ECE 407 - Final Project - Spring 2022

I. Introduction

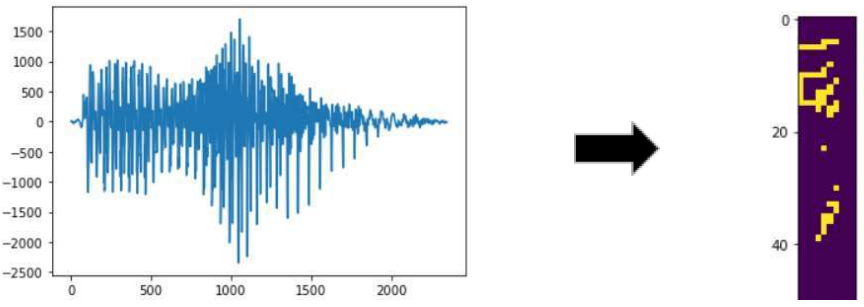
This project explores the Free Spoken Digit Dataset (FSDD). This is similar to the MNIST dataset, however, instead of image files, the data takes the form of audio sampled at 8 kHz. It is important to note that the dataset contains only 3,000 files total compared with MNIST's 60,000 training and 10,000 test images.

The objective of the project was to classify spoken digits using a generative model. The model was based on spectrograms that were generated from the raw audio data. A spectrogram is generated by taking a small slice of an audio signal and performing a fourier transform on it. This is repeated until the entire length of the signal has been covered. Spectrograms are used because it would be advantageous to see how a signal's frequency content varies with time. The team hypothesized that if a spoken digit is repeated, then its spectrograms would share similar characteristics.

II. Method Description

Setup:
Python was selected as the programming language and it was supported by some key libraries including openCV, Matplotlib, Numpy, and Scipy. These libraries greatly simplified data processing and model development. Jupyter Notebooks was the programming environment.

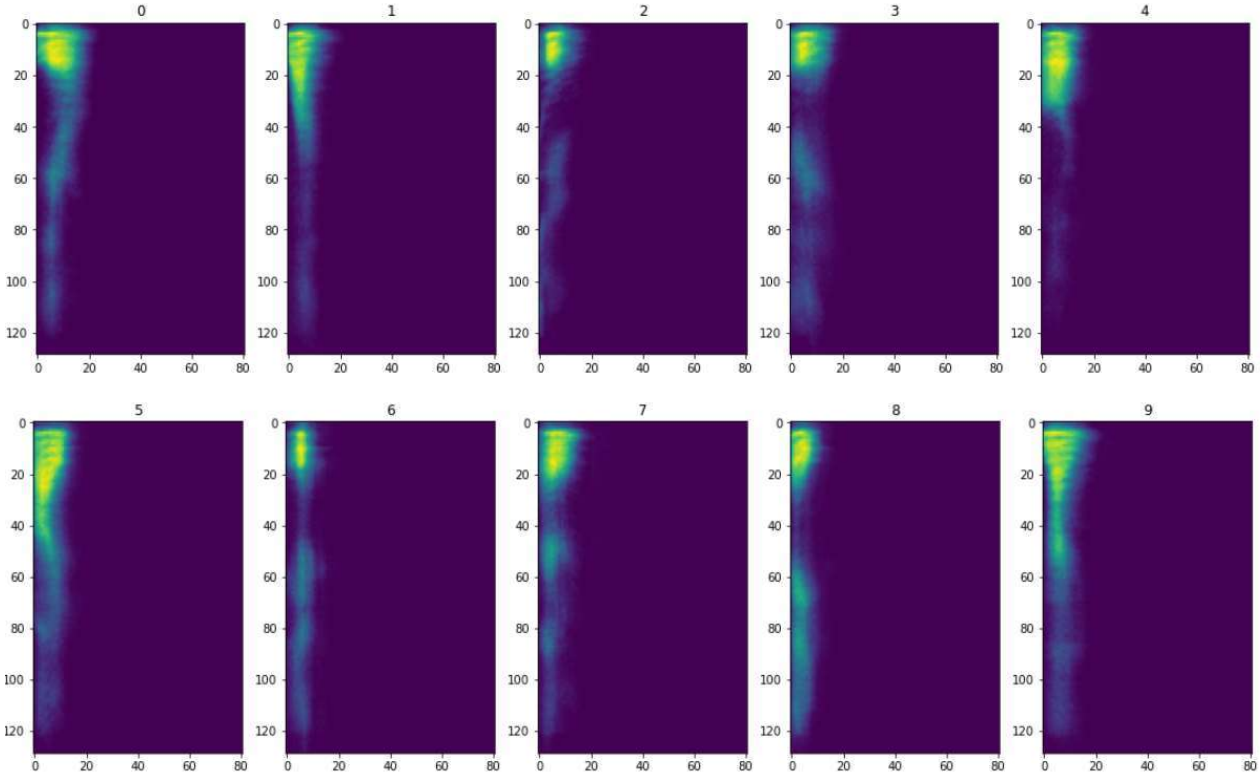
Data Preparation:
The first step of the project was to download the dataset and ensure that the data was accessible. After inspecting the data, the spectrograms were generated. This process began with finding the longest audio recording, creating a zero-array of its size, and then populating copies of the zero-array with samples from every audio signal in the dataset. This was done in order to ensure that every spectrogram would have the same size regardless of differences in audio signal lengths. Scipy was used to actually generate the spectrograms. In order to simplify the code structure necessary for the training of the model, each 2-D spectrogram was converted to a 1-D array. Then each 1-D array was binarized. Finally, the dataset was split into training and testing sets.



Model Generation:

Each class (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) was modeled using the Bernoulli distribution. This was accomplished by looping through each 1-D array in a certain class, performing the summation on the number of active pixels at each index, and dividing by the total number of 1-D arrays in the class. This yielded an array containing the probabilities of an active pixel in each position. Note that since each class had an equal amount of data, all class probabilities were equal. In order to view the results, the 1-D arrays representing each class were reshaped into the size of the original spectrograms. The results are shown in the figure below.

Spectrograms representing each class:



The figure shows that different spoken digits are represented by different patterns in the frequency content. Bayes' rule can now be used to classify spoken digits from the training and testing sets.

Signal Classification:

In order to classify an audio signal, a function takes in the signal's spectrogram and the array containing the spectrograms that represent each class. The classifier can be represented by the following:

$$\arg\max_j \quad \underbrace{\log \pi_j + \sum_{i=1} (x_i \log p_{ji} + (1 - x_i) \log(1 - p_{ji}))}$$

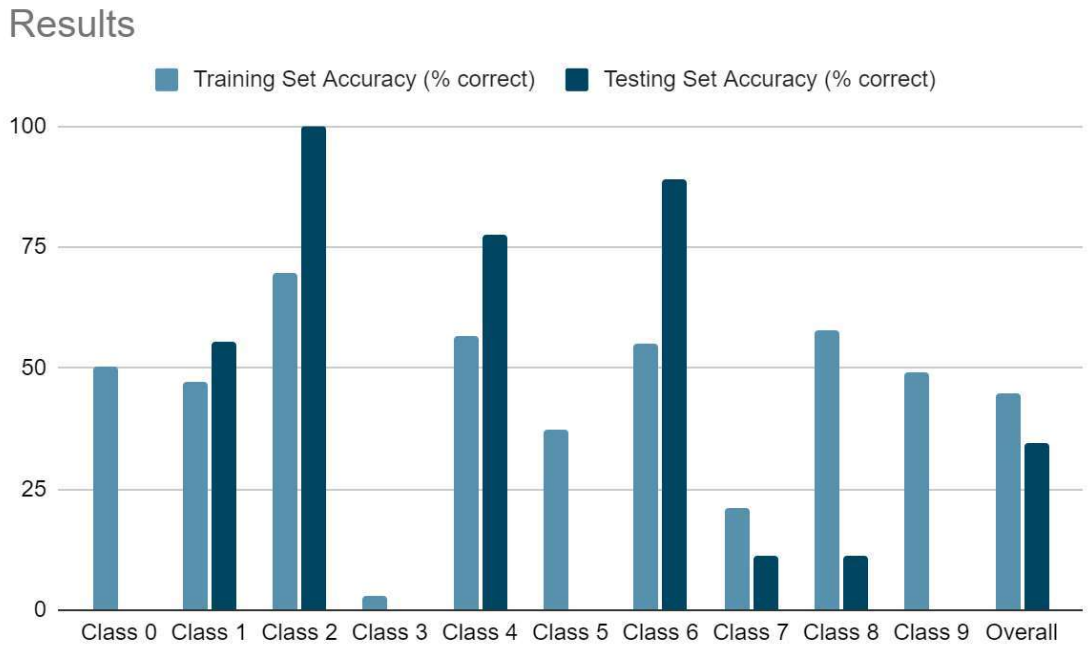
In the equation above π_j represents the prior probability of a certain class. X_i is either 1 or 0, this is dependent on the spectrogram of the signal being classified. Lastly, p_{ji} is the probability of having an active pixel at a certain location for some class. This calculation is repeated for all the generated probability distributions. Finally, the signal receives the label corresponding to the probability distribution that yielded the highest result (argmax). Note that log is used in order to prevent underflow during the calculations.

Computing Accuracy:

The code was structured so the model's ability to predict certain classes could be observed. A function is passed an array containing all the training/testing data of a certain class. Accuracy was computed by looping over the data, classifying, and checking if the predicted output matches the label of the class. If the class was correctly predicted, a variable tracking the total number of correct predictions was incremented by one. Finally, the actual accuracy value was calculated as the total number of correct predictions in the class divided by the total number of elements in the class. In addition to accuracy predictions for each class, overall accuracy for both the training and testing sets were computed. The results are shown below.

III. Experimental Results

Class	*Training Set Accuracy (% correct)*	*Testing Set Accuracy (% correct)*
0	50.173	0.000
1	47.059	55.556
2	69.550	100.000
3	2.768	0.000
4	56.747	77.778
5	37.370	0.000
6	55.017	88.889
7	21.107	11.111
8	57.785	11.111
9	49.135	0.000
Overall	44.671	34.444



IV. Conclusion

The experimental results showed that the model had some success in classifying the digits. Highlights include 69.55% accuracy in classifying the digit 2 for the training set and 100% accuracy for the testing set. However, overall accuracy was only 44.671% for the training set and 34.444% for the testing set. This can be viewed as low when compared to ~95% accuracy achieved by Convolutional Neural Networks in performing the same task.

A possible cause for the low accuracy can be attributed to how similar the spectrograms are to each other. The classification approach used in this project was shown to perform adequately when the data used for generating the class models was somewhat unique to each class (eg. MINST).

Overall, this project was a useful introduction to audio classification. Although the model did not show strong performance, much of the work done to prepare the data could be used for an alternative approach (eg. neural networks).