**APPENDIX B: Functions that used in Speech Recognition system.**

functiontrain\_Callback(~,~)

% -----------------------------------------------------------------------

% Name: training function.

% Author: Hussein M. Mohammed.

% Description: train the neural network with the matrix data that obtained

% from data coefficients function and with the target matrix.

% -----------------------------------------------------------------------

if (current\_method == 0)

msgbox ('Erroe: you must choose a recognizing method','Error','error');

sum = -1;

else

temp = current\_method;

files\_no = dir('\*.wav'); % read all sound files to make a training for

% them.

 Data = coeff(files\_no); % create a matrix to save the sounds' coeffetionts

 % as a columns in it.

sounds = length(files\_no); % number of sound files that was read.

 Target = eye(sounds,sounds); % target of the neural network.

fet = trnng\_files(Data,Target,sounds); % trained sounds.

end

end

function data = coeff(files)

% -----------------------------------------------------------------------

% Name: Data coefficients function.

% Author: Hussein M. Mohammed.

% Description: create a matrix of features that obtained from features

% extraction function.

% -----------------------------------------------------------------------

fori = 1:length(files)

% Read the file to take the coeffecionts for it and save it in a matrix

% data.

filename = files(i).name;

 AA = features\_data(filename);

% The last matrix of data used as an input for the neural network.

for k = 1:length(AA)

data(k,i) = AA(k);

end

end

end

function A = features\_data(file)

% -----------------------------------------------------------------------

% Name: Features extraction function.

% Author: Hussein M. Mohammed.

% Description: Extract features from speech signal depending on the method

% that was already chosen by the user.

% -----------------------------------------------------------------------

[ypre,fs] = wavread(file);

N = 240;

step = 240;

if (test == 0)

% endpoints detection so more processing on the signal will be reduced.

 [~,points] = locatespeech(ypre,N,step,fs);

xpre = ypre(points(1):points(2));

stend = points;

else

xpre = ypre(stend(1):stend(2));

end

if (current\_method == 1) % LPC method.

% Frame blocking.

 l = length(xpre);

 n = 240; % frame size

 m = 120; % overlapping (50%)

frames = floor((l-n)/m) + 1;

for I = 1:n

for J = 1:frames

M(I, J) = xpre(((J - 1) \* m) + I);

end

end

% Hamming for every fraem.

 w = hamming(n);

fori = 1:frames

xw(:,i) = w.\* M(:,i);

end

 p = 12;

fori = 1:frames

lpc\_coef(:,i) = lpc(xw(:,i),p);

end

rr = lpc\_coef(2:end,:);

% Make the data as a one column to put it to the coeffetionts' matrix.

 A = reshape(rr,[],1);

elseif (current\_method == 2)% MFCC method.

% Frame blocking.

 l = length(xpre);

 n = 240; % frame size

 m = 120; % overlapping

frames = floor((l-n)/m)+ 1;

for I = 1:n

for J = 1:frames

M(I, J) = xpre(((J - 1) \* m) + I);

end

end

% Hamming for every fraem.

 w = hamming(n);

fori = 1:frames

xw(:,i) = w.\* M(:,i);

end

fori = 1:frames

M2(:,i) = fft(xw(:, i));

end

mfcc = melfunc(M2);

% take the Cepsetral for the coeffetionts.

rr = dct(mfcc);

% Make the data as a one column to put it to the coeffetionts' matrix.

 A = reshape(rr,[],1);

elseif (current\_method == 3) % Spectrogram method.

 w = hamming(240);

 S = spectrogram(xpre,w,120,240,fs);

% Take the absolute value for the coeffecionts.

 R = abs(S);

% Make the data as a one column to put it to the coeffetionts' matrix.

 A = reshape(R,[],1);

end

end

function [mag,pts] = locatespeech(sig,N,step,fs)

% -----------------------------------------------------------------------

% Name: Zero-Crossing function.

% Author: Hussein M. Mohammed.

% Description: Detect start and end points of the spoken word.

% -----------------------------------------------------------------------

% 1) Remove DC offset

sig\_no\_dc = filter([1, -0.97], 1, sig);

% 2) Compute Avg. Mag and Zero-X rate of sig

m = avgmag(sig\_no\_dc,N,step);

z = zero\_crossing(sig\_no\_dc,N,step);

% 3) Compute mag and zero-crossing of noise (first 100 msec of sig) - already

% computed,

% just cut it out of m and z above

hundredmsec\_rel = round((fs\*.2)/step); % # samples that equals 100ms

% Ends of these may be corrupted due to zero padding -- chop off N/step

% samples from each side

chop = ceil((N/2-step)/step); % round up for safety

noise\_m = m(2+chop:hundredmsec\_rel-chop);

noise\_z = z(2+chop:hundredmsec\_rel-chop);

% Compute means and st. deviations of each, so we can develop thresholds.

noise\_m\_mean = mean(noise\_m);

noise\_m\_std =std(noise\_m);

noise\_z\_mean = mean(noise\_z);

noise\_z\_std =std(noise\_z);

% Set lower thresholds

fudge = 5;

ITL = noise\_m\_mean + fudge\*noise\_m\_std;

IZCT = noise\_z\_mean + fudge\*noise\_z\_std;

% Define upper threshold for avg mag.

ITU = 3.2\*noise\_m\_mean; % since std<< mean, twice the mean should cover it

% find place where sig consistently tops ITU.

start = 3; % since window goes back two spots, start at 3rd sample

avg\_last3pts = 0; % ITU won't be topped in the first 3 pts, so initialize to 0

while avg\_last3pts < ITU,

start = start + 1;

 avg\_last3pts = (m(start) + m(start-1) + m(start-2))/3;

end

% move backwards to find where we first go under ITL

while m(start) > ITL,

start = start - 1;

end

% See if need to move start back due to zero-crossing

below\_izct\_count = 0;

first\_below = -999;

if start > 25

fori = start:-1:start-25,

if z(i) < IZCT

below\_izct\_count = below\_izct\_count + 1;

iffirst\_below == -999

first\_below = i;

end

end

end

ifbelow\_izct\_count>= 3

start = first\_below;

end

end

% Now do the same process backwards for the end;

endpt = length(m)-2;

avg\_last3pts = 0; % threshhold won't occur in first 3 pts, so initialize to 0

while avg\_last3pts < ITU,

endpt = endpt - 1;

 avg\_last3pts = (m(endpt) + m(endpt+1) + m(endpt+2))/3;

end

% move forwards to find where we first go under ITL

while m(start) > ITL,

endpt = endpt + 1;

end

% See if need to move start back due to zero-crossing

below\_izct\_count = 0;

first\_below = -999;

if (endpt-length(z)) > 25

fori = endpt:1:endpt+25,

if z(i) < IZCT

below\_izct\_count = below\_izct\_count + 1;

iffirst\_below == -999

first\_below = i;

end

end

end

ifbelow\_izct\_count>= 3

endpt = first\_below;

end

end

% Return values (multiply endpoints by step so that it is scaled

% appropriately for the actual signal

mag = m(start:endpt);

pts = [start\*step endpt\*step];

end

function out = trnng\_files(coeffdata,T,n\_sounds)

% -----------------------------------------------------------------------

% Name: Artificial neural network function.

% Author: Hussein M. Mohammed.

% Description: Create the neural network that is used to recognize a spoken

% words and initialize all the parameters of the neural network.

% -----------------------------------------------------------------------

% create a neural network to recognize the sound.

[row,col] = size(coeffdata);

hidd = floor((row.\*col)/(5.\*(col + n\_sounds))) + 1;

net = newff(minmax(coeffdata),[hiddn\_sounds],{'logsig''logsig'},

'traingdx');

net.performFcn = 'sse';

net.trainParam.goal = 0.1;

net.trainParam.lr = 0.001;

net.trainParam.show = 20;

net.trainParam.epochs = 1000;

net.trainParam.mc = 0.025;

out = train(net,coeffdata,T);

end

functioncomput\_noise\_Callback(source,~)

% -----------------------------------------------------------------------

% Name: Adding noise function.

% Author: Hussein M. Mohammed.

% Description: Create the popup menu button that contain the values of SNR to

% be added to the speech signal by the user.

% -----------------------------------------------------------------------

SNR = 100;

str = get(source, 'String');

val = get(source,'Value');

% Signal to Noise Ratio in dB.

switchstr{val};

case'No Noise'

 SNR = 100;

case'30 dB'

 SNR = 30;

case'25 dB'

 SNR = 25;

case'20 dB'

 SNR = 20;

case'15 dB'

 SNR = 15;

case'10 dB'

 SNR = 10;

case'5 dB'

 SNR = 5;

end

end

functiontest\_Callback(~,~)

% -----------------------------------------------------------------------

% Name: testing function.

% Author: Hussein M. Mohammed.

% Description: Test a spoken word and give the decision if there is matching

% or there is no matching with a stored spoken word.

% -----------------------------------------------------------------------

clc

if (current\_method == 0) % the user didn`t choose a method.

msgbox ('Erroe: you must choose a recognizing method','Error','error');

elseif (temp ~= sum) % the user didn`t train the network.

msgbox('Error: you must train the network then test it','Error','error');

else

legal\_No = 1;

whilelegal\_No> 0

choice = input('Enter a No. to recognize a word, No. 1 for not trained

words: No. 2 for trained words:');

if (choice == 1)% not trained words testing will be done.

 [filename, pathname] = uigetfile('C:\Users\Ghaith\Desktop\not trained

 words\\*.wav','Select the sound file

that you want to recognize it');

sp = fullfile(pathname, filename);

machindex = dir('C:\Users\Ghaith\Desktop\not trained words\\*.wav');

legal\_No = -1;

elseif (choice == 2) % trained words testing will be done.

set(noise,'visible','on');

set(noise\_ratio,'visible','on');

set(OK,'visible','on');

% select the sound file to be recognized by traind neural

% network.

filename = uigetfile('\*.wav','Select the sound file that you

want to recognize it');

machindex = dir('\*.wav');

 [x,fs] = wavread(filename);

 SNR = 200;

while SNR > 100

figure(1)

plot(x);

xlabel('Samples')

ylabel('Amplitude')

title('Original signal');

end

if (SNR == 100)

sp = filename;

else

 Ok = 0;

while Ok < 1

 y = awgn(x,SNR,'measured');%Add Gausian Noise as a S/N ratio.

figure(1)

plot(y);

xlabel('Samples')

ylabel('Amplitude')

title('Signal with noise');

end

% Denoising the speech signal.

wname = 'coif5'; lev = 10;

tree = wpdec(y,lev,wname);

 det1 = wpcoef(tree,(1:8000));

sigma = median(abs(det1))/0.6745;

alpha = 1.8;

thr = wpbmpen(tree,sigma,alpha);

keepapp = 1;

xd = wpdencmp(tree,'s','nobest',thr,keepapp);

noisy = 'C:\Users\Ghaith\Desktop\noisy10';

wavwrite(xd,fs,noisy);

sp = 'C:\Users\Ghaith\Desktop\noisy10';

end

legal\_No = -1;

else

disp('error: wrong No.');

legal\_No = 1;

end

end

test = 1;

 B = features\_data(sp); % take the coeffecionts of chosen sound file.

 Y = fet(B); % test chosed file by the neural network and take the output.

 K = 1;

max = Y(1);

fori = 2:length(Y)

if(Y(i) > max)

max = Y(i);

 K = i;

end

end

test\_files = dir('\*.wav');

fori = 1:length(machindex)

file = strcmp(filename,machindex(i).name);

if(file == 1)

 Index = i;

end

end

 c = []; % create a matrix for the network output

last\_data = Data;

 o = length(test\_files);

fori = 1:o

c(:,i) = fet(last\_data(:,i));

end

 there\_is\_voice1 = c(K,1);

fori = 1:o

if(c(K,i) > there\_is\_voice1)

 there\_is\_voice1 = c(K,i);

end

end

if(Index == K)

message = strcat('there is matching with: ',test\_files(K).name);

disp(message);

sound(wavread(test\_files(K).name));

figure(1)

plot(wavread(test\_files(K).name));

xlabel('samples')

ylabel('Amplitude')

title('Original signal');

else

msgbox('There is no matching word');

end

end

set(noise,'visible','off');

set(noise\_ratio,'visible','off');

set(OK,'visible','off');

end

functionquit\_Callback(~,~)

closeall;

end