Machine Learning Final Report

Macey Cohn, Santiago Solis, Jhonny Velasquez

Topic/Task:

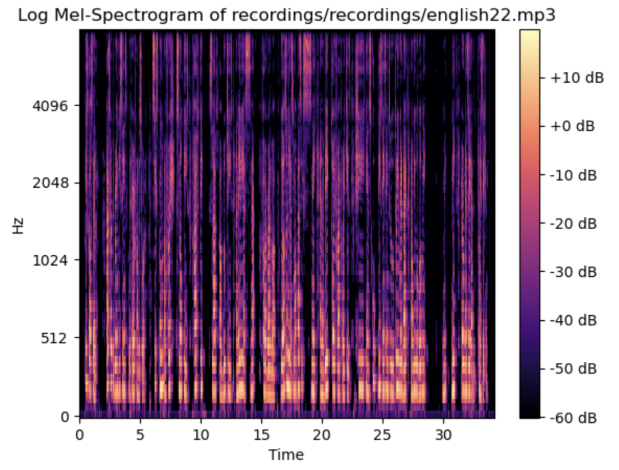
We decided to work on improving speech recognition systems. Speech recognition is notorious for having difficulty understanding accents from non-native English speakers. We hoped to aid these speech recognition systems by separating speakers based on their accents, allowing the system to better accommodate non-native English. accents. In order to differentiate between the two, our group aimed to build binary classifiers to determine if an individual is more likely to be a native English speaker using two different machine learning algorithms.

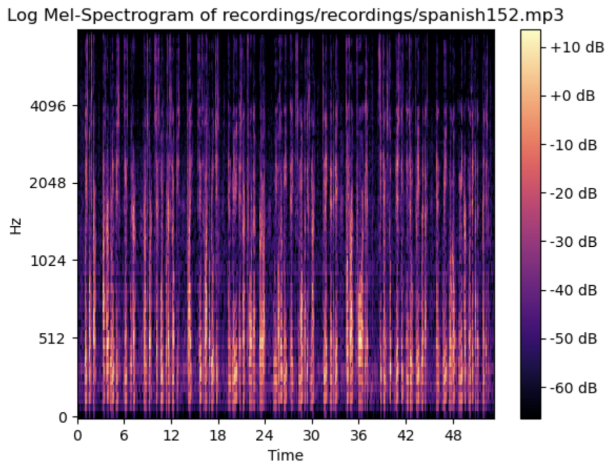
The dataset that we are using holds more than 2000 recordings of people throughout the world with various accents saying a few sentences in English. It provides the backgrounds of each individual in the dataset, such as age, sex, birthplace, native language, and country. The sentences are the same for each individual speaking them, which provides consistency within our dataset. If we were working without a consistent dataset, there would be variations within our dataset that would not be beneficial for the purposes of this project. Before implementing each model, data cleaning will occur by removing entries that are missing from the audio dataset. From there, we will be using the available audio datasets provided to us in addition to various datasets created through augmentation.

Background:

The MFCC coefficients come from the Mel Scale, which describes the transformation of an audio signal's frequency transformation. More specifically, since humans are more capable of identifying lower frequencies, the Mel Scale serves to relate the human-perceivable frequency of a sound to the actual frequency. By transforming the audio files using the Mel Scale, we are able to work with data that is more realistic to the human experience. The MFCCs can then be used as features upon which to train the models.

A spectrogram is a type of graph that displays the intensity of an audio file's frequencies throughout time. The methods of acquiring these spectrograms include performing Fast Fourier Transform (FFT) and Short-Time Fourier Transform (STFT). FFT works by analyzing the intensities of the frequencies throughout an entire audio file and outputting a spectrogram of the given frequencies, and STFT works by dividing an audio file into individual segments and performing an FFT on each segment. Once the spectrograms for each audio file are generated, they are normalized by using min-max normalization. The benefit of normalizing the data in this context is to remove any bias that spectrograms may possess and to improve the accuracy of the models since variations in signal amplitude will be minimized. This also ensures that the range of pixel values for the spectrograms is between 0 and 1 for each spectrogram. It should be noted that we chose to use a log-mel spectrogram instead of a traditional spectrogram. The difference is that the latter uses a Mel frequency scale and then takes the log of the power spectrogram, the end result being something better scaled for human auditory perception.





Native English Speaking Sample **Non-Native English Speaking Sample**

Data Collection, Transformation, Augmentation:

As mentioned in the Topic/Task section, the Speech Accent Archive contains varying-length audio clips of a sentence being spoken, by many different people of varying ethnicities, ages, and gender. We decided to work primarily with data in the frequency domain for this project, given the relatively large amounts of data that even a 30-second audio sample can have when sampled even at relatively low sampling rates for speech, usually around 20,000 Hz since this is around the Nyquist rate for the frequency range that human beings hear in. Spectrograms, and their many varieties such as Mel-Frequency Cepstral Coefficients, are a great way of reducing the overall amount of data of an audio sample while still preserving the temporal relationships of the energies of the frequencies at different portions of the audio sample.

In the original data set, there were exactly 2172 audio files, but after cleaning up the data we were left with 1487 audio files, 908 non-native English speakers, and 579 native English speakers. Since there is an imbalance between the number of native and non-native speakers, we decided to minimize the number of non-native speakers from 908 to 579. The criteria for cleaning up the data include removing audio files from countries that had less than 20 speakers and removing random audio files that pertain to countries with a high amount of speakers. This was done in order to have a balance between the number of native and non-native speakers in order to prevent a bias towards classifying audio files wrongly; the likeliness of a model misclassifying data increases if that model has more experience classifying one class compared to another.

After data cleansing, the data was separated into training and testing data. 100 sample from both groups was used for validating the model, and the rest were used for training the model. For each training audio sample, there were an additional 5 audio samples that were generated to augment the data. This adds more variance to the model which will make it more capable of classifying data. This was not done to the validation because it creates false results that the model would deem to be true. The 5 different ways in which the audio samples were created were by pitching up, pitching down, adding background noise, pitching up with background noise, and pitching down with background noise. Pitching up and pitching down were done by two semi-tones respectively, and background noise added was gaussian. Adding this variability of audio for testing aids in preventing overfitting of voice patterns and microphone characteristics since not everyone has the same voice pattern and different microphones were used most of the time. For the labels, the native English speakers were given a value of 1, and the non-native English speakers were given a value of 0.

Machine Learning Models:

Logistic regression was chosen as a baseline performance model for this project because it is a linear, simple, and easy-to-implement classifier. This provides the benefit of being able to fit a model with relatively little computational resources and time. Since one of the main limitations of logistic regression models is not being able to use high dimensional data, MFCC coefficients from each audio file were utilized for this model because they are very popular with speech recognition analysis due to their ability to compactly represent a speech signal while preserving information useful at the frequencies of human speech.

A convolutional neural network (CNN) was used as our second machine-learning model. We decided on this as a group because we wanted to gain a deeper understanding of how neural networks function. Additionally, the structure of a CNN is well-suited for picking up on differences between images, or spectrograms for this scenario. For this model, spectrograms were generated for each audio file by performing STFT on each audio file. In order to have consistent spectrograms for each audio file since spectrograms are dependent on the length of an audio file, each was truncated relative to the shortest spectrogram in both the training and validation sets. This was especially important for the CNN because this model can only train with spectrograms of the same dimensions.

As mentioned previously, deep neural networks have a lot more hyperparameters available to tune the performance of a model on. Specifically for our CNN architecture, we had 3 convolutional layers with a max-pooling layer after each, followed by 2 fully connected layers, all with a ReLU activation function, which returns the input if the value is positive and zero if it is negative. The input layer received a one-dimensional image of size (1, 128, 1418) for the truncated spectrograms. The size and number of filters for each convolutional layer, respectively, are 32 with a dimension of (3, 9), 64 with a dimension of (3, 7), and 128 with a dimension of (3, 5). The length of the first fully connected layer had 5120 neurons, with 50% dropout. The second hidden layer had 256 neurons, also with 50% dropout. Both of these layers also had a ReLU activation function. The final layer was a single neuron, for binary classification, with sigmoid activation. Due to the rectangular shape of the spectrograms generated for these audio samples, we made sure to use rectangular filters as opposed to the traditionally used square filters that are used for square images. By using rectangular filters that resemble the shape of a spectrogram, we can make sure that we are discarding information in the horizontal direction at a roughly faster pace than in the vertical direction so that important frequency information is not discarded. The descending filter sizes were used so that progressively smaller features could be learned during the training process. The loss function chosen for this task was the Adam optimizer as it was able to converge relatively quickly, within about 20 epochs on average. The SGD optimizer was not able to converge after experimenting with a range of learning rates and weight decays. All training was done using a batch size of 16.

The filter sizes were not a major focus of training the CNN since preliminary prototyping showed little improvements in loss and accuracy relative to changes in the type of optimizer, learning rate, weight decay, and learning rate schedule. Overall, it was found that a small learning rate was best for this problem. Experimental data showed that a starting learning rate of about *1e-4* helped the model converge the fastest. We hypothesize that a small learning rate for this problem helps the model learn the nuanced features and differences of speech from people with different accuracies, that larger learning rates could not learn. A major problem encountered during training was overfitting the training data after loss values on validation data had already converged. It was found that adding a value of *4e-4* to the weight_decay parameter in PyTorch's optimizer constructor, a form of L2 regularization, helped combat this the best. At a high level, this works by penalizing the model when the entire model is learning weights that have large magnitudes, a potential sign of overfitting on noisy patterns in data instead of general patterns. Additionally, to continue learning at later epochs in training, an exponential learning rate scheduler was used with a gamma of 0.78.

Performance:

For the logistic regression model, we tested different values for the hyperparameter, C. The purpose of this hyperparameter is to prevent any overfitting that is to occur in the model. It is inversely related to the strength: the lower the value of C, the stronger it will be against overfitting.

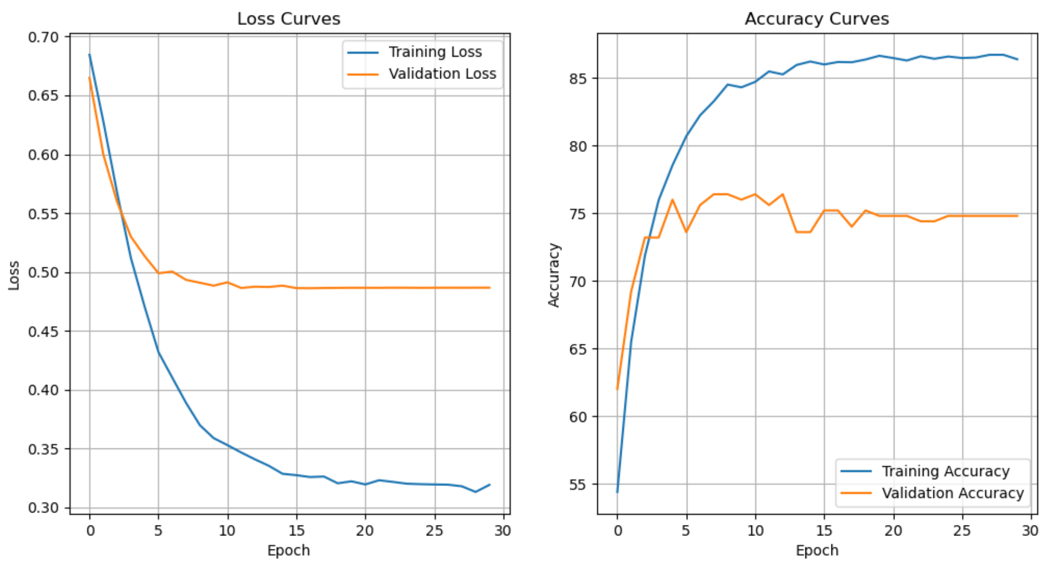
Validation Accuracies for Logistic Regression on MFCC Data

Fold #	**C = 0.1**	**C = 1.0**	**C = 10.0**	**C = 20.0**	**C = 50.0**
1	0.585	0.588	0.586	0.587	0.587
2	0.646	0.646	0.648	0.645	0.647
3	0.560	0.561	0.561	0.561	0.561
4	0.552	0.550	0.550	0.551	0.550
5	0.557	0.556	0.556	0.556	0.557

Overall Validation Accuracy: 60%
Time Elapsed: 14.05 seconds

For the CNN, loss curves and accuracy curves were generated during training.

Loss Curves and Accuracy Curves for CNN on Spectrogram Data (Training and Validation)



Overall Validation Accuracy: 74.8%
Time Elapsed: 4 minutes 30 seconds

Conclusion:

In conclusion, this report examined the effectiveness of a baseline logistic regression model and a higher parameter CNN model for the task of classifying speech accents from native English speakers to non-native English speakers. The baseline model showed an accuracy of 60% on the validation data that was described above, while the higher parameter model showed an accuracy of 74.8%. Although the higher parameter took a longer time, relative to the baseline model, the increased accuracy performance significantly outweighs the low accuracy from the baseline model and is thus preferable for speech accent recognition systems.

Member Contributions:

Santiago was the team member to originally came up with our concept. He focused mainly on the research behind the topic, taking time to understand the way we would be analyzing the audio data. He also took lead on the writing of the final paper. On the code side of things, he worked on some of the Python notebooks to help with performance modeling and bringing everything together.

Jhonny was the one who originally recommended using the MFCCs for analyzing the audio data. He did the neural network code and data augmentation. He also created the logistic regression file we ended up using.

Macey was in charge of organizing team meetings, keeping goals on track, and task assignment. She did a lot of the initial work that ended up being scrapped later. She worked on logistic regression code, Python notebooks, and cleanup in addition to compiling everything for the README.

Sources:

[1] "Learning from audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral coefficients," *Medium*, 14-Apr-2021. [Online]. Available: https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8. [Accessed: 21-Apr-2023].

[2] "Short-time fourier transform," *Short-time Fourier Transform > Frequency Domain Analysis > Math > General > Modules > Setup | Dewesoft X Manual EN*. [Online]. Available: https://manual.dewesoft.com/x/setupmodule/modules/general/math/freqdomainanalysis/stft. [Accessed: 21-Apr-2023].

[3] Weinberger, Steven. (2015). Speech Accent Archive. George Mason University. Retrieved from http://accent.gmu.edu