Scaly Neural Networks for Speech Recognition Using DTW and Time Alignment Algorithms

S.karpagavalli[*], P.V.Sabitha[**]

[*] Senior Lecturer, Department of Computer science (PG), PSGR Krishnnammal College for Women, Coimbatore
[**] Mphil Research Scholar, Department of Computer science, PSGR Krishnammal College for Women, Coimbatore

Abstract- Speech recognition has been an active research topic for more than 50 years. Interacting with the computer through speech is one of the active scientific research fields particularly for the disable community who face variety of difficulties to use the computer. Such research in Automatic Speech Recognition (ASR) is investigated for different languages because each language has its specific features. Neural Networks are, in essence, biologically inspired networks since they are based on the current understanding of the biological nervous system. In essence they are comprised of a network of densely interconnected simple processing elements, which perform in a manner analogous to the most development of neural networks, and a basic introduction to their theory is outlined in this elementary functions of a biological neuron. Reduced connectivity neural networks are discussed and the scaly architecture neural network is described. Various algorithms are available to perform this time alignment of the input pattern to the neural network and the performance of the neural network is dependent upon the performance of the time alignment algorithm used. In this chapter, the various types of time alignment algorithms are described and their operation is outlined in detail.

Index Terms- The Perception, Multi-Layer Perception, Error Back Propagation Algorithm, Scaly Neural Network Architecture, Experimental Procedure, Time Alignment Algorithms, Linear Time Alignment, Dynamic Time Warping, Trace Segmentation.

I. Introduction

In addition, the published research work also provides a big weight-age to get admissions in reputed varsity. Now, here we enlist the proven steps to publish the research paper in a journal. Automatic Speech Recognition (ASR) is investigated for different languages because each language has its specific features.In this paper we present scaly neural network for speech recognition and preprocessing of speech using time alignment algorithm. Two of the major problems in speech recognition systems have been due to the fluctuations in the speech pattern time axis and spectral pattern variation [13]. Speech is greatly affected by differences in the speaker such as age and sexes as well their physical and psychological condition. The pitch and speed of the speaker will be altered by the way they feel at the time. If they are agitated, angry or short of time they will most likely speak at a faster rate and higher pitch than when they are calm and relaxed [7].

Speaking rate variation results in non-linear fluctuations in the speech pattern time axis. The length of the input pattern to the neural network in question is constrained by the number of input neurons to the neural network since this type of network architecture cannot be varied once trained. The input pattern vectors must be modified to fit the neural network while still retaining all their discriminating features.

Various algorithms are available to perform this time alignment of the input pattern to the neural network and the performance of the neural network is dependent upon the performance of the time alignment algorithm used. In this chapter, the various types of time alignment algorithms are described and their operation is outlined in detail.

II. The Perception

The idea of the simple neuron model first emerged in the 1940s with the work of McCulloch and Pitts [11]. The cybernetics movement, which ensued, attempted to combine biology, psychology, engineering and mathematics resulting in architectures for networks of neurons, which would perform a number of tasks. In 1949, Herb's book [5] put forward the theory of neural networks developing "internal representations" related to experience.

In the 1950s, research continued initially into the development of networks to perform specific tasks but this changed and the goal became to develop machines that could learn. By the end of that decade there had been a lack of significant developments and work in this field diminished considerably.

In the 1960s, interest was revived with the publication of a book by Rosenblatt [12] where he defined the concept of the perceptions and laid down many theories about them. In their simplest form, these processing elements, also known as, nodes or artificial neurons have the structure illustrated in figure 2.1. It was proved theoretically that a perceptron could learn to perform any task as long as it was possible to program it to do so.

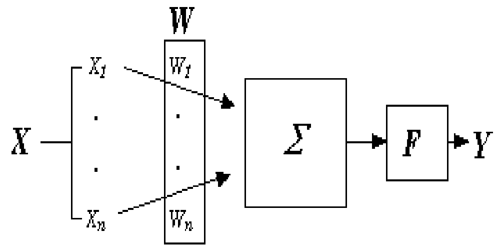


Figure 2.1 : The General Structure Of A Perceptron

A set of inputs (X1 to Xn) is applied to each node representing the inputs from the outside world or, alternatively, they may be outputs from other nodes. Each input is multiplied by a weight (W1 to Wn) associated with the node input to which it is connected and the weighted inputs are then summed, A threshold value (C) local for each node is added to the weighted summation and the resulting sum is then passed through a hard limiting activation function (F). The output of a node is therefore

The perception effectively splits the input patterns into two distinct regions with one region being represented by a 1 on the output and the other a 0. Rosenblatt's training algorithm for the perceptron would converge if the input patterns to the perceptron were linearly separable. The perceptron would therefore approximate the decision boundary between the two classes of outputs.

Perceptions were successfully trained to perform certain tasks but there were failures that could not be overcome. Minsk and Paper pointed out the serious shortcomings of perceptions [15] and interest in the study of neural networks again declined.

The Exclusive-Or (Ex-Or) function is a major illustration of the limitation of perceptions. For the ex-or function an output of 1 is generated if the inputs are {0,1} or {1,0} and an output of 0 is generated if the inputs are {0,0} or {1,1}. This is not a linearly separable function so the perceptron cannot learn it. A more complicated decision surface is required here and it was found that a curved decision surface is required to separate the two classes of inputs.

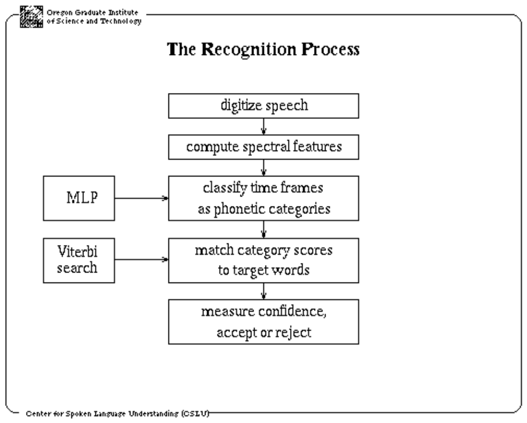
2.2 The Multi-Layer Perceptron.

Minsk and Paper had proposed a solution to the problem posed by functions such as the ex-or. They suggested that an extra layer of nodes with non-linear activation functions could be introduced. The output would now be a non-linear combination of the inputs so more complicated decision surfaces could be represented. The problem that remained was that no training algorithm was available to train such a network of perceptions at time

During the 1970s more research turned towards the representation of knowledge and away from learning and many new ideas were developed. Then, in the 1980s, there was a resurgence of interest in neural networks and it was during this time that an effective algorithm, called back propagation, for the training of multi-layer perceptron (MLP) structures was developed [16].

2.3 The Recognition Process using MLP's

There are four basic steps to performing recognition. Each step will be explained briefly


Oregon Graduate Institute of Science and Technology
The Recognition Process
digitize speech
compute spectral features
classify time frames as phonetic categories
match category scores to target words
measure confidence, accept or reject
MLP
Viterbi search
Center for Spoken Language Understanding (CSLU)

First, we digitize the speech that we want to recognize; for telephone speech the sampling rate is 8000 samples per second. Second, we compute features that represent the spectral-domain content of the speech (regions of strong energy at particular frequencies).

These features are computed every 10 msec, with one 10-msec section called a *frame*. Third, a neural network (also called an ANN, multi-layer perceptron, or MLP) is used to classify a set of these features into phonetic-based categories at each frame. Fourth, a Viterbi search is used to match the neural-network output scores to the target words (the words that are assumed to be in the input speech), in order to determine the word that was most likely uttered[21].

2.3.1 The Error Back Propagation Algorithm

Error back propagation is a gradient descent algorithm where weights and biases are adjusted to minimize a cost function equal to the mean square error in the network. For a 3-layer neural network with *N* input nodes and *M* output nodes, the network's weights are initially set to small random values. An input/output vector pair *p* is presented to the network with input vector

x_{p0} , x_{p1}, ..., x_{pN-1}
and target output vector
t_0 , t_1, ..., t_{M-1}

From this input vector an output vector is produced by the network which can then be compared to the target output vector. If there is no difference between the produced and target output vectors no learning takes place. Otherwise the weights are changed to reduce the difference. The weights are adapted using a recursive algorithm which starts at the output nodes and works back to the hidden layer.

First, we digitize the speech that we want to recognize; for telephone speech the sampling rate is 8000 samples per second. Second, we compute features that represent the spectral-domain content of the speech (regions of strong energy at particular frequencies).

These features are computed every 10 msec, with one 10-msec section called a *frame*. Third, a neural network (also called an ANN, multi-layer perceptron, or MLP) is used to classify a set of these features into phonetic-based categories at each frame. Fourth, a Viterbi search is used to match the neural-network output scores to the target words (the words that are assumed to be in the input speech), in order to determine the word that was most likely uttered[21].

The error in the network when training pair *p* is presented is defined as;

$$E_p = 1/2\ \Sigma(t_{pj} - o_{pj})^2 \qquad (2.1)$$

Where:

t_{pj} is the target value for the *jth* element of the output pattern from the training pair *p*.
o_{pj} is the actual value produced by the network for the j_{th} element of the output pattern when the input pattern from the training pair *p* is presented to its input.

The overall error is therefore

$$E = \sum_p E_p \qquad (2.2)$$

The input to node *j* is

$$net_{pj} = \sum_i w_{ji} o_{pi} \qquad (2.3)$$

Where:

wji *(t)*is the weight from the *itch* node of the previous layer to the *jth*node at time *t*than the input/output pair *p* is presented to the network.
Opiis the output of the *itch* node of the previous layer.

A non-linear activation function is employed in each node such that the output of node *j*

To implement a gradient descent the negative derivative of $E_p$with respect to w_{ij} must be proportional to the change in the weight w_{ij}, Δ_{pwji}. Therefore,

$$\Delta_p w_{ji}\ \alpha - \frac{\delta E_p}{\delta w_{ji}} \qquad (2.5)$$

Applying the chain rule to (2.5) gives

$$\frac{\delta E_p}{\delta w_{ji}} = \frac{\delta E_p}{\delta net_{pj}} \frac{\delta net_{pj}}{\delta w_{ji}} \qquad (2.6)$$

From (2.3) it can be seen that

$$\frac{\delta net_{pj}}{\delta w_{ji}} = \frac{\delta}{\delta w_{ji}} \sum_i w_{ji} o_{pi} = o_{pi} \qquad (2.7)$$

Applying the chain rule to (2.6) gives

$$\frac{\delta E_p}{\delta net_{pj}} = \frac{\delta E_p}{\delta o_{pj}} \frac{\delta o_{pj}}{\delta net_{pj}} \qquad (2.8)$$

From (2.1) it can be seen that

$$\frac{\delta E_p}{\delta o_{pj}} = \frac{\delta}{\delta o_{pj}} 1/2\ \Sigma(t_{pj} - o_{pj})^2 = -(t_{pj} - o_{pj}) = -\delta_{pj} \qquad (2.9)$$

From (2.4) it can be seen that

$$\frac{\delta o_{pj}}{\delta net_{pj}} = f_j'(net_{pj}) \qquad (2.10)$$

Substituting (2.7), (2.9) and (2.10) into (2.6) gives

$$-\frac{\delta E_p}{\delta w_{ji}} = -(-\delta_{pj}) f_j'(net_{pj}) o_{pi} = \delta_{pj} f_j'(net_{pj}) o_{pi} \qquad (2.11)$$

As mentioned earlier, the negative derivative of E_p with respect to w_{ij} must be proportional to the change in the weight w_{ij}, D_p to implement a gradient descent.

Therefore,

$$\Delta_p w_{ji}\ \alpha\ \delta_{pj} f_j'(net_{pj}) o_{pi} \qquad (2.12)$$

Let,

$$\Delta_p w_{ji} = \eta \delta_{pj} f_j'(net_{pj}) o_{pi} \qquad (2.13)$$

Where:
η is a small constant known as the learning rate.

Then,

$$w_{ji}(t+1) = w_{ji}(t) + \eta \delta_{pj} f_j'(net_{pj}) o_{pi} \qquad (2.14)$$

$w_{ji}\ (t+1) = w_{ji}\ (t) + hd_{pjfj'}(net_{pj})o_{pi}$

Where:
w_{ji} *(t+1)* is the weight from the *ith* node to the *jth* after adjustment.

Equation (2.14) is known as the standard delta rule and defines how weights are changed after the presentation of a training pair. The activation function must be non-linear since, otherwise, the neural network would perform a linear transformation at each layer and could therefore be reduced to its equivalent single layer network. The effectiveness of the extra layer of perceptions is then lost. The activation must also be differentiable as required by equation (2.11). The sigmoid function is the one most often used since it meets all the requirements

2.3.2 The Scaly Neural Network Architecture

A problem with fully connected neural networks is their size. When it comes to the practical implementation of neural networks size becomes an important factor. In hardware implementation this becomes a problem of scaling and the number of components required. In the case of computer simulation the problem comes with the large computational cost. The larger and more complex the network the longer it takes to train and once trained it takes longer for the network to perform its recognition task. For this work the concern is with the computational cost.

In general, it is not known what size of network works best for a given task and this problem is unlikely to be resolved since each task demands different capabilities from the network [6]. As with the decision about the number of hidden units, the size of network tends to be decided on a trial and error basis and the designers own experience. One such approach is to start with a fully connected network and then gradually Prune it. This involves removing those weights, which are close to zero and contribute little to the solution [4]. Another approach is to base the network architecture on prior knowledge of the structure of the input data. The network topology can then be arranged to reflect the data structure. Examples of this approach are Demichelis et al [1], Hampshire and Waibel. All of these structures are similar in that the input nodes are grouped into zones, which are connected to one node or a group of nodes in the hidden layer. Krause and Hackbut used' scaly type architecture to reduce the number of connections in a network. They showed that performance of a neural network does not necessarily improve as the number of connections between the nodes is increased. This architecture uses overlapped zones and was shown to support high recognition rates for isolated word recognition. Figure 2.3 shows an example of a neural network where scaly architecture has been applied between the input layer and the hidden layer. This is the approach adopted for the work here since the localized structure of the input zones is somewhat analogous to the cochlear processingwhich occurs, in the human ear [4]. A preliminary investigation into the ability of a scaly architecture neural network was carried out by A.D. Smith at the University of Newcastle Upon Tyne [14]. Smith's work suggested that further investigation of this network was required to better determine the effect of changing the parameters of the network. The work in this thesis is the continuation of that preliminary work.

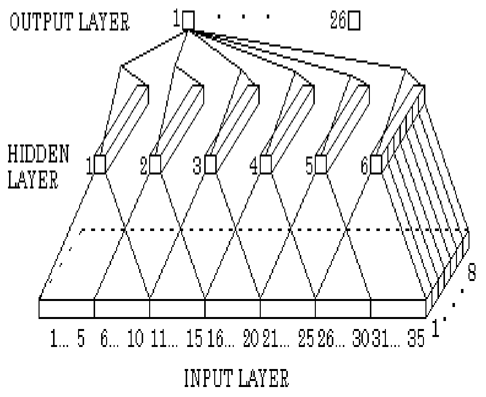


Figure 2.3 : Scaly Neural Network Architecture

The input data is processed and presented to the network such that successive feature vectors or frames are presented to the network inputs, each coefficient of the feature vectors being presented to one of the input nodes. The hidden nodes are also grouped into frames of the same size as those in the input layer. The input frames are further grouped into input zones with each frame of hidden nodes being assigned to a zone of input nodes. Each hidden node is connected to its equivalent node in the input frames of the zone associated with that hidden frame. The input zones overlap such that some of the input frames connected to one hidden frame will also be connected to the adjacent frames.

In the example network shown in figure 2.3, the network is capable of taking input patterns of 35 feature vectors containing 8 coefficients. To accommodate this, architecture with 280 input nodes is required. The scaly neural network therefore consists of an input layer with 35 frames each of 8 nodes. The number of frames in a zone is taken as 10 frames with an overlap of 5 frames so the number of frames required in the hidden layer is 6. The hidden layer therefore consists of 6 frames each of 8 nodes equaling 48 nodes in total. The number of output classes is 26, so 26 nodes are required in the output layer.

A disadvantage of the reduced connectivity is that some of the robustness of the neural network is lost but large savings in computational cost are gained. The output of each node is calculated using equation 2.1 and the number of operations required is shown in table 2.1. In this case, the computational cost is reduced by nearly two-thirds by employing the scaly architecture.

III. Preprocessing of Speech Data using time alignment algorithm

As mentioned in section 2.3, two of the major problems in speech recognition systems have been due to the fluctuations in the speech pattern time axis and spectral pattern variation [13]. Speech is greatly affected by differences in the speaker such as age and sexes as well their physical and psychological condition. The pitch and speed of the speaker will be altered by the way they feel at the time. If they are agitated, angry or short of time they will most likely speak at a faster rate and higher pitch than when they are calm and relaxed [7].

Speaking rate variation results in non-linear fluctuations in the speech pattern time axis. The length of the input pattern to the neural network in question is constrained by the number of input neurons to the neural network since this type of network architecture cannot be varied once trained. The input pattern vectors must be modified to fit the neural network while still retaining all their discriminating features.

Various algorithms are available to perform this time alignment of the input pattern to the neural network and the performance of the neural network is dependent upon the performance of the time alignment algorithm used. In this chapter, the various types of time alignment algorithms are described and their operation is outlined in detail.

3.2 Time Alignment Algorithms

The simplest time alignment algorithms are linear. However, they take no account of the importance of the feature vectors within the pattern vector when deleting or duplicating them to shorten or lengthen the pattern vector if required. Important features, therefore, may be lost in the process.

On the other hand, non-linear time alignment algorithms are more complicated and involve higher computational expenditure. Their advantage lies in the fact that they recognize important features and attempt to retain these features in the time aligned pattern vector. Two non-linear time alignment algorithms are dynamic time warping (DTW) and trace segmentation (TS)

3.2.1 Linear Time Alignment

Linear time alignment algorithms are the simplest algorithms to implement and they can be used for both expansion and compression of the speech pattern vector. There are various ways of implementing linear algorithms but all use the basic method of deleting feature vectors to shorten the speech pattern and duplicating feature vectors to length the speech pattern. An example is to duplicate or delete vectors at regular intervals along the pattern vector until the speech pattern is the correct size. An example of a linear algorithm used in conjunction with a neural network is that of Woodland [17]. Woodland achieved recognition rates of 91% for multiple speaker recognition and 88.3% for speaker independent recognition. And while we implement in javasapi4/5 we can get around 94%.

3.2.2 Dynamic Time Warping

The DTW algorithm removes timing differences between speech patterns by warping the time axis of one speech pattern until it maximally coincides with the other. All pattern vectors are warped against a reference pattern vector of the same category, which has the same number of feature vectors, as there are frames in the input layer of the neural network.

After the relevant feature extraction has taken place, speech patterns can be represented as a sequence of feature vectors,

$A = a_1, a_2,..., a_i, ..., a_K$ $B = b_1, b_2,..., b_j, ..., b_M$

Let *A* be the reference speech pattern and *B* be the pattern vector to be aligned against *A*. Figure 3.1 shows *A* and *B* developed against the *i* and *j* axes.

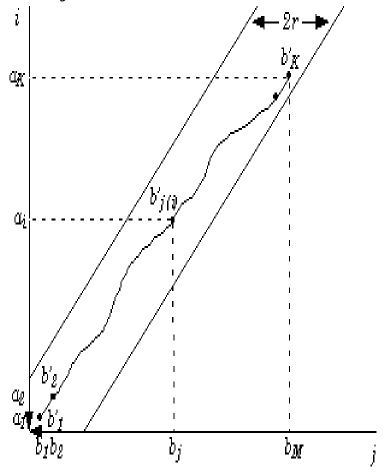


Figure 3.1 : Warping Function And Adjustment Window

Consider a warping function *F* between the input pattern time *j* and the reference pattern time *i*, where

$j = j(i)$

A measure of the difference between the two-feature vectors a_i and $b_j$is the distance

$d(i, j) = || a_i - b_j||$

When the warping function is applied to B this distance becomes

$d(i, j(i)) = || a_i - b'_j||$

Where:

b'j is the jth element of *B* after the warping function has been applied.

The weighted summation of these distances on the warping function is

$$E(F) = \sum_{i=1}^{K} d(i, j(i)) * w(i)$$

Where:

w (i) is a nonnegative weighting coefficient.

E reaches a minimal value when the warping function is determined to optimally align the two pattern vectors.

The minimum residual distance between *A* and *B* is the distance still remaining between them after minimizing the timing differences between them. The time-normalized difference is defined as follows:

$$D(A,B) = \underset{F}{\mathrm{Min}} \left[\frac{\sum_{i=1}^{K} d(i, j(i)) * w(i)}{\sum_{i=1}^{K} w(i)} \right]$$

Certain restrictions are applied to the warping function to ensure that it approximates the properties of actual time axis fluctuation. This means it should preserve all the significant linguistic features present in the speech pattern being warped. Such properties are monotonic and continuity [12]. These can be realized by imposing the following conditions on the warping function *E*.

Monotonic conditions
$J (i-1) <= j(i)$

Continuity conditions
$j (i) - j (i-1) <= 1$

Boundary conditions are imposed as follows;
$j(i) = 1$

$j(K) = M$

An adjustment window is implemented such that
$| i - j(i) | <= r$

Where *r* is a positive integer. The adjustment window condition is imposed since the time axis fluctuation does not yield excessive timing differences, therefore the algorithm must do likewise.

The final constraint imposed is the slope constraint condition. The results of this condition is that if $b_{j(i)}$ moves forward in one direction, *m* times consecutively, then it must step

n times in the diagonal direction before it can step any further in that direction. This ensures a realistic relation between *A* and *B* by ensuring that relatively short segments of one are not mapped to relatively long segments of the other. The intensity of slope constraint is measured as follows;
P = n/m

Increasing P more rigidly restricts the warping function slope but if it is too severe then time normalization is not effective.

The denominator of the time normalized distance equation can be defined as:

$$N = \sum_{i=1}^{K} w(i)$$

Since *N* is independent of the warping function *F* it can be put out of the bracket in *E(F)* simplifying the equation as follows:

$$D(A,B) = \frac{1}{N} \min_{F} \left[\sum_{i=1}^{K} d(i, j(i)) * w(i) \right]$$

Minimization can be achieved by applying dynamic programming principles. There are two typical weighting coefficient definitions, which allow this simplification, one for symmetric time warping, and one for asymmetric. In symmetric time warping the summation of distances is carried out along a temporarily defined time axis *l = i + j*. The previous discussion has described asymmetric time warping where the summation is carried out along the *i* axis warping *B* to be of the same size as *A*. In asymmetric time warping the weighting coefficient is defined by the following

W (i) = j(i) - j(i - 1)

When the warping function attempts to step in the direction of the *j* axis the weighting coefficient reduces to 0, since
J(i)=j(i-1)

Therefore,
W (i) = 0

And when the warping function steps in the direction of the *i* axis or the diagonal, then,
W(i) = 1

Then
N=K

Applying Dynamic programming principles to the simplified time normalization equation gives the following algorithm for calculating the minimal value of the summation:

The dynamic programming equation is

$$g_i(i, j(i)) = \min [g_{i-1}(i-1, j(i-1)) + d(i, j(i)) . w(i)]$$

The time-normalized distance is

$$D(A,B) = \frac{1}{N} g_K(i(K), j(K))$$

The initial condition is

*g1(1, 1) = d(1, 1) * w(1) = d(1,1)*

The dynamic programming equation for *P = 0* is

$$g(i, j(i)) = \min \begin{cases} g(i-1, j-1) + d(i, j(i)) \\ g(i-1, j) + d(i, j(i)) \end{cases}$$

The permissible paths through which the warping functions may move under this slope constraint are shown in figure 3.2(a).

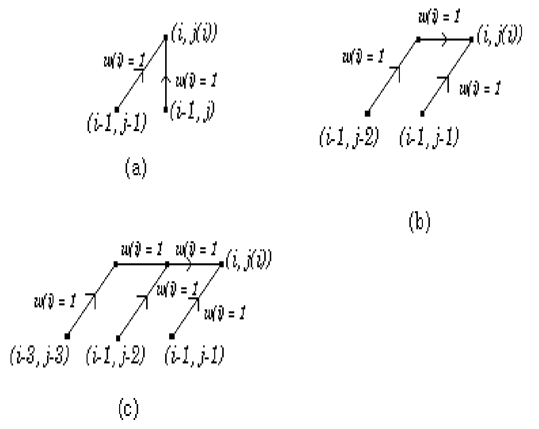


Figure 3.2 : Possible Warping Function Paths Under Different Slope Constraints

For *P = 1* the dynamic programming equation will be

$$g(i, j(i)) = \min \begin{cases} g(i-1, j-2) + (d(i, j-1) + d(i,j))/2 \\ g(i-1, j-1) + d(i, j) \end{cases}$$

The permissible paths through which the warping functions may move under this slope constraint are shown in figure 3.2(b).

For *P=2* the dynamic programming equation will be

$$g(i, j(i)) = \min \begin{cases} g(i-1, j-1) + d(i, j) \\ g(i-1, j-2) + (d(i, j-1) + d(i, j))/2 \\ g(i-1, j-3) + (d(i, j-2) + d(i, j-1) + d(i, j))/3 \end{cases}$$

The permissible paths through which the warping functions may move under this slope constraint are shown in figure 3.2(c).

The entailing result is that, for the initial condition, the first feature vector of *B* is taken as the first feature vector of the warped pattern vector b_1'. Subsequent feature vectors for the warped pattern vector are chosen such that the *nth* feature vector is that feature vector from the input pattern vector *B* closest to the *nth* feature vector of the reference pattern vector.

The asymmetric dynamic time warping algorithm only provides compression of speech patterns. This means that a linear algorithm must be used with any speech patterns that need to be expanded. This is acceptable since no feature vectors are being deleted when a linear algorithm is used in this form and there is no danger of losing important features of the speech pattern.

3.2.3 Trace Segmentation

Trace segmentation (TS) was introduced by Kuhn et al in 1981 and was used on its own and in conjunction with dynamic programming (DP) methods for isolated word recognition [9]. Two databases were used to train and test the speech recognition systems. The TS algorithm on its own performed worse than DTW on its own yielding error rates of around 10%. When the TS algorithm was used as a preprocessing step to DP it performed better than the DP alone. This method was also found to offer savings in computational expenditure of a factor of 10 or more over the DP algorithm used on its own. TS is based on the assumption that despite timing differences, for speech signals of the same category, fluctuations in the frequency spectrum with time will occur in the same sequence but over different lengths of time.

Assuming speech patterns of the same form as *A* and *B* in the previous section e.g.
$A = a_1, a_2, a_i, a_K$

And also assuming that each feature vector contains *N* features
$a_i = (a_{i1}, a_{i2}, ..., a_{il}, ..., a_{iN})$

Which can be represented as a point in *N*-dimensional space, the speech pattern can therefore be seen as a trace of points in *N*-dimensional space.

Where there is no change in frequency there will be a high density of points and where the frequency changes are rapid the points will be widely spaced. Removing those, which occur during the stationary portions of the speech pattern, can reduce the number of feature vectors in a trace. Kuhn et al achieved this by summing the Euclidian distances between successive feature vectors of a pattern vector to give the total length *D*off the trace [9]. If *F* feature vectors are required in the time aligned pattern vector then *D* is divided into *F-1* segments of length *L* where$L = D/(F-1)$

The most suitable vectors from the pattern vector are selected as follows. The first feature vector of the input pattern vector is taken as the first feature vector of the time aligned pattern vector. Successive feature vectors of the time aligned pattern vector are then chosen such that the Euclidian distance between each of them is as close to *L* as possible. The final time aligned pattern vector should therefore consist of *F* feature vectors with Euclidian distances between them of approximately *L*.

Lienard and Soong used the TS algorithm for the recognition of the P-set from the English alphabet which consists of the letters "P', "B", "T", "D", "V" and "Z" spoken by 4 speakers and obtained promising results [10]. Nadeu et al also applied TS to isolated word recognition of the ten Catalan digit words spoken by one speaker [11]. It was found that the recognition did not significantly degrade when TS was applied to the speech signal prior to recognition unless the number of frames removed from the speech signal by the algorithm was very high.

Gauvin and Mariani applied the TS algorithm to connected speech recognition comparing it to a linear time alignment algorithm and another non-linear time alignment algorithm, comparing fixed length and variable length versions of the algorithms [2]. The variable length trace segmentation algorithm gave the best recognition results. The TS algorithm was used in conjunction with a neural network by Demichelis et al [1] for the recognition of isolated digits. They achieved 86.4% performance and compared this to an hidden Markov model (HMM) approach where a 95% recognition rate was achieved. TS is used in conjunction with a neural network for the research . The DTW time alignment algorithm and the TS algorithm are compared when used in conjunction with neural networks.

As with the asymmetric DTW, trace segmentation algorithm only provides compression of speech patterns. This means that a linear algorithm must be used with any speech patterns that need to be expanded. As mentioned previously, this is acceptable since no feature vectors are being deleted when a linear algorithm is used in this form and there is no danger of losing important features of the speech pattern.

IV. Methodology

Using the multilayer back propagation algorithm used to test the network of the spoken words for the five speakers. Each speaker has to test the network by 11 words repeated four times. Each speaker, tests the word four times and the node with the higher number in the output will be the winner node. Comparing this node with the input word to the network will indicate the correct answer.

As a result, vocalizations can vary widely in terms of their accent, pronunciation, articulation, roughness, nasality, pitch, volume, and speed; moreover, during transmission, our irregular speech patterns can be further distorted by background noise and echoes, as well as electrical characteristics (if telephones or other electronic equipment are used). All these sources of variability make speech recognition, even more than speech generation, a very complex problem.

V. Implementation and Testing

Cloud Garden has produced a full implementation of Sun's Java Speech API, allowing a large range of SAPI4 and SAPI5 compliant Text-To-Speech and Speech-Recognition engines (in many different languages) to be programmed using the standard Java Speech API[19]. The Java TM Speech API, developed by Sun Microsystems in cooperation with speech technology companies, defines a software interface that allows developers to take advantage of speech technology for personal and enterprise computing. By averaging the inherent strengths of the Java platform, the Java Speech API enables developers of speechenabled applications to incorporate more sophisticated and natural user interfaces into Java applications and applets that can be deployed on a wide range of platforms.

Testing

The Java Speech API defines a standard, easy-to-use, and cross-platform software interface to state-of-the-art speech technology. Two core speech technologies are supported through the Java Speech API: speech recognition and speech synthesis. Speech recognition provides computers with the ability to listen to spoken language and to determine what has been said. In other words, it processes audio input containing speech by converting it to text. Speech synthesis provides the reverse process of producing synthetic speech from text generated by an application, an applet or a user. It is often referred to as text-to-speech technology [20].

Microphone: Desktop speech recognition systems get audio input through a microphone. Some recognizers, especially dictation systems, are sensitive to the microphone and most recognition products recommend particular microphones. Headset microphones usually provide best performance, especially in noisy environments. Table-top microphones can be used in some environments for some applications.

VI. Results and Discussion

Using the same multilayer back propagation algorithm to test the network of the spoken words for the five speakers. Each speaker has to test the network by 11 words repeated Four times. Each speaker, tests the word four times and the node with the higher number in the output will be the winner node. Comparing this node with the input word to the network will indicate the correct answer. So by testing the words said by each speaker the performance can be found by this equation.

Performance = Total succeeded number of testing words/Total number of words *100%.

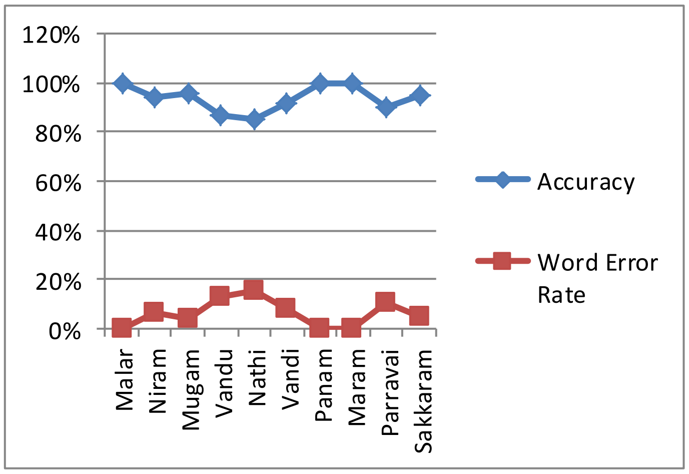
Word recognition or word accuracy percentage rates for each condition were found using the formula:

$$\text{Word Accuracy} = \frac{\text{\# of words correctly recognized} * 100}{(100 - \text{\# of words/commands skipped} - \text{\# of words mispronounced})}$$

Table 1: Results of Neural Network Based Speech Recognitions:

Word	Malar	Niram	Mugam	Vandu	Nathi	Vandi	Panam	Maram	Parravai	Sakkaram
Accuracy	100%	94%	96%	87%	85%	92%	100%	100%	90%	95%
Word Error Rate	0%	6%	4%	13%	15%	8%	0%	0%	10%	5%

1.1 Graph Represents the Accuracy and Error Rate of Words



VII. Summary

An introduction to the basic theory of neural networks in general and scaly neural networks in particular has been given in this chapter. This thesis deals with the performance of scaly neural networks for isolated word recegonition . The next we will deal with the specific problem of preprocessing the speech signals so that they can be presented in an appropriate form to the neural network inputs.

The concept of time alignment algorithms for preprocessing the speech pattern before it is presented to a neural network was introduced the two types of algorithm available, linear and non-linear, were described. Possible implementations of a linear algorithm were outlined and two non-linear algorithms were described in detail. The first was the widely used dynamic time warping algorithm which has been used both with and without neural networks for speech recognition and the less well known trace segmentation algorithm. Non-linear algorithms take into account the importance of feature vectors when adding or discarding them to change the length of speech patterns. This means they offer less danger of losing important features of the speech signal. For this reason, non-linear algorithms are the main interest of this thesis. The trace segmentation algorithm offers great computational savings over the dynamic time warping algorithm so in chapter 4 the performances of the two are compared to find out if there is any loss in performance incurred by the TS algorithm

VIII. Conclusions

The scaly type architecture neural network has been shown to be suitable for the recognition of isolated words in the form of letters of the English alphabet achieving high recognition rates for small vocabularies. When it came to the harder task of recognizing the whole English Words the scaly architecture could not compete with the fully connected architecture. Many more permutations of the scaly architecture than were looked at here are available and further investigation of the scaly architecture is carried

The trace segmentation was shown to be a good choice for use as a time alignment algorithm with the neural Network. It offers high computational saving over the dynamic time warping algorithm with very little or no drop in performance. In some cases it actually outperforms the dynamic time warping algorithm. Recognition of the words was carried out in speaker independent mode. In this mode the tested data is presented to the network are different from the trained data. The linear prediction coefficient (LPC8) with 12 parameters from each frame improves a good feature extraction method for the spoken words.

Since the first 12 in the cepstrum represent most of the formant information. In all speech recognition systems, the data is recorded using a noise-cancelling microphone, since this type of microphone is not available, the data was recorded using a normal microphone, but recorded in a closed room without any type of noise. Hence using the first type of microphone would give even better results.

Recognition of the letters was carried out in multiple speaker modes. The data will be used in speaker independent mode since this is very much suitable and useful in real world situations. In this mode the test speakers never before heard by the network have spoken data presented to the network. This is a much more practical implementation when we use high quality microphones in closed room we can achieve around 96% accuracy and performs better results for future use.

References

[1] Demichelis, P., Fissore, L., Laface, P., Micca, G., and Piccolo, E., "On The Use Of Neural networks For Speaker Independent Isolated Word Recognition", *Proc. Of The IEEE Int. Conf. On Acoustics, Speech and Signal Processing* , ICASSP '89, pp. 314-317, 1989.

[2] Gauvain, J.-L., and Mariani, J., "Evaluation Of Time Compression For Connected Word Recognition", *Proc. Of The IEEE Int. Conf. On Acoustics, Speech and Signal Processing* , ICASSP '84, Vol. 3, pp. 1-35, 1984.

[3] Hampshire, J. B., and Waibel, A. H., "A Novel Objective Function For Improved Phoneme Recognition Using Time Delay Neural Networks", *IEEE Trans. on Neural Networks* , vol. 1, No.2, pp. 216-228, June 1990.

[4] Harrison, T. D., and Fallside, F., ""A Connectionist Model For Phoneme Recognition In Continuous Speech", *Proc. Of The IEEE Int. Conf. On Acoustics, Speech and Signal Processing* , ICASSP '89, pp. 417- 420, Glasgow 1989.

[5] Hebb, D. O., "The Organization Of Behaviour", John Wiley & Sons, New York 1949.

[6] Hush, D. R., and Horne, B. G., "Progress In Supervised Neural Networks : What's New Since Lippmann?", *IEEE Signal Processing Magazine* , pp. 8-39, January 1993.

[7] Kirschning, I. "Continuous Speech Recognition Using The Time-Sliced Paradigm", MEng. Dissertation, University Of Tokushima, 1995.

[8] Krause, A., and Hackbarth, H., "Scaly Artificial Neural Networks For Speaker-Independent Recognition Of Isolated Words", *Proc. Of The IEEE Int. Conf. On Acoustics, Speech and Signale Processing* , ICASSP '89, pp.21-24, 1989.

[9] Kuhn, M. H., Tomaschewski, H., and Ney Hermann, "Fast Nonlinear Time Alignment For Isolated Word Recognition", *Proc. Of The IEEE Int. Conf. On Acoustics, Speech And Signal Processing* , ICASSP '81, pp. 736-740, 1981.

[10] Lienard, J-S., and Soong, F.K., "On The Use Of Transient Information In Speech Recognition", *Proc. Of The Int. Conf. On Acoustics, Speech And Signal Processing, ICASSP '84* , Vol. 2, pp. 1-17, 1984.

[11] Nadeu, C., Lieida, E, and Santamaria, M. E., "Trace Segemntation In A LPC-Based Isolated Word Recognition System", *Proc. Of MELECON 1985 : Volume 2 : Digital Signal Processing* , pp. 111-113, Madrid 1985

[12] Sakoe, H, Isotani, R., Yoshida, K., Iso, K. and Watanabe, T., "Speaker-Independent Word Recognition Using Dynamic Programming Neural Networks", *Proc. Of The IEEE Int. Conf. On Acoustics, Speech and Signal Processing* , ICASSP '89, pp. 29-32, 1989.

[13] Rosenblatt, R. "Principles Of Neurodynamics", Spartan Books, New York 1959.

[14] Smith, A. D. "Isolated Word Recognition Using Gradient Back-Propagation Neural Networks", MEng. Dissertation, University Of Newcastle Upon Tyne, 1990.

[15] Minsky, M. L., and Papert, S. A., "Perceptrons (Expanded Edition)" , MIT Press, 1988.

[16] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations By Error Propagation", *Parallel Distributed Processing: Explorations In The Microstructure Of Cognition. Vol. 1. Foundations* , pp. 318-362, MIT Press, 1986.

[17] Woodland, P.C., "Isolated Word Speech Recogniton Based On Connectionist Techniques", *Br. Telecom. Technol. J.* , vol. 8, no.2, pp. 61-66, April 1990.

[18] Kevin Gurney “An Introduction to Neural Networks” Ucl, Press Limited Taylor & Francis Group London, 1997.

[19] Raman, T.V. *Auditory User Interfaces: Towards the Speaking Computer*. Kluwer Academic Publishers, Boston, MA, 1997.

[20] Roe, D.B. and N.M. Wilpon, editors. *Voice Communication Between Humans and Machines*. National Academy Press, Washington D.C., 1994.

[21] HTTP://WWW.cslu.ogi.edu/tutordemos/nnet_recog/overview.ps,HTTP://WWW.Google.com

Authors

First Author – S.karpagavalli, Senior Lecturer, Department of Computer science (PG), PSGR Krishnnammal College for Women, Coimbatore, karpagam@grgsact.com
Second Author – P.V.Sabitha, Mphil Research Scholar, Department of Computer science, PSGR Krishnammal College for Women, Coimbatore, Sabitha.mphil@gmail.com