# VOICE EXTRACTION FOR SPEECH RECOGNITION BASED ON NEURAL NETWORK

Ummu Salmah Mohamad Hussin [1] and Saira Banu Omar Khan[2] Siti Nur Izyandiyana Ab Hadi[3]

[1] *Universiti Sultan Azlan Shah, dr_ummu@usas.edu.my, 05-7732257*
[2]*Universiti Pendidikan Sultan Idris, sairabanu@fskik.upsi.edu.my*
[3] *Universiti Sultan Azlan Shah, izyandiyana@usas.edu.my, 05-7732356*

**ABSTRACT :** *This paper presents the framework of malay isolated digit speech recognition system. The framework design is based on neural network method. One of the very famous methods to develop speech recognition system is a neural network (NN). NN is a computational paradigm model that consists of interconnected nerve cells. The NN is capable to classify noisy data, various pattern data, variable data streams, multiple data and overlapping, interacting and incomplete cues. It has been used for many different tasks because its capability in solving the non-linear problems. Thus, the versatility of NN makes them significant for automatic speech recognition (ASR). In speech recognition, the speech signal needs to be processed before being applied to neural network classifier and this process is known as preprocessing phase. It is well known preprocessing that is one of the data trimming tools used in preprocessing phase. In this study we will discuss fundamental design of preprocessing for speech recognition based on NN.*

**KEYWORDS** : *neural network, non-linear problems, automatic speech recognition, neural network classifier, preprocessing.*

## 1. INTRODUCTION

An automatic speech recognition (ASR) system is a goal in speech research for more than 6 decades. The ASR research began since 1900's and started to attract much interest in the market in the past couple of years because of the advancements of its methods, algorithm and related technology (Srinivasan and Brown, 2002; Padmanabhan and Picheny, 2002). Several interesting and useful software/hardware were produced to simplify the human tasks, such as the IBM via voice, Dragon Naturally Speaking, L&H's voice Xpress, Philip's Free Speech, Sensory Circuit and etc.

One of the very famous methods to develop speech recognition system is a Neural Network (NN). NN is a computational paradigm model that consists of interconnected nerve cells (Kevans and Rodman, 1997). The NN is capable to classify noisy data, various pattern data, variable data streams, multiple data and overlapping, interacting and incomplete cues. It has been used for many different tasks because its capability in solving the non-linear problems. Thus, the versatility of NN makes them significant for ASR.

In speech recognition, the speech signal needs to be processed before being applied to NN classifier and this process is known as preprocessing phase. In this study we will discuss fundamental design of preprocessing for speech recognition based on NN.

## 2. PREPROCESSING

The preprocessing module prepares the speech signal in a digitized form before the NN module processes it. Usually, the preprocessing module consists of endpoint detection, time axis normalization, and feature extraction and normalization process. Each process is discussed below.

### Endpoint Detection

Endpoint detection plays an important role in speech application and has been studied for several decades. It is an

essential task in speech recognition systems to separate the speech segments from non-speech segments (Zhang *et al.*, 1997). Non-speech segments consist of speech, silence and other background noises. The method of detection of the speech signal embedded in various types of non-silence and background noise is also known as speech detection or speech activity detection (Li *et al.*, 2002). The process of inaccurate detection of the beginning and ending boundaries of test and reference pattern will be the cause of errors in speech recognition (Shin *et al.*, 2000; Ying *et al.*, 1993). The main function of the endpoint detection is to discard the extraneous data in order to increase the recognition rate and to accelerate the computation time (Hahn and Park, 1992).

According to Rabiner and Sambur (1975), it is difficult to detect the beginning and end of an utterance, especially when there are (Rabiner and Sambur, 1975): -

- Weak fricatives (/f/, /th/, /h/) at the beginning or end.
- Weak plosive bursts (/p/, /t/, /k/) at the beginning or end.
- Nasals at the end.
- Voiced fricatives that become devoiced at the end of words.
- Trailing off of vowel sounds at the end of an utterance.

There are various methods to perform the task of endpoint detection. A short list includes energy threshold, pitch detection, spectrum analysis, zero crossing rate, periodicity measure, hybrid detection and fusion. Two widely used methods are energy and energy and zero crossing (conventional method), and these methods are the earliest methods, introduced by Rabiner and Sambur (1975) to detect the speech boundary.

In this study, the endpoint detection is emphasized in high signal to noise ratio (SNR). In other words, the endpoint detection needs to remove the silence speech only because the entire recording is done in silence condition. Few researchers have proposed several related works on enhancing algorithm for high SNR condition.

**Time Axis Normalization**

The NN structure requires a fixed number of input neurons. However, in human speech, the uttered word varies in duration. To cope with this problem, time-axis normalization is implemented. It is done after the starting and endpoints have been detected. There are two types of time axis normalization, these include linear and non-linear. The simplest is linear time axis normalization. Non-linear alignment is more complicated and involves higher computational method. The advantage of linear is faster, simple and can be used for both expansion and compression of the speech pattern vector. Meanwhile non-linear time axis normalization attempts to retain the important features in the time aligned pattern vector. We employed linear time axis normalization because of its simplicity. Another advantage is that it performs faster than non-linear method (Creaney, 1996).

**Feature Extraction**

Various types of feature extraction methods are used in ASR to represent the speech model. These include fast fourier transform, wavelet, auditory preprocess, etc. The most common short-term spectral measurements currently used are the linear prediction coding (LPC) (Woo *et al.*, 2000). The LPC has been a popular and efficient feature extraction method to represent speech model at low bit rate and is widely used (Barnwell, 1996). The main advantages of this method are: -

- Robust, reliable and accurate method to extract important speech parameter.
- Convenience to implement and use a small storage because it consists of simple and concise formula, thus it is easy to be produced in both software and hardware.
- Powerful method to isolate the speech to frequency resonant and amplitude, and do consider the useful features needed to obtain compressed speech data.

Based on above reasons LPC has been used in a large number of recognizers. For instance, the recent work using LPC is in real task as in speech recognition chip for monosyllable (Nakamura *et al.*, 2001) and distributed speech recognition system (Raj *et al.*, 2001). Matsumo and Moroto (2001) proposed improved LPC namely mel-LPC. They adopted the LPC in large vocabulary continuous speech recognition. Other applications that used LPC are in isolated vowel recognition (Byorick *et al.*, 2002) and Chinese syllable recognition (Xiaoming and Baoyu, 1998). Other works include Mei and Sun (2000) and Kil *et al.* (2002). In addition, most of the researches in Malay speech employ the LPC method as a

feature extraction tool. Therefore, it is believed that LPC still a powerful method to represent the speech features accurately and efficiently.

**Normalization**

The normalization stage is the final stage in preprocessing before the speech data is fed into NN module. Normalization is a data transformation into certain range. It can be linear or non linear, depending on the distribution of the data (Rafiq *et al.*, 2001). Normalization is utilized in this study because, the multilayer perceptron (MLP) is able to process the data in a certain domain, usually 0,1 or –1,1 (Rafiq *et al.*, 2001). This is because the neurons in the MLP should be in the interval between 0 and 1 or –1 and 1 to fit the activity domain of the neurons that performed by the activation function which interval usually from –1,1 and 0,1. Moreover the input variables should be kept small in order to avoid saturation caused by the sigmoid function (Rafiq *et al.*, 2001). In other words the normalization is done to accelerate learning and to avoid computational problems. Rafiq *et al.* (2001) claim that normalization of the inputs to the range [-1, +1] greatly improves the learning time of the NN, as these values fall in the region of the sigmoid transfer function where the output is most sensitive to variations of the input values. This claimed is proved by our experiments, where the normalization that have a range between –1 and 1 converge very fast compared to range 0 and 1.

The most widely used normalization method for NN is (Zhang *et al*., 1998): -

- Linear transformation
- Statistical normalization
- Simple normalization

Several researchers have also proposed a new normalization in various applications. Rafiq *et al.* (2001) proposed a normalization method to keep the normalized data away from the sigmoid extreme boundaries of 1 and 0 for engineering applications. Baron and Girau (1998) introduced parameterized normalization that applied to wavelet networks.

Even though the normalization method is widely used in NN for speech application, it is seldom described in experiments reports. Data are assumed as already normalized such that the authors do not even mention it. Several studies by a few researchers use normalization in speech recognition including Salleh *et al.* (1994) and Hussain (1997). They used linear transformation to transform the speech data before feeding to NN. Pedersen (1997) and Vieira *et al.* (1997) used variance normalization (statistical normalization).

One of the reasons is that the speech does not emphasized in normalization method, the researchers do not alert that input normalization has an effect on learning and recognition phase. Throughout this study, it shows that performance of MLP network is greatly influenced by normalization. As such, this study investigates how to produce the fastest and highest recognition rate normalization method compared to conventional methods (range I, range II, statistic and simple method). Finally a new normalization method for speech recognition is presented.

## 3. FRAMEWORK

This section presents the framework of the isolated digit speech recognition system. The system is a prototype and offline consisting of 3 main phases namely; preprocessing, learning and recognition. A MLP classifier with Backpropagation algorithm and LPC speech-processing method is adopted in this system. In this study TI46 data sets data sets is used.

**System Design**

The system design shown in Figure 1**,** consists of preprocessing, learning and recognition phase. The preprocessing is intended to process the data set before being fed to NN.

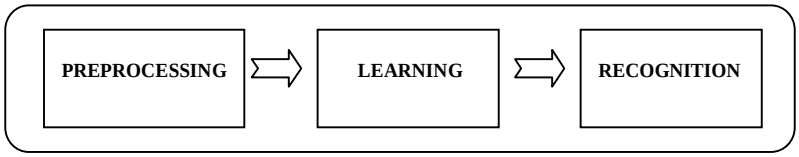

**Figure 1**. The General System Design of Isolated Digit Speech Recognition System

In learning phase, the NN is trained to use the pattern of the given speech data while in recognition phase, NN recognizes the speech data. The detailed flow of the system design is shown in Figure 2. The following section discusses the flow of each phase in detailed.

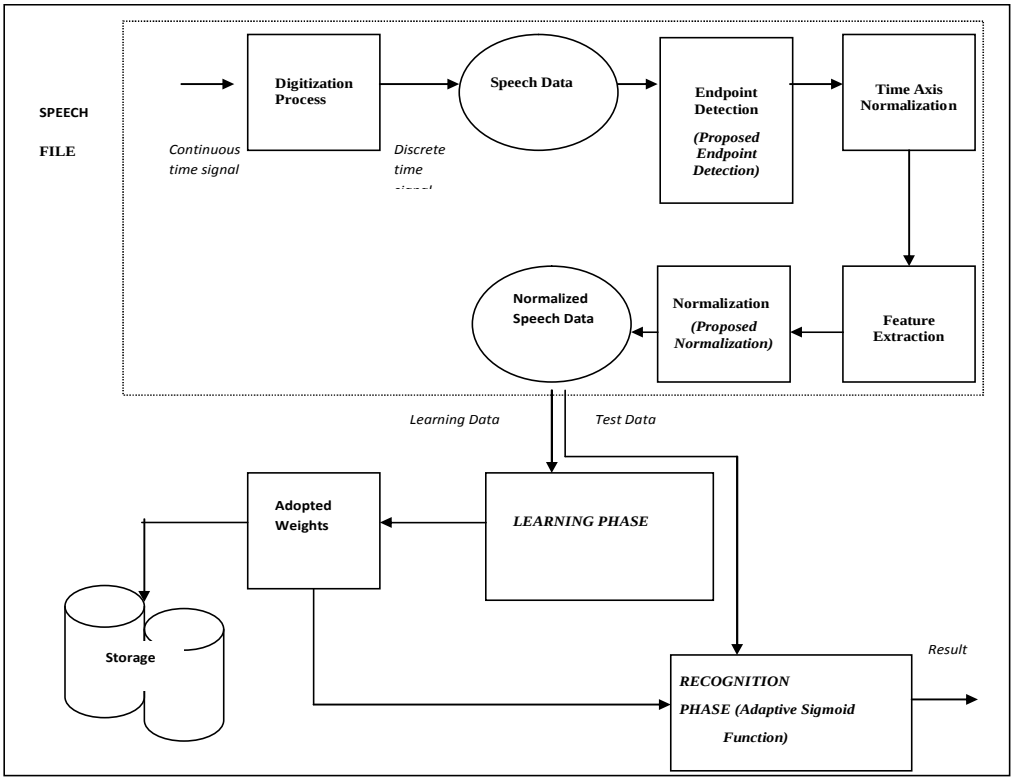


**Figure 2**. Detailed Flow of Malay Isolated Digit Speech Recognition Prototype

**Preprocessing**

This phase comprises of five stages: digitization process, endpoint detection, time axis normalization, feature extraction and normalization. These stages are explained accordingly.

**Digitization Process**

The analog form of utterance of Malay and TI46 data set was digitized into 8 bits (mono) and 16 bits (stereo) resolution respectively. The digital pattern yielded was converted to samples as shown in Figure 3. The value of these samples will be utilized further in the following stages. At the second final stage of the preprocessing phase these samples were converted to LPC parameter before being fed to the NN.

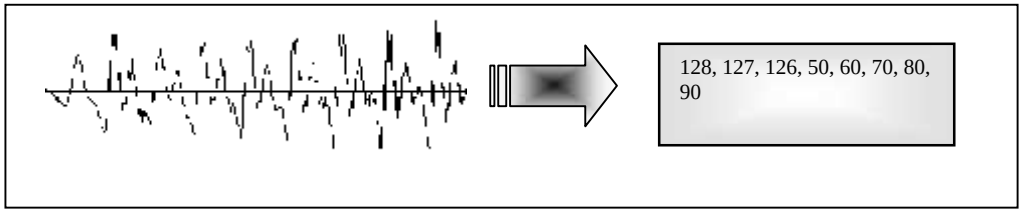


**Figure 3**. Digitized Spoken Input

**Endpoint Detection**

For the purpose of this study, 3 methods were used. These included the Rabiner and Sambur approach (energy and energy and zero crossing methods) and the variance method (proposed method). The measurements of the speech signal using those methods are time-domain measurements and calculated on a frame-to-frame basis. The time domain measurements directly involve the waveform (Figure 4)**,** where  is discrete-time signal and *n* takes integer values. Furthermore, energy, energy and zero crossing and variance method will be discussed in detail.

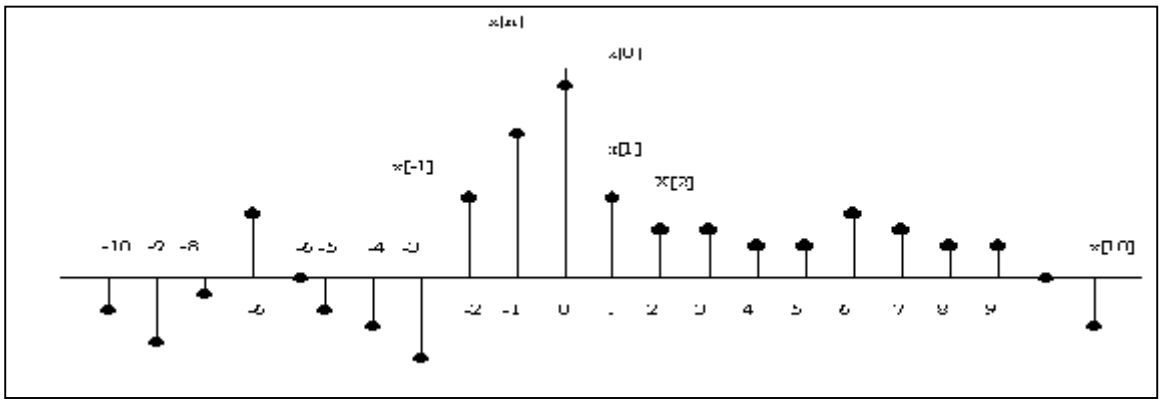

**Figure 4**. Graphical Illustration of Discrete-Time Signals

**Energy Method**

Energy measures the loudness of the sounds and provides a basis to differentiate voiced speech segments from the unvoiced speech segments. Commonly, the energy for voiced data such as **a**, **e** and **o** is much higher than silence. The energy of unvoiced for instance, **s**, **and f** is lower than voice sounds but higher than silence. The energy rate of the speech is defined in this study as the number of average magnitude of energy per 30ms interval and overlapping at 10ms. This type of interval is sufficient to capture important speech features (Rabiner and Juang, 1975).

$$x_l \; \mathrm{X} \sum_{n\mathrm{X}0}^{N} \left| S(ml\,\Gamma\, n) \right| \qquad (3.1)$$

$$M_l \; \mathrm{X} \frac{x_l}{N\;\mathrm{Z}1} \qquad (3.1a)$$

where,

| | |
|---|---|
| *S* | indicates the speech samples. |
| m | speech samples overlapping at 10ms. |
| M | average magnitude of energy. |
| n | 0,1,…N-1 (N speech samples) |
| l | 0,1,…L-1 (Intervals) |
| N | speech samples at 30ms per interval |

Before the endpoint detection process, the mean of the average magnitude was computed to give a statistical characterization of the background noise. It was assumed that the first 100ms of interval contained no speech. This information was further used to compute the peak energy $fIMX A$ for the entire interval in each speech sample and the silence energy ( $IMN$ ). Subsequently the $IMX$ and $IMN$ were used to set two thresholds: upper threshold ( $ITL$ ) and lower threshold ( $ITU$ ), by using:

$$I1 \; X \; 0.03 \mid fIMX \; Z \; IMN \; A \Gamma \; IMN \tag{3.2}$$

$$I1 \; X \; 0.03 \mid fIMX \; Z \; IMN \; A \Gamma \; IMN \tag{3.3}$$

$$ITL \; X \; MIN \; fI1, I2 A \tag{3.4}$$

$$ITU \; X \; 5 \mid ITL \tag{3.5}$$

**Equation (3.2)** shows *I1* to be a level, which is 3% of the peak energy, whereas **Equation (3.3)** shows *I2* to be a level set at four times the silence energy. The lower threshold, *ITL* (**Equation 3.4**), is the minimum of this conservative energy threshold, and the upper threshold, *ITU* (**Equation 3.4**), is five times the lower threshold (Rabiner and Sambur, 1975).

The average magnitude profile was searched to find the interval, in which it exceeded the conservative threshold, $ITL$ . It is assumed that the beginning and end points lie outside this interval. Then working backwards from the point at which $E_n$ first exceeded the threshold $ITU$ , the point (labeled *N1* in Figure 5) where $E_n$ first fell below a lower threshold *ITL*, is selected as the beginning point.

A similar procedure was followed to find the tentative endpoint *N2*. This double threshold (ITL and ITU) was performed to avoid dips in the average magnitude function. It is to ensure that it does not falsely signal the endpoint. For *energy* and *zero crossing* algorithms, at this stage, it is reasonably safe to assume that the beginning and ending points are not within the interval *N1* to *N2*. The final result was gained after implementing the zero crossing algorithms (Rabiner and Sambur, 1975).

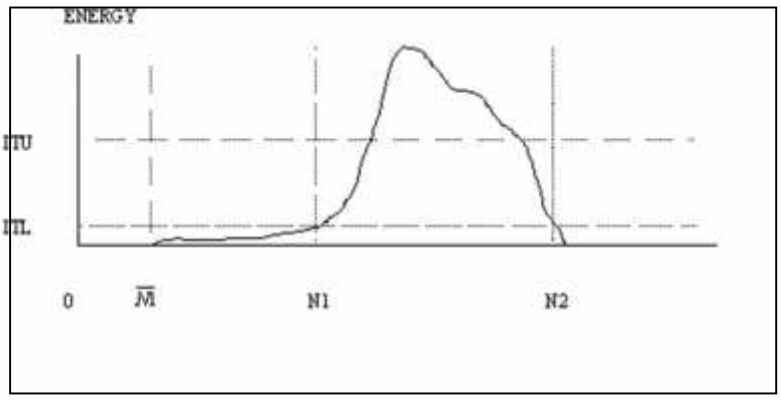


**Figure 5.** Typical Example of Energy for a Word Beginning with a Strong Fricative

**Zero Crossing Method**

This method was used to count the frequent of the signal that crosses the zero axes. In other words, a zero crossing occurs if successive samples have different algebraic signs. Zero crossing is very useful method for detecting the occurrence of unvoiced speech. The unvoiced speech is produced due to excitation of the vocal tract by the noise-like source at a point of construction in the interior of the vocal tract and shows a high zero crossing count. Due to this reason, the endpoint detection algorithm was refined by zero crossing after the energy measurements located the actual begin or end point. Throughout this study, the zero (level) crossing rate of the speech, $Z_l$, was defined as the number of zero (level) crossing per 30ms interval and overlapping at 10ms.

$$Z_l = \frac{1}{2}\sum_{n=0}^{N} \left| \operatorname{sgn}[S(ml+n)] - \operatorname{sgn}[S(ml+n-1)] \right| \tag{3.6}$$

where,

$$\operatorname{sgn}[S(ml+n)] = \begin{cases} 1 & S(ml+n) \geq 0 \\ -1 & S(ml+n) < 0 \end{cases}$$

Z zero crossing rate

m speech samples overlapping at 10ms.

n 0,1,…N-1 (N speech samples)

l 0,1,…L-1 (Intervals)

N speech samples at 30ms per interval

The zero crossing count of silence was expected to be lower than unvoiced speech, but relatively comparable to that of voiced speech. The standard deviation zero crossing was analyzed to obtain statistical characterization of the background noise as in energy method. This information was used to compute the zero crossing threshold (*IZCT)* by assuming that the first 100msec contained no speech or silence speech. A zero crossing threshold (*IZCT)* for unvoiced speech was chosen as the minimum of a fixed threshold. The *IZCT* was obtained by the sum of the mean zero crossing rate during silence ( $\overline{IZC}$ ), plus twice the standard deviation of the zero crossing rate during silence (**Equation (3.7)**). Rabiner and Sambur (1975) give clear explanation about zero crossing method.

$$IZCT \; \mathrm{X} \, MIN \{IF, \; \overline{IZC} \; \Gamma \; 2\pm \overline{IZC} \} \tag{3.7}$$

Using the zero crossing algorithms, the searching starts backwards from *N1* (forward from *N2*) comparing the zero crossing rates to a threshold *IZCT* (determined from the statistics of the zero crossing rates for the background noise) as shown in Figure 6. This is limited to 25 frames preceding *N1* (following *N2*). If the zero crossing rates exceeded the threshold by 3 or more times, the beginning point *N1* will move back to the first point at which the zero crossing threshold exceeded ( $\overline{N1}$ ). Otherwise, *N1* is defined as the beginning. A similar procedure is followed at the end of utterance to remove the silence samples.

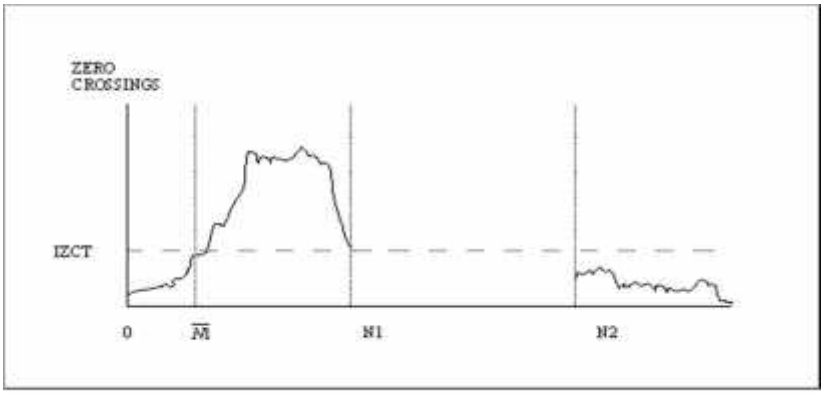


**Figure 6**. Typical Example of Zero Crossing Rate for a Word Beginning with a Strong Fricative

**Time-Axis Normalization**

The time-axis normalization was done after the starting point and endpoint had been detected. In this stage, the average speech length was calculated in order to obtain a modification factor. This stage led to the compression or enlargement of the sampled waveform, which relying on the modification factor. As such, the whole speech signals with the same number of frames were obtained as shown in Figure 7. Thus, fixed size frames were obtained from each utterance. The algorithm for time-scale is described below.

**Step 1**: Compute the modification factor:

a = ¦ / ±

The calculation for the length of samples is as follows

~ = ² (in milliseconds) *X* © (KHz)

**Step 2**: Compute the new analysis frame size:

¡ = *(Samples) X* © (KHz) *X* ª (KHz)

**Step 3**: Compute the overlapping portion

¤ = · *(samples) X* © (KHz) *X* ª (KHz)

where,

ª modification factor

¦ actual length of samples

± normalized length of samples (based on average speech length)
~ length of samples.
² speech duration.
© sampling rate.
¡ new window.
normalized window.
¤ new overlapping portion.
· normalized overlapping portion.

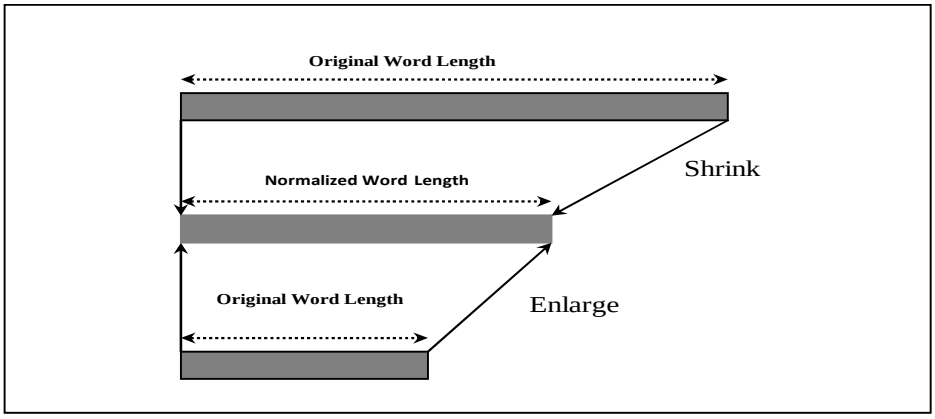


**Figure 7**. Time Axis Normalizations of The Speech Length

**Feature Extraction**

The feature extraction was performed in this stage to represent the speech model. In this study, the LPC was used as a feature extraction method to represent the input data and decomposed into main five steps as given below:

- Pre-emphasis.
- Frame blocking.
- Windowing.
- Autocorrelation analysis.
- LPC analysis.

These steps were implemented to prepare the speech signals for further used in MLP.

The function of pre-emphasis is to flatten the speech signals. Here, the speech signals were sent to low pass filter. The formulation is given as,

$$\overline{S}(n) = S(n) - a\,S(n-1) \qquad (3.8)$$

Where,

$\overline{S}$ speech signal after pre-emphasis,
$S$ speech signal prior to pre-emphasis,

$n$ number of samples,

$\bar{a}$ is the gain of value, normally between 0.9-1.0. In this study $a = 0.95$ was used because this was commonly used value (Rabiner, 1975).

The second step in LPC was to block the preemphasized signals, $\bar{S}(n)$ into frames. These frames consisted of *N* samples, with the adjacent frames, which were separated by *M* samples. Figure 8 shows the frame blocking process, where M= (1/3) x N**.** The first frame comprises of the first *N* speech samples. The second frame is begun with M samples after the first frame and overlapped by N-M samples. Similarly, the third frame is begun with 2M samples after the first frame and overlapped by N – 2M samples. This process was repeated until the entire speech signals are computed for all frames.

*S* denotes the $l_{th}$ frame of the speech signal. Let *L* frames represent the entire speech signal, hence the general equation for frame blocking is:

$$x_l(n) = \bar{S}(Ml + n) \tag{3.9}$$

where

$x_l$ $l_{th}$ frame of speech .

$n$ 0,1,…, N-1 samples.

$l$ 0, 1, …, L-1 frames.

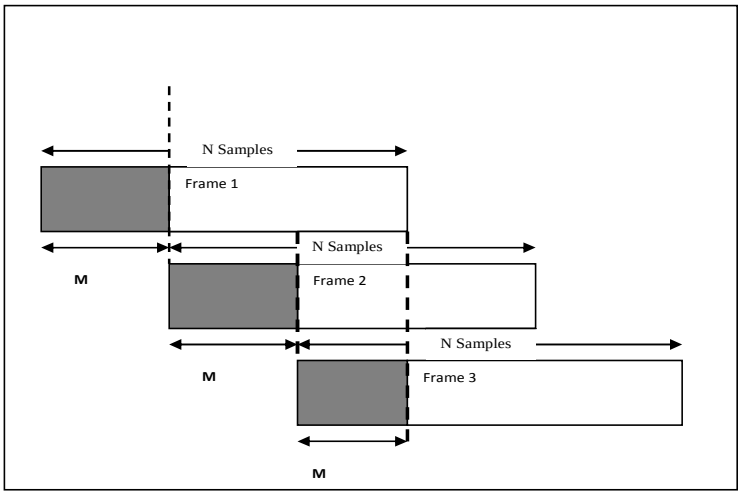


**Figure 8.** Frame Blocking of Speech into Overlapping Frames (after Rabiner, 1993)

Next, the windowing process was performed to ensure that the speech produced was the same as the original speech. The process in this step was employed by overlapped adjacent frames to minimize the signal discontinuities at the beginning and end of each frame. Otherwise, the loss of some speech data would be encountered if there were no overlap between frames occurs. This process was formulated by multiplying the blocked-frame signal, $x_l(n)$ with Hamming window function $w(n)$. The formula for this function is given as,

$$\bar{x}_l(n) = x_l(n)\, w(n), \qquad 0 \le n \le N-1 \tag{3.10}$$

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1 \tag{3.11}$$

where,

$\bar{x}$ windowed signal .
$w$ Hamming window function .
$l$ 0, 1, …, L-1 frames.

The function of autocorrelation analysis used to auto-correlate each frame of the windowed signal is shown as,

$$r_l(m) = \sum_{n=0}^{N-1-m} \bar{x}_l(n)\, \bar{x}_l(n+m), \qquad m = 0,1,\ldots p \tag{3.12}$$

.

where,

$p$ is the order of the LPC analysis.
$r$ autocorrelated windowed signal .
$l$ 0, 1, …, L-1 frames.

$p$ is the highest autocorrelation value. Typical values of $p$ range from 8 to 16. In other word, $p$ represents the human vocal tract. The LPC analysis was performed to convert each frame of $p+1$ autocorrelations into an "LPC parameter set". LPC coefficients used the LPC parameter. The formal conversion method known as Durbin's method was applied to produce this parameter. The description of this method is given below. Note that for convenient subscript of $r$ was omitted. The equation of LPC is given below:

$$E^{(0)} = r(0) \tag{3.13}$$

.

where,

$E$ error prediction
$r$ autocorrelerated windowed signal

$$k_i = \frac{r(i) - \sum_{j=1}^{i-1} \tilde{}_j^{(i-1)}\, r(|i-j|)}{E^{(i-1)}}, \qquad 1 \le i \le p \quad \text{and} \quad 1 \le j \le i-1 \tag{3.14}$$

where,

$k$ reflection coefficients
$\tilde{}$ LPC coefficients
$E$ error prediction
$r$ v bv autocorrelerated windowed signal

$$\tilde{}_j^{(i)} = k_i \quad , \qquad 1 \le i \le p \quad \text{and} \quad 1 \le j \le i-1 \tag{3.15}$$

where,

$\tilde{}$ LPC coefficients

$k$ reflection coefficients

$$\tilde{}_j^{(i)} = \tilde{}_j^{(i-1)} - k_i \tilde{}_{i-j}^{(i-1)}, \qquad 1 \le i \le p \quad \text{and} \quad 1 \le j \le i-1 \tag{3.16}$$

where,

$\tilde{}$ LPC coefficients

$k$ reflection coefficients

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}, \qquad 1 \le i \le p \tag{3.17}$$

where,

$E$ error prediction

$k$ reflection coefficients

**Equation (3.14)-(3.17)** are solved recursively for *i = 1,2…,p* and the final solution is given as: -

$$a_m = \text{LPC coefficients} = \tilde{}_m^{(p)}; \quad 1 \le m \le p \tag{3.18}$$

The LPC coefficients further are normalized between 0 to 1 or –1 to 1 before fed to MLP.

**Data Normalization**

Normalization was performed to scale the speech data (LPC coefficients) into certain range before being fed to NN. Usually the range is between [–1,1] and [0,1] to suit with the output from neurons by using a linear scale to transform the data in the [0,1] (Aksoy, 2000). The equation is given as,

$$x' = \frac{x - l}{u - l} \tag{3.19}$$

where,

$x'$ is normalized feature,

$x$ is unnormalized feature,

$u$ is upper bound of the unnormalized feature,

$l$ is lower bound of the unnormalized feature.

Finally the normalized data will be fed to NN.

**Learning**

The learning task was carried out in this stage by using standard BP algorithm. The BP algorithm was adopted to minimize the error of the output computed by the neural net, which is based on supervised learning. In other words, the BP algorithm was learned by adjusting the weights using gradient descend method.

**Architecture**

In this study, the neural network architecture consists of three-layered MLP network (input layer, hidden layer and output layer) with error back propagation learning method (supervised learning). Figure 3.9 shows the architecture of the MLP. The neuron $X$ , $Z$ and $Y$ represent the input layer, hidden layer and output layer respectively. The neurons labelled as $b$ are the bias neuron. A circle represents each neuron and interconnections (weight) between neurons. Arrows represent these interconnections. The output of the bias neuron is always 1. For instances $v_{1j}$ , $v_{ij}$ , and $v_{nj}$ are a weight for $Z_j$ neuron. Meanwhile the weight for $Y_m$ are represented by $w_{1m}$ and $w_{jm}$ . The bias on $Z_j$ , $Y_m$ neuron is $v_{01}$ and $w_{01}$ respectively.

The nomenclature that is used for the BP algorithm in the following discussion is listed as follows.

$x$ input pattern $x = (x_1,...,x_i,...,x_n)$

$t$ output target: $t = (t_1,...,t_k,...,t_m)$

$\delta_k$ error information regarding to the error at neuron $y_k$

$\delta_j$ error information for hidden neuron $z_j$ due to the backs propagation error information from the output layer

$\alpha$ learning rate (0 to 1)

$\mu$ momentum (0 to 1)

$X_i$ input neuron $i$ :

for an input neuron, the input and output are same, namely, $x_i$

$v_{0j}$ bias on hidden unit $j$

$Z_j$ hidden unit $j$

$w_{ok}$ bias on output unit $k$

$Y_k$ output unit $k$

**Back propagation Learning Process and Algorithm**

The learning process for BP comprises of 3 stages, they are:

- Feedforward of the input-learning pattern.
- The BP of the associated error.
- The adjustment of the weights.

The input neuron $fX_i, i\ \mathrm{X}1,...,nA$ received an input signal and sent it to the hidden neurons $fZ_j, j\ \mathrm{X}1,...,jA$. Then, the hidden neuron computed its activation function and broadcasted $z_j$ to each output unit. At output layer, each output neuron $fY_k, k\ \mathrm{X}1,...,mA$ computed its activation $y_k$. As a result, the obtained activation was the response of the net for the given input pattern.

In this stage, each output neuron compared its activation $y_k$ with its target value $t_k$. Such process was done to identify the error for that pattern with that neuron. Subsequently, the factor $¢_k$ was computed based on this error. The function of $¢_k$ was divided into two tasks. First, it was used to distribute the error at output unit $Y_k$ back to all neurons in the previous layer. Secondly, it is used to update the weights between the output and the hidden layer. A similar procedure was followed to compute the factor $¢_j$ for each hidden unit $Z_j$. It was not required to propagate the $¢_j$ to the input layer. Thus $¢_j$ was used to update the weights between the hidden layer and the input layer only.

The $¢$ factors were used to adjust the weight for all layers simultaneously. $¢_k$ and $z_j$ were used to adjust the weight $w_{jk}$ (from hidden unit $Z_j$ to $Y_k$). Meanwhile, the adjustment for weight $v_{ij}$ (from input neuron $X_i$ to hidden neuron $Z_j$) was based on factor $¢_j$ and activation $x_i$ of the input neuron.

**Standard BP algorithm**

**Step 0**: Initialize the connections weights
**Step 1**: While stopping condition is false, do steps 2-10.
**Step 2**: For each input patterns, do steps 3-8.
**Feedforward:**
**Step 3**: The input from each input neuron $fX_i, i\ \mathrm{X}1,...,nA$ is sent to hidden layer.
**Step 4**: Compute the net input to each hidden neuron

$fZ_j, j\ \mathrm{X}1,...,jA$ by summing all the inputs and multiply with weight.

$$z\_in_j\ \mathrm{X}b_j\ \Gamma\ \sum_{i\mathrm{X}1}^{n} x_i v_{ij} \tag{3.20}$$

Compute the activation function of a hidden neuron and broadcast it to output layer.

$$z_j\ \mathrm{X}\ f\,fz\_in_j A \tag{3.21}$$

**Step 5**: Compute net input to each output neuron $(Y_k, k = 1,...,m)$ sums its weighted input signals.

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \tag{3.22}$$

Compute the activation function of an output neuron

$$y_k = f(y\_in_k) \tag{3.23}$$

**Back propagation of error:**

**Step 6**: The actual output $y_k$ was compared with the target output $t_k$. The actual output is the result from feedforward calculations. The error term for standard BP is

$$e_k = \sum_{k=1}^{m} (t_k - y_k) \tag{3.24}$$

**Step 7**: To compute error signal for each output neuron $(Y_k, k = 1,...,m)$, multiply the derivative of its activation function.

$$\delta_k = (t_k - y_k) f'(y\_in_k) \tag{3.25}$$

Compute the weight and bias change (connection from output layer to hidden layer) by

$$\Delta w_{jk} = \alpha \delta_k z_j \tag{3.26}$$

$$\Delta w_{0k} = \alpha \delta_k \tag{3.27}$$

and send $\delta_k$ to hidden layer.

**Step 8**: Compute the error signal for each hidden unit $(Z_j, j = 1,..., j)$,

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \tag{3.28}$$

Multiplies by the derivative of its activation function

$$\delta_j = \delta\_in_j f'(z\_in_j) \tag{3.29}$$

Compute weight and bias change (connection from hidden layer to input layer)

$$\zeta v_{ij} \; X\sim_{j} x_i \; \Gamma\,\dagger\,\zeta v_{ij} \tag{3.30}$$

$$\zeta v_{0j} \; X\sim\cent_{j} \; \Gamma\,\dagger\,\zeta v_{0j} \tag{3.31}$$

**Step 9**: Adjust weights and biases by starting from the output layer backward to the hidden layer.

$$w_{0k} \; fnewAX \, w_{0k}\,(old)\,\Gamma\,\zeta w_{0k} \tag{3.32}$$

$$w_{jk} \; fnewAX \, w_{jk} \; foldA\Gamma\,\zeta w_{jk} \tag{3.33}$$

**Step 10**: Adjust weights and biases by starting from the hidden layer backward to the input layer.

$$v_{0j} \; fnewAX \, v_{0j} \; foldA\Gamma\,\zeta v_{0j} \tag{3.34}$$

$$v_{ij} \; fnewAX \, v_{ij} \; foldA\Gamma\,\zeta v_{ij} \tag{3.35}$$

**Step 11**: Test stopping condition

An epoch denotes one cycle for the entire set of learning input patterns. In other words, weight updates were computed after each learning pattern is presented.

**Recognition**

In this phase the backpropagation used the weight information to generalize the speech data (learning data and testing data). The results can be categorized into two possibilities, either a recognized word or unrecognized word. . The input pattern is recognized if the output neuron produces value that greater than 0.5 threshold. In other words, if the value of the output neuron is greater than 0.5 then it will consider as 1 and if it is less then 0.5, consider as 0, which means the input pattern is unrecognized. The threshold value of 0.5 is sufficient to MLP capable for recognized any patterns (Fausett, 1994). The process for the recognition procedure is described below.

**Step 0**: Restore weights that are obtained from learning process.

**Step 1**: For each input patterns, do steps 2.

**Step 2**: Perform the Feedforward (refer to previous standard learning algorithm for standard BP)

**Adaptive Sigmoid Function**

In conventional NN, the output computed by the neuron during learning and generalization is performed by fixed activation function. In this study, a new approach to generalize the speech data in recognition phase is introduced. In this sense, the fixed activation function was used only in learning process while in the recognition phase, the adaptive sigmoid function was employed.

The fixed sigmoid function is defined as: -

$$f(net)\ \mathrm{X}\frac{1}{1\Gamma\exp(\mathrm{Z}net)} \tag{3.36}$$

The Adaptive Sigmoid Function is defined as: -

$$f(net)\ \mathrm{X}\frac{1}{1\Gamma\exp(\mathrm{Z}\ net)} \tag{3.37}$$

where is a slope parameter of the sigmoid function in the region.

Throughout this study, the steepness of the slope of sigmoid function was adjusted in the generalization process to get the best fitting properties. Therefore, the highest recognition accuracy will be produced. **Figure 9** shows the sigmoid activation function with different slope. This slope is control by parameter.

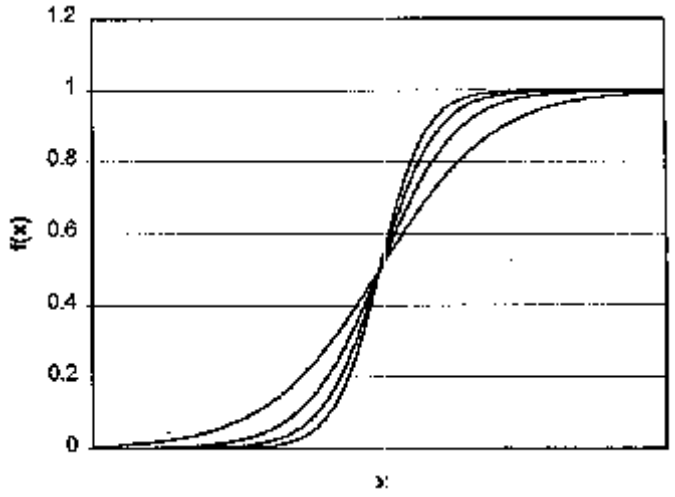


**Figure 9**: Activation Function with Different Slope

The higher the is, the steeper will be the slope**.** In this study the was tested between 1.0 to 2.0. Other than this value made the recognition rate is low. From the experiment, the value 2.0 contributes to the highest recognition for most of the experiments. The algorithm to perform the adaptive sigmoid function is given below.

**Step 0**: Restore weights that are obtained from learning process.

**Step 1**: Define the value for slope of parameter of sigmoid function (refer **Equation 3.37**)

**Step 2**: For each input patterns, do step 3.

**Step 3**: Perform the feed forward (refer to previous standard learning algorithm for standard BP).

**Step 4:** Select the highest recognition accuracy (according to the best value of the slope of parameter of sigmoid function).

**Step 5**: Repeat the **step 1** for other value of parameter of sigmoid function.

## 4.0 CONCLUSION

This paper presents the framework of malay isolated digit speech recognition system. The detailed description of data sets and the flow of framework system has also been described. In preprocessing phase the endpoint detection, time axis normalization, feature extraction and normalization stage was discussed in detailed. Future work will concentrate on experiments.

## REFERENCES

Bauer, N., Pathirana, P., & Hodgson, P. 2006. Robust Optical Flow with Combined Lucas-Kanade/Horn-Schunck and Automatic Neighborhood Selection. *Interternational Conference on Information and Automation, 2006. ICIA 2006.*

Djeraba, C., Lablack, A., & Benabbas, Y. 2010. Abnormal Event Detection. *In Multi-Modal User Interactions in Controlled Environments*, Springer US, 34, pp. 11-58.

Fleet, D. J., & Jepson, A. D. 1989. Computation of normal velocity from local phase information. *International Conference Computer Vision and Pattern Recognition, 1989. Proceedings CVPR '89., IEEE Computer Society*

Horn, B., & Schunck, B. (1981). Determining Optical Flow. ARTIFICAL INTELLIGENCE, 17, pp.185-203.

Jinnian, G., Xinyu, W., Zhi, Z., Shiqi, Y., Yangsheng, X., & Jianwei, Z. 2009. An intelligent surveillance system based on RANSAC algorithm. *Int. Conf. on Mechatronics and Automation, 2009. ICMA 2009.*

Lee, D., Papageorgiou, A., & Wasilkowski, G. W. 1988. Computational Aspects Of Determining Optical Flow. 2nd *International Conference on Computer Vision.*

Lee Yee, S., Mokri, S. S., Hussain, A., Ibrahim, N., & Mustafa, M. M. 2009. Motion detection using Lucas Kanade algorithm and application enhancement. *International Conference on the Electrical Engineering and Informatics, 2009. ICEEI '09.*

Lu, X., & Manduchi, R. 2011. Fast image motion segmentation for surveillance applications. Image and Vision Computing, 29(2-3), pp.104-116.

Lucas, B. D., & Kanade, T. 1981. An Iterative Image Registration Technique with an Application to Stereo Vision. *Paper presented at the IJCAI81.*

Meng, L., Chengdong, W., & Yunzhou, Z. 2008. Multi-resolution optical flow tracking algorithm based on multi-scale Harris corner points feature. *Paper presented at the Control and Decision Conference, 2008. CCDC 2008. Chinese.*

Shobhit, S., Fran, ois, B., mond, Monnique, T., & Ruihua, M. 2008. Crowd Behavior Recognition for Video Surveillance. *Paper presented at the Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems.*

Tian, C., Xinyu, W., Jinnian, G., Shiqi, Y., & Yangsheng, X. 2009. Abnormal crowd motion analysis. *International Conference on Robotics and Biomimetics (ROBIO), 2009 IEEE.*

Weiming, H., Tieniu, T., Liang, W., & Maybank, S. 2004. A survey on visual surveillance of object motion and behaviors. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 34(3), pp. 334-352.