$ and the size of $\eta(x)$.

5.6.2. Modified Viterbi algorithm

From the definition in Equation 5.30, the modified Viterbi algorithm can be computed as below:

$$\delta_i(t) = \pi_i w_{i}(o_t) \quad 1 \leq i \leq N$$  \hspace{1cm} (5.31)$$

$$\delta_i(j) = \max_{1 \leq i \leq N} [\delta_{i-1}(t) a_{ij} w_{j}(o_t)] \quad 2 \leq t \leq T, \ i \leq j \leq N$$  \hspace{1cm} (5.32)$$

and the result is

$$\text{Prob}(O)^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$  \hspace{1cm} (5.33)$$

which is the usual output of an HMM.
5.7. **VQ/HMM HYBRID MODEL**

Given the VQ codebook index $j$, the probability density function $f(x | j)$ can be estimated by non-parametric or heuristic methods, such as multi-labeling. The estimation of $f(x | j)$ is crucial in the development of the hybrid system design.

A basic model of the hybrid system combining VQ and HMM is given in Figure 5.8.

![Diagram of Basic Model of Hybrid System (VQ and HMM)](image)

**Fig. 5.8 : Basic Model of Hybrid System (VQ and HMM)**

The labels obtained after VQ are the observations for HMM. Clustering is only used in VQ codebook generation and the clustering technique used is the modified k-means algorithm described in Chapter 4. Apart from reducing search space, the added advantage of this approach is that each HMM state must model a finite number of discrete labels. This typically results in faster HMM computation.

**5.7.1. Multi-Dimensional Labeling (Multiple Codebooks)**

Using multiple VQ codebooks is a unique technique in HMM for speech recognition. It is well known that parameter velocity (first derivatives) and parameter acceleration (second derivatives) are effective means of describing the speech dynamics (Wilpon et al., 1991). As, the spectral parameters, their first and second derivatives can be assumed independent, it is possible to make quantization
separately, and then code them into a single label. On the basis of this labeling, multiple features are simultaneously evaluated in a HMM with a conventional formulation. Since these features are appreciably independent of each other, VQ distortion can be significantly minimized by partitioning the parameters into separate codebooks. The observation probability is computed in the following way:

\[
w_i(x) = w_i(z) w_i(v) w_i(a) = b_i(z) b_i(v) b_i(a)
\]

(5.34)

where vectors z, v and a are the speech parameter vector, its first derivatives and its second derivative respectively, and \(f(xl) = 1\). This method is effective at reducing both memory space and processing time for labeling compared with a method which quantizes a vector of multi-features as a whole. On the other hand, the HMM with a labeling of multi-features may require much memory space to save the output probabilities compared with the HMM with labeling based on only spectral features alone.

5.8. MLP AND FFNN

In recent years, there has been considerable interest in the use of Artificial Neural Networks for speech recognition. Because Multi-Layer Perceptrons (MLPs) have excellent pattern recognition properties and Hidden Markov Models (HMMs) have powerful dynamic time warping capabilities, many researchers have tried to combine MLPs with HMM in a hybrid fashion. The benchmark work was done by Bourlard and Wellekens in 1990. Most research on hybrid neural network/HMM systems until now has concentrated on the use of MLPs or other types of networks (Recurrent Neural Networks, Radial Basis Functions (RBFs), etc.) as probability estimators (Yi and Tan, 2004; Mesbahi and Benyettou, 2004). In the present research work, instead of using MLPs as probability estimators, MLPs are used as labelers for discrete parameter HMMs.
5.8.1. Output Feed Forward Neural Network (FFNN)

While recording speech, before speech begins, the coarticulation value for noise is high. When the speech begins, the coarticulation value of speech data increases, while that of noise decreases. As time goes on, the value for the phoneme increases up to a saturation point and then decreases. As value for the phoneme decreases, the value for noise increases. Speech recognition accuracy may be increased if this basic coarticulation characteristic is represented inside a neural network rather than in the MLP input by using the weighted training data. This can be implemented by using a feed forward connection in the output layer that connects the units where coarticulation occurs. The FFNN takes the coarticulation into account by connecting the units in the output layer. The basic structure is shown in Figure 5.9. In this structure, the current phoneme and its adjacent previous phoneme are only considered since it is in these places where the coarticulation occurs. There is no connection between the current class output unit and other units.

![Output Feed Forward Neural Network](image)

**Fig. 5.9 : Output Feed Forward Neural Network**
Although the FFNN has shown great promises for static frame recognition, it has not reached the level for the whole word recognition as the FFNN itself lacks some of the most important HMM features such as dynamic time warping. To overcome this problem, the present research work uses FFNN non-linear mapping and multiple feature input capabilities to develop FFNN/HMM hybrid CSR systems.

5.8.2. FFNN/HMM Hybrid System

The MLP labeler extends the phonetic frame classification to the real speech recognition. The output FFNN described in Figure 5.9 is used. The inputs of the output FFNN are one or multiple adjacent frames of speech parameters. Each output unit corresponds to a phonetic label. The target value of the output unit corresponding to the current phoneme is set to one, the target values of the other output units are set to zero. Training is done by the modified error back-propagation algorithm. The training database is obtained by using both random data selection and training data weighting. Once the FFNN is fully trained, the weights are kept fixed. For each frame, the label corresponding to the highest scoring output of the FFNN is passed on to the discrete parameter HMM system. Thus a supervised VQ which contains phonetic information is achieved. The HMMs are then trained using a classical algorithm like the Viterbi algorithm. Figure 5.10 shows a pictorial representation of the system.

![Fig. 5.10 : Basic Structure of FFNN/HMM Hybrid System](image)

In classical HMM training algorithms the models are trained to maximize the likelihood of producing their training examples, but no training is done to minimize the probability produced by other examples in the model. Several
researchers have, therefore, investigated discriminative training methods for HMMs, most notably the Maximum Mutual Information Criterion (Cetin, 2004) and Corrective Training (Chien and Wu, 2007/2008; Xiuyang and Wayne, 1999). FFNN, however, incorporates automatic discrimination. In the conventional MLP, used for pattern classification, the number of output units corresponds to the number of pattern classes present. During training, the output unit corresponding to the class of a pattern vector is kept clamped at state 1 while the others are clamped to state 0. Hence the components of the desired output vector take on two crisp state values. During test, a winner-take-all mechanism caused the test pattern to be classified as belonging to that class corresponding to the output unit with the highest activation. When the FFNN is trained for frame classification, it is explicitly demanded that one output is maximal and the other outputs are zero. This provides a discriminating effect.

5.8.3. Multi-Dimensional Labeling FFNN/HMM System

In the multi-dimensional labeling method, three labeling techniques are combined. All the three methods are grouped into one category as they all need the HMM with multiple codebooks. The three techniques are

1. Top-N Method
2. Modified Top-N Method proposed by Ma and Van Compernolle (1992)

A simple way to incorporate the information of output units into the system is to use not only the top scoring output, but also the N top scoring outputs as labels. This method is called as Top-N. For example, if N = 2, then 2 labels are passed on to the HMM system. These labels correspond to the highest and the second highest output of the MLP. This method has the advantage of using information from N output units instead of one. There are now N label streams from the FFNN to the HMM instead of one. In this manner, the FFNN input
parameter space is much more clearly described. The HMM can then use these labels as independent observation variables, just as labels from multiple codebooks. The top-N method has the disadvantage - the assumption that the labels are independent is invalid. It is clear, for example, that the first and second label can never be the same.

To overcome this difficulty, another method was proposed again by Ma and Compernolle (1992) where the idea was to give every combination of highest and the second highest scoring MLP output a separate label. This means that the number of possible labels would now be equal to the square of the number of phonemes. For a vocabulary with 22 phonemes, for example, this means that the number of possible labels would be 484. Normally, this is very high, and the label probabilities need large amounts of training data to be estimated properly. This method is labeled as MC-Top N Method.

To solve the above problem Cerf and Van Compernolle (1993) described another modified Top-N method, which reduced the number of possible labels by taking the most occurring second choices, since in practice, only a few FFNN outputs other than the highest scoring output have values significantly different from zero and the equivalent result was produced. This method is labeled as CC-Top N Method. However, the independent assumption is still invalid for both methods, as using different labels do not change the parameter dependency. The three methods Top-N, MC-Top and CC-Top are grouped together to form a multi-dimensional labeling Network. The system architecture of hybrid multi-dimensional FFNN/HMM system is shown in Figure 5.11.

![Fig. 5.11: Hybrid Multi-Dimensional FFNN/HMM System](image-url)
5.9. IMPLEMENTATION AND EXPERIMENTAL RESULTS

This section describes the implementation and experiments performed to investigate the neural network for continuous speech recognition of medical terminology. The recogniser will recognise continuous medical terms spoken by healthcare professionals. Although the present work is designed for medical field, it is designed in a general purpose fashion and hence can be applied to other applications also.

5.9.1. Simulated Experimental Environment

- Data Preparation

Speech data is needed for both training and testing. For the experimental purpose, the system is built by recording all the speech from scratch by using a long-established paradigm prompt-and-response method where the user is prompted for each sentence. In the case of the test data, these prompts provide the reference transcriptions against which the recogniser's performance can be measured. They also provide a convenient way to create them and to use the task grammar as a random generator. In the case of the training data, the prompt scripts will be used in conjunction with a pronunciation dictionary to provide the initial level transcriptions needed to start the HMM training process. Since the application requires the arbitrary terms to be added to the recogniser, training data with good phonetic balance and coverage is needed. It follows from the above that before the data can be recorded, a terminology set must be defined, a dictionary must be constructed to cover both training and testing and a task grammar must be defined. The terminology set in the present research work is taken as general medical field consisting of 1500 words.
Task Grammar

The grammar used consists of a set of variable definitions followed by a regular expression describing the words to recognize. Sample grammar used in the experimental setup is given below:

\[
\begin{align*}
$digit &= \text{ONE | TWO | THREE | FOUR | FIVE | SIX | SEVEN | EIGHT | NINE | OH | ZERO;} \\
$name &= \text{CROCIN | METACINE | GELUSIL} \\
$type &= \text{TAB | LIQ} \\
$timing &= \text{[ AM | PM ] DAILY}
\end{align*}
\]

{SENT-START

{PRESCRIBE <$name> <$digit> <$type> <$timing>}

SENT-END}

where the vertical bars denote alternatives, the square brackets denote optional items and parenthesis \{ \} indicates repetition.

Dictionary

The first step in building a dictionary is to create a sorted list of the required words. The CSR task needs to be trained with a large set of sentences containing words that are phonetically balanced. For this reason, the training data consists of English sentences that are unrelated to the recognition task. Below, a short example of creating a word list from sentence prompts is given. The first few items might be as follows

S0001 TWO OTHER CASES ALSO WERE ALSO CONSIDERED
S0002 BOTH REQUIRED FOR LONG PERIOD DIAGNOSIS
S0003 BOTH FIGURES WOULD GO HIGHER IN LATER YEARS
S0004 THIS IS NOT A PROGRAM OF SOCIALIZED MEDICINE
etc.

The desired training word list is then extracted automatically from these sentences. Before processing, all the white spaces are replaced by new line characters and are sorted alphabetically with one word per line. An HMM is
estimated for each training and test data after it has been recorded. An example of
the dictionary is shown below:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>ah</td>
<td>sp</td>
</tr>
<tr>
<td>a</td>
<td>ey</td>
<td>sp</td>
</tr>
<tr>
<td>cough</td>
<td>ka</td>
<td>ff</td>
</tr>
<tr>
<td>crocin</td>
<td>kr</td>
<td>ow</td>
</tr>
<tr>
<td>eight</td>
<td>ey</td>
<td>t</td>
</tr>
<tr>
<td>daily</td>
<td>da</td>
<td>ie</td>
</tr>
<tr>
<td>phone</td>
<td>f</td>
<td>ow</td>
</tr>
<tr>
<td>sent-end []</td>
<td>sp</td>
<td></td>
</tr>
<tr>
<td>sent-start []</td>
<td>sp</td>
<td></td>
</tr>
<tr>
<td>seven</td>
<td>s</td>
<td>eh</td>
</tr>
<tr>
<td>sick</td>
<td>c</td>
<td>ie</td>
</tr>
<tr>
<td>ten</td>
<td>ta</td>
<td>en</td>
</tr>
<tr>
<td>to</td>
<td>t</td>
<td>ax</td>
</tr>
<tr>
<td>to</td>
<td>t</td>
<td>uw</td>
</tr>
<tr>
<td>two</td>
<td>t</td>
<td>ow</td>
</tr>
<tr>
<td>visit</td>
<td>v</td>
<td>ie</td>
</tr>
<tr>
<td>zero</td>
<td>z</td>
<td>ia</td>
</tr>
</tbody>
</table>

The entries for SENT-START and SENT-END have a silence model ‘sp’ as their pronunciations are null output symbols. Having completed prompts for test data, before recording training data, prompts for training data has to be generated. For explanation purposes a small vocabulary set with phonetic transcription as shown below is used.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>am</td>
<td>a</td>
<td>em</td>
</tr>
<tr>
<td>and</td>
<td>aa</td>
<td>n</td>
</tr>
<tr>
<td>at</td>
<td>aa</td>
<td>t</td>
</tr>
<tr>
<td>crocin</td>
<td>kr</td>
<td>ow</td>
</tr>
<tr>
<td>daily</td>
<td>da</td>
<td>ie</td>
</tr>
<tr>
<td>pm</td>
<td>p</td>
<td>em</td>
</tr>
<tr>
<td>sick</td>
<td>c</td>
<td>ie</td>
</tr>
<tr>
<td>take</td>
<td>taa</td>
<td>ie</td>
</tr>
<tr>
<td>two</td>
<td>t</td>
<td>ow</td>
</tr>
<tr>
<td>times</td>
<td>ta</td>
<td>ie</td>
</tr>
<tr>
<td>visit</td>
<td>v</td>
<td>ie</td>
</tr>
</tbody>
</table>
The above set has 20 phonemes and thus 21 phonetic classes including noise or silence denoted by ‘sp’. Several combinations of continuous sentences can be formulated with the above 20 phonemes. For example, “visit daily at two pm” “take crocin daily two times”.

- **Speech Database**

  The database constructed consisted of the experimental words uttered by 400 speakers (200 males and 200 females). The experimental set contains 21 different phonemes. A subsample of 100 males and 100 females were used for training the HMM parameters and the FFNN connectionist weights, and the rest were used for testing.

- **Speech Signal Processing**

  Both the test data and training data consisted of medical terms spoken by 400 speakers with a sampling rate of 8 Khz. The speech signal preprocessing is based on Mel-spectral filter bands The sequence of operations is as follows:
  - Sampling at 8 kHz;
  - Applying the Hamming window with a window length of 30 msec
  - Using a 10 msec frame shift;
  - Calculating 256 points FFT;
  - Computing the summation of energy values into 15 contiguous bands spanning the frequency range from 200 Hz to 3125 Hz;
  - Converting to log-energy parameters

  The input Mel-spectra are normalized between [-1, 1] for two reasons. The first is to remove the variations in the overall magnitudes of the spectral coefficients which enable the network to focus on the linguistic information in the speech signal, such as the formant frequencies and the second is to move the mean of the input vector close to zero, so that the classification process is improved.
Moreover, the normalized values are near the transition regions of the sigmoid function, where learning is faster than it is in the saturation regions. However, the overall magnitudes of the spectral coefficients cannot be removed, since they often contain relevant linguistic information.

5.9.2. Number of Hidden Units and Frame Classification

The network's capability to capture the underlying characteristics of the input data can be affected by the number of hidden units. The classification accuracy is a function of both numbers of hidden units and the amount of training data. With the inclusion of more hidden units, performance can be typically improved. However, the performance degrades when there are too many hidden units. As long as the number of hidden units is reasonably chosen, increasing the size of the training set typically improves the performance of the network. According to Mirchandani and Cao (1989), the number of hidden units in a feed-forward neural network depends on the number of input training patterns. The purpose of this experiment is to determine the optimal number of units in the hidden layer and to improve the classification accuracy.

As mentioned previously, a three-layered network is used in the FFNN network. Five frames are used in the input layer. Assuming that 15 channels are required to cover the bandwidth of the network, the number of units in the input layer is $5 \times 15 = 75$ units. The output layer has 21 units corresponding to the 21 phonemes taken in the example. The next step is to find the optimal number of units in the hidden layer that promotes the performance of classification. The training and test results are illustrated in Figure 5.12.

The experimental results show that the performance on the training data constantly improves as the number of hidden units increases compared to that of the test data. The performance started decreasing when the number of hidden units went beyond 80 and hence the reasonable size is 30-80 hidden units and the
experimental results reported in this chapter uses 30 as the number of units in the hidden layer.

5.9.3. FFNN/HMM Hybrid System

The system training consists of three parts, that is, the HMM training for phonetic segmentation, the FFNN training for labeling and the second HMM training for speech recognition.

5.9.3.1. HMM Training for Phonetic Segmentation

Since the connectionist networks require an extremely large labeled training corpus, and since the training of the system requires that an utterance be labeled at the frame level (windowed frames), it is necessary to use an automatic alignment procedure to generate initial labels for supervising the neural network targets for the training set. The Hidden Markov Models are trained on this task first, and a forced alignment procedure is used to generate phoneme or noise (silence) labels for each frame of speech in the training database. The proposed HMM system is a discrete parameter system that uses a VQ codebook size of 200 and 21 phonetic models. The Viterbi style training is run over a sufficient number of passes to obtain full convergence.

Building the FFNN Training Database

Initially, a Viterbi alignment against the computed phonetic models is used to produce a phonetically labeled database frame by frame. The frames at the boundaries of each phonemic segment are deleted in order to avoid labeling errors around phonemic boundaries. The phonemic transitions from the MLP training is excluded as advised by Weiye (1999). The remaining frames are used in an overlapping fashion. This procedure reduced the number of representations of the phonemes. For example, during experimentation, 21 phonemes produced 8000 representations in total. Application of Viterbi alignment procedure reduced the
number of representations of the 21 phonemes by 5000, or on an average 250 representations per phoneme. The result obtained using varied number of phonemes and their representation before and after Viterbi process is shown in Figure 5.13.

**FFNN Training**

As mentioned earlier, the FFNN structure consists of 3 layers namely input layer (75 units), hidden layer (30 units) and output layer (21 units). The input data is normalized between [-1, 1] and are trained using BP algorithm. The connection weights are randomly initialized to ±0.3. All training samples are presented to make the weight adaptation. In order to do the network scaling, the training is further divided into three stages:

- A short training with one representation per phoneme occurrence in a small network
- A short training with all training material in the small network
- A full training with all training material in the scaled network.

First a small network (3 x 15 units at input layer) is initialized. The training starts from the random initial weights with one representation per phoneme. After 20 iterations, all the training data are applied and the training goes on. After 50 iterations, the network is rescaled by adding one frame at both sides of the small network in the input layer and adjusting the weights. The training of the enlarged network starts from those redistributed weights using all training data. The training stops after 2,000 iterations. The values of connectionist weights are stored every 100 training iterations for later evaluation. Here, the network first learns some initial knowledge to find the optimum convergence direction to save searching time with small training data in the initial stage. It is very difficult to converge to scaling the network after the initial stage because the small network reaches its own local minima which are quite different from the local minima of the enlarged network.
Fig. 5.12: Number of Hidden Units

Fig. 5.13: Number of phonemes and number of representation before and after Viterbi Process

145
5.9.3.2. FFNN Training for Labeling

The most straightforward implementation of FFNN labeling is to use the phonetic label corresponding to the highest analog output value as a discrete output. Because of the sigmoid function, the analog outputs are in the range [0, 1]. Empirically, the confidence level is 0.5 for the highest analog output.

5.9.3.3. Second HMM Training

In the second HMM training, both word models and phonetic models are applicable. In this experiment, phonetic models are chosen. The same HMM phonetic models and training methods are used as in the original Viterbi segmentation. However, instead of Euclidean VQ with a codebook, the MLP labeler is used.

5.9.4. Results and Discussion

The FFNN labeler in the proposed system is used for improving the standard vector quantization for HMM recognizers. During experimentation, the 21 parameter FFNN labeler is compared with VQ of the codebook size. The results revealed that FFNN labeler produced 99% accuracy, while VQ produced 93% accuracy (Fig.5.14). This shows that the intrinsic acoustic phonetic distortion provided by the FFNN labeling is significantly smaller than the distortion provided by VQ labeling.

5.9.5. Random Frame Selection and Weighted Training Data

During frame selection, the following should be considered:

1. When the size of the training data is small, the whole data set can be used to update weight changes each time. But this might not suit the large training
database, which is the case of most CRS. Therefore, the input to the MLP should be a small subset of data from the large training data.

2. The phonetic balance between short and long phonemes should be maintained to avoid over training on vowels and consonants.

To solve the above two problems, the proposed system uses weighted training and random frame selection methods. Weights are applied to the training data. As context independent phonemes are used, the frames in the middle part of each phoneme are more important than those on the boundaries. The purpose is to emphasize the middle frames and de-emphasize the boundary frames so that the network is robust to the middle part of each phoneme when compared to the boundary part.

This is done by applying a window function, which has large values in the middle and small values near the ends, that is, a Hamming window. An example of how to put the weighting factors in a phonetic label file is shown in Table 5.2.

<table>
<thead>
<tr>
<th>PHONETIC LABELS</th>
<th>WEIGHTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp</td>
<td>0.1</td>
</tr>
<tr>
<td>sp</td>
<td>0.2</td>
</tr>
<tr>
<td>sp</td>
<td>0.1</td>
</tr>
<tr>
<td>a</td>
<td>0.4</td>
</tr>
<tr>
<td>a</td>
<td>0.6</td>
</tr>
<tr>
<td>a</td>
<td>0.6</td>
</tr>
<tr>
<td>a</td>
<td>0.4</td>
</tr>
</tbody>
</table>
For example, the third frame “sp” has a weight factor of 0.1. If this weight factor is repeated in all frames, then the training of data is not needed. The new connectionist weights are calculated using the formula,

\[ w_{\text{new}} = \frac{1}{2} w_{ij} \] (5.35)

Hamming window allots small weights to the boundary frames and thus avoids the deletion of the boundary frame.

To this calculated weight data, random frame selection is applied. For example, where the hand-labeled training data contains a total of 16,300 frames, the shortest phoneme has 103 frames, the longest one has 1,562 frames and the noise data has 6,337 frames. It is evident that the noise occupies more than one third of the training database and the reason is that the training data is not phonetically balanced.

To generate a phonetically balanced training data random frame selection method is used. The steps performed are explained below:

1. Frames for each phoneme are extracted from the whole data. Thus with the experimental data, a total of 21 collections are extracted that correspond to each phoneme.

2. Two hundred frames are randomly selected from each collection. Thus the training data set results with 4200 frames in total.

The modified Back-Propagation algorithm is applied to train the MLP. The connectionist weights (including the offsets) are randomly initialized between [-0.3, +0.3]. A new training set is generated in each iteration. The results with weighting or without weighting are illustrated in Figure 5.15.
Fig. 5.14 Vector Quantization and FFNN Labeling

Fig. 5.15: Recognition Rate using data weighted method
The results reveal that when the number of iterations is small, the network is under-trained and the performance on both the training and test data is directly proportional to the number of iterations. When the network is well-trained, the performance of training data is better than that of the test data when the number of iterations increases. When the number of iterations is very large, the network is over-trained and the test data performance degrades while the improvement of training data is very small. The figure further shows that the results of using weighted method produces better result than without weight method. Further the experiments revealed that the intrinsic acoustic phonetic distortion provided by spectral derivative labeling is smaller in FFNN labeling than VQ labeling. The FFNN/HMM has very high discriminative power and performs well in capturing speech features.

5.9.6. Multi-Dimensional Labeling FFNN/HMM System

The performance of FFNN Multi-dimensional labeling is discussed in this section.

Training Database

In order to supervise the FFNN learning, the automatic phonetic segmentation is obtained by using Viterbi algorithm. The phonetically labeled database is obtained from Viterbi alignments. This procedure results in 150,000 frames representing 21 phonemes. The shortest phoneme has 1100 frames, the longest one has 17,700 frames. The noise which has 55,000 frames occupies more than one third of the training database. 600 frames of each phoneme are selected randomly from the whole training database for each FFNN training iteration. On an average, there are 3 representations per phoneme per speaker, since the training data includes 200 speakers. The random frame selection method is used for every iteration during the connectionist weight adaptation.
System Training

Let $S_{\text{int}}$ be the initial random seed, $N_{\text{stop}}$ and $N_{\text{rept}}$ be the number of iterations for stopping and repeating the FFNN training data respectively. The seed ($S$) is initially set to $S_{\text{int}}$ and the iteration counter and repetition counters are set to 0. The system randomly generates $600 \times 21$ frames as training data from the training database. Forward and backward propagation are used to adapt the weight. Counter is incremented by 1. The MLP training is stopped when counter is greater than $N_{\text{stop}}$. The new $S$ value is set to $S_{\text{int}}$ and if $r = N_{\text{rept}}$, a new seed is generated and the whole process is repeated for the new seed. After the network is ready, more than one output label is taken as HMM input labels.

Top-N Labels

As discussed, the top-N labels are used as label vector by the HMM system. The elements of the label vector are assumed to be statistically independent. The 21 labeled outputs of the neural network are sorted from large to small. The number of FFNN labeled output is kept less than four. The top-N labels are considered as observation vector for the HMM system. If the vector length is 2, then there are 441 output vectors ($21 \times 21$). When the length is more than 2, then the number of combinations also becomes high. To avoid this problem, combinations with null observation are pruned out. The experimental result comparing the different number of labels with recognition rate is illustrated in Figure 5.16.

From the figure, it can be observed that the recognition rate is directly proportional to the number of labels. A subsample space of 21 labels was selected for the experiment. The system reached a steady state when the number of labels is between four and seven for multi-dimensional with a recognition rate of 97% and gradually increased to 99% when the number of labels reached its maximum.
Fig. 5.16. Number of Labels Vs. Recognition Rate
5.10. CSR EXPERIMENTS CONDUCTED IN HOSPITAL

To compare the performance of the standard VQ/HMM hybrid model, FFNN Multi-dimensional/HMM hybrid model and FFNN/HMM hybrid model in real time environment, a local hospital, Raagee’s Clinic, Madurai, Tamil Nadu, India was used. The clinic has 20 doctors and five doctors were selected on the basis of their availability. Care was taken to make sure that they have minimum accent or speech impediment and all underwent routine training and enrolment process. The vocabulary consisted of 25,000 words specifically designed for general physicians and technicians. A total of 5000 reports were dictated by these doctors. Care was taken to make sure that the report was small and the training phase is complete. The parameters used for the performance study are recognition rate and time taken to produce reports before and after implementation of CSR and execution speed.

5.10.1. Recognition Rate

The term accuracy is associated with the correct word recognition rate. The correct word recognition rate is measured with the formula;

\[ 100 \times \frac{(N-D-S-I)}{N} \]  

where N denotes the total number of words in recognized sentences, D denotes deletions, S denotes substitutions and I denotes insertions. The system performance was measured in terms of Correct Word Recognition Rate and Correct Sentence Recognition Rate.

The doctors were asked to spontaneously dictate cases using continuous speech recognition. For each doctor, the total number of words spoken and the number of words correctly recognized by the system were recorded. Subsequently, a 124-word test report was developed that contained a variety of words commonly used in dictation (Appendix II). Each doctor was asked to read this test report
All testing was performed in reading room, with no attempt made to control background noise. For the purposes of accuracy determination, punctuation marks and formatting commands (e.g., "new paragraph", "full stop") were considered as single words. Numbers, dates, and other compound words were counted as single words, regardless of the number of utterances it took to produce them. A word was considered correctly recognized only if it was transcribed exactly as intended. Errors caused by mispronunciations, homonyms (words with similar pronunciations but different spellings, such as "to," "too," and "two"), and out-of-vocabulary words (words not included in the computer’s 25,000-word vocabulary) were considered incorrect. The result of word recognition rate for five doctors using 80 reports is shown in Table 5.3 and Figure 5.17.

**TABLE 5.3 WORD RECOGNITION RATE OF CONTINUOUS SPEECH RECOGNITION FOR FIVE DOCTORS**

<table>
<thead>
<tr>
<th></th>
<th>NO. OF WORDS SPOKEN (80 REPORTS)</th>
<th>RECOGNITION RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>VQ/HMM</td>
</tr>
<tr>
<td>Doctor 1</td>
<td>1115</td>
<td>92.8</td>
</tr>
<tr>
<td>Doctor 2</td>
<td>856</td>
<td>93.1</td>
</tr>
<tr>
<td>Doctor 3</td>
<td>846</td>
<td>92.2</td>
</tr>
<tr>
<td>Doctor 4</td>
<td>879</td>
<td>93.3</td>
</tr>
<tr>
<td>Doctor 5</td>
<td>993</td>
<td>91.9</td>
</tr>
<tr>
<td>Total</td>
<td>4689</td>
<td>92.7</td>
</tr>
</tbody>
</table>

From the results, it is clear that the hybrid models are superior to that of the traditional VQ/HMM model. On an average the multi-dimensional FFNN/HMM model produced around 99% accuracy, followed by 97% (FFNN/HMM) and 93% (VQ/HMM) models. Eventhough the performance of both the hybrid models are more or less the same, the FFNN/HMM model is slightly better than multi-dimensional FFNN/HMM model.
5.10.2. Reports Availability

The number of reports prepared by five doctors was 80 over a period of 7 days. The time taken to prepare the report was analyzed and the results are shown in Table 5.4.

**TABLE 5.4 REPORT AVAILABILITY BEFORE AND AFTER APPLYING CSR SYSTEM**

<table>
<thead>
<tr>
<th>NO. OF WORDS SPOKEN (80 REPORTS)</th>
<th>REPORT AVAILABILITY (% OF TOTAL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BEFORE</td>
</tr>
<tr>
<td></td>
<td>VQ/HMM</td>
</tr>
<tr>
<td>8 Hours</td>
<td>2.4</td>
</tr>
<tr>
<td>24 Hours</td>
<td>10.5</td>
</tr>
<tr>
<td>48 Hours</td>
<td>32.1</td>
</tr>
<tr>
<td>72 Hours</td>
<td>54.2</td>
</tr>
</tbody>
</table>

It is clearly evident that the usage CSR has definitely improved the report preparation speed. Of the three HMM techniques, the best performance was produced by Multidimensional label FFNN/HMM technique.

5.10.3. Execution Speed

The hybrid systems developed were tested with respect to the time taken for recognition with varying number of doctors in the training dataset. The result is depicted in Figure 5.18.
Fig. 5.17 Recognition Rate of VQ/HMM and Proposed Hybrid Models

Fig. 5.18: Recognition Speed of VQ/HMM and Proposed Hybrid Models
It is clearly evident that the performance of multi-dimensional labeling system is superior to that FFNN/HMM system with respect to speed of recognition. This might be due to the more number of computations required in the formation of codebooks and labels. The difference in performance arises only when the number of speakers increases. When the number of speakers is less, both the systems perform more or less in the same manner with respect to execution speed.

5.11. SUMMARY

In this chapter, two new hybrid models using MLP and HMM were explained and the performance of both the systems was analyzed and compared with VQ/HMM hybrid system. The results are concluded in the next chapter, Conclusion and Future Recommendations.