

Cohesive Modeling, Analysis and Simulation for Spoken Urdu Language Numbers with Fourier Descriptors and Neural Networks

Engr. S K Hasnain, *Member IET, Assistant Professor,*
Pakistan Navy Engineering College, Karachi
National University of Engineering & Technology, Pakistan
hasnain@pnec.edu.pk

Dr. Azam Beg, *Member IEEE, Assistant Professor*
College of Information Technology,
UAE University Al-Ain, United Arab Emirate
abeg@uaeu.ac.ae

ABSTRACT

This unified research describes spoken Urdu numbers investigative analysis from 'siffar' (zero) to 'nau' (nine) for making a concrete foundation in recognition of Urdu language. Sound samples from multiple speakers were utilized to extract different features using Fourier descriptor and Neural networks. Initial processing of data, i.e. normalizing and time-slicing was done using a combination of Simulink in MATLAB. Afterwards, the MATLAB tool box commands were used for calculation of Fourier descriptions and correlations. The correlation allowed comparison of the same words spoken by the same and different speakers.

The analysis presented in this paper laid a foundation step in exploring Urdu language in developing an Urdu speech recognition system. In this paper the speech recognition feed-forward neural network models in Matlab were developed. The models and algorithm exhibited high training and testing accuracies. Our major goal work involves in the future use of TI TMS320C6000 DSK series or linear predictive coding. Such a system can be potentially utilized in implementation of a voice-driven help setup in different systems. Such as multi media, voice controlled tele-customer services.

Keywords: Spoken number, Fourier, Correlation, Feature extraction, Feed-forward neural networks, Learning rate

1. INTRODUCTION

Automatic speech recognition has been an active research topic for more than four decades. With the advent of digital computing and signal processing, the problem of speech recognition was clearly posed and thoroughly studied. These developments were complemented with an increased awareness of the advantages of conversational systems. The range of the possible applications is wide and includes: voice-controlled appliances, fully featured speech-to-text software, automation of operator-assisted services, and voice recognition aids for the handicapped [1]. The speech recognition problem has sometimes been treated as a speech-to-text conversion problem.

Many researchers have worked in this regard. Some commercial software is also available in the market for speech recognition, but mainly in English and other European languages. While the humans are speaking, the formats vary depending on the position of jaws, tongue and other parts of the vocal tract. Two related key factors are: bandwidth of each format, and format membership in a known bandwidth. The vowel for all human beings tends to be similar [2].

Sounds are mainly categorized into these groups: voiced sounds (e.g., vowels and nasals), unvoiced sounds (e.g., fricatives), and stop-consonants (e.g., plosives). The speech starts in lungs but is actually formed when the air passes through larynx and vocal tracts. Depending on the status of vocal fold in larynx the sound can be grouped into: voiced sound that is

time-periodic in nature and harmonic in frequency; the unvoiced is more noise-like. Speech modeling can be divided into two types of coding: *waveform* and *source*. In the beginning, the researchers tried to mimic the sounds as it is, meaning, waveform coding. This method tries to retain the original waveform using quantization and redundancy. An alternative approach makes use of breaking the sound up into individual components which are later modeled, separately. This method of utilizing parameters is referred to as source coding. Different characteristics of speech can be used to identify the spoken words, the gender of the speaker, and/or the identity of the speaker. Two important features of speech are *pitch* and *formant frequencies*:

(a) Pitch is a significant distinguishing factor among male and female speakers. The frequency of vibration of vocal folds determines the pitch, for example, 300 times per second oscillation of the folds results in 300 Hz pitch. Harmonics (integer multiples of fundamental frequency) are also created while the air passes through the folds. The age also affects the pitch. Just before puberty, the pitch is around 250 Hz. For adult males, the average pitch is 60 to 120 Hz, and for females, it is 120 to 200 Hz.

(b) The vocal tract, consisting of oral cavity, nasal cavity, velum, epiglottis, and tongue, modulates the pitch harmonics created by the pitch generator. The modulations depend on the diameter and length of the cavities. These reverberations are called formant frequencies (or resonances). The harmonics closer to the formant frequencies get larger while others are attenuated. While the humans are speaking, the formants vary depending on the positions of the tongue, jaw, velum and other parts of the vocal tract. Two related key factors are: bandwidth of each formant, and formant's membership in a known bandwidth. The vowels for all human beings tend to be similar. Each vowel uttered by a person generates different formants. So we can say that the vocal tract is a variable filter, whose inputs are (1) the pitch, and (2) the pitch harmonics. The output of the filter is the gain or the attenuation of the harmonics falling in different formant frequencies. The filter is called *variable filter* model. The transfer function for the filter is determined by the formant frequencies [3].

Correlation exists between objects, phenomena, or signals and occurs in such a way that it cannot be by chance alone. Unconsciously, the correlation is used in everyday life. When one looks at a person, car or house, one's brain tries to match the incoming image with hundreds (or thousands) of images that are

already stored in memory [4]. We based our current work on the premise that same word spoken by different speakers is correlated in frequency domain.

In the speech recognition research literature, no work has been reported on Urdu speech processing. So we consider our work to be the first such attempt in this direction. The analysis has been limited to number recognition. The process involves extraction of some distinct characteristics of individual words by utilizing discrete (Fourier) transforms and their correlations. The system is speaker-independent and is moderately tolerant to background noise.

2. REVIEW OF DISCRETE TRANSFORMATION & ITS MATLAB IMPLEMENTATION

Discrete Fourier transform (DFT) is itself a sequence rather than a function of continuous variable and it corresponds to equally spaced frequency samples of discrete time Fourier transform of a signal. Fourier series representation of the periodic sequence corresponds to discrete Fourier transform of finite length sequence. So we can say that DFT is used for transforming discrete time sequence $x(n)$ of finite length into discrete frequency sequence $X[k]$ of finite length. This means that by using DFT, the discrete time sequence $x(n)$ is transformed into corresponding discrete frequency sequence $X[k]$ [4]-[5].

DFT is a function of complex frequency. Usually the data sequence being transformed is real. A waveform is sampled at regular time intervals T to produce the sample sequence of N sample values, where n is the sample number from $n=0$ to $n=N-1$.

$$\{x(nT)\} = x(0), x(T), \dots, x[(N-1)T]$$

The data values $x(nT)$ will be real only when representing the values of a time series such as a voltage waveform. The DFT of $x(nT)$ is then defined as the sequence $\{X[k\omega]\} = X(0), X(\omega), \dots, X[(N-1)\omega]$ in the frequency domain, where ω is the first harmonic frequency given by $\omega = 2\pi / NT$.

Thus $X[k\omega]$ has real and imaginary components in general, so that for the k th harmonic

$$X(k) = R(k) + jI(k) \\ |X(k)| = [R^2(k) + I^2(k)]^{1/2} \quad (2.1)$$

and $X[k]$ has the associated phase angle

$$\varphi(k) = \tan^{-1} [I(k) / R(k)] \quad (2.2)$$

where $X[k]$ is understood to represent $X[k]$. These equations are therefore analogous to those for the Fourier transform. Note that N real data values (in time domain) transform to N complex DFT values (in frequency domain). The DFT values, $X[k]$, are given by:

$$F_D [x(nT)] = \sum_{n=0}^{N-1} x(nT) e^{-jk\omega nT}, \quad k=0,1,\dots,N-1 \quad (2.3)$$

where $\omega=2\pi/NT$ and F_D denotes the DFT.

$$X[k] = \sum_{n=0}^{N-1} x(nT) e^{-jk2\pi n/N} \quad (2.4)$$

The Fast Fourier transform (FFT) eliminates most of the repeated complex products in DFT. In C version of signal processing algorithm, there are several different routines for real and complex versions of the DFT and FFT. When these routines are coded into the MATLAB language, they are very slow compared with the MATLAB *fft* routine, which are coded much more efficiently. Furthermore, the MATLAB routines are flexible and may be used to transform real or complex vector of arbitrary length. They meet the requirements of nearly all signal processing applications; consequently, in this paper, the *fft* routines are preferred over all discrete transform operations. MATLAB's *fft* routine produces a one-dimensional DFT using the FFT algorithm; that is when $[X_K]$ is a real sequence, *fft* produces the complex DFT sequence $[X_m]$. In MATLAB, the length N of the vector $[X_K]$ may not be given. Thus both of the following are legal expressions:

$$\begin{aligned} X &= \text{fft}(x) \\ X &= \text{fft}(x, N) \end{aligned} \quad (2.5)$$

The first expression in (2.5) produces a DFT with same number of elements as in $[X_K]$, regardless of whether $[X_K]$ is real or complex. In the usual case where $[X_K]$ is real and length N , the last $N/2$ complex elements of the DFT are conjugates of the first $N/2$ elements in the reverse order, in accordance with (2.4). In the unusual case where $[X_K]$ is complex, the DFT consists of N independent complex elements. For example, the results of the following commands

with $N=4$ can be easily verified using definition in (2.6).

$$\frac{\text{FFT computing time}}{\text{DFT computing time}} = \frac{1}{2N} \log_2 N \quad (2.6)$$

The results of the following commands with $N=4$ can be for example easily verified with [5]:

```
x=[1 0 0 1];
X=fft(x)
```

In this example, the DFT components $[X_m] = [2, 1-j, 0, 1+j]$ are found from (2.4). The second expression in (2.6) specifies the value of N in (2.4), which effectively overrides any previous specification of the length of the vector x , thus the following commands produce the same result:

```
x=[1 0 0 1 3];
X=fft(x,4)
```

The DFT, $x = [X_m]$ has length = 4 is the same as in previous example.

```
x=[1 1];
X=fft(x,4)
```

$$[X_m] = [2, 1-j, 0, 1+j]$$

The result here is same because, when N is greater than the length of x ; X is the DFT of a vector consisting of x extended with zeros on the right, from the length of x to N . (The length of vector x itself is not increased in this process). The MATLAB library also includes a two dimensional *fft* routine called *fft2*. The routine computes two-dimensional FFT of any matrix, whose element may be, for example, samples (pixel values) of a two dimensional image.

Usually, some *recognition* occurs when the incoming images bear a strong correlation with an image in memory that "best" corresponds to fit or is most similar to it. A similar approach is used in this investigation, to measure the similarity between two signals. This process is known as *autocorrelation* if the two signals are exactly same and as *cross-correlation* if the two signals are different. Since correlation measures the similarity between two signals, it is quite useful in identifying a signal by comparing it with a set of known reference signals. The reference signal that results in the lowest value of the correlation with the unknown signals is most likely the identity of the unknown object [9].

Correlation involves shifting, multiplication and addition (accumulation). The *cross-correlation function* (CCF) is a measure of the similarities or shared properties between two signals. Application of CCF includes cross spectral density, detection and recovery of signals buried in noise, for example the detection return signals, pattern, and delay measurement. The general formula for cross-correlation $r_{xy}(n)$ between two data sequences $x(n)$ and $y(n)$ each containing N data might therefore be written as:

$$r_{xy} = \sum_{n=0}^{N-1} x(n) y(n) \quad (2.7)$$

The *autocorrelation function* (ACF) involves only one signal and provides information about the structure of the signal or its behaviour in the time domain. It is special form of CCF and is used in similar applications. It is particularly useful in identifying hidden properties.

$$r_{xx} = \sum_{n=0}^{N-1} x(n) x(n) \quad (2.8)$$

3. FEEDFORWARD VS RECURRENT NETWORKS

Neural networks have proven to be a power tool for solving problem of prediction, classification and pattern recognition. [12]-[17].

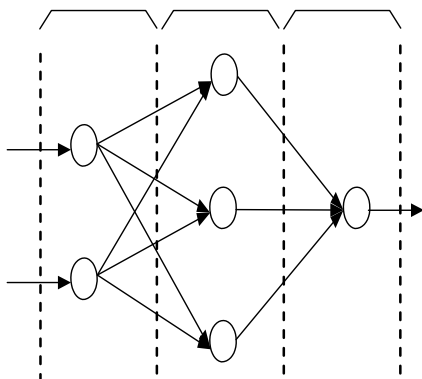


Fig.1 Neural network model for feedforward multilayer Perceptron for analysis

Neural network architecture can be divided into two principal types: recurrent and non-recurrent networks. An important sub-class of non-recurrent NN consists of architectures in which cells are

organized into layers, and only unidirectional connections are permitted between adjacent layers. This is known as a feed-forward multi layer Perceptron (MLP) architecture. This architecture is shown in Fig 1.

4. DATA ACQUISITION AND PROCESSING

The system presented in this model was limited to processing of individual Urdu numerals (0 to 9). The data was acquired by speaking the numerals into a microphone connected to MS-Windows-XP based PC. We recorded the data for fifteen speakers who uttered the same number set (0 to 9), specifically, *siffar, aik, do, teen, chaar, paanch, chay, saat, aath, and nau*. Each sound sample was curtailed to 0.9 minute.

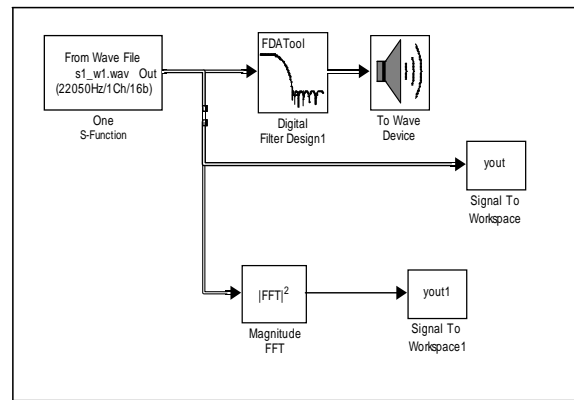


Fig.2 Simulink model for data of numbers extraction for analysis

One of the obvious methods of speech data acquisition is to have a person speak into an audio device such as microphone or telephone. This act of speaking produces a sound pressure wave that forms an acoustic signal. The microphone or telephone receives the acoustic signal and converts it into an analog signal that can be understood by an electronic system. Finally, in order to store the analog signal on a computer, it must be converted to a digital signal [6],[8].

The data in this paper is acquired by speaking Urdu numbers into a microphone connected to MS-Windows-XP based PC. The data is saved into '.wav' format files. The sound files are processed after passing through a (Simulink) filter, and are saved for further analysis. We recorded the data for fifteen speakers who spoke the same number set, i.e. zero to nine.

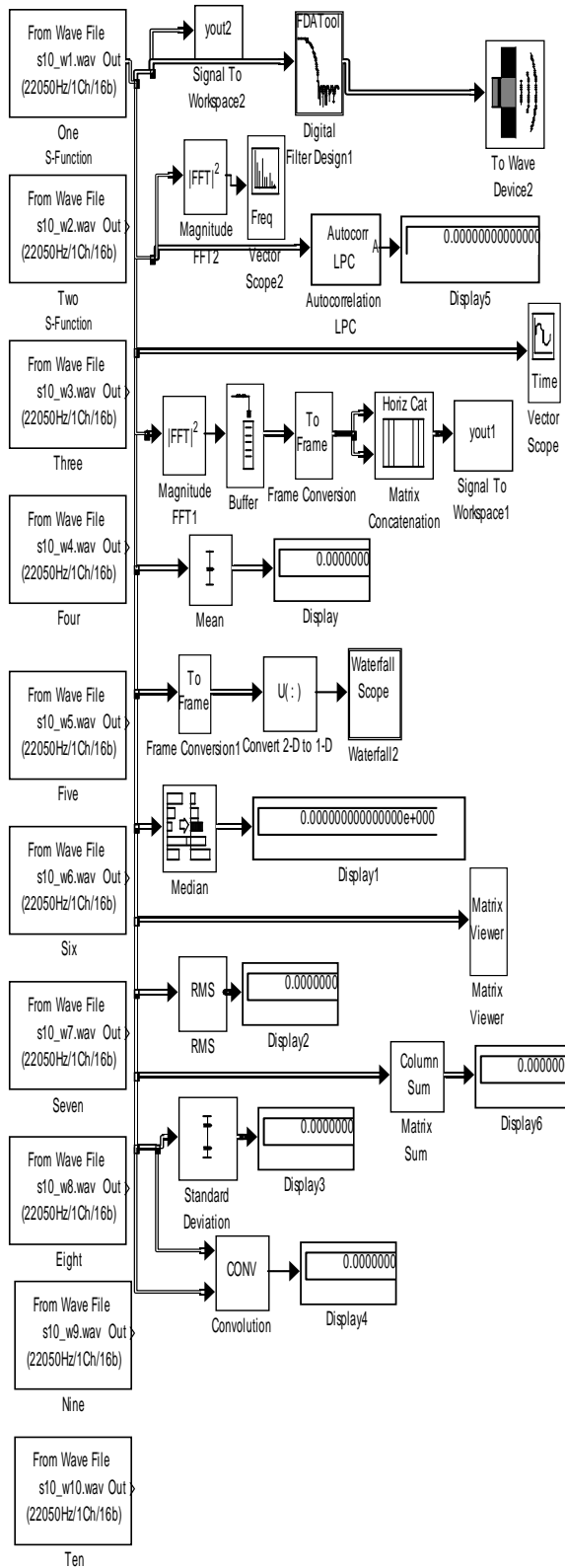


Fig.3 Extended Simulink model for analysis of Urdu spoken numbers

In general, the digitized speech waveform has a high dynamic range, and can suffer from additive noise. So first, a Simulink model was used to extract and analyze the acquired data as shown in Fig.3. Another Simulink model was developed for performing analysis such as standard deviation, mean, median, autocorrelation, magnitude of FFT and data matrix correlation. We also tried a few other statistical techniques, however, most of them failed to provide us any useful insight into the data characteristics. We would also like to mention that we had started our experiments by using Simulink, but found this GUI-based tool to be somewhat limited because we did not find it easy to create multiple models containing variations among them. This iterative and variable-nature of models eventually led us to MATLAB's (text-based) .m files. We created these files semi-automatically by using a PERL-language script; the script was developed specifically for this purpose. Three main data pre-processing steps were required before the data could be used for analysis:

4.1 Pre-Emphasis

By pre-emphasis [7], we imply the application of a *normalization* technique, which is performed by dividing the speech data vector by its highest magnitude.

4.2 Data Length Adjustment

FFT execution time depends on exact number of the samples (N) in the data sequence $[X_K]$, and the execution time is minimal and proportional to $N \cdot \log_2(N)$, where N is a power of two. Therefore, it is often useful to choose the data length equal to a power of two [10].

4.3 Endpoint Detection

The goal of endpoint detection is to isolate the word to be detected from the background noise. It is necessary to trim the word utterance to its tightest limits, in order to avoid errors in the modeling of subsequent utterances of the same word. As we can see from the upper part of Fig. 4, a threshold has been applied at both ends of the waveform. The front threshold is normalized to a value that all the spoken numbers trim to a maximum value. These values were obtained after observing the behavior of the waveform and noise in a particular environment. We can see the difference in frequency characteristics of the words *aik* (one) to *paanch*

(five) in Fig 4–8 respectively.

4.4 Windowing

Speech signal analysis also involves application of a window with a time less than the complete signal. The window first starts with beginning of the signal and then shifted until it reaches the end. Each application of the window to the part of the speech signal results in a spectral vector.

4.5 Frame Blocking

Since the vocal tract moves mechanically slow, speech can be assumed to be a random process with slowly varying properties. Hence the speech is divided into overlapping frames of 100 ms. The speech signal is assumed to be stationary over each frame and this property will prove useful in further operations [18].

4.6 Fourier Transform

The MATLAB algorithm for the two dimensional FFT routine is as follows [9]:

```
fft2(x) =fft(fft(x), `);
```

Thus the two dimensional FFT is computed by first computing the FFT of x, that is, the FFT of each column of x, and then computing the FFT of each row of the result. Note that as the application of *fft2* command produced even symmetric data, we only show lower half of the frequency spectrum in our graphs.

4.7 Correlation

Calculations for correlation coefficients of different speakers were performed [2]-[3]. As expected, the cross-correlation of the same speaker for the same word did come out to be 1. The correlation matrix of a spoken number was generated in a three-dimensional form for generating different simulations and graphs.

Figures 4 - 8 show the combined correlation extracted for fifteen different speakers for the Urdu number siffar (zero) to paanch (five), that differentiate the number spoken by different speaker for detailed analysis.

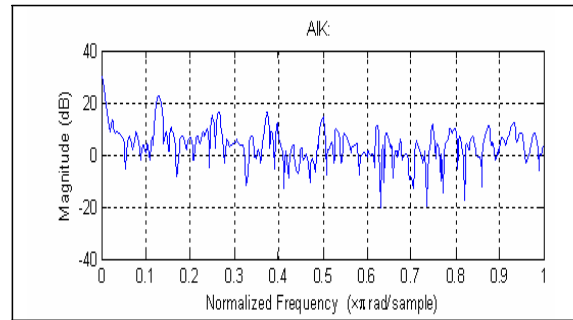


Fig.4 Correlation of the spoken Urdu number aik (one)

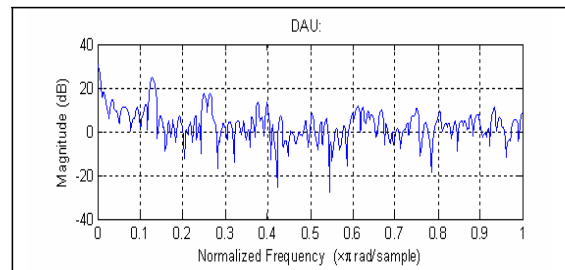


Fig.5 Correlation of the spoken Urdu number dau (two)

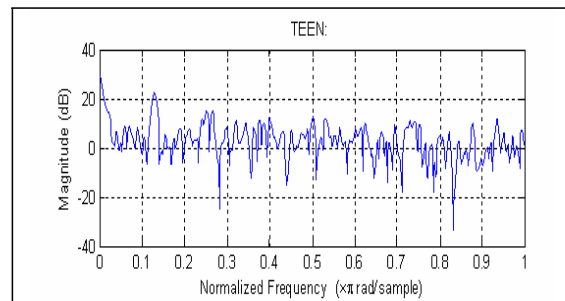


Fig.6 Correlation of the spoken Urdu number teen (three)

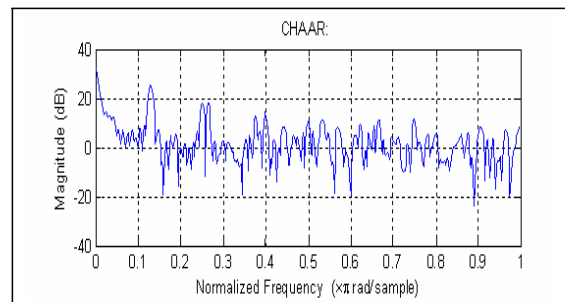


Fig.7 Correlation of the spoken Urdu number char (four)

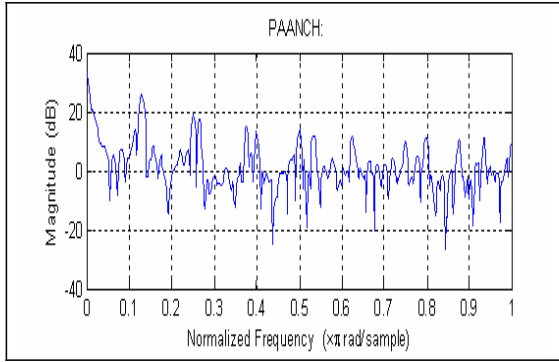


Fig.8 Correlation of the spoken Urdu number paanch (five)

Figures 9 - 14 show the FFT magnitude spectrum extracted for fifteen different speakers for the Urdu number siffar (zero) to paanch (five), that again differentiate the number spoken by different speaker for detailed analysis.

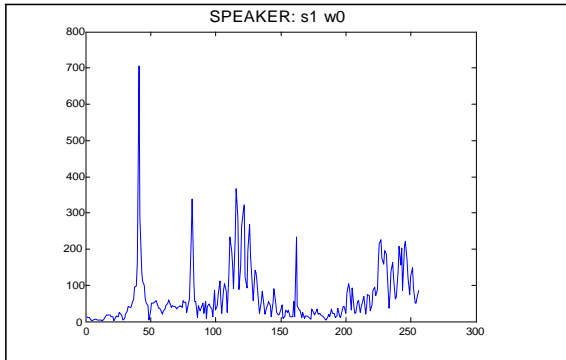


Fig.9 FFT magnitude spectrum for the spoken Urdu number siffar (zero)

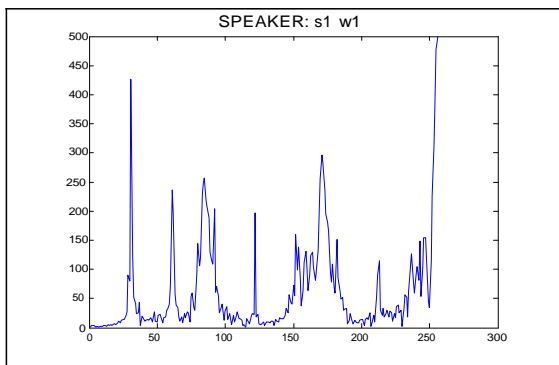


Fig.10 FFT magnitude spectrum for spoken Urdu number aik (one)

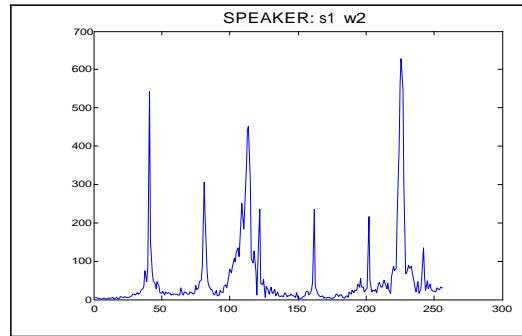


Fig.11. FFT magnitude spectrum for spoken Urdu number dau (two)

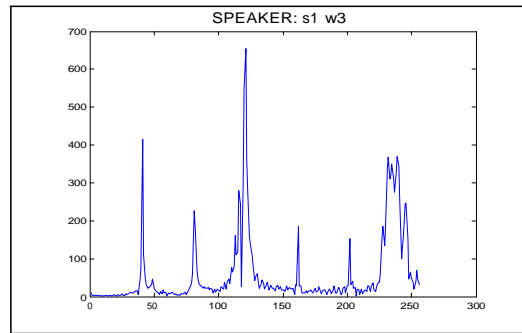


Fig.12 FFT magnitude spectrum for spoken Urdu number teen (three)

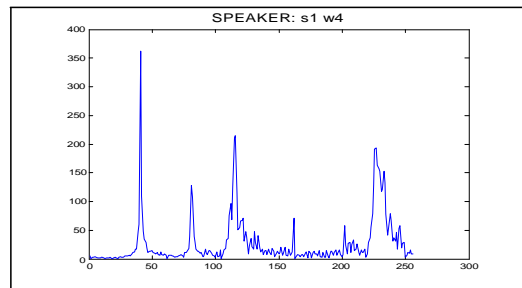


Fig.13 FFT magnitude spectrum for spoken Urdu number char (four)

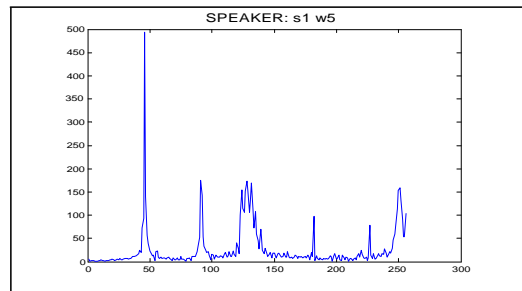


Fig.14 FFT magnitude spectrum for spoken Urdu number paanch (five)

Figures 15-19 show the combined magnitude spectrum of correlation extracted for fifteen different speakers for the Urdu number siffar (zero) to paanch (five), shown here for brevity which differentiate the number spoken by different speaker for detailed analysis, while Fig 20 shows surface graph of the number aik (one).

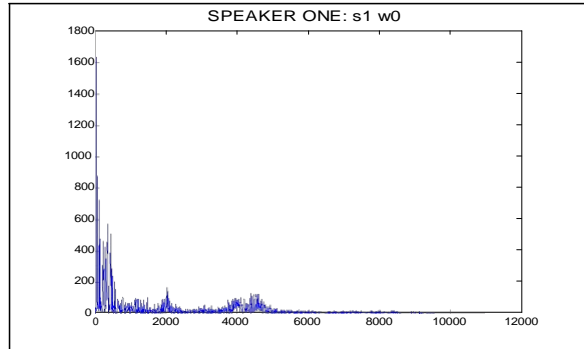


Fig.15 The waveform of the correlation of the spoken Urdu number *siffar* (zero) by speaker-1

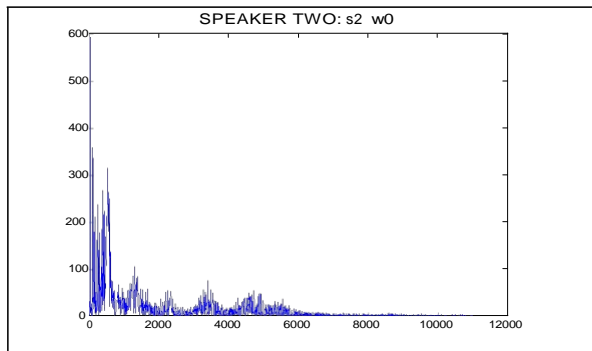


Fig.16 The waveform of the correlation of the spoken Urdu number *siffar* (zero) by speaker-2

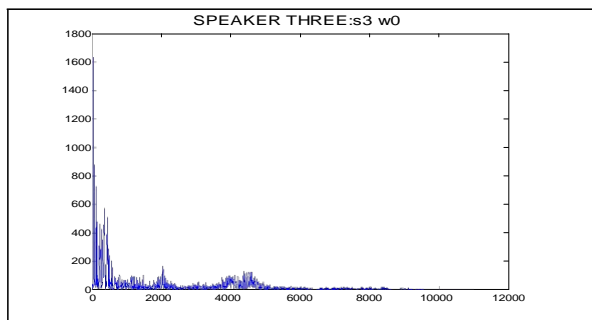


Fig.17 Magnitude spectrum of the spoken Urdu number *siffar* (zero) by speaker-3

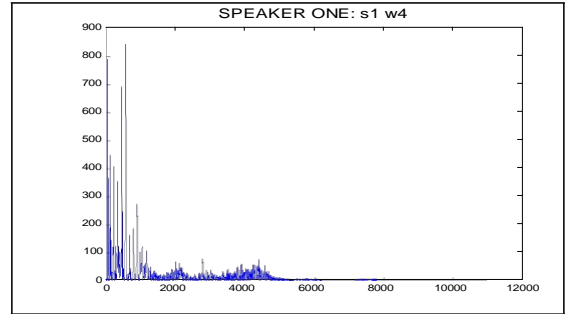


Fig.18 Magnitude spectrum of the correlation of the spoken Urdu number spoken *chaar*(four) by speaker-1

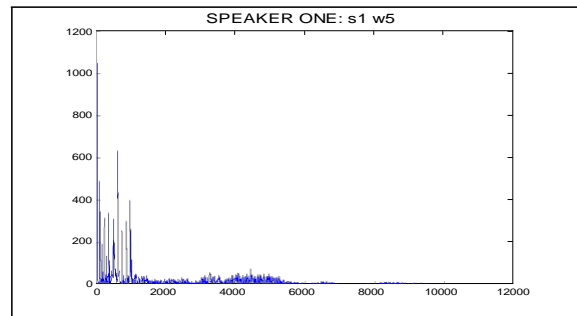


Fig.19 Magnitude spectrum of the correlation of the spoken Urdu number spoken *paanch* (five) by speaker-1

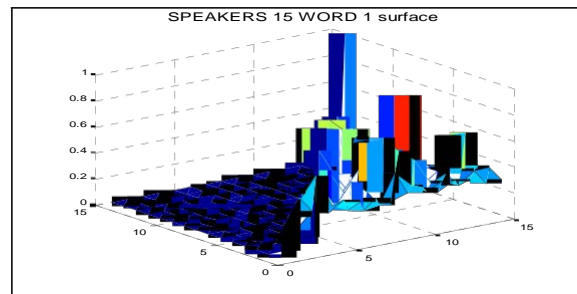


Fig.20 The surface plot of the correlation of the spoken Urdu numbers spoken *aik* (one) by speaker-15

4.8 Creating a Network

To create a feedforward neural network (MLP) object in an interactive way to use the command `ntool` to open the Neural Network Manager. We can then import or define data and train our network in the GUI or export our network to the command line for training [5].

```
net = newff(MxMx, [S1 S2 ... SK], {TF1 TF2 ... TFK}, BTF, BLF, PF)
```

%where MnMx ($p - 1 \times 2$) matrix of min and max values for x input vectors.
 %Si Size of ith layer, for K layers.
 %TFi Transfer function of ith layer, default = tansig.
 %BTF Backpropagation network training function, default = traingdx.
 %BLF Backpropagation weight/bias learning function, default = learngdm.
 %PF Performance function, default = mse.

4.9 Network Structure and Parameters

```
%Neural Network object:
...
%subobject structures:
inputs: {1x1 cell} of inputs
layers: {2x1 cell} of layers
outputs: {1x2 cell} containing 1 output
targets: {1x2 cell} containing 1 target
%biases: {2x1 cell} containing 2 biases
inputWeights: {2x1 cell} containing 1 input weight
layerWeights: {2x2 cell} containing 1 layer weight
...
%weight and bias values:
IW: {2x1 cell} containing 1 input weight matrix
LW: {2x2 cell} containing 1 layer weight matrix
b: {2x1 cell} containing 2 bias vectors
```

4.10 Network Training / Testing

In incremental training the weights and biases of the network are updated each time an input is presented to the network. The cell array format is most convenient to be used for networks with multiple inputs and outputs, and allows sequences of inputs to be presented. The matrix format can be used if only one time step is to be simulated, so that the sequences of inputs cannot be described. There are two basic functions available for the incremental training: learngd and learngdm. Refer to the manual for details.

```
% learning rate
net.trainParam.lr = 0.01;
net.trainParam.epochs = 6000;
% tolerance
net.trainParam.goal = 0.01;
net=train(net,p,t);
%test the network with
y_start = sim(net,p);
figure(200),plot(t,'g');
```

4.11 System Workflow

This section describes the system workflow for creating, initializing, training and running the network on test data. The following sub sections demonstrate the Word Network to demonstrate the system workflow of the demo program.

The first step is to create and initialize the Neural Network and set up the network configuration.

```
net=newff(minmax_of_p,[64, 32, 32, 1],
{'tansig','tansig','tansig','purelin'},'traingd');
// this is input layer.
Layer = 64
// this is hidden layer no 1.
Layer = 35
// this is hidden layer no. 2
Layer = 35
// this is output layer.
Layer = 1
```

4.12 Data Structure Training

Data training is an array list of a structure which is presented. Each of the initial four string values are part of the input vector provided to the network. The last string variable represents the target output for the network in the code fragmented below:

```
net.trainParam.show = 50;
% x-axis of graph is 50 values apart
net.trainParam.lr = 0.01;
% learning rate
net.trainParam.epochs = 6000;
net.trainParam.goal = 0.01;
% tolerance
net=train(net,p,t);
```

4.13 Setting and Loading Training Data

Following code loads up the default training data into the training data list. The default training data includes hard-coded input and output patterns for a number of different homonyms word scenarios along with their corresponding suggested output.

```
% Minimum array size has been kept 1e10
min_arr_size = 1e10;

% Read data from .wav files and calculate
magnitudes from FFTs
% 15 speakers have spoken the same word
pronounced as "siffar"
s1_w0 = wavread('s1_w0.wav');
```

```

s1_w0_norm = s1_w0/max(s1_w0);
s1_w0_fft0 = abs(fft2(s1_w0_norm));
min_arr_size = min(min_arr_size,
size(s1_w0_norm));
% Minimum array size has been kept 1e10

min_arr_size = 1e10;

% Read data from .wav files and calculate
magnitudes from FFTs
% 15 speakers have spoken the same word
pronounced as "siffar"
s1_w0 = wavread('s1_w0.wav');
s1_w0_norm = s1_w0/max(s1_w0);
s1_w0_fft0 = abs(fft2(s1_w0_norm));
min_arr_size= min(min_arr_size,
size(s1_w0_norm));
.....

s15_w0 = wavread('s15_w9.wav');
s15_w0_norm = s15_w0/max(s15_w9);
s15_w0_fft0 = abs(fft2(s15_w9_norm));
min_arr_size=min(min_arr_size,
size(s15_w9_norm))

```

4.14 Network Training

Before using the Neural Networks in the demo program to detect and classify correct word structure as well as able to find out the alternate word for the incorrectly recognized word (by the Speech Recognition engine), we need to train the Word Network with the training data. The train network uses the object to train the word frequency and network 6000 and 1500 times respectively. The following code snippet shows how the word network gets trained.

```

cutoff = 64;
for i=1:150;
    for j=3:cutoff+2;
        ptrans(i,j-2) = log(alldat(i,j));
    end
end

p = transpose(ptrans); % inputs
t = transpose(alldat(:,2)); % target array

for i = 1:cutoff
    minmax_of_p(i,1) = min(p(i,:));
end

for i = 1:cutoff
    minmax_of_p(i,2) = max(p(i,:));
end

```

In the above code, notice that the first line in the function is the training of the word structure network, which the Word Network uses to first detect the problem in the network and then try to identify the alternate word for the incorrect word in word engine.

4.15 Pattern Detecting

Once the network is trained, it can be used to detect the correct output for the given input vector. The following functions take the input pattern as an array and use the function to retrieve the output from the network.

% Creating new array 'alldat (:,:)' for NN use ...

```

alldat(1,1) = 1;
alldat(1,2) = 1;
alldat(1,3:258) = s1_w1_fft0(1:256);

alldat(2,1) = 1;
alldat(2,2) = 2;
alldat(2,3:258) = s1_w2_fft0(1:256);

alldat(3,1) = 1;
alldat(3,2) = 3;
alldat(3,3:258) = s1_w3_fft0(1:256);

```

4.16 Setting and Loading Test Data

The demo program is capable of loading the required input for the neural network from the wave file. The wave file that contains the input sentences to be detected by the neural network, however should be in a specific format so that the program can correctly load and feed the input data to the neural network and generate the correct output.

load 'neural_network.mat'

**% The data of 15 speakers spoken in Urdu from siffar (zero) to nau (nine).
% Minimum array size has been kept 1e10**

```

min_arr_size = 1e10;

% Read data from .wav files and calculate
magnitudes from FFTs
% 15 speakers have spoken the same word
pronounced as "siffar"
s1_w0 = wavread('s1_w0.wav');
s1_w0_norm = s1_w0/max(s1_w0);
s1_w0_fft0 = abs(fft2(s1_w0_norm));
min_arr_size= min(min_arr_size,
size(s1_w0_norm));

```

`% Creating new array 'alldat(:,:)' for NN use ...`

```
alldat(1,1) = 1;
alldat(1,2) = 1;
alldat(1,3:258) = s1_w1_fft0(1:256);
alldat(2,1) = 1;
alldat(2,2) = 2;
alldat(2,3:258) = s1_w2_fft0(1:256);
```

```
alldat(3,1) = 1;
alldat(3,2) = 3;
alldat(3,3:258) = s1_w3_fft0(1:256);
```

`% min_arr_size = min_arr_size/2;`

```
min_arr_size = 11000;
```

`% 15 speakers have spoken the same word pronounced as "siffar".
% To find min the array length so all arrays are clipped to it.`

```
s1_w0_fft(1:min_arr_size,1)=s1_w0_fft0(1:min_arr_size, 1);
s2_w0_fft(1:min_arr_size,1)=s2_w0_fft0(1:min_arr_size, 1);
s3_w0_fft(1:min_arr_size,1)=s3_w0_fft0(1:min_arr_size, 1);
s4_w0_fft(1:min_arr_size,1)=s4_w0_fft0(1:min_arr_size, 1);
```

`% testing the network`

```
y_start = sim(net,p);
figure(200),plot(t,'g');
hold on;
plot(y_start,'b');
legend('actual','predicted');
hold off
```

For the above model we can examine structure of the created network `nnet1` and its all initial parameters. Typing in `nnet1` on the command line gives the structure of the top level neural network object. The extract of the structure is as follows:

5. ANALYSIS & RESULTS

When we compared the frequency content of the same word by different speakers, we found striking similarities among them. This helped us to get more confidence in our initial hypothesis that a single word uttered by a diverse set of speakers would exhibit similar characteristics. Additionally, Fig.23 shows surface graph for the correlation of frequency content among different speakers, for words *aik* (one).

The effect of learning rate (α) on training is shown in Fig.25-38. In our experiments, a larger value of α took lesser time to train, although convergence to the steady state value was noisy. It shows that there exists a trade off between learning rate (α), network parameter setting and the epoch for which the algorithm was used.

We observed that Fourier descriptor feature was independent of the spoken numbers, with the combination of Fourier transform and correlation technique commands used in MATLAB, a high accuracy recognition system can be realized. Recorded data was used in Simulink model for introductory analysis. The feedforward neural network was trained for different learning rates, goal and epochs. It was found that there is a trade off between learning rate, goal and epochs. The network was trained and tested as shown in Fig 21-30.

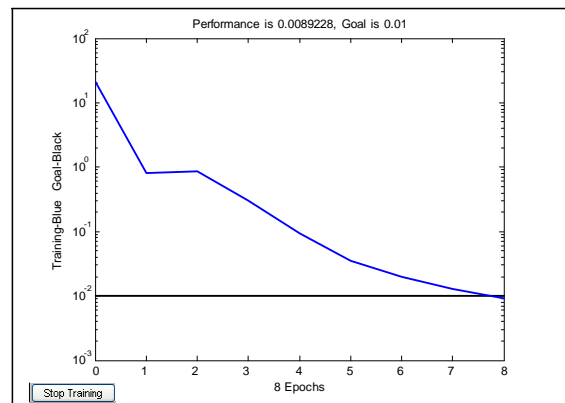


Fig.21 Training on Class 5 with 12 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

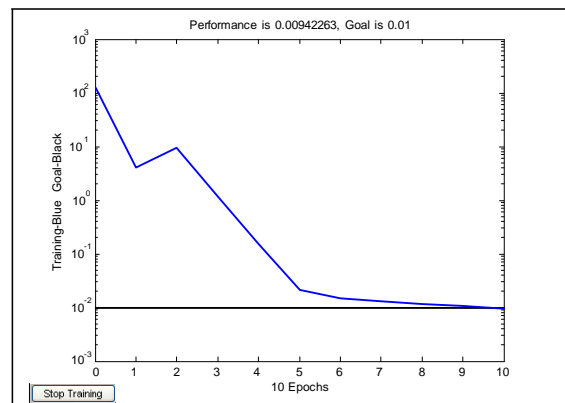


Fig.22 Training on number 6 with 12 speakers
Layers (64,35, 35,1)
Learning rate 0.01

Goal 0.01

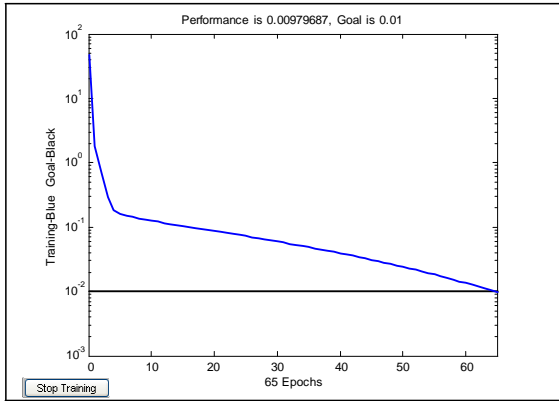


Fig.23 Training on number 7 with 12 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

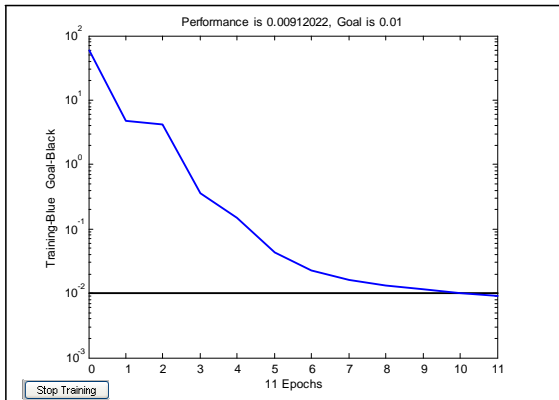


Fig.24 Training on number 8 with 12 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

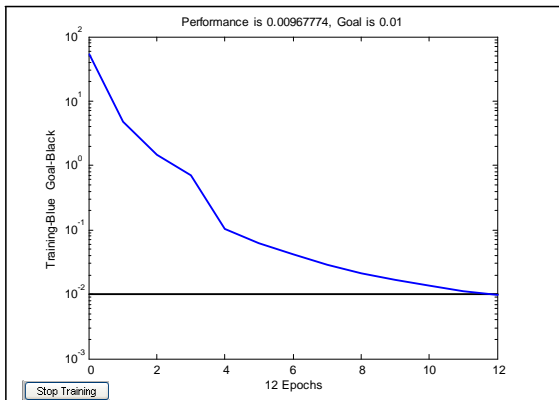


Fig.25 Training on number 9 with 12 speakers
Layers (64,35, 35,1)

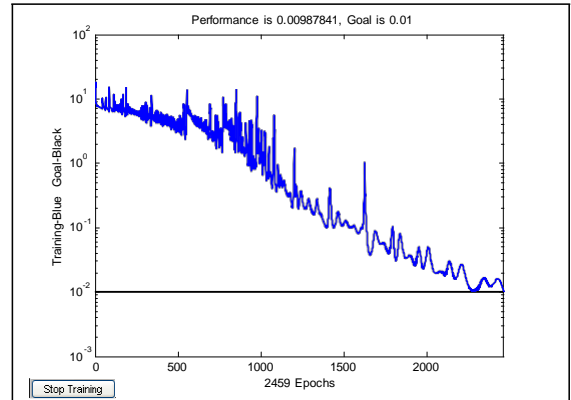


Fig.26 Training on 15 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

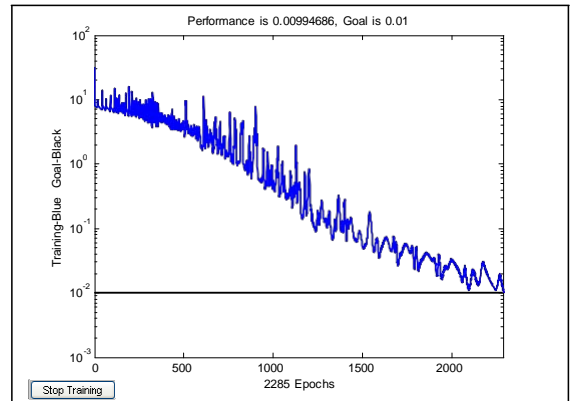


Fig.27 Training on 12 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

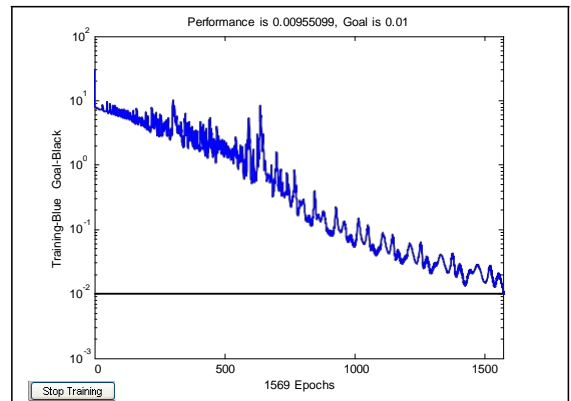


Fig.28 Training on 10 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

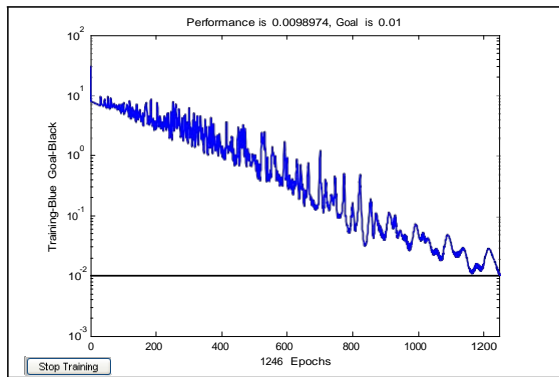


Fig.29 Training on 7 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

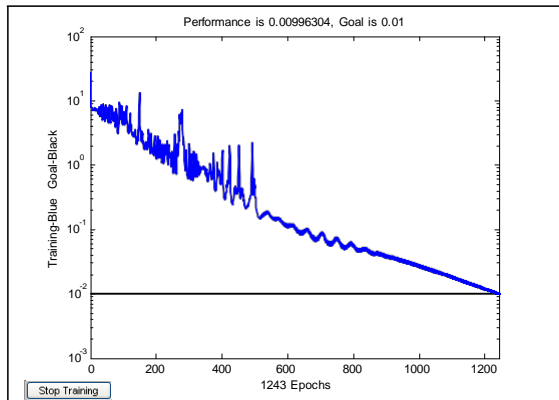


Fig.30 Training on 5 speakers
Layers (64,35, 35,1)
Learning rate 0.01
Goal 0.01

We created and tested the networks in different configurations, especially the hidden layer size. The following table shows the learning accuracy with some of the networks. A maximum accuracy of 94% was achieved with double hidden layers network (64, 35, 35 1). Therefore double-hidden layers should suffice for application as shown in table 1.

S No	Neuron count	Learning accuracy
1	64, 10, 10	81.48%
2	64, 20, 10	81.48%
3	64, 30, 10	72.59%
4	64, 40, 10	83.70%
5	64, 60, 10	62.96%
6	64, 80, 10	45.93%

7	64, 10, 10, 10	64.44%
8	64, 15, 15, 10	69.63%
9	64, 20, 20, 10	58.52%
10	64, 25, 20, 10	71.85%
11	64, 25, 25, 10	71.85%
12	64, 30, 30, 1	72.59%
13	64, 35, 35, 1	94%

Table 1. Training of Neural Network with different number of speakers

The feedforward neural network was trained for number of speakers with same layers, learning rate, goal and epochs. It was found that there is trade off between neural network, learning rate, goal and epochs as shown in table-2.

S No	No of Training speakers	Number of Testing Speaker	Accuracy Rate %
1	15	5	94
2	12	4	95
3	10	3	96.7
4	7	2	95
5	5	2	100

Table 2. Training of Neural Network with different number of speakers showing accuracy rate.

Relationship between network layers and training data can be formulated from the effect of learning rate (α) on training is shown in Fig 21-30. In experiments, a large value of α took lesser time to train, although convergence to the steady state value was noisy. It shows that there exists a trade off between layers of the network (network parameter setting), learning rate (α), and the epoch for which the algorithm was used.

6. CONCLUSION

In this paper, we presented recognition analysis of Urdu numbers siffar to nau (one to nine), which is totally an unique idea. The data was acquired in moderate noisy environment by word utterances of 15 different speakers. FFT algorithm was used in MATLAB to analyze the data. As expected, we found high correlation among frequency contents of the same word, when spoken by many different speakers.

We have investigated creation of neural network models for automatically recognizing individual Urdu numbers to be specific. The feedforward neural network was trained for different learning rates; combined and individually for different goals and

epochs. It was found that there is a trade off between learning rate, goal and epochs. It means all values to be adjusted at a level of learning rate, parameter goals and epochs. We are still in further development stage of the system using TI TMS 320C6713. This recognition system could be of many potential applications, for example, voice-driven menu selection in a telephone-based customer service in Urdu speaking countries such as Pakistan/India.

7. REFERENCES

- [1] S K Hasnain, Azam Beg and Samiullah Awan, "Frequency Analysis of Urdu Spoken Numbers Using MATLAB and Simulink" Journal of Science & Technology PAF KIET ISSN 1994-862x, Karachi, Dec. 2007.
- [2] D. O'Shaughnessy. *Speech Communication: Human and Machine*. Addison Wesley Publishing Co., 1987.
- [3] T. Parsons. *Voice and Speech Processing*. McGraw-Hill College Div., Inc, 1986.
- [4] S K Hasnain, Pervez Akhter, "Digital Signal Processing, Theory and Worked Examples," 2007.
- [5] MATLAB User's Guide. Mathworks Inc., 2006.
- [6] DSP Blockset (For use with Simulink) User's Guide Mathworks Inc., 2007.
- [7] Samuel D Stearns, Ruth A David, "Signal Processing Algorithms in MATLAB," Prentice Hall, 1996.
- [8] "TMS320C6713 DSK User's Guide," Texas Instruments Inc., 2005.
- [9] G. C. M. Fant. *Acoustic Theory of Speech Production*. Mouton, Gravenhage, 1960.
- [10] S K Hasnain, Aisha Tahir, "Digital Signal Processing Laboratory Workbook", 2006.
- [11] S. Varho. *New Linear Predictive Systems for Digital Speech Processing*. PhD dissertation, Helsinki University of Technology, Finland, 2001.
- [12] A. Beg, W. Ibrahim. An Online Tool for Teaching Design Trade-offs in Computer Architecture. *In Proc. International Conference on Engineering Education*, Coimbra, Portugal, September 3-7, 2007.
- [13] A. Beg. Predicting Processor Performance with a Machine Learnt Model. *IEEE International Midwest Symposium on Circuits and Systems, MWSCAS/NEWCAS 2007*, Montreal, Canada, August 5-8, 2007, pp. 1098-1101.
- [14] A. Beg, P.W.C. Prasad, S.M.N.A. Senanayake. Learning Monte Carlo Data for Circuit Path Length. *In Proc. International Conference on Computers, Communications & Control Technologies, CCCT 2007*, Orlando, Florida, July 12-14, 2007.
- [15] A. Beg, P.W.C. Prasad, M. Arshad, S K Hasnain. Using Recurrent Neural Networks for Circuit Complexity Modeling", *In Proc. IEEE INMIC Conference*, Islamabad, Pakistan, December 23-24, 2006, pp. 194-197.
- [16] S K Hasnain, Nighat Jamil, "Implementation of Digital Signal Processing real time Concepts Using Code Composer Studio 3.1, TI DSK TMS 320C6713 and DSP Simulink Blocksets," IC-4 conference, Pakistan Navy Engineering College, Karachi, Nov. 2007
- [17] M M El Choubassi, H E El Khoury, C E Jabra Alagha, J A Skaf, M A AL Alaoui, "Arabic Speech Recognition Using Recurrent Neural Networks," Symp. Signal Processing & Info. Tech., 2003, ISSPIT 2003, Dec. 2003, pp. 543-547.
- [18] M A Al-Alaoui, R Mouci, M M Mansour, R Ferzli, "A Cloning Approach to Classifier Training," *IEEE Trans. Systems, Man and Cybernetics – Part A: Systems and Humans*, vol. 32, no. 6, pp. 746-752, 2002.