Speaker Recognition

EEC 201 Winter 2024 Final Project

Logan Simmons (919158205)
Benjamin Fung (918498539)

Approaches

To start, we used the haythamfayek link posted in the project2024.pdf. This was a bit difficult to understand since the programming language used in the paper was in Python, while we used Matlab to code our project. The link proved useful in helping us to better conceptualize framing, and the steps needed in order to ensure your frames are appropriately sized (you don't want decimals). We opted to avoid pre-emphasis though, as it did not seem necessary for the samples we were given.

Following our specifications of N and M, both dependent upon our desired time span of a frame (about 30 ms); we needed to window each frame and find the fourier transform of each windowed frame. This was done by using the Matlab function: "stft," which broke each audio file up into L frames, each N samples in length, With N-M overlap; windowed those frames, and performed the stft of each frame.

Next, to find the mel-frequency wrapped spectrum of our signal we used the melfb function provided in the Final Project folder, as well as a basic for loop. At first, we decided to use the standard mel-spaced filter number of 20. We changed the filter number as time went on, but 20 proved satisfactory when simply testing the samples given, that were not the ones recorded in class.

In order to ensure that our mel-frequency wrapped values were correct, we compared the spectrogram of our computed mel-frequency spectrum to the spectrogram given by the matlab function: "melspectrogram". The two appeared nearly identical in nature, and after specifying the correct number of filters in the "spectrogram" function, the difference in appearance decreased a bit more.

Next, in order to find the mfcc, we applied the dct to the log of the mel-frequency spectrum using a basic for loop. This was done to get the mel-coefficients back to time domain.

To generate the codebooks for each speaker we used an LBG algorithm. The algorithm works by generating a 1 vector codebook and iterating through a loop k times, to generate 2^k codewords per frame of each speaker.

Finally, the testing phase simply catalogues the average minimum Euclidian distances between each speaker's codewords and the mfccs of a test speaker, and determines that the person with the smallest average minimum distance must be the test speaker.

Tests

TEST 1: To begin, use the voice files of "zero" that we provided (not from the class). Play each sound file in the TRAIN folder. Can you distinguish the voices of the given 11 speakers in the database? Next play each sound in the TEST folder in a random order without looking at the groundtruth and try to identify the\speaker manually. Record what is your (human performance) recognition rate. Use this result as a later benchmark.

TEST 2: In Matlab one can play the sound file using "sound". Record the sampling rate and compute how many milliseconds of speech are contained in a block of 256 samples? Now plot the signal to view it in the time domain.

Use STFT to generate periodogram. Locate the region in the plot that contains most of the energy, in time (msec) and frequency (in Hz) of the input speech signal. Try different frame size: for example N = 128, 256 and 512. In each case, set the frame increment M to be about N/3.

TEST 3: Plot the mel-spaced filter bank responses. Compare them with theoretical responses (e.g. triangle shape that you expected). Compute and plot the spectrum of a speech file before and after the mel- frequency wrapping step. Describe and explain the impact of the melfb.m or melfbown.m program.

TEST 4: Complete the "Cepstrum" step and put all pieces together into a single Matlab function, e.g., mfcc.m

TEST 5: To check whether the program is working, inspect the acoustic space (MFCC vectors) in any two dimensions in a 2D plane to observe the results from different speakers. Are they in clusters? Now write a function that trains a VQ codebook using the LGB algorithm.

TEST 6: Plot the resulting VQ codewords using the same two dimensions over the plot of in TEST 5.

TEST 7: Record the results. What recognition rate can your system achieve? Compare this with human accuracy. Experiment and find the reason if high error rate persists. Record more voices of yourself and your teammates/friend. Each new speaker can provide one speech file for training and one for testing. Record the results.

TEST 8: Use notch filters on the voice signals to generate another test set. Test your system on the accuracy after voice signals have passed different notch filters that may have suppressed distinct features of the original voice signal. Report the robustness of your system.
From this point on, we are recording our own speeches as tests.

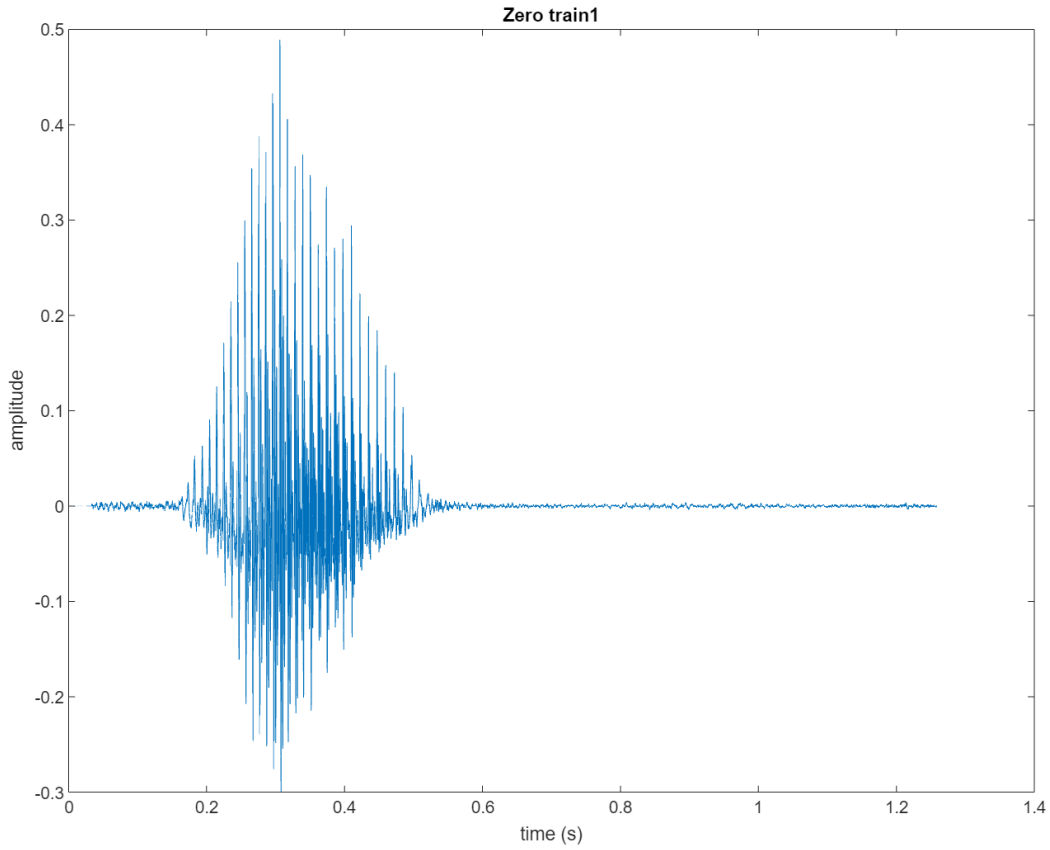
TEST 9: Assuming that your system has been trained with the non-students' "zero" speeches, we now augment the problem by adding 10 students' voice as follows: (a) Randomly select speech "zero" of 10 students from EEC 201 we recorded twice: one for training and one for recognition test. (b) Next, retrain your system by adding our own recorded speech to our existing speakers' samples. (c) Test the accuracy of your system and compare the accuracy obtained previously. NOW this system can identify more speakers that include the original set of speakers + the 10 students.

TEST 10: Use all samples of EEC 201 students saying "zero" and "twelve". Retrain the speech recognition system and test the accuracy in terms of the accuracy to identify the different "zero" and "twelve" sounds. Question 1: If we use "twelve" to identify speakers, what is the accuracy versus the system that uses "zero"? Question 2: If we train a whole system that tries to identify a) which speaker, and b) whether the speech is "zero" or "twelve", how accurate is your system?

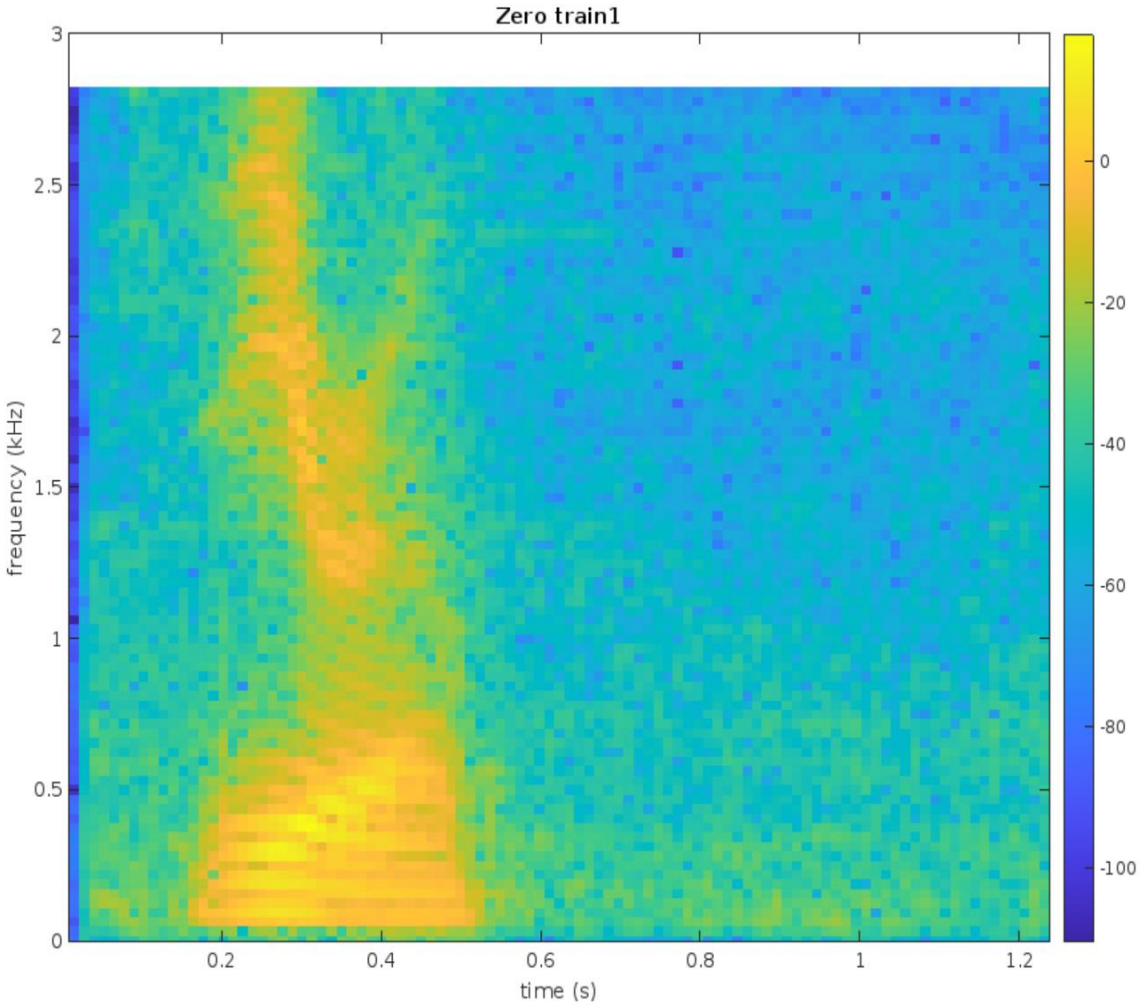
Test Results

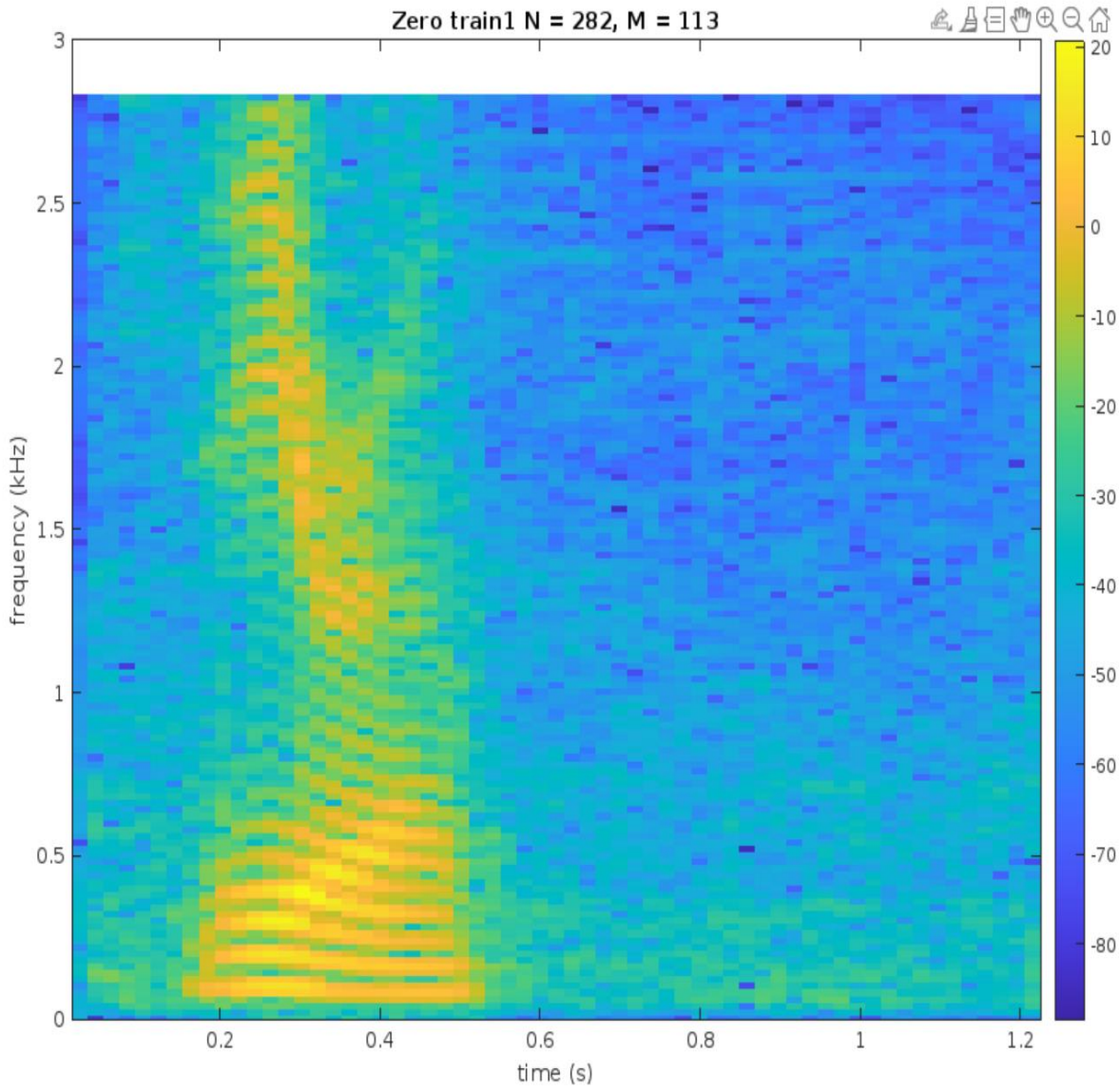
1) For Test 2 I was able to distinguish most speakers from each other, however, I had difficulty distinguishing between speakers 3, 6 and 8. When playing each file in the TEST folder, I was able to identify all speakers except speakers 3 and 8. I confused speaker 3 for speaker 6 and speaker 8 for speaker 6. So, my human performance recognition rate was 75 percent.

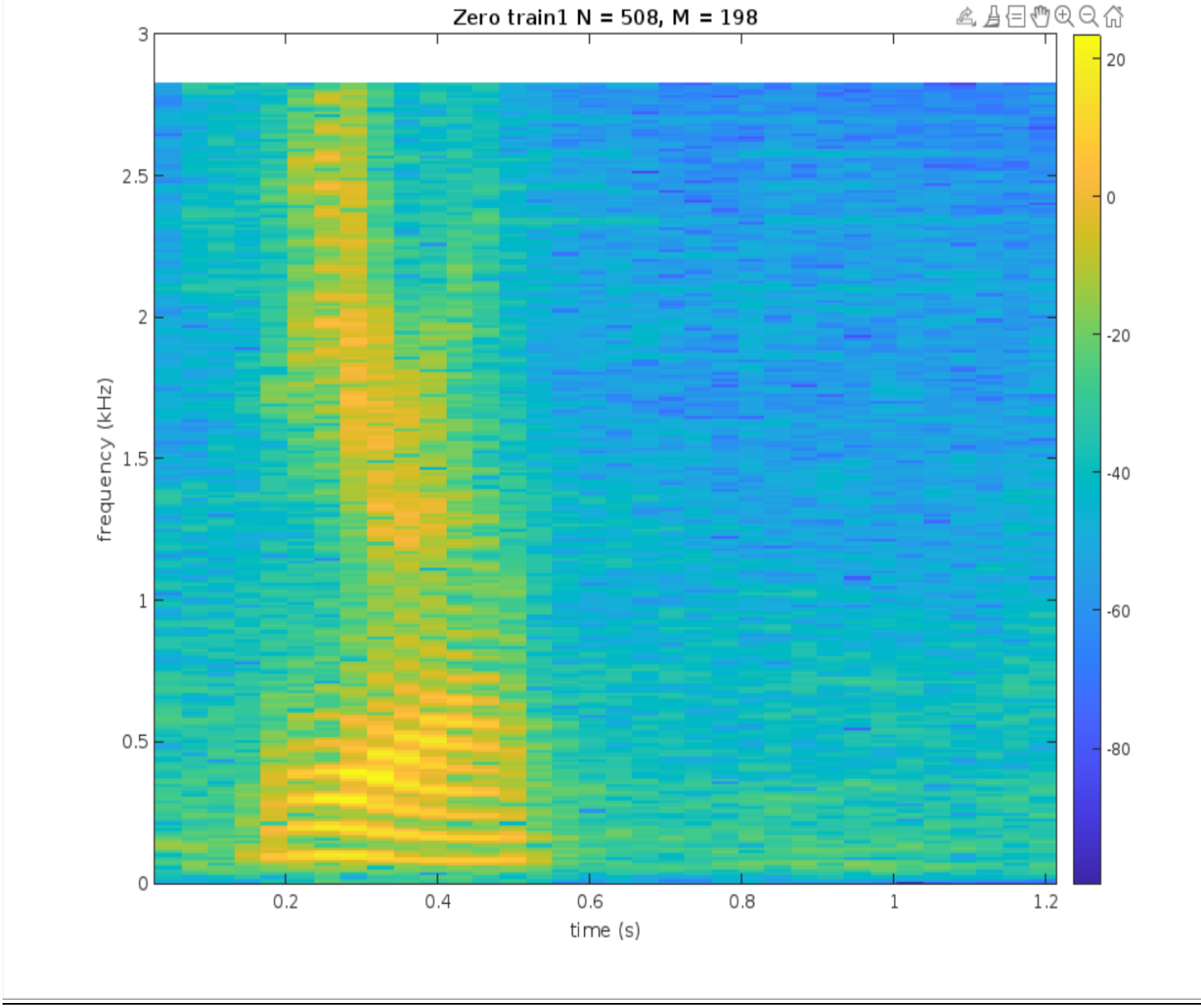
2) For test 2 we resampled the Zero_train1 wav file to get a sampling frequency of 5647 Hz. Upon doing this, a frame of 256 samples corresponds to about 45 ms.



N = 181, M = 171


Zero train1
frequency (kHz)
time (s)
3
2.5
2
1.5
1
0.5
0
0.2
0.4
0.6
0.8
1
1.2
0
-20
-40
-60
-80
-100

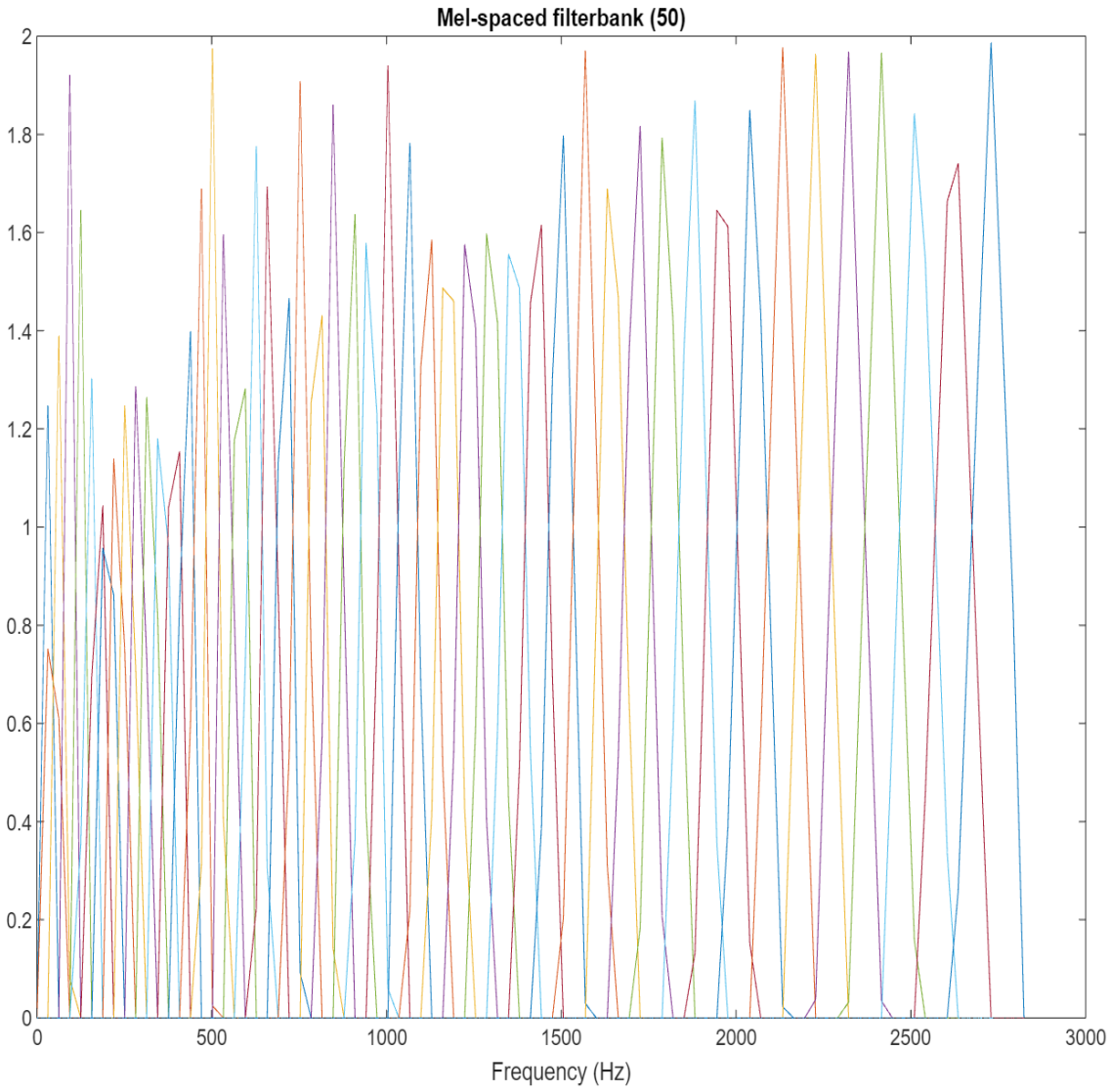

Zero train1 N = 282, M = 113
frequency (kHz)
time (s)
0
0.5
1
1.5
2
2.5
3
0.2
0.4
0.6
0.8
1
1.2
20
10
0
-10
-20
-30
-40
-50
-60
-70
-80

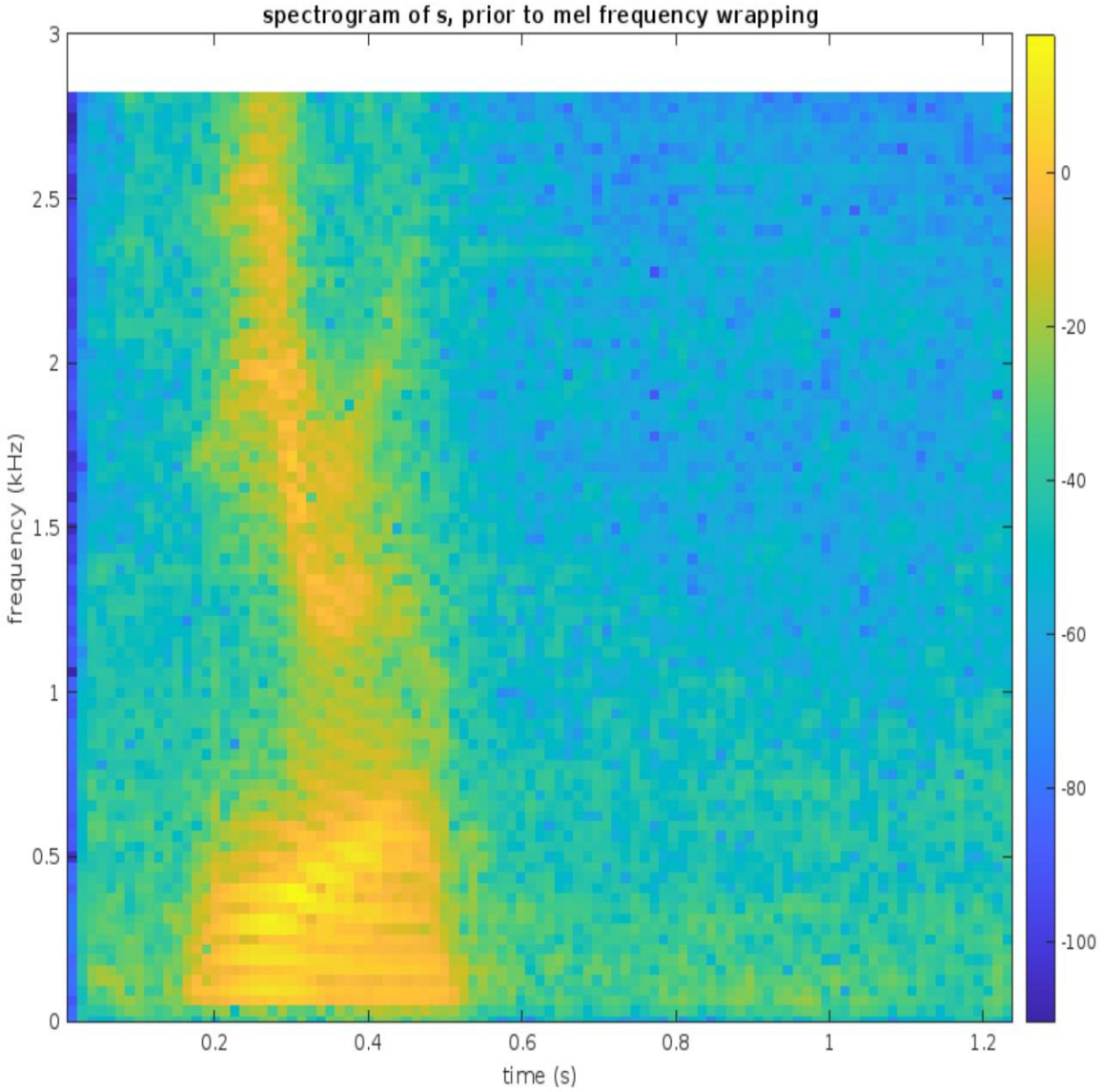


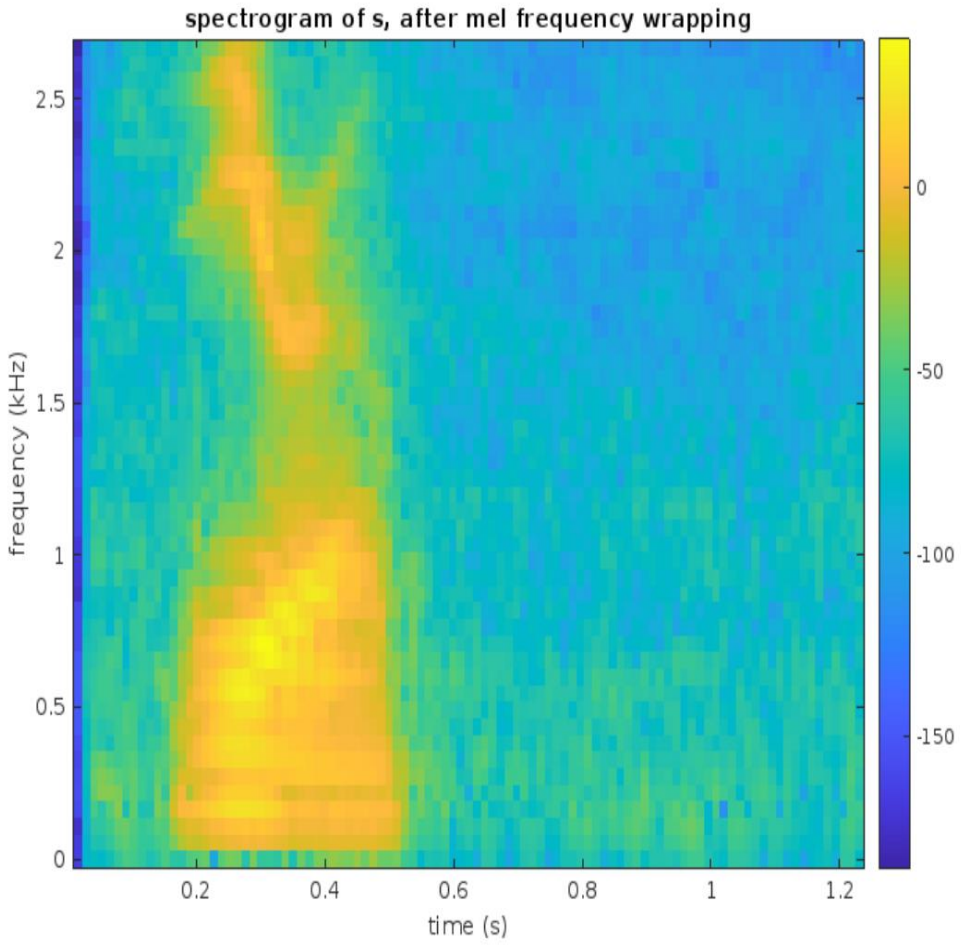
Energy Density:

In each spectrogram the greatest energy density seems to correspond to about .18 to .55 seconds, and about 0 to .8 KHz.

3)


Mel-spaced filterbank (50)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
3000
Frequency (Hz)

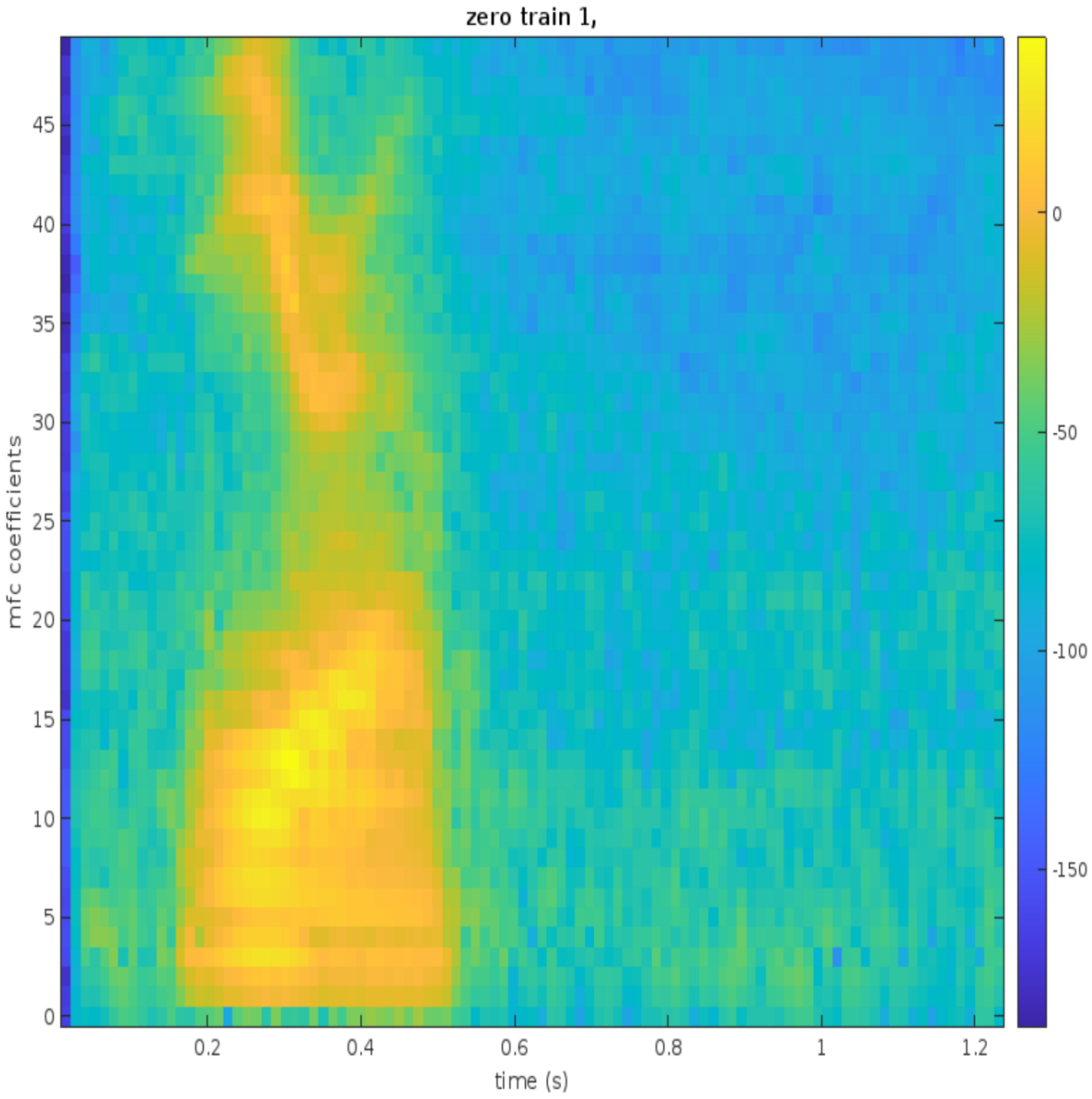

spectrogram of s, prior to mel frequency wrapping
frequency (kHz)
time (s)
0
0.5
1
1.5
2
2.5
3
0.2
0.4
0.6
0.8
1
1.2
0
-20
-40
-60
-80
-100



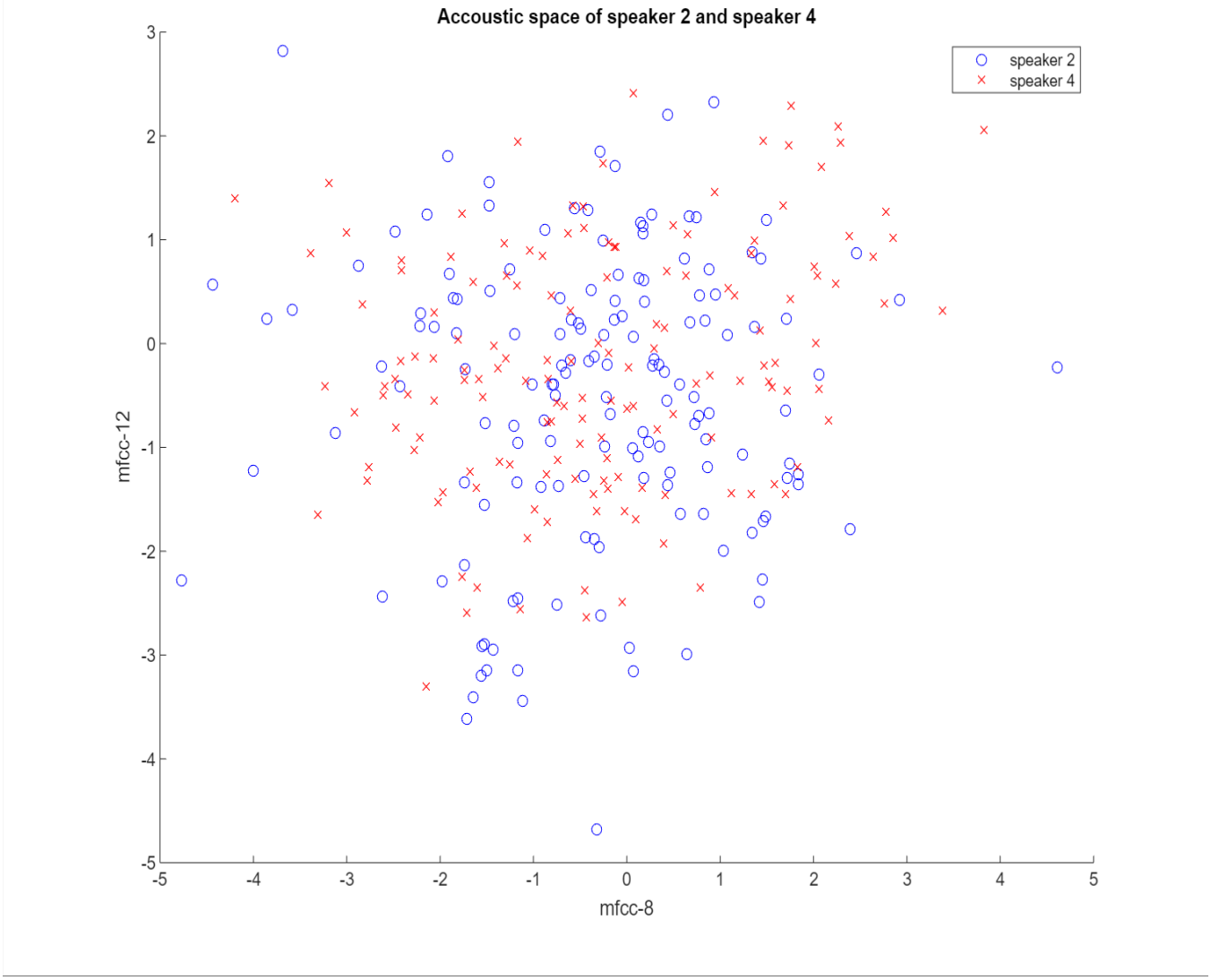
Effect:

The Mel-frequency wrapped spectrogram is smudged, giving the impression of a more even distribution of energy across frequency and time. Also worth noting, is the difference in power, observable by the color bar to the right of the plot. This change in appearance is due to the application of mel-spaced filter banks to the original audio, which is designed to scale linearly at 1000 Hz, and logarithmically above 1000 Hz; in accordance with human auditory perception of frequency.

4)

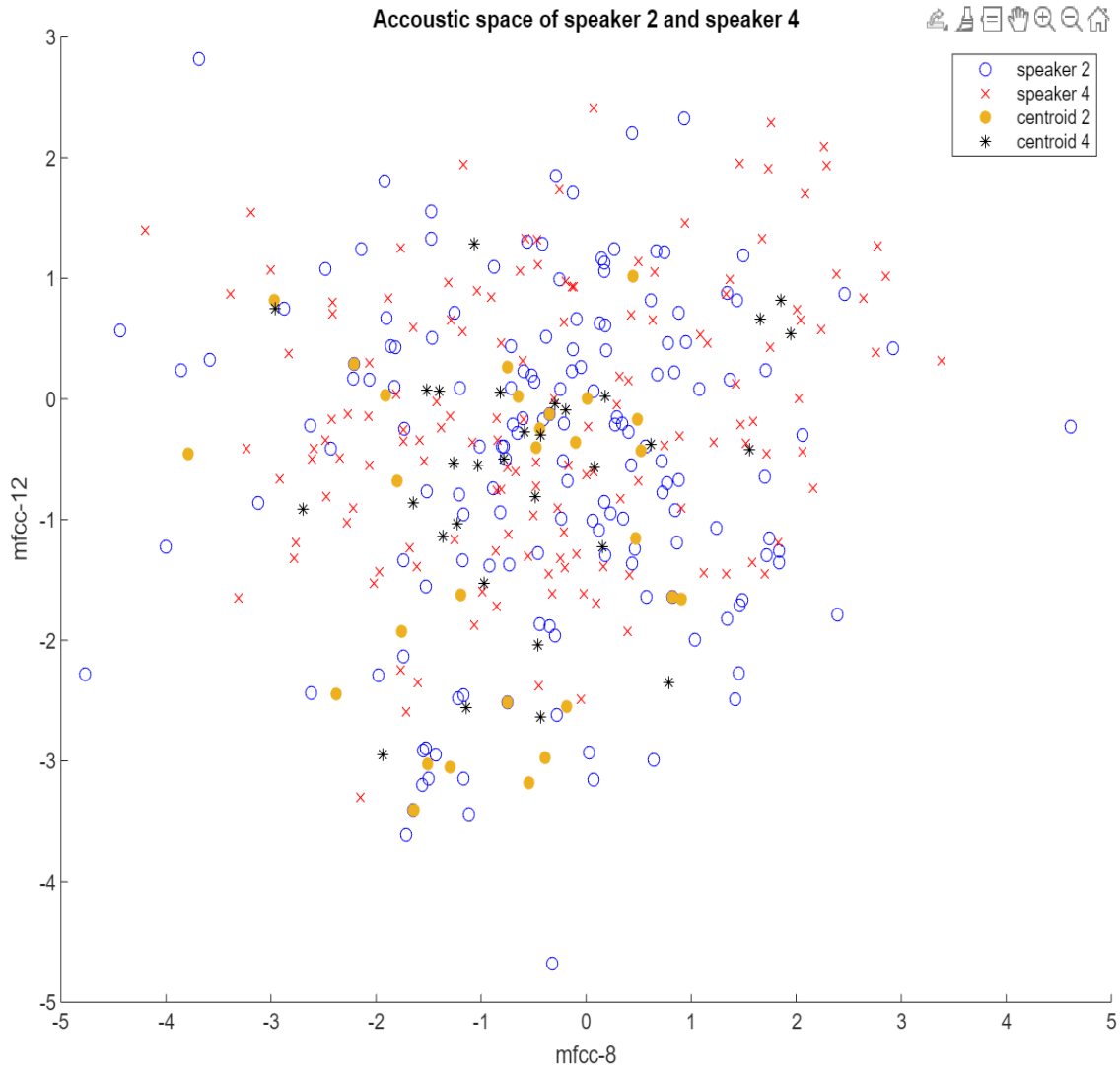

zero train 1,
mfc coefficients
time (s)
0
5
10
15
20
25
30
35
40
45
0.2
0.4
0.6
0.8
1
1.2
0
-50
-100
-150

5)



Though it might not appear to be the case at first glance, the vectors are clustered.

6)


Accoustic space of speaker 2 and speaker 4
speaker 2
speaker 4
centroid 2
centroid 4
mfcc-12
mfcc-8
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

7) The recognition rate achieved by our system for the training and testing samples provided was 100 percent. This is a whole 25 percent better than the human recognition rate.

Results of running provided testing and training samples through program

```
>> senior_design
Test 1 corresponds to train 1
Test 2 corresponds to train 2
Test 3 corresponds to train 3
Test 4 corresponds to train 4
Test 5 corresponds to train 5
Test 6 corresponds to train 6
Test 7 corresponds to train 7
Test 8 corresponds to train 8
>>
```

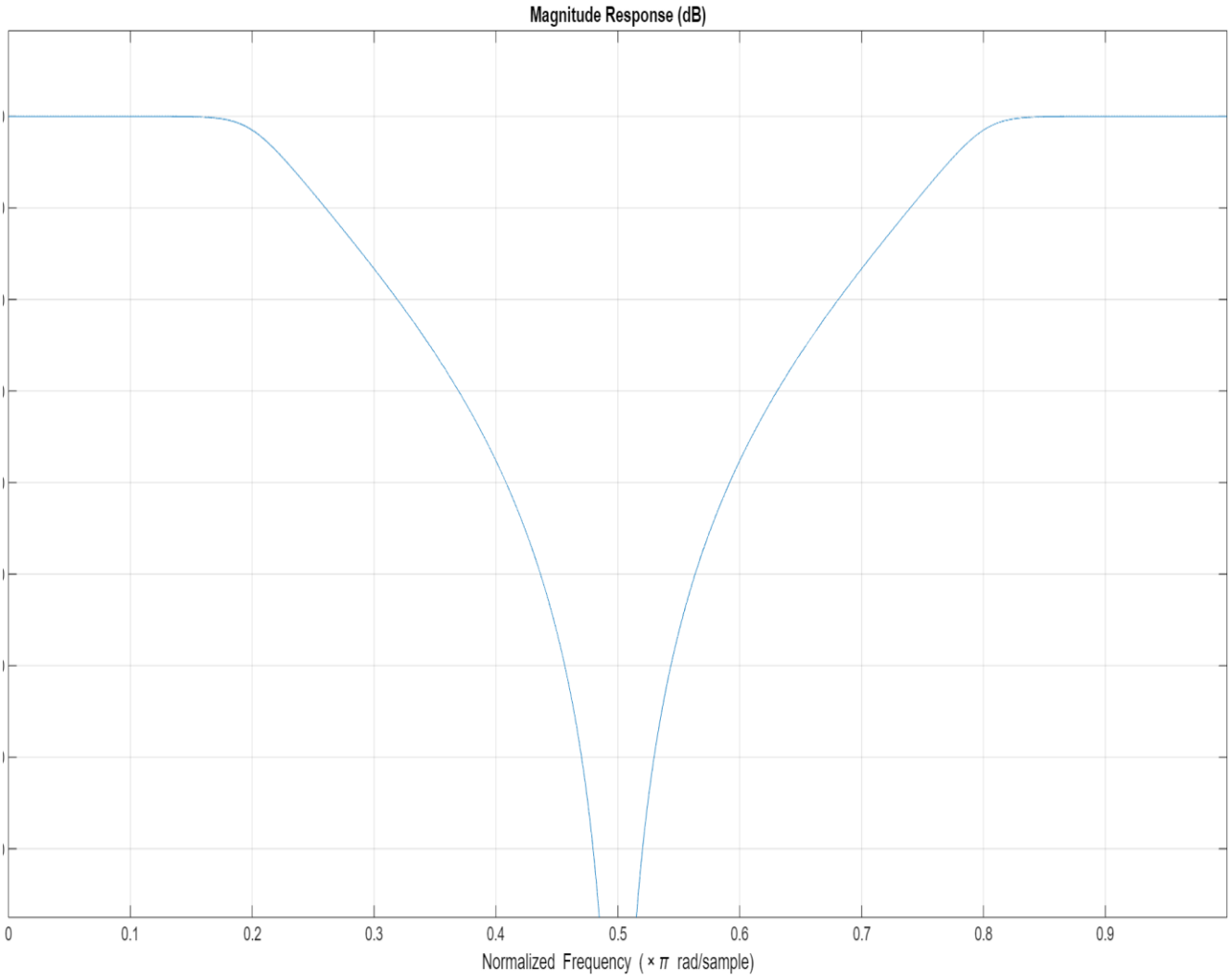
For the next part I tested the program using the audio files for the first 5 speakers for the Zero-training folder. Prior to modification, our program had a success rate of 80 percent. However, when we changed the hamming window to a hanning window, increased the centroid number to 32, and resampled the files at about 5600 Hz, we achieved a 100 percent success rate.

Results of running the first 5 speakers in the and Zero-training folder after changes

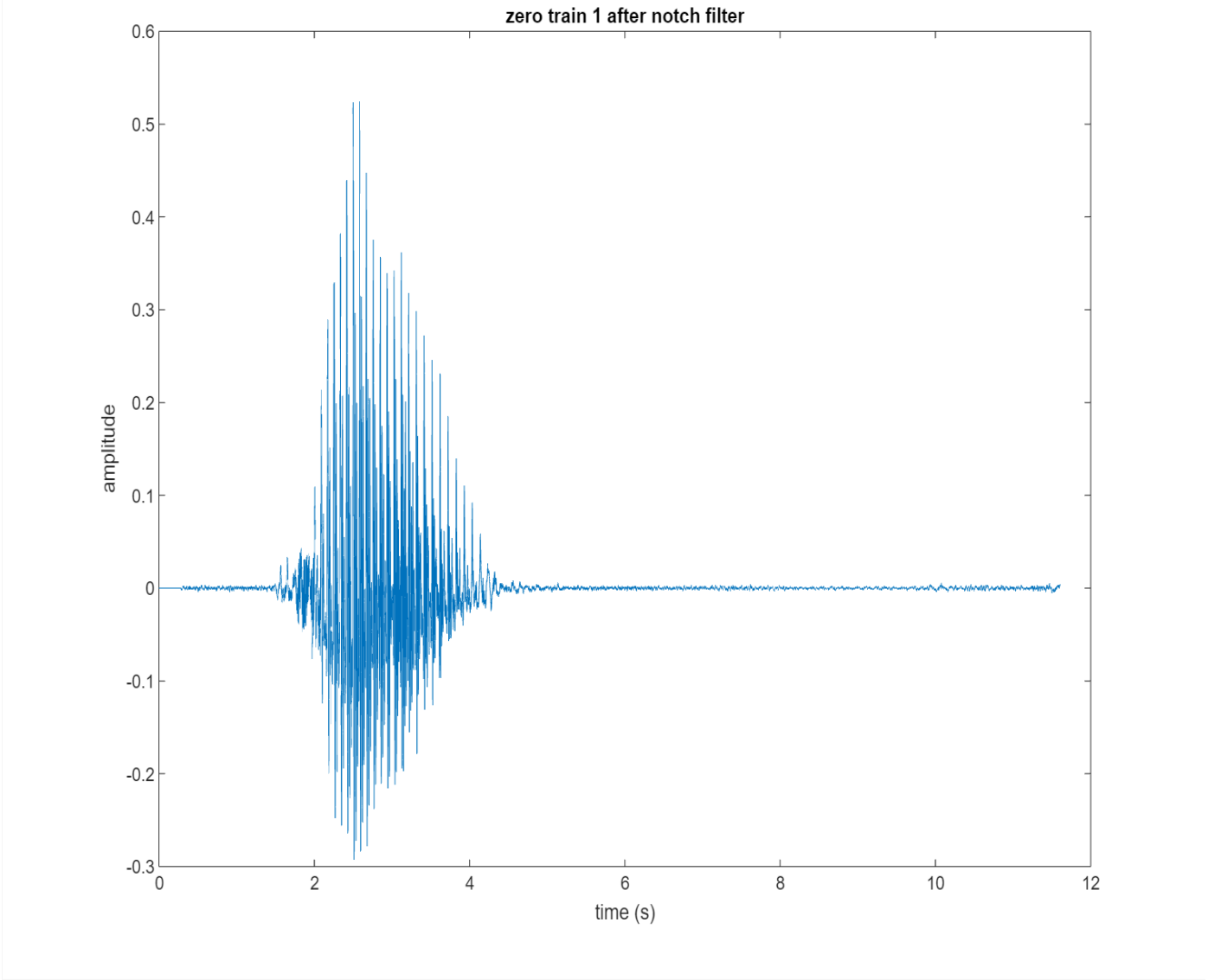
```
>> senior_design
Test 1 corresponds to train 1
Test 2 corresponds to train 2
Test 3 corresponds to train 3
Test 4 corresponds to train 4
Test 5 corresponds to train 5
>>
```

8)

Plot of notch filter used



Plot of audio file after notch filter application



Robustness:

Even after applying the notch filter, the accuracy of the system is still 100 percent when testing using the first 10 samples of the Zero-train, so it would appear that this is a very robust method of voice recognition.

```
>> senior_design
Test 1 corresponds to train 1
Test 2 corresponds to train 2
Test 3 corresponds to train 3
Test 4 corresponds to train 4
Test 5 corresponds to train 5
Test 6 corresponds to train 6
Test 7 corresponds to train 7
Test 8 corresponds to train 8
Test 9 corresponds to train 9
Test 10 corresponds to train 10
>>
```

9) The accuracy after augmenting the given training and testing zero files with an additional 10 zero training and testing files from speakers in class is still 100 percent.

Results:

```
>> senior_design
Test 1 corresponds to train 1
Test 2 corresponds to train 2
Test 3 corresponds to train 3
Test 4 corresponds to train 4
Test 5 corresponds to train 5
Test 6 corresponds to train 6
Test 7 corresponds to train 7
Test 8 corresponds to train 8
Test 9 corresponds to train 9
Test 10 corresponds to train 10
Test 11 corresponds to train 11
Test 12 corresponds to train 12
Test 13 corresponds to train 13
Test 14 corresponds to train 14
Test 15 corresponds to train 15
Test 16 corresponds to train 16
Test 17 corresponds to train 17
Test 18 corresponds to train 18
>>
```

10)

1) The accuracy of the twelve test and the accuracy of the zero test are both 100 percent.

Results for twelve and zero tests

```
>> senior_design
Test 1 corresponds to train 1
Test 2 corresponds to train 2
Test 3 corresponds to train 3
Test 4 corresponds to train 4
Test 5 corresponds to train 5
Test 6 corresponds to train 6
Test 7 corresponds to train 7
Test 8 corresponds to train 8
Test 9 corresponds to train 9
Test 10 corresponds to train 10
Test 11 corresponds to train 11
Test 12 corresponds to train 12
Test 13 corresponds to train 13
Test 14 corresponds to train 14
Test 15 corresponds to train 15
Test 16 corresponds to train 16
Test 17 corresponds to train 17
Test 18 corresponds to train 18
>>
```

2) For this part we created 1 training pool and 1 testing pool. The training pool is composed of files titled: z1 to z36. Z1 to z18 correspond to the twelve training data for speakers 1 through 18, while z19 to 36 correspond to zero training data for speakers 1-18. The testing pool is composed of files titled: tw1 to tw36. Tw1 to tw18 correspond to the twelve testing data for speakers 1 through 18, while tw19 to tw36 corresponds to zero testing data for speakers 1 through 18.

a) The system correctly identified the speaker 35/36 times

b) The system correctly identified the word 34/36 times

Results

```
>> senior_design
Test 1 corresponds to train 1
Test 2 corresponds to train 2
Test 3 corresponds to train 3
Test 4 corresponds to train 4
Test 5 corresponds to train 5
Test 6 corresponds to train 6
Test 7 corresponds to train 7
Test 8 corresponds to train 8
Test 9 corresponds to train 9
Test 10 corresponds to train 10
Test 11 corresponds to train 11
Test 12 corresponds to train 12
Test 13 corresponds to train 13
Test 14 corresponds to train 14
Test 15 corresponds to train 15
Test 16 corresponds to train 16
Test 17 corresponds to train 17
Test 18 corresponds to train 18
Test 19 corresponds to train 19
Test 20 corresponds to train 20
Test 21 corresponds to train 21
Test 22 corresponds to train 22
Test 23 corresponds to train 23
Test 24 corresponds to train 2
Test 25 corresponds to train 25
Test 26 corresponds to train 26
Test 27 corresponds to train 27
Test 28 corresponds to train 28
Test 29 corresponds to train 29
Test 30 corresponds to train 30
Test 31 corresponds to train 31
Test 32 corresponds to train 32
Test 33 corresponds to train 15
Test 34 corresponds to train 34
Test 35 corresponds to train 35
Test 36 corresponds to train 36
>>
```

Brief elaboration:

Using this test, so long as the test number and train number match, both the speaker and word are correctly identified. If you observe test 33, you can see that it corresponds to train 15. This means that a zero was confused for a twelve. However, the system correctly identified the speaker. The same cannot be said for test 24 though, in which a zero was confused for a twelve, and additionally, the speakers were confused as well.

Unique Efforts

When we first tested our system, the accuracy was 100 percent for the given training and testing samples that did not belong to our classmates. But, when we tested the data provided to us by classmates the accuracy dropped down to about 80 percent. We considered that it might be the values of N and M, so we changed those, but still the error persisted.

Next, we tried changing the hamming window to a hanning window in the stft, as we recalled that the professor said an adjustment in window type might increase accuracy due to different side lobe characteristics of certain windows. This did improve accuracy, but we still were not achieving 100 percent accuracy with all of the samples provided by the 18 speakers from class.

Following that change, we elected to specify a higher number of centroids and filters, for the mel-spaced filterbanks. This too increased accuracy, but there were still a couple misidentifications that persisted no matter what we changed in both the zero and twelve tests.

Finally, we tried resampling the data as a last-ditch effort to achieve 100 percent accuracy for all samples provided. This did the trick. We noticed that when we lowered the original sampling rate of the data provided (48000 Hz) we were able to eradicate some of the errors. After playing around with the data a bit, we found that the sweet spot to achieve 100 percent accuracy was a sampling frequency of about 5600 Hz.

When we changed the sampling frequency though, this caused sizing issues for the mel-frequency wrapping step, and resulted in Matlab throwing error codes. In order to circumvent this issue we also needed to redefine the N2 value for the melfb function, depending upon the sampling frequency. These quirks can be seen in the mffc function in the matlab code.