EE264 Final Project Report

Project Members: Elaina Chai
echai@stanford.edu

Proposal Due Date: February 9, 2015
Final Project Due Date: March 19, 2015

Introduction

Speech recognition continues to this day to be a major area of research and commercial importance. With the rise of mobile applications and Internet of Things (IoT), the importance of fast and energy-efficient speech recognition algorithms will only continue to grow in the coming decades.

In this work, I implement a pre-processing front-end speech dependent key word recognizer on a combined c55 DSP/Matlab platform. The DSP platform extracts the periodogram in real-time. The periodogram is used to compute the mel-frequency cepstrum coefficients (mfcc) [1][2], which are used as input features to a machine learning classifier implemented in Matlab.

The training data used will be self-generated voice clips. The clips will be voice recordings captured using the DSP shield + microphone. These voice clips will be pre-processed to reduce noise and dead time. The final testing data will be these voice clips augmented with noise recordings

Timeline

(Original) Dates	Milestones
Week 1 (2/16 – 2/22)	Speech Recognizer Matlab Back-End Setup/Training
Week 2 (2/23 – 3/01)	Digital Filter Design/Implementation in Matlab
Week 3 (3/02 – 3/08)	Speech Recognizer Inference/Demo Preparation
Week 4 (3/09 – 3/13)	Demo + Report Write-up

(Actual) Dates	Milestones
Week 1 (2/16 – 2/22)	-
Week 2 (2/23 – 3/01)	Keyword Recognizer Back-End Setup/Implementation in Matlab, Data Collection I
Week 3 (3/02 – 3/08)	Speech Recognizer Inference/Demo Preparation I, Data Collection II
Week 4 (3/09 – 3/13)	Demo Preparation II
Week 5 (3/16-3/19)	Report Write-up

Class Concepts Used

1) FFT
2) DFT
3) Q15 Fixed Point

   The c55 DSP can only perform fixed-point operations. Understanding the hazards from overflow/underflow and quantization was crucial to ensuring that a valid output was produced by the DSP
4) Periodogram

   MFCC are based on the weighted power spectrum of the speech signal. The periodogram provides an estimate of this function. Understanding time averaging allows for processing the speech segment in segments.
5) Short-time Fourier Transforms

   Short-time Fourier Transforms/Time-Dependent Fourier Transforms were used to process the speech signal, instead of a single FFT over the entire speech signal.

Implementation

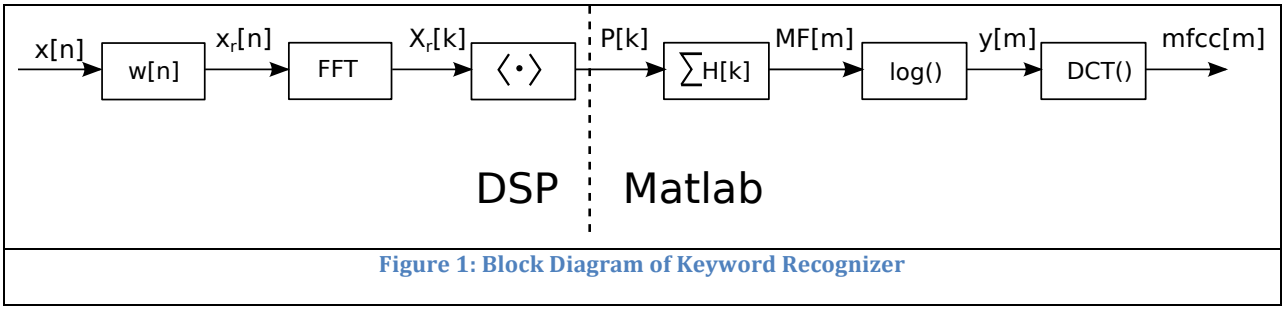

Figure 1: Block Diagram of Keyword Recognizer

Pre-processing Chain Equations

The following is a description of the algorithm used in the pre-processing/classification chain. Essentially, I am using the DSP to compute the average of the periodogram in real-time, by averaging across non-overlapping samples of the speech signal:

1) A speech signal is sampled at 8kHz by the DSP audio codec to produce sequence x[n]
2) The sequence is split into L segments, modeled as a windowing operation. A time-dependent Fourier transform is performed on segment $x_r[n]$ to produce $X_r[k]$, the DFT for segment r (for $0 < r < L-1$):

$$x_r[n] = x[rR + n]w[n]$$

$$X_r[k] = \sum_{n=0}^{N-1} x_r[n]W_N^{kn}$$

3) The periodogram for segment r is calculated, then averaged across all L segments [7]:

$$S_r[k] = \frac{1}{N}|X_r[k]|^2$$

$$P[k] = \frac{1}{L}\sum_{r=0}^{L-1} S_r[k]$$

4) P[k] is returned to the Matlab for final processing. The sequence P[k] is multiplied by the weighting functions and summed to produce the mel-spectrum:

$$MF[m] = \sum_{k=0}^{k=N} P[k]H_m[k]$$

5) A logarithm and a discrete cosine transform (DCT) is performed to extract the final mfcc [6]:

$$mfcc[n] = \frac{1}{M}\sum_{m=1}^{M} \log(MF[m]) \cos\left[\frac{2\pi}{M}\left(m + \frac{1}{2}\right)n\right]$$

mfcc[m] is used as the input features to my classifier

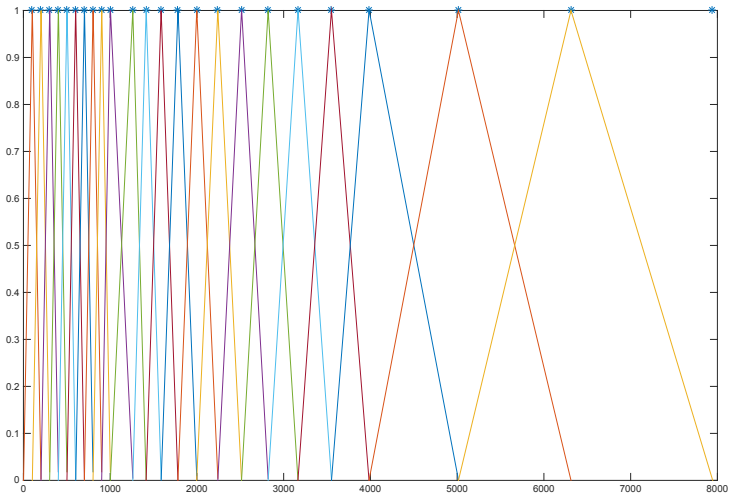

Figure 2: Filter Banks for MFCC Extraction

Data Collection/Generation

For this implementation, I used a very limited word list, as follows:

Word List	
1	apple
2	Computer
3	Windows

4	Digital
5	laptop

To collect data for training and testing, I used the DSP to record my voice at 48kHz (`AudioRecorder.ino`). I repeated the above words 12 times each, and these recordings were parsed and downsampled to 8kHz. The clean versions of the recordings extracted using `word_parse.m` (see Appendix C) and were used as training data.

Parameter Training and Classification

The classifier used was a soft-max regression classifier, trained used a gradient descent algorithm.[3] The script used for training can be found in `mfcc_matlab_echai.m` (see Appendix C). The trained parameters can be found in Appendix A. Below is a color map of these parameters:

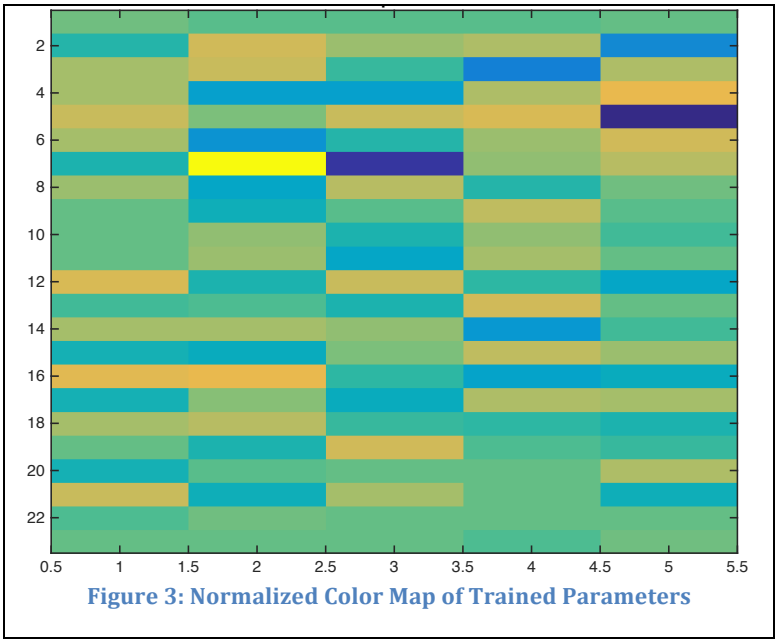

Figure 3: Normalized Color Map of Trained Parameters

In `test_data.m` (see Appendix C) I test the accuracy of my system against noisy input. For test data, I augmented the original voice recordings with noise samples [5] of increasing scale.

Design Considerations

To realize real-time operation of the pre-processing front-end, I had to take into account the following design considerations:

1) Functions available in c55 DSP library

Any function available for use in the DSP Programmer's Reference, would be naturally optimized for use on the c55 DSP platform. As a result, they would be faster than any equivalent

function I could implement manually in C++. Hence, I tried to ensure that as much of my implementation as possible used these available libraries.

Therefore, if I were to migrate the final steps of the mfcc extraction, i.e. the logarithm and DCT, onto the DSP, this would have required implementing look-up tables. In the current implementation, due to limitations on the project timeline, and the fact that the logarithm and DCT do not need to be computed in real time, I chose to leave this implementation in Matlab.

2) Clock speed/ Buffer length limits of DSP

In the final implementation, the size of the speech segments, and by extension the size of the DFT and sampling frequency, was set by the limitations of the c55 DSP.

The `cfft()` function, used to compute the DFT of the speech signal can only perform a maximum of N =2048 DFT of my speech signal. Additional tests showed that to maintain real-time performance (processing of signals without noticeable skipping on the output), the actual maximum size of my DFT was N =512 at 8kHz.

3) Overflow dangers from using Q15 format

4) Quantization/Underflow from fixed point

Results

Accuracy Tests

For testing, I augmented the clean speech data with noise samples [5] of increasing scale. This augmented data was used to test classification accuracy, whose results are shown below:

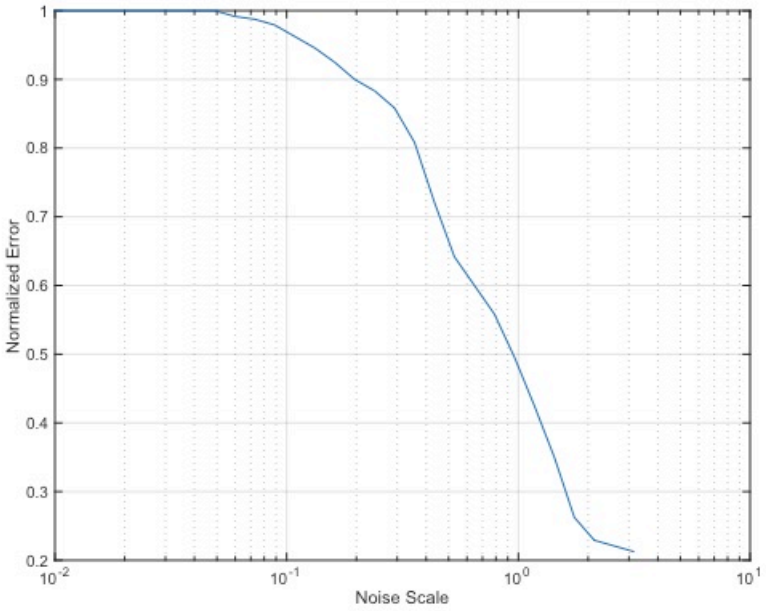

Figure 4: Normalized Accuracy vs. Noise Scale over Entire Test Set

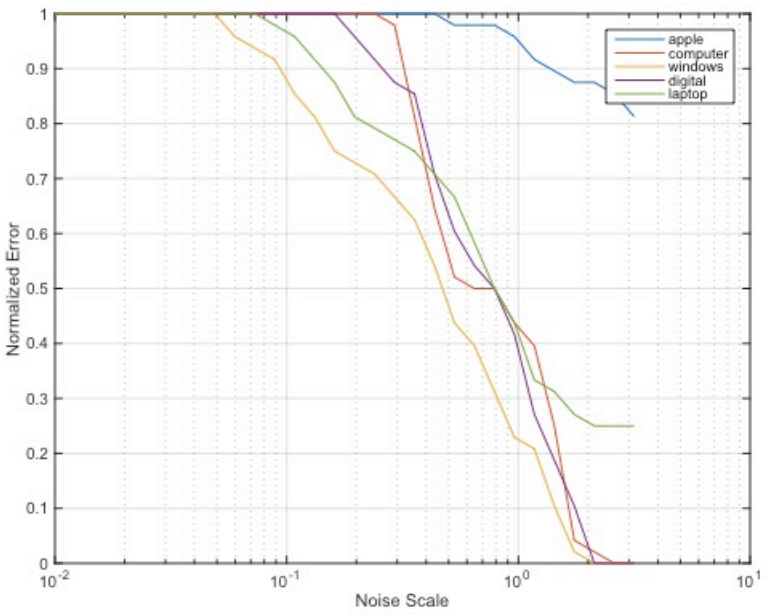

Figure 5: Normalized Accuracy vs. Noise Scale per Word

From the above data, we can see that the system is very sensitive to noise. Additionally, as the noise increases, the classifier "converges" on two of the 5 words available.

Using the DSP front end and the microphone as input, I had limited success. Though this is harder to quantify with a graph, I did observe that of all the words, "windows" was the most likely to be misclassified. This agrees with the above graph, which shows that the accuracy of the word "windows" falls off the fastest of all the possible words.

Real Time Periodogram Estimation using DSP

Below are the periodogram estimate output using the DSP front-end implementation:

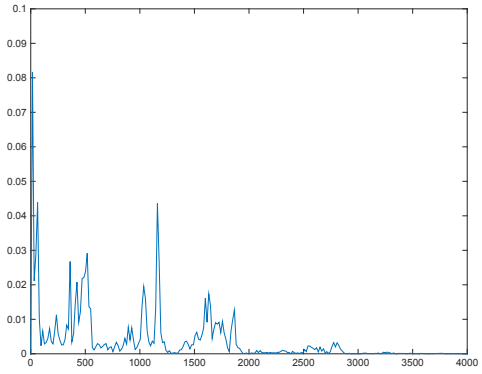

Figure 6a: Periodogram of "Apple"

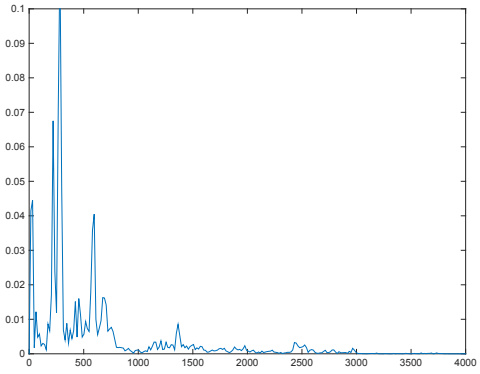

Figure 7: Periodogram of "Computer"

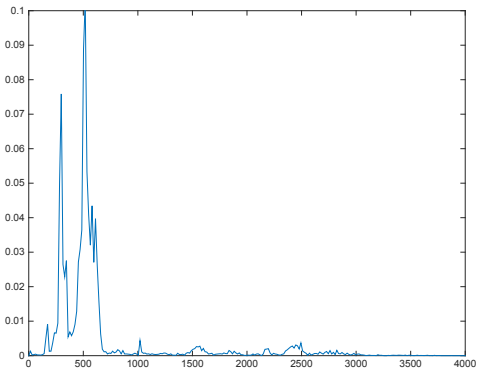

Figure 8: Periodogram of "Windows"

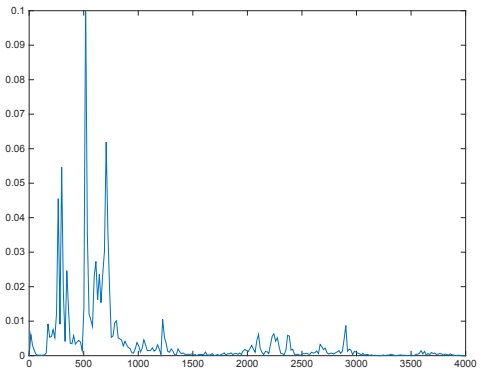

Figure 9a: Periodogram of "Digital"

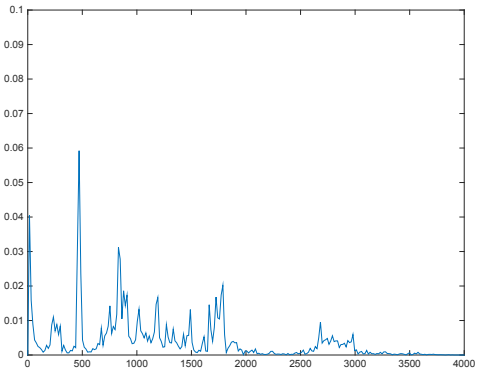

Figure 10: Periodogram of "Laptop"

Effects not Captured by this System

1) Time Dependent Effects: The MFCC is extracted from an estimation of the power spectrum, averaged over the entire speech signal. This means that multi-syllabic words are naturally difficult to classify, as time-dependent effects are not captured in the features. Indeed, in [1], the authors admitted that while their method works reasonably well for monosyllabic words, multi-syllabic words still remain a challenge.

2) The Lombard Effect and other vocal inflections/stresses: The Lombard Effect is the human tendency to add stress or inflections to words spoken in different environments. The speech recordings used to the train the system were taken in the same environment. Using the DSP front-end in different environments can lead to stresses/inflections in the speech data that can affect the accuracy of the system.

Limits on Scaling to larger Word Lists

The current system was trained to identify a keyword from a list of 5 words. While there were attempts to expand this system to larger word lists, these effort lead to failure. For larger number of classes (i.e. larger word lists), the gradient descent algorithm failed to converge on a

set of parameters that could even classify the training data. Why this is the case (an error in the implementation, limited number of features, etc) requires further investigation.

Comparisons to State-of-the-Art

Many of the problems encountered are addressed in more modern speech recognition systems. "Deep Speech" by Baidu Research[4], trains on noisy data, using the spectrogram of the speech data as input. This system uses short-time Fourier transform, with a window length of 20ms, overlap length of 10ms, at a sampling frequency of 16kHz.

Future Work

Possible future work:

- Experiment with different windowing functions and observe the effect on the output
- Use the spectrogram as the input into the classifier
- Augment the training data to capture the effects due to voice inflections/stresses

I will not keep the Lab in a Box

References

[1] Davis, S and Mermelstein, P. (1980) "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoke Sentences" *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), pp. 357–366

[2] Lyons, James (2015, February 24). "Mel Frequency Cepstral Coefficient (MFCC) tutorial". Retrieved from http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/#computing-the-mel-filterbank

[3] Ng, A. (2015, February 24) "Supervised Learning, Discriminative Algorithms". Retrieved from http://cs229.stanford.edu/notes/cs229-notes1.pdf

[4] Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y. Ng. "Deep Speech: Scaling up end-to-end speech recognition." CoRR abs/1412.5567 (2014)

[5] Ellis, Dan (2015 March 19). "Noise". Retrieved from http://www.ee.columbia.edu/~dpwe/sounds/noise/

[6] Schafer, R. (2008). Homomorphic Systems and Cepstrum Analysis of Speech. In *Springer Handbook of Speech Processing* (pp. 161-180). Springer.

[7] Oppenheim, A., Schafer, R. (2010) Fourier Analysis of Stationary Random Signals: the Periodogram. In *Discrete-Time Signal Processing*. (pp 836-843). Prentice Hall.

Appendix A: Trained Parameters

7.6320	-1.4052	-6.4492	-2.7143	2.8251
-31.5035	69.5479	30.0847	41.6172	-109.8716
37.8801	58.3501	-19.3556	-119.0427	41.9070
37.7914	-80.7746	-83.5532	43.1848	83.4221
60.6803	13.7081	63.9062	75.0960	-213.5021
39.5757	-101.2430	-35.5759	29.2183	68.0034
-38.8994	164.4327	-201.6585	24.6101	51.3707
34.4876	-65.8832	51.2159	-30.7359	10.8959
2.3055	-50.9836	-1.2023	53.4418	-3.7397
0.9665	25.8106	-36.5852	23.8264	-13.8234
0.4031	33.0652	-70.9773	34.9714	2.3831
75.3349	-41.4530	60.0881	-28.0759	-65.9752
-18.5059	-8.1072	-37.7020	65.0948	-0.6112
39.4060	40.19048	26.0758	-92.1824	-13.4975
-44.5174	-59.4673	15.8181	53.8546	34.5044
77.7403	85.8325	-29.7002	-76.1764	-57.6864
-46.4501	20.3607	-56.2952	46.1604	35.8111
38.1917	50.2336	-21.8637	-26.3099	-39.9001
1.2156	-36.9871	65.0550	-9.0047	-20.3924
-42.8326	-6.0529	4.2920	1.7221	43.1626
59.8255	-50.1173	39.6213	3.9556	-53.2341
-11.2289	7.6793	0.4133	0.9447	2.1137
0.6198	0.4635	2.4826	-12.2897	8.3673
7.6320	-1.4052	-6.4492	-2.7143	2.8251

Appendix B: DSP Code API for Calculating Periodogram

Audio Periodogram

Author

Elaina Chai

Reads data from codec audio IN, process the data and sends the output to the codec OUT which can be listened on headphone.

Commands included to calculate periodogram using short-time fourier transform

Five Commands to process data:

cmd 30:

- Receive window of 512 real-valued Q15 intergers from Matlab
- Save in buffer window[BufferLength]

cmd 31:

- Retrieve data in Left and Right Audio Buffers
- Each buffer of length 512
- Output raw buffers to Matlab, Left first, then Right

cmd 32:

- Retrieve data in Left and Right Audio Buffers
- Each buffer of length 512
- Multiply with window[BufferLength]
- Save in AudioLeft[BufferLength] and AudioRight[BufferLength] respectively
- Output audioLeft and AudioRight to Matlab, AudioLeft first, then AudioRight

cmd 33:

- Retrieve data in Left and Right Audio Buffers
- Each buffer of length 512
- Multiply with window[BufferLength]
- Save in AudioLeft[BufferLength] and AudioRight[BufferLength] respectively
- Perform real FFT on AudioLeft and AudioRight.
    1. Resulting Buffers contain BufferLength/2 samples, with interleaved real and imaginary values
    2. Recall FFT of real values is symmetric
    3. Reconstruction of complete DFT to be done in Matlab
- Output audioLeft and AudioRight to Matlab, AudioLeft first, then AudioRight

cmd 34:

- Retrieve data in Left and Right Audio Buffers
- Save in AudioLeft[BufferLength] and AudioRight[BufferLength] respectively
- Save in interleaved format
- Perform FFT using cfft()
- Accumulate Periodogram Estimate

cmd 94:

- Return Periodogram Estimate left audio buffer to matlab
- Return Periodogram Estimate left audio buffer to matlab

Additional Functions:

- mult(int* a, int*b, int *c, int length)
    1. Multiply two real vectors *a* and *b* of size 'length', and return in *c*
    2. Q15 format assumed for all input and output
- PeriodEst(int *x, int x_dataLength, int *output)
    1. Compute PeriodGram estimate of of data and accumulate in output
    2. X has x_datalength samples, or 2*x_datalength entries, because it is complex interleaved
    3. iterate over x_dataLength/2
    4. output has x_datalength/2 samples, nad contains only real values
    5. User MUST RESET Output before using

Appendix C: Matlab Code API

mfcc_matlab_echai

```
  Elaina Chai
  EE264
  Final Project
  March 2, 2015

 ************************************************************************
  Keyword Recognizer
  This is a Matlab implementation of a keyword recognizer
  Key steps are
  1) Generate MFCC filterbanks
  2) Extract MFCC from sound files
  3) Separate training and testing data
  4) Train on training data using gradient descent
  5) Classification on testing data

  For the final DSP implementation, the MFCC filterbanks will be saved on
  the DSP, and step 2 and step 5, i.e. extracting the MFCC from the sound
  and final classification, will implemented on the DSP

  Future work:  Add acceleration/decceleration coefficients to feature vector
                Noise reduction of raw input from microphone
                Dead-time removal of sound


  Train in floating point
  At times the training intermediary values, such as gradient
  becomes very small
  Testing is done using rounding and Q15 format
 ************************************************************************
```

Test_Data

```
  Elaina Chai
  EE264
  Final Project
  Test_Data.m
*************************************************************************
  Script to test keyword recognizer using pre-recorded speech data.
  Key steps are as follows:
  1) Load clean speech data
  2) Augment with noise data
  3) Extract mel-frequency cepstrum coefficient
  4) Classify using pre-trained parameters
  5) Compute and plot accuracy vs noise scale

*************************************************************************
```

word_parse

```
  Elaina Chai
  EE264
  Final Project
  word_parse.m
 ************************************************************************
  Script to extract speech data from raw file created by Audio Recorder

  Input data are speech recordings cpatured using Audio Recorder running
  on c55 DSP. A single speech recording will have >10 recordings of words
  This script will automatically locate and extract these words

  Option to augment with noise or generate with clean speech data

  Key steps are as follows:
    1) Load raw sound file
    2) Locate sound bites with words
    3) Extract word clips from words
    3a) Augment with noise if desired
    4) Save words to individual .wav files at 48kHz
```

AudioPeriodgramTest

Contents of **AudioPeriodgramTest**:

AudioPeriodgramTest is both a directory and a function.

```
  Elaina Chai
  EE264
  Final Project
  AudioPeriodgramTest.m
 ************************************************************************
  Script to test DSP implementation of Periodogram Estimator
  Key steps are as follows:
    1) Estiablish serial connection
    2) Initiate Periodgram Estimator (cmd 34). Script pauses at this point
    3) On DSP, LED2 will blink three times before staying on
    4) Speak word into microphone while LED2 is steadily on.
    5) When LED2 turns off, continue matlab program
    6) Return Periodogram (cmd 94)
    7) Run final classification to determine word spoken
 ************************************************************************
```

matlab_dsp_prototype

```
  Elaina Chai
  EE264
  Final Project
  matlab_dsp_prototype.m
*************************************************************************
  Matlab implementation of DSP Periodogram Estimator and Final
  Classification

  Can be used to test trained parameters using any pre-recorded speech
  data, including sound files not part of original training set
  Key steps are as follows:
    1) Load sound file
    2) Downsample to 8 kHz
    3) Perform short-time FFT
    4) Calculate Periodogram
    5) Calculate mel-frequency cepstrum coefficients using log() and DCT
    6) Perform final classification


*************************************************************************
```

PeriodEst

```
  Elaina Chai
  EE264
  Final Project
  PeriodEst.m
*************************************************************************
  function output = PeriodEst(record,N, MFCC_filter_Q15, mf_index, mf_center)
  Function computes the mel-frequency cepstrum coefficients from a given
  sound file
  Coefficients are stored in 'output' vector
  Parameters:
      1) record: sound file to be processed
      2) MFCC_filter_Q15: Filter banks to compute Periodogram Estimate
      3) mf_index: vector to index into filter bank matrix
      4) mf_center: Vector to help with indexing into filter bank matrix

      Parameters MFCC_filter,Q15, mf_index, mf_center are saved in mfcc_param_#.mat file
*************************************************************************
```