



THE GEORGE  
WASHINGTON  
UNIVERSITY

WASHINGTON, DC

Data Science Program

Capstone Report - Spring 2022

## Speech Engine

Harish Ram,  
Jiachen Sands,  
Sisi Zhang,  
Zequi Zhang

supervised by  
Amir Jafari

### Abstract

Our study on speech classification shows the results of several types of convolutional neural networks and transformer models with the use of pseudo labeling, augmentation and an autoencoder for pre-training to increase model performance. Since the human aspect of the data must remain the central focus of analysis, allowing users to classify their speech requires a certain degree of data manipulation as to not distort the quality of data. As a result, we designed a Speech Engine to classify emotions, race, accent, age, sex and Parkinson's disease with high accuracy.

# Contents

|     |                          |
|-----|--------------------------|
| 1   | Introduction             |
| 2   | Problem Statement        |
| 3   | Related Work             |
| 4   | Solution and Methodology |
| 5   | Results and Discussion   |
| 5.1 | Experimentation protocol |
| 5.2 | Data tables              |
| 5.3 | Graphs                   |
| 6   | Discussion               |
| 7   | Conclusion               |
| 8   | Bibliography             |

## Introduction

Speech Recognition focuses on interpreting the sounds of voices to extract speaker-related features in an effort to provide extra demographic information to the context of what is being said. Therefore, audio classification will be performed on the emotion, accent, race, age and sex of the speaker. The purpose of this study is to build a speech engine that can derive these features with high accuracy and use this to build a GUI (Graphical Application Interface) allowing the upload of any audio files containing speech for classification. With the diverse set of data that we have collected from multiple voice studies that contain participants from numerous countries, backgrounds and ethnicities, we can extract a higher number of features within each category such as more types of emotions and a wider range of age groups to make the speech engine more applicable to more types of voices. The primary obstacle that is presented in this project is preprocessing the data into a viable format for training machine learning models all the while maintaining the integrity of the data; therefore, it is necessary to use many preprocessing techniques such as , cleaning, wrangling and augmentation properly to maintain a balanced class size throughout the project. With a topic like speech classification, when manipulating the data to increase model performance, we must always make sure not to distort the audio too much so that it would be unrecognizable from the original audio. Keeping all these factors in mind, we have successfully generated enough data to effectively train convolutional neural networks and transformers to achieve higher success on our GUI.

We have designed a speech engine to let users input raw audio files and output classification results. To build a speech engine, the first step is to collect data which contains all the features we need. We decided to use raw audio files in .wav and .mp3 format then generate Mel Spectrograms to accomplish different classification tasks. We tuned a different number of evenly spaced frequencies ( $n\_mels$ ), the number of samples between each successive FFT window ( $hop\_length$ ) and the number of output frequency bins ( $n\_fft$ ). We, then, used the best combination of parameters to generate the Mel Spectrograms. For different categories, we used data augmentation to make sure the number of class labels were balanced. After preprocessing, we were able to get a reasonable baseline score from our CNN model, which we used as a benchmark to compare to transformers that we plan to pretrain with our data.

## Problem Statement

The two largest challenges for this project can be categorized into data manipulation and model development. Different datasets were used for different categories and processing can be a complex obstacle when dealing with the data. For Parkinson's Disease and accent data, the biggest issue would be that the .wav and .mp3 files would be too large to preprocess, so we would need to split the files into smaller segments and remove the silence (in case some of the audio segments are silent). Furthermore, the data is quite imbalanced for some categories, which is the primary setback for classification problems.

The model parameters for different categories can be different, so tuning them can take a lot of effort. Therefore, when testing the performance of CNN models on our data, we need to generate Mel Spectrograms to feed into the model which can pose an issue of time.

## **Related Work**

The team started with a CNN model from George Washington University's professor Amir Jafari as a basic model for CNN. Using the MNIST dataset, the author constructed a rudimentary CNN model. Our team has added extra convolution layers to 9 and 11 layers as beginning points to improve model accuracy. In addition, we used pre-trained models such as ResNet18, ResNet34, Vgg16, and Efficient b2 to evaluate model performance in each category. Our team built data augmentation code to handle data imbalance issues prior to the CNN implementation. We attempted a variety of augmentation approaches in this section of the code, such as adding white noise, time shift, and pitch scale to generate extra Mel Spectrograms. Finally, we mixed the Mel Spectrograms from the original audio file with the newly created Mel Spectrograms and used them in all CNN models.

## **Solution and Methodology**

### *Data Augmentation*

Data augmentation is a technique to generate new data by manipulating small parts of the original data points to add to the original dataset. Augmentation is best suited for unbalanced data. Speech data augmentation works in a very similar way. In this project, we have worked with several augmentation methods coming from the librosa library to modify the signals in the original audio files such as time shift, pitch shift, adding white noise, implementing random gain and time stretch. For different classification tasks in our project (emotions, age, accents, etc.), different augmentation methods yield the best performance. In all these augmentation techniques, there are corresponding parameters that can be changed as well. For example, in time shift, adjusting the factor will change how much the audio signal is shifted and in what direction. The figure above shows how the corresponding Mel Spectrograms look before (left) and after (right) time shift is applied. Other augmentation techniques listed work in similar ways by altering the values in the audio signal array through non-linear calculations.

### *Mel Spectrograms*

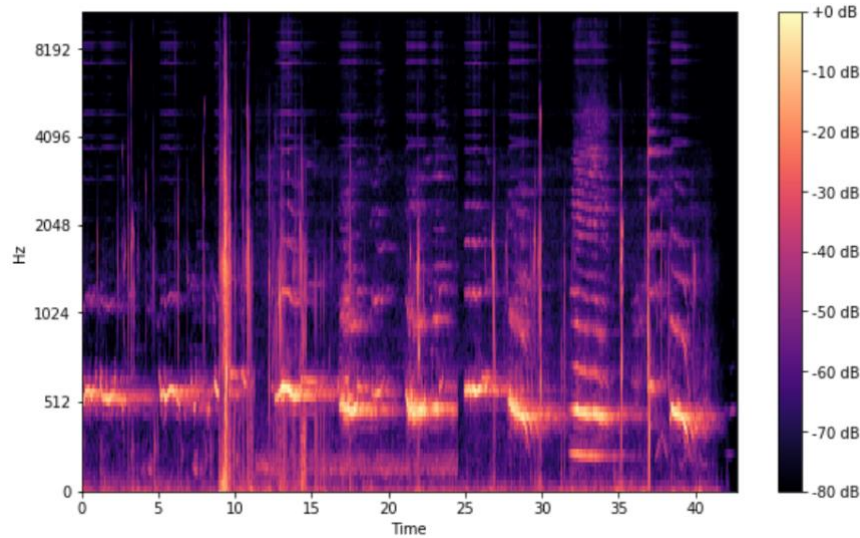


Figure 1

A spectrogram is a visual representation of an audio signal over time at various frequencies. The color in the spectrogram indicates the amplitude of the sound at that point in time - the louder the sound, the brighter the color as is signified in the legend on the right measured in decibels. In this project, we use Mel Spectrograms which are similar to normal spectrograms; however, the y-axis is calculated on the Mel Scale. For CNN, we use Mel Spectrograms for image classification. To generate the spectrograms, we input the audio signal which is in an array format and the sample rate of the audio files which we standardized to 16,000 hz as this value is used for most VoIP (Voice over Internet Protocol) communications along with several parameters such as hop length, number of evenly spaced frequencies and number of output frequency bins. As CNN is most efficiently used for image analysis, we used Mel Spectrograms as input for our benchmark value and will continue to use the audio signals for transformer use.

## CNN

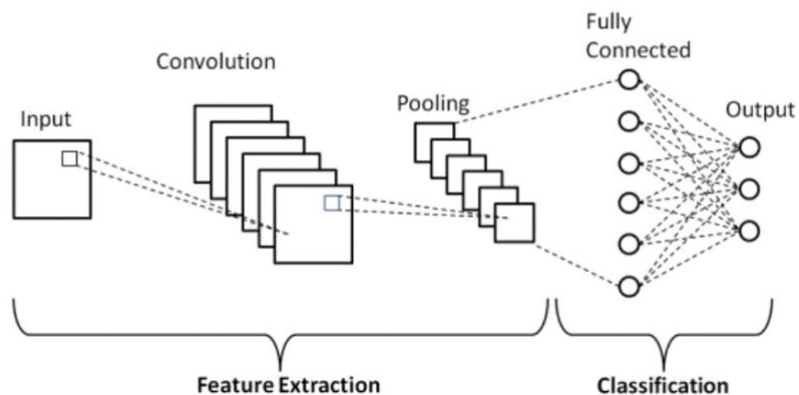


Figure 2

A Convolutional Neural Network consists of several types of layers that all perform a specific type of feature extraction from the input image, which in this project, is a Mel Spectrogram. CNN architecture works best for Mel Spectrograms as opposed to a basic MLP (Multi-layer Perceptron) because CNN can capture the spatial and temporal dependencies of the pixels in the image. Since Mel Spectrograms are RGB images meaning they contain multiple channels, the concept of pixels depending on adjacent pixels is important. Moving on to the architecture of CNN, the convolutional layer which comprises the kernel and stride. The kernel is a matrix of reduced size compared to the input image that traverses the entire image and performs matrix multiplication in the portions it occupies each stride to extract a representation of the image in a one channel convolutional feature output to pass along to the next convolutional layer. In terms of the actual size of the image (height and width), we use the same padding feature to maintain the same dimensionality throughout the network. To allow more layers in the CNN architecture and reduce computational power required by the model, pooling layers are necessary to take the maximum value of each portion of the image the kernel covers. Finally, a fully connected layer is a linear layer to make sure that the dimension of output is the same as the number of labels. The activation functions in the form of the activation function ReLU to perform gradient calculation as defined by  $f(x) = \max(0, x)$  to speed up training.

### *Autoencoder*

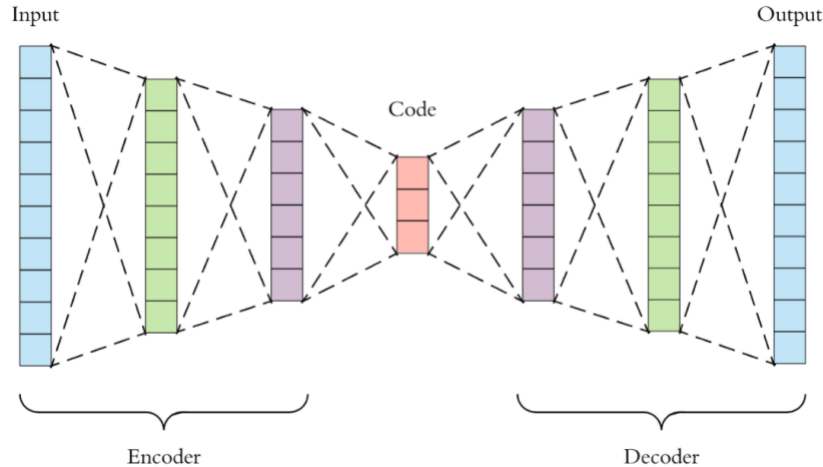


Figure 3

An autoencoder is a neural network that compresses the input (an image) into a downgraded representation and reconstructs the image as the output. In Figure 1 above, the encoder is where the original input, which in our case is a Mel Spectrogram, is compressed and produces an encoder representation of the image. The decoder then uses this representation to reconstruct the image as the same size and parameters as the input. However, this new image will not be exactly the same as the input. For this project, it is important that the colors of the Mel Spectrograms are carefully analyzed by the model as they are representations of the actual audio. The purpose of using an autoencoder, for this

project, is for pre-training our data in order to use the weights and biases as a starting point for a CNN model. Specifically, we pretrained the autoencoder with both the encoder and decoder; however, only saved the encoder model parameters. By doing this, we can load the model from the autoencoder before beginning training on CNN. This will increase the performance of the CNN model, making our benchmark score more accurate.

## **Pseudo-labeling**

Since data augmentation failed to improve the model accuracy further and we have not yet used an autoencoder for pre-training with a more complex architecture, we also tried pseudo-labeling using semi-supervised learning. The main idea behind pseudo-labeling is to improve model accuracy by training labeled and unlabeled data simultaneously. The benefit of pseudo-labeling is seen mainly when there is a lack of labeled data available entirely or for specific minority classes. Ideally, the unlabeled data should have a balanced distribution; however, this is impossible to tell as the data is unlabeled. Furthermore, the subject matter of the data should be similar to the labeled data as well. For example, if the goal is to improve model performance using Race data, then the unlabeled data should contain speakers with a diverse distribution of races.

The following four steps will explain this idea in more detail. First, we have trained multiple CNNs, pretrained models and saved the best model that has the highest accuracy for each category. Next, using the parameters from the best model, the labels that are predicted on the unlabeled data become the pseudo labels. Then, these pseudo labels are used to train the model on the same unlabeled data again after which the unlabeled loss is calculated. In every epoch, the model is updated by learning from backpropagation using the unlabeled loss of the pseudo labels.

However, within each epoch, the labeled data is also trained on the model every several batches of unlabeled data. By using this semi-supervised learning technique, the model uses both the labeled and unlabeled loss to perform backpropagation to allow the model to learn from both labeled and unlabeled data.

There are three important hyperparameters in this process that heavily influence the performance of training. Alpha is used to control the weight applied to the unlabeled loss. Essentially, this determines how much influence the unlabeled loss has in backpropagation. T1 refers to the number of steps used to train on labeled data from the previous supervised training; while, T2 is the number of steps until alpha reaches its maximum value. Before reaching T2 steps, the influence of unlabeled data is slowly increasing from 0. In other words, the unlabeled data consistently becomes more effective in the training process as the number of steps increases to T2.

Loss per Batch = Labeled Loss + *Weight* \* Unlabeled Loss

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1} \alpha_f & T_1 \leq t < T_2 \\ \alpha_f & T_2 \leq t \end{cases}$$

Mnist experiments

Our experiments

## Results and Discussion

*The results section details your metrics and experiments for the assessment of your solution. It then provides experimental validation for your approach with visual aids such as data tables and graphs. In particular, it allows you to compare your idea with other approaches you've tested, for example solutions you've mentioned in your related work section.*

The results that we have collected throughout this project reflect the level of complexity that data processing in the speech recognition field requires. In this section, we will go through the metrics that our experiments have yielded, explain the reasoning behind these scores and how they can be improved with future work.

### 5.1. Experimentation protocol

*It is of the utmost importance to describe how you came up with the measurements and results that supports your evaluation.*

The process of making better predictions in terms of speech classification begins with formatting the data in such a way to yield accurate results. To determine an acceptable benchmark performance, we decided to use a CNN model. We first experimented with using solely linear layers which did not prove very accurate as our input - Mel Spectrograms - contained 3 channels (RGB). A preliminary step was to tune the parameters of the Mel Spectrograms to produce the best images. Thus, we explored using convolutional, pooling, activation and fully connected layers as these have been noted to work more effectively for images with color and pixel complexity. The issue with this approach was how the number of layers to use before accuracy began diminishing. On account of there being several classification tasks that we are observing, we tested using 7, 9 and 11 convolutional layers with varying results for different tasks. After demonstrating the capability of our own CNN model, we used pretrained models - Resnet18, Resnet34, VGG16 and EfficientNet\_b2 to compare results. Similarly, different tasks produced different



results. Our next approach involved building an autoencoder model to pre-train our data first on an encoder and decoder, save the model weights and biases of the encoder and begin training CNN with said model weights and biases as the starting point. The objective of doing this was to use an autoencoder to extract features from spectrograms with reduced dimensions to allow for higher performance with a subsequent model.

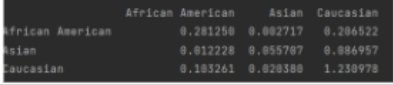
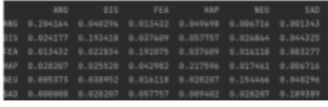

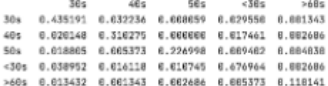
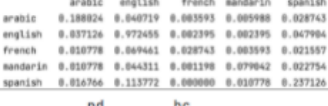
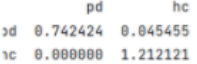
The measurements that we primarily observed were the F1 score and confusion matrix as our project revolves around accurate classification.

## 5.2 Data tables

*Every data table should be numbered, have a brief description as its title, and specify the units Used. As an example, Table 1 compares the average latencies of native application calls to networked services. The experiments were conducted on an Apple MacBook Air 2010 with a CPU speed of 1.4GHz and a bus speed of 800MHz. Each data point is a mean over 20 instances of each call, after discarding both the lowest and the highest measurement.*

| <b>Tasks with best parameters</b> | <b>n_fft</b> | <b>hop_length</b> | <b>n_mels</b> | <b>Accuracy</b> | <b>F1</b> |
|-----------------------------------|--------------|-------------------|---------------|-----------------|-----------|
| Race                              | 1024         | 512               | 160           | 78.40%          | 77.16%    |
| <b>Emotion</b>                    | 1536         | 512               | 160           | 57.56%          | 57.63%    |
| Sex                               | 1024         | 512               | 128           | 97.71%          | 97.82%    |
| Age                               | 1024         | 512               | 128           | 87.97%          | 88.00%    |
| Accent                            | 1024         | 512               | 128           | 62.8%           | 75.2%     |
| Disease                           | 1024         | 512               | 160           | 97.73%          | 98.16%    |

The table below demonstrates the accuracy and F1 score associated with the best set of parameters for Mel Spectrograms.

| Task    | F1 Score | Model           | Confusion Matrix   | Distribution  |
|---------|----------|-----------------|--|---|
| Race    | 77.17%   | CNN-9           |  | Caucasian - 4988<br>Asian - 562<br>African American - 1804                    |
| Emotion | 57.63%   | CNN-9           |   | HAP - 1271<br>SAD - 1271<br>DIS - 1271<br>NEU - 1087                          |
| Sex     | 97.82%   | Resnet18        |   | Female: 3512<br>Male: 3930  |
| Age     | 88.00%   | CNN-9           |   | <30s: 2774<br>30s: 1886<br>40s: 1306<br>50s: 984<br>>60s: 492                 |
| Accent  | 62.8%    | <u>Resnet34</u> |   | Arabic: 744<br>English: 2956<br>French: 374<br>Mandarin: 439<br>Spanish: 1053 |
| Disease | 98.16%   | <u>Resnet34</u> |   | hc: 399<br>pd: 258  |

The table below details the highest F1 scores with the corresponding model. From this data below, we can observe that different audio containing unique voice lines, types of voices and audio quality can expect better results from different models, regardless of the number of layers. The confusion matrix is also a major factor in understanding the results of the model in that it allows us to examine which class labels the model is learning the best or the worst. In the distribution column, the value counts of the classes for each classification task is noted and provides insight to how undersampled classes are associated with low prediction success in the confusion matrix. For example, the Asian class in the Race task is almost 10 times lower than the majority class which indicates a low value in the confusion matrix.

| Task   | Model Tested On | Pre-Augmentation F1 Score | Augmentation Method                                 | Post-Augmentation F1 Score |
|--------|-----------------|---------------------------|---|----------------------------|
| Race   | CNN-9           | 77.16%                    | Time Shift  | 79.70%                     |
| Age    | Restnet18       | 87.84%                    | Time Stretch, White Noise, Random Gain, Pitch Shift | 86.03%                     |
| Accent | <u>Resnet34</u> | 62.8%                     | Time Shift, White Noise                             | 62.0%                      |

### 5.3. Graphs - \*add loss graph of autoencoder?

For example, Figure 3 compares the efficiency of three different service architectures in eliminating adversarial behaviors. Every data point gives the probability that k faulty/malicious nodes managed to participate in a computation that involves 32 nodes. In the absence of at least one reliable node ( $k = 32$ ), the failure will go undetected; but the results show that this case is extremely unlikely, regardless of the architecture. The most significant result pertains to  $k = 16$ : the reliable nodes detect the failure, but cannot

reach a majority to recover. The graph shows that the CORPS 5% architecture is much more resilient than the DHT 30% architecture, by a magnitude of  $10^{11}$ .

## Discussion

*The discussion section focuses on the main challenges/issues you had to overcome during the project. Outline what your approach does better than the ones you mentioned in your related work, and explain why. Do the same with issues where other solutions outperform your own. Are there limitations to your approach? If so, what would you recommend towards removing/mitigating them? Given the experience you've gathered working on this project, are there other approaches that you feel are worth exploring?*

## Conclusion

*Give a clear, short, and informative summary of all your important results. Answer the initial question(s) or respond to what you wanted to do, as stated in your introduction. It can be a short table or a list, and possibly one or two short comments or explanations. Target a reader who may not have time to read the whole report yet, but needs the results or the conclusions immediately. This is a typical situation in real life. Some readers will read your introduction and skip to your conclusion first, and read the whole report only later (if at all). You may also draw perspectives. What's missing? In what directions could your work be extended?*

## Bibliography

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

<https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798>

<https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0>

[https://github.com/peimengsui/semi\\_supervised\\_mnist](https://github.com/peimengsui/semi_supervised_mnist)

<https://towardsdatascience.com/pseudo-labeling-to-deal-with-small-datasets-what-why-how-fd6f903213af>