ECE 272 Final Project Presentation
Spoken 4-Function Calculator

Jordan Floyd

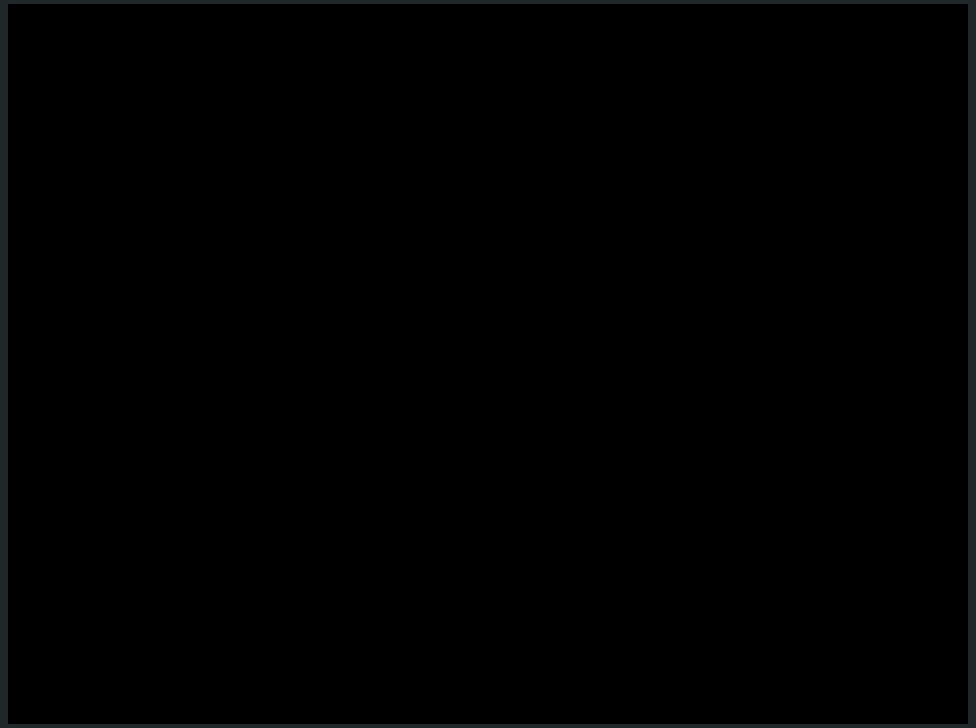
Motivation

- Voice assistants are becoming increasingly popular
- MFCCs are often used in speech recognition
- Classification ECOCs (Error-Correcting Output Codes) are useful for training models that involve simple sounds
    - How well do they work with more complex sounds?
    - Can they still be helpful in recognizing these sounds?

How to Use the Spoken 4-Function Calculator

- Think of a single-digit expression that uses one of the following functions: plus, minus, times, or divided by.
- Run “spoken_calculator_v2.m.”
- Wait until you see “RECORDING.” in the command prompt.
- Say all 3 parts of the expression out loud. Take your time, and say the words clearly for best results.
- Wait until you see “DONE RECORDING.” in the command prompt.
- Look at the command prompt to see the expression that the program thinks you said and the result of that expression.

Demonstration

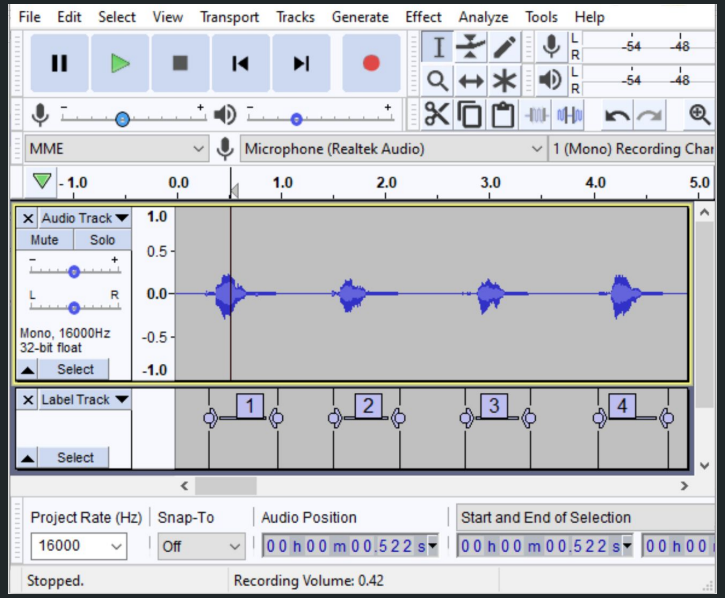


Overview of Implementation

- Downloaded free spoken-digit dataset from GitHub
- Created spoken-function dataset
- Organized these datasets into folders based on label and train/test sets
- Added relevant files from Homework 5 to this project's folder & edited them
- Trained 2 models (digits and functions) by generating MFCCs of each *.wav* file and by using the "fitcecoc" MATLAB function
- Tested both models individually using test sets and "predict" MATLAB function
- Wrote "spoken_calculator_v2.m", which takes in live input, divides the input into 3 segments (digit #1, function, and digit #2), and returns an answer

Importing/Creating and Organizing Datasets

- Imported free spoken-digit dataset from GitHub
  - https://github.com/Jakobovski/free-spoken-digit-dataset
- Created spoken-function dataset using my 4 family members' voices
  - Did not use my voice in the train or test sets
- Organized data into train/test folders and labels based on spoken content



Above: example of how I recorded and labeled snippets of me saying "plus" in Audacity

Model Training

- Used/edited the following functions from Homework 5:
  - my_mfcc.m
    - Unchanged
  - my_evaluation.m - For testing purposes
    - Made 2 versions from this: one for functions and one from digits
  - p3_test_live.m - For testing purposes
    - Made 2 versions from this: one for functions and one from digits
- Referenced p3_train.m and p3_test.m when making train and test scripts for functions and digits
  - Needed to resample spoken-digit .wav files from 8 kHz to 16 kHz so it would work with my_mfcc.m
  - Made all digit arrays 4500 samples long (0.281s) and all function arrays 6000 samples long (0.375s) - some trial and error to see what produced the best results
    - Zero-padding or cropping of each signal

Digits Model Testing: Test Set

%%%%%%%% confusion matrix %%%%%%%%

	0	1	2	3	4	5	6	7	8	9
0	47.88%	1.92%	0.58%	15.38%	5.00%	3.27%	6.54%	0.19%	16.35%	2.88%
1	10.58%	51.73%	13.85%	0.77%	0.00%	11.15%	1.73%	0.96%	0.58%	8.65%
2	1.73%	16.35%	30.38%	1.15%	0.19%	9.62%	4.04%	13.85%	2.12%	20.58%
3	7.50%	0.19%	1.35%	82.12%	2.69%	1.35%	2.50%	0.19%	1.92%	0.19%
4	0.96%	0.38%	0.96%	2.12%	73.08%	4.42%	12.12%	0.77%	4.81%	0.38%
5	7.31%	14.04%	5.19%	0.58%	0.77%	49.04%	7.50%	10.19%	0.77%	4.62%
6	1.92%	1.54%	2.88%	0.00%	7.12%	13.85%	63.27%	0.00%	7.88%	1.54%
7	1.92%	7.31%	4.81%	0.58%	2.69%	17.50%	2.69%	56.92%	2.12%	3.46%
8	9.23%	0.38%	2.12%	0.00%	3.46%	4.04%	4.62%	0.96%	69.62%	5.58%
9	3.46%	7.88%	20.96%	7.12%	0.38%	3.27%	2.31%	2.50%	1.92%	50.19%

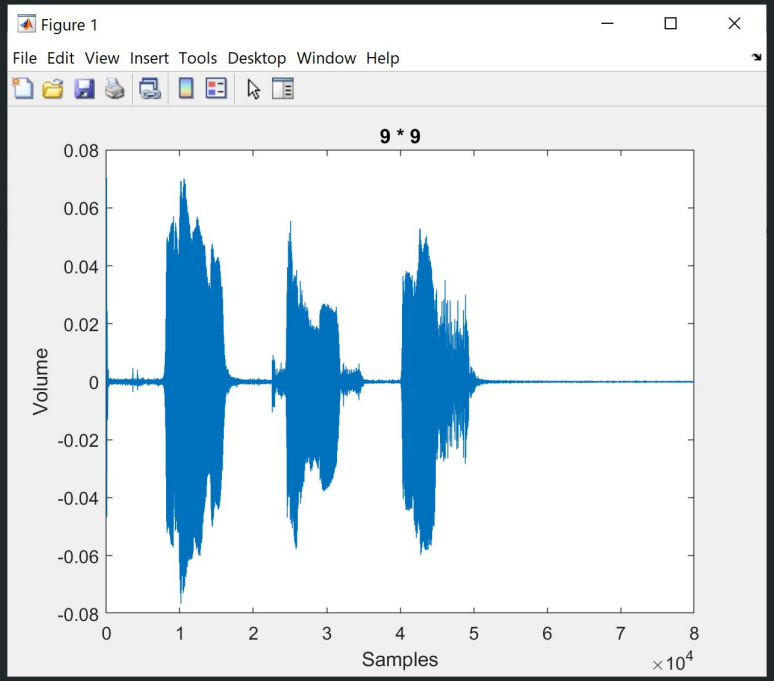
Functions Model Testing: Test Set

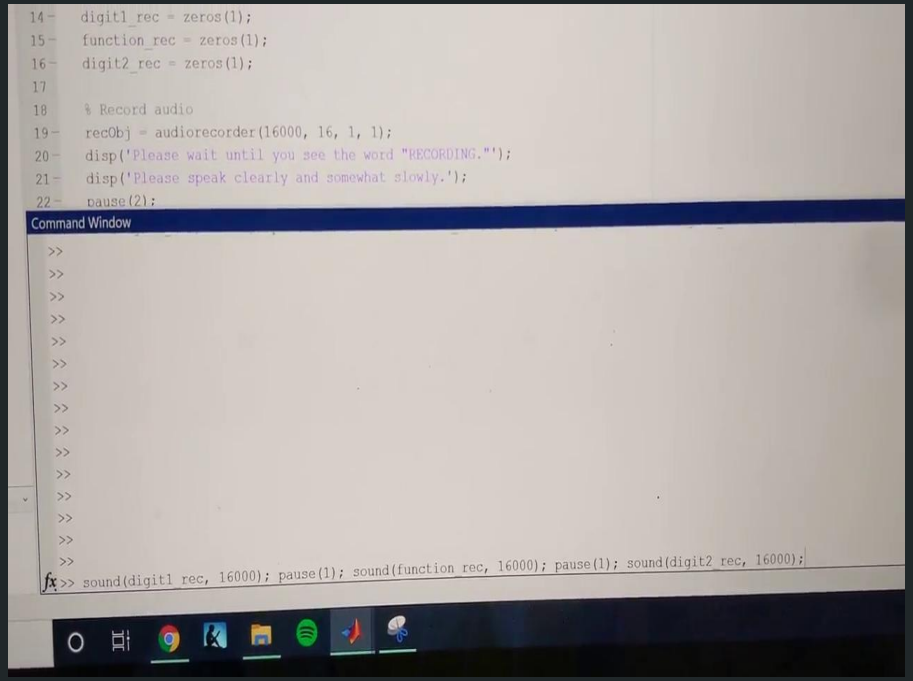
%%%%%%%% confusion matrix %%%%%%%%

	+	-	*	/
+	75.28%	7.08%	7.50%	10.14%
-	2.27%	77.40%	13.26%	7.07%
*	21.69%	16.27%	51.72%	10.32%
/	13.47%	18.33%	10.83%	57.36%

Putting It All Together: spoken_calculator_v2.m (1/2)

1. Record the input audio for 5 seconds.
2. **Separate the audio into 3 segments (digit #1, function, and digit #2) based on volume levels.**





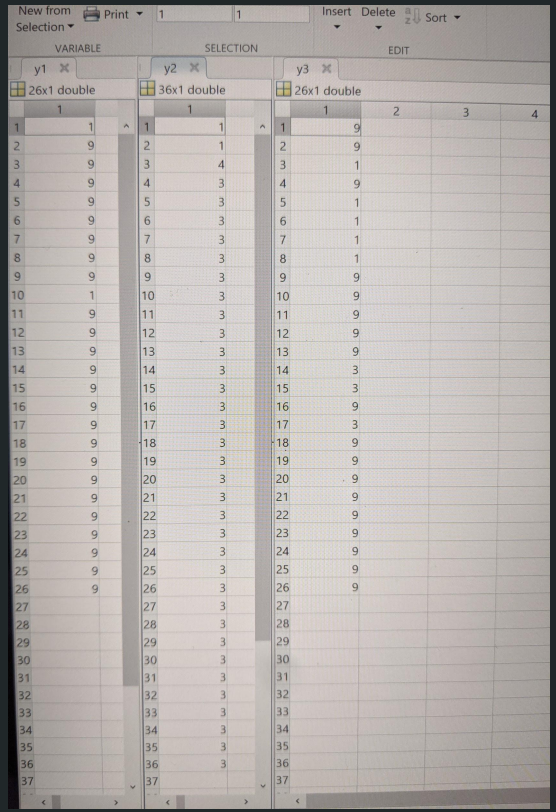
Putting It All Together: spoken_calculator_v2.m (2/2)

3. **Use the “predict” function to classify each segment as a number/function.**
4. Use a switch statement for the value of the function.
5. Apply that function to the two digits and display the full equation.

Photo on Right:

- y1 = labels for digit #1 predictions
- y2 = labels for function predictions
  - (1 = plus, 2 = minus, 3 = times, 4 = divided by)
- y3 = labels for digit #2 predictions

The final result is the mode of these predictions.



Reflection on Results / Room for Improvement

- Add to datasets to improve accuracy
  - More speakers and more speaker variety
- Revise approach
  - The “fitcecoc” function works best on simple sounds such as single phonemes
  - Training the models differently could lead to better results
- Expand capabilities of calculator (allow 2-digit numbers?)
- Implement a wake word so the calculator could be hands-free
- Text-to-speech: Have a voice read the answer

Q&A