Digit Recognition using Speech Spectrograms

Jay Menon

Arizona State University

EEE 591: Digital Signal Processing

Professor Andreas Spanias

November 25, 2024

Digit Recognition using Speech Spectrograms

Digit classification based on speech has emerged as a crucial component in voice recognition technologies. Here we recognize spoken digits through the application of audio processing and machine learning methodologies. Contemporary voice-activated systems, such as Google Assistant and Siri, significantly depend on these algorithms to accurately interpret spoken instructions.

Speech signals, being inherently non-stationary, necessitate segmentation into shorter, quasi-stationary frames for effective processing. Utilizing spectrograms—visual depictions of frequency content over time—facilitates targeted analysis of speech signals. This method effectively extracts features, especially when used with supervised machine learning techniques like artificial neural networks (ANNs). Studies have shown that multi-layer perceptrons (MLPs) excel in classification by understanding complex, non-linear relationships between features and labels. The emergence of deep learning has markedly enhanced the field of spoken digit recognition. Convolutional Neural Networks (CNNs) have shown exceptional accuracy within this area. By transforming audio signals into spectrograms—a visual representation of frequency spectra over time—CNNs can adeptly learn spatial hierarchies of features, resulting in superior recognition performance. [3] [4] [5] [6]

Results

Deliverables for Part A

A1

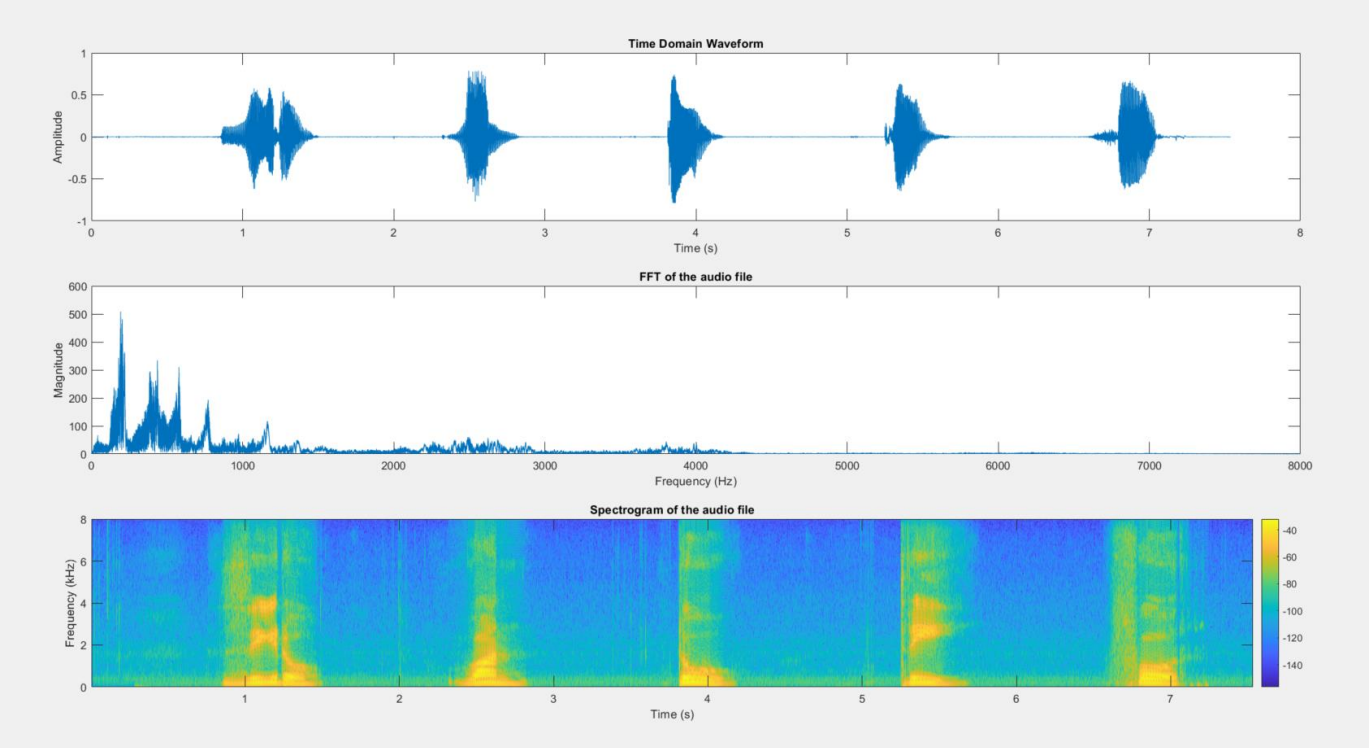

Figure 1

A2

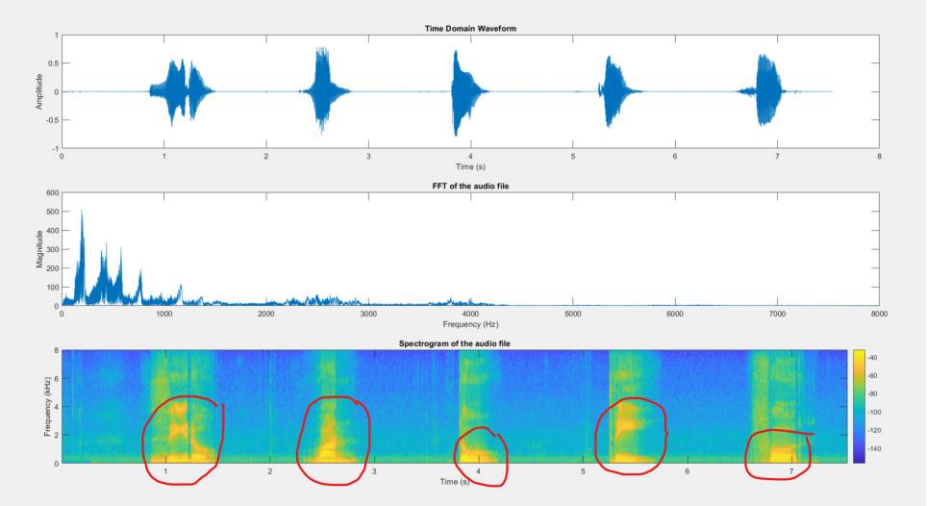

Figure 2

The generated spectrogram reveals yellow areas where strong energy is present in the 0 to 2-4 kHz range, indicating vowels, which are represented by bright, continuous bands in the lower frequency range due to the open vocal tract that produces this sustained energy.

A3

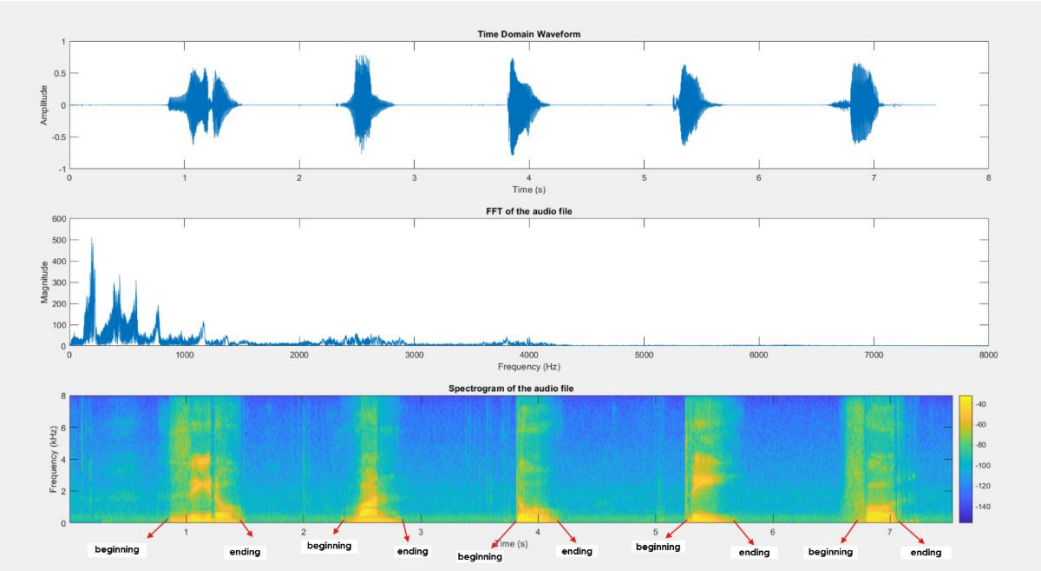

Figure 3

The fundamental frequency for the first vowel is around **200 Hz**, which is visible at the lower end of the frequency axis where the energy is concentrated.

A4

The time-domain plot shows the variations in the amplitude of the speech signal over time, and the FFT plot displays the dominant frequencies, which is helpful for understanding the overall frequency behavior of the signal. The spectrogram shows the localized frequency information over time. By identifying the brighter regions, you can differentiate between vowels and other speech sounds since vowels typically have sustained energy at specific frequencies.

Deliverables for Part B

XOR_dataset.m plots

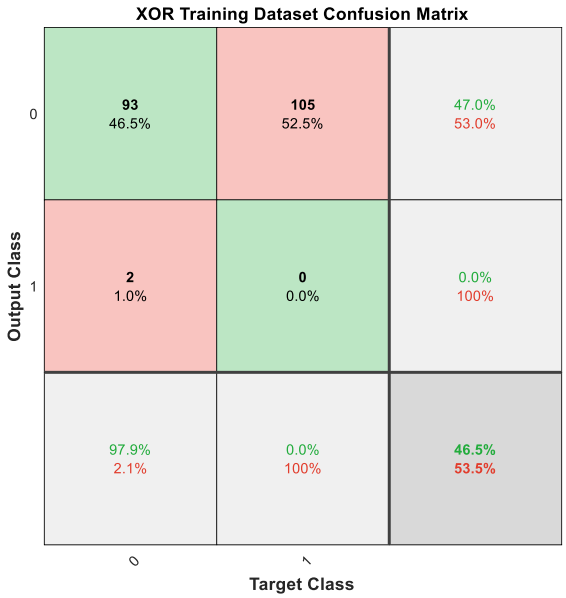

Figure 4

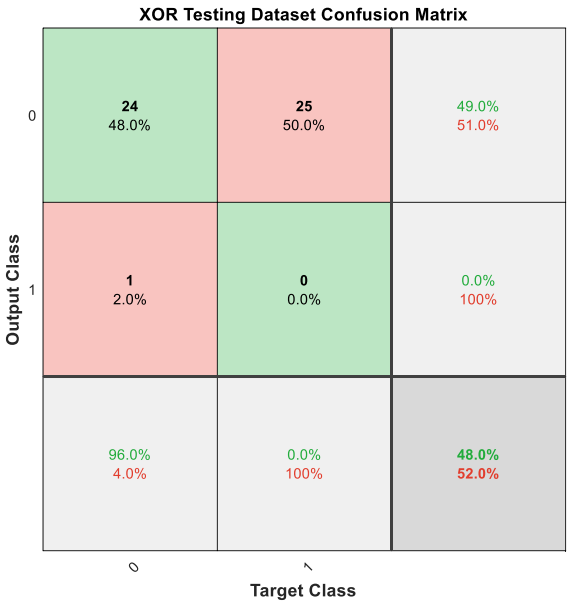

Figure 5

B1

>> part_b

Test Accuracy: **100.00%**

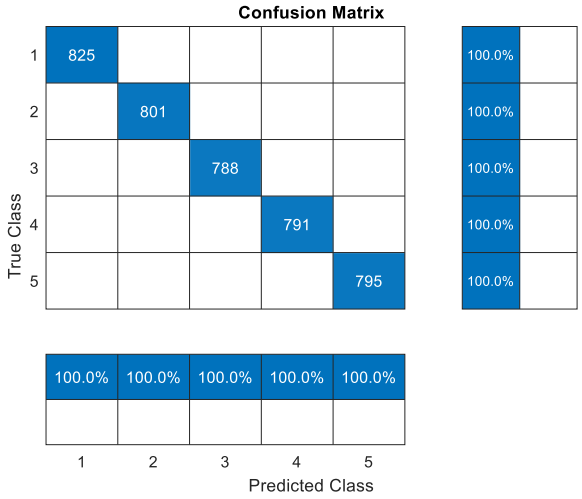

Figure 6

B2

>> part_b

Test Accuracy: **88.33%**

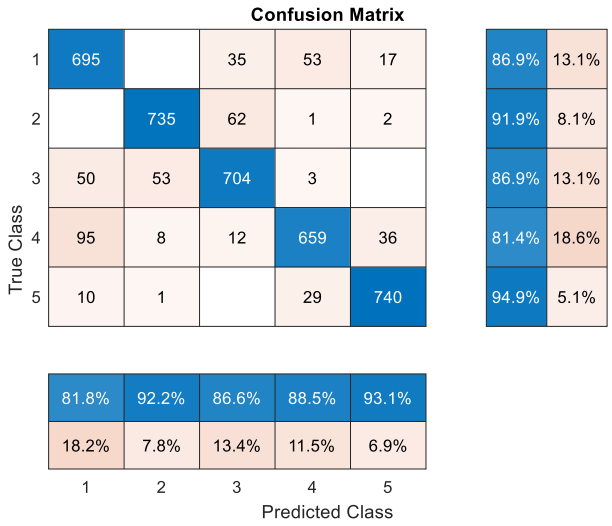

Figure 7

B4

We observed in B1 & B2 that the results showed the impact of data quality on neural network performance. The model achieved a high unrealistic accuracy of 100% on clean data, indicating effective feature learning, but accuracy dropped to around 88.33% when noise was introduced, showing how a noisy dataset can change the perception. Confusion matrices revealed that while clean data predictions mostly aligned along the diagonal, indicating correct classifications, noisy data produced increased misclassification rates, with certain digits being confused more frequently. We learn here the important role of data preprocessing techniques to enhance the model performance. Analyzing confusion matrices also provided insights for targeted model improvements and data collection efforts to tackle specific classification challenges.

Conclusion

This study successfully showed that recognizing spoken digits can be done by using speech spectrograms and machine learning techniques. We initially transformed speech signals into spectrograms to capture the key frequency and timing information needed for digit classification. However, once we added noise, it significantly affected the model's performance, reducing its accuracy from an ideal 100% down to 88.33%. This drop showcases how important data quality and preprocessing are for building effective speech recognition systems. Finally, the confusion matrices revealed specific parts where noise led to misclassifications.

References

[1] A. Spanias, Digital Signal Processing; An Interactive Approach – 2nd Edition, 403 pages,

Textbook with JAVA exercises, ISBN 978-1-4675-9892-7, Lulu Press On-demand Publishers

Morrisville, NC, May 2014.

[2] U. Shanthamallu, A. Spanias, C. Tepedelenlioglu, M. Stanley, "A Brief Survey of Machine Learning Methods and their Sensor and IoT Applications," Proceedings 8th International Conference on Information, Intelligence, Systems and Applications (IEEE IISA 2017), Larnaca, August 2017.

[3] A. Khemani, "Spoken Digit Recognition (Speech Recognition)," Stanford University CS230 Project Report, Fall 2020. https://cs230.stanford.edu/projects_fall_2020/reports/55617928.pdf

[4] S.I. Safie, “Spoken Digit Recognition Using Convolutional Neural Network,” pp. 24–27, May 2022, doi: https://doi.org/10.1109/aiic54368.2022.9914596.

[5] A. J. M. Adoptante, A. M. Baes, J. C. A. Catilo, P. K. L. Lucero, A. L. P. De Ocampo, A. S. Alon, and R. M. Dellosa, "Spoken-digit classification using artificial neural network," ASEAN Engineering Journal, vol. 13, no. 1, Mar. 2023, doi: 10.11113/aej.v13.18388.

[6] MathWorks, "Spoken digit recognition with wavelet scattering and deep learning," MathWorks Documentation, 2023. [Online]. Available: https://www.mathworks.com/help/wavelet/ug/spoken-digit-recognition-with-wavelet-scattering-and-deep-learning.html.

Appendix

Code for Part A:

```
[x, fs] = audioread('cleanspeech.wav');
t = (0:length(x)-1) / fs;

figure;

subplot(3, 1, 1);
plot(t, x);
title('Time Domain Waveform');
xlabel('Time (s)');
ylabel('Amplitude');

N = length(x);
X_fft = fft(x);
f = (0:N-1)*(fs/N);
magnitude = abs(X_fft);

subplot(3, 1, 2);
plot(f(1:floor(N/2)), magnitude(1:floor(N/2)));
title('FFT of the audio file');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3, 1, 3);
window_length = 128;
overlap = 120;
nfft = 1024;

spectrogram(x, window_length, overlap, nfft, fs, 'yaxis');
title('Spectrogram of the audio file');
colorbar;
```

Code for Part B:

```
load noisydigitrecognition.mat;
% load digitrecognition.mat;
load labels.mat;

rand_ind = randperm(10000);
data = noisydigitrecognition(rand_ind, :);
% data = digitrecognition(rand_ind, :);
labels = labels(rand_ind, :);

figure;
num_samples_to_plot = 10;
for i = 1:num_samples_to_plot
    subplot(2, 5, i);
    imagesc(reshape(data(i, :), [28, 28]));
    % colormap('gray');
    axis off;
    title(['Sample ', num2str(i)]);
end
sgtitle('Visualization of Noisy Digit Recognition Data');
```

```
% Test train split
num_train = round(0.6 * size(data, 1));
X_train = data(1:num_train, :);
y_train = labels(1:num_train, :);
X_test = data(num_train+1:end, :);
y_test = labels(num_train+1:end, :);

trainFcn = 'trainscg';
hiddenLayerSize = [12 16 12];
net = patternnet(hiddenLayerSize, trainFcn);

net.trainParam.showWindow = 0;
net.divideParam.trainRatio = 0.8;
net.divideParam.valRatio = 0.2;
net.divideParam.testRatio = 0;

[net, tr] = train(net, X_train', y_train');

% Testing
y_pred = net(X_test');
y_pred_classes = vec2ind(y_pred); % Prediction
y_test_classes = vec2ind(y_test');

accuracy = sum(y_pred_classes == y_test_classes) / length(y_test_classes) * 100;
fprintf('Test Accuracy: %.2f%%\n', accuracy);

figure;
plotperform(tr);
title('Training Performance'); % optional

% Confusion matrix
figure;
confMat = confusionchart(y_test_classes, y_pred_classes);
confMat.Title = 'Confusion Matrix';
confMat.RowSummary = 'row-normalized';
confMat.ColumnSummary = 'column-normalized';
```

Code for Part B – XOR_dataset.m:

```
clc;
clear all;
close all;

No_train = 200;     % No. of training points
No_test = 50;       %No. of test points
hiddenlayersize = [6 6];


[X_train,y_train] = generate_dataset(No_train);
gscatter(X_train(:,1), X_train(:,2), y_train);
title('XOR Train Dataset');
figure;

[X_test,y_test] = generate_dataset(No_test);
gscatter(X_test(:,1), X_test(:,2), y_test);
title('XOR Test Dataset');
```

```
figure;

net = neural_network(hiddenlayersize, X_train, y_train);

confusionmatrix(net, X_train, y_train, 'Training');
figure;
confusionmatrix(net, X_test, y_test, 'Testing');


function confusionmatrix(net,X,y,name)

    %Neural network output
    outputs = net(X');

    %Training Dataset Confusion Matrix
    plotconfusion(y',outputs);
    title(sprintf('XOR %s Dataset Confusion Matrix', name));
end


function net = neural_network(hiddenlayersize, X, y)
    %% Neural network training
    trainFcn = 'trainscg';

    % Hidden Layer dimensions (Can be customized)
    hiddenLayerSize = hiddenlayersize;

    %Neural network object
    net = patternnet(hiddenLayerSize, trainFcn);
    net.trainParam.showWindow = 0;

    %Training parameters
    %Here we are training on the entire dataset
    net.divideParam.trainRatio = 1;

    %Training process
    [net,tr] = train(net,X',y');
end

function [xor_dataset,labels] = generate_dataset(N)

    %N no. of points generated from a normal distribution
    xor_dataset = randn(N,2);

    %Performing XOR operation on the columns and converting logical variables to double data
type
    y_xor = double(xor((xor_dataset(:,1)>0), (xor_dataset(:,2)>0)));

    %Labels of the dataset
    labels = y_xor;

end
```