Elec 378 Spring 2023
Final Project - Speech Emotion Classification

Olamide Adeshola, Yurie Han, Natalia Perey, Hamza ElMokhtar Shili

02 May 2023

1. Introduction

When humans communicate with each other, they use sounds to share information. One important form of sound is speech, which involves using words to convey information. However, speech is not just about the words we use - the way we say those words (the "acoustic properties" of speech) also carry important emotional meaning, which is transmitted from one person to another during a conversation. Speech carries a rich and complex set of data, and speech emotion classification is simply an application of pattern recognition. Therefore, through three different techniques of machine learning – Support Vector Machines (SVM), Convolution Neural Networks (CNN), and Gaussian Mixture Modelling (GMM) – our group attempted to apply these techniques to a dataset consisting of sound files across 8 emotions to classify each speech clip into the correct emotion of the speaker. The general steps we took consisted of feature extraction, feature selection, preprocessing, training, validation, and testing for each technique.

2. Feature Extraction

SVM/GMM

Some features of an audio clip include bit rate, sampling rate, duration, volume, etc. Bit rate describes the amount of data transferred into audio. Bit rate and audio quality go hand in hand. A higher bit rate typically means higher audio quality. Duration is the length of time it takes to play the audio clip. Volume is defined as the loudness of the audio signal. The sampling rate is defined as the number of samples taken every second to represent the signal. Sampling rate also directly influences the audio quality.

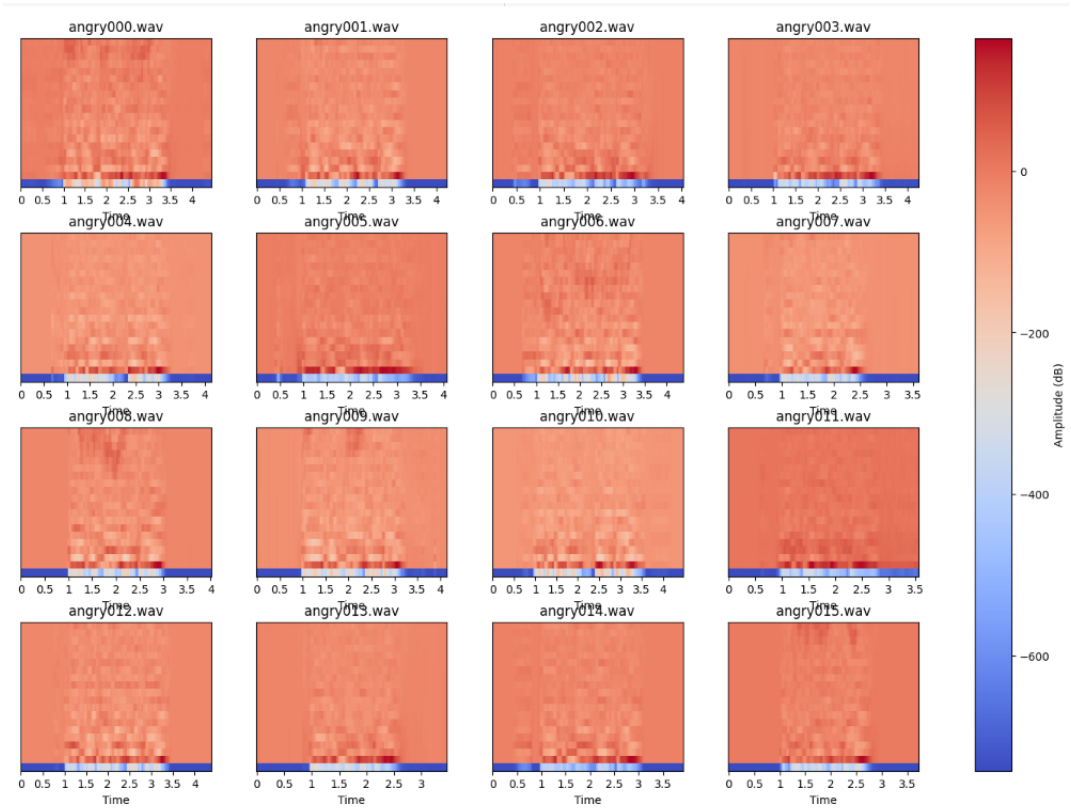


Figure 1: Heat Maps for the MFCCs

For our project, we believe that frequency range, volume, and dynamic range will be the most useful in

our experiments for emotional classification. Frequency range is defined as an indication of the range of frequencies in an audio signal which are generally between 20 Hz and 20,000 Hz. The dynamic range of a signal is defined as maximum signal-to-noise ratio. Better defined, it is the difference between the loudest and quietest volume present in the audio signal.

The features extracted for the SVM and GMM were the Mel-frequency cepstral coefficients (MFCCs) of the training data set through the Librosa Library. MFCCs are features used for audio recognition tasks. These coefficients provide extensive information about the variation of individual features in a speech signal. Specifically, MFCCs can measure loudness, timbre, pitch, spectral envelope, and other audio features. Timbre is defined as its quality which comes from the spectral content of the audio signal. Pitch is defined as the frequency of the sound in that moment. The spectral envelope is defined as the shape of the power spectrum of a sound. Because we chose 20 MFCCs, these will typically represent lower-level spectral features of the audio signal such as its shape, pitch, spectral envelope or high and low frequency balance.

CNN

For our implementation of CNN, we extracted the Mel-Spectrogram bands using the Librosa Library. A spectrogram is a visual representation of a signal's amplitude as it varies over different frequencies. A Mel-Spectrogram is when a spectrogram is displayed over the Mel scale for frequencies. The Mel scale distributes the frequencies so that equal distances in pitch sound equally distant to a person. In this case we generated 128 Mel bands and used those as our features in our training data.

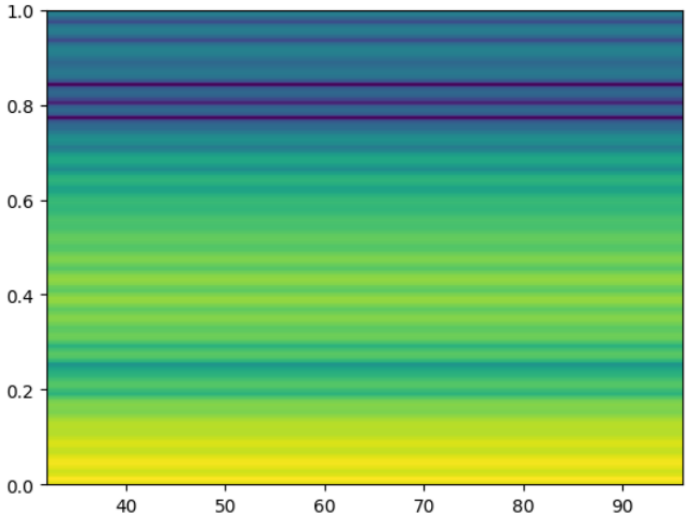


Figure 2: Mel-Spectrogram features that were extracted

3. Model Selection

GMM

Justification: We chose a Gaussian Mixture model because it is an unsupervised learning algorithm that extracts clustering relationships between data points that are not easily seen. Additionally, it is a probabilistic algorithm that grants us the distribution of data for each class.

Strengths:

- Ran the quickest of all the models

- Can be trained efficiently on large datasets
- Interpretability and its ability to capture complex data distributions

Weaknesses:

- Assumes that the data is of a Gaussian distribution, which may not be true for all cases
- Assumes that features are independent of each other

References: To read on GMM, we referred to the article, "Speech Emotion Recognition based on Gaussian Mixture Models and Deep Neural Networks" by Ivan Tashev and Zhong-Qiu Wang.

SVM

Justification: SVM is a supervised learning method that operates on high-dimensional vectors and is based on specific assumptions about emotions. It assumes that the presence or absence of a particular feature in a class is not dependent on the presence or absence of all other features. We chose an SVM model with MFCC implementation because MFCCs grant us an extensive amount of information about the variability of audio signals. MFCCs can obtain a substantial amount of spectral information from speech signals that relate to vocal tract characteristics and ways of speaking.

Strengths:

- Simple to program and execute
- Can handle high-dimensional data

Weaknesses:

- Computationally expensive, as the run-time took nearly two hours with the large dataset
- Limited as a binary classification model

References: The hyperparameter tuning code is based on BayesSearchCV from scikit-optimize, which can be found here.

CNN

Justification: We chose a CNN model because CNNs have different network layers, which are effective at learning distinct speech features. Generally, the lower layers learn the simple structures of the speech signal such as its frequency components, pitch, timing, and other features. In contrast, the higher layers learn the complex structures of the speech signal such as words, phrases, prosody, semantics, and other features.

Strengths:

- Automatically detects important features
- Ran relatively quick, as it only took 10 minutes even though CNNs are usually supposed to be computationally expensive
- Ability to learn complex patterns

Weaknesses:

- Requires large amount of computational resources and training data
- Prone to overfitting

References: We referred to Andrew Ng's Coursera to learn about CNN.

Eijaz Allibhai's article, "Building a Convolutional Neural Network (CNN) in Keras" helped us define the model using the Keras library.

3. Complete Pipeline

1. We identified features of an audio signal that are most important for accurate classification
2. We began by building our GMM model that implements MFCCs to capture important characteristics of speech signals amplifier
   (a) Split our data set into a validation and training set
   (b) Train a GMM model for each emotion class of the op amp
   (c) Classify the testing data using the trained GMM models
   (d) Save the predictions to a CSV file
3. Next, we built our SVM model that implements the same MFCCs as those we used in GMM
   (a) Attempt 1:
      i. Split our data set into a validation and training set
      ii. Initialized the SVM classifier with 'linear' set as the hyperparameter
      iii. Train the SVM classifier on the training data
      iv. Extract MFCCs from the test data and append to X_test matrix
      v. Predict labels of the test data (X_test) using the SVM classifier
      vi. Save the predictions to a CSV file
   (b) Attempt 2:
      i. Feature scaled the data set to specifically have zero mean and unit variance
      ii. Found the best hyperparameters using BayesSearchCV
      iii. Initialize the SVM classifier with the hyperparameters found in the previous step
      iv. Train the SVM classifiers on the training data using the newly found hyperparameters
      v. Extract MFCCs from the test data (X_test)
      vi. From the MFCCs, determine the mean and standard deviation of the features
      vii. Append the mean and standard deviation MFCCs to the test data (X_test)
      viii. Scale the test data to have zero mean and unit variance
      ix. Using newly trained SVM classifier predict values of the test data (X_test)
      x. Save the predictions to a CSV file
4. We built our final CNN model
   (a) Determined the number of Mel-Spectrograms that we wanted to generate
   (b) Iterate through all the .wav files to make them the same length
   (c) Add the mel spectrogram features to the data matrix (X_train)
   (d) Converted our data into numpy arrays while expanding the dimensions
   (e) Encode the labels using [LabelEncoder]
   (f) Encode the labels and split the data set into training and validation sets
   (g) Created the CNN model and compiled it with a 'Sparse Categorical Cross Entropy" Loss
   (h) Set up callbacks
   (i) Made predictions of the emotion label for each audio file
   (j) Convert predicted labels from integer values to emotion strings using label encoder from step (d)
   (k) Save the predictions to a CSV file

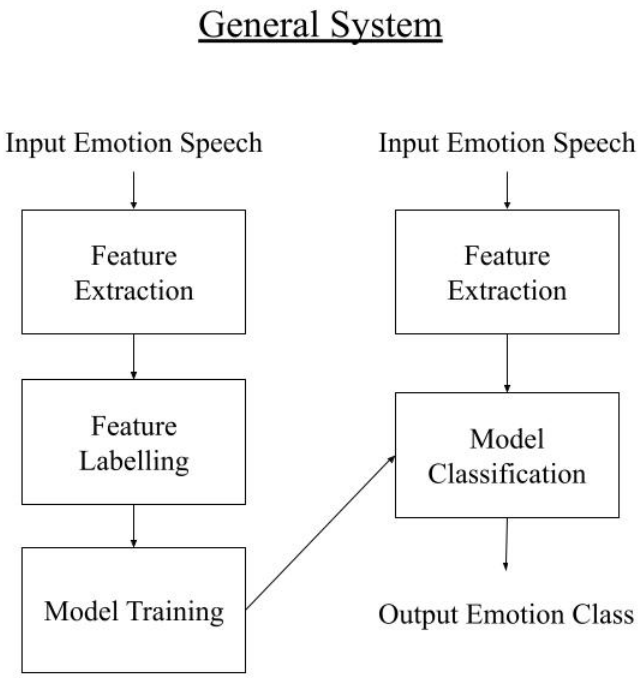


Figure 3: General system of classification for each model

4. Conclusion

Due to limited time and experience, the execution of each of these models were not highly accurate. For instance, determining the values for our hyperparameters were based on time-efficiency. Setting different values would have yielded different, and perhaps better, results. To enhance performance of the emotion classification process, extracting effective and accurate speech features can be enhanced.

However, our results were reasonably good for some models such as the SVM model. On our first attempt at using SVM we used 'linear' as our hyperparameter and did not try and optimize this feature. This meant that our results were not as good as we saw later on.

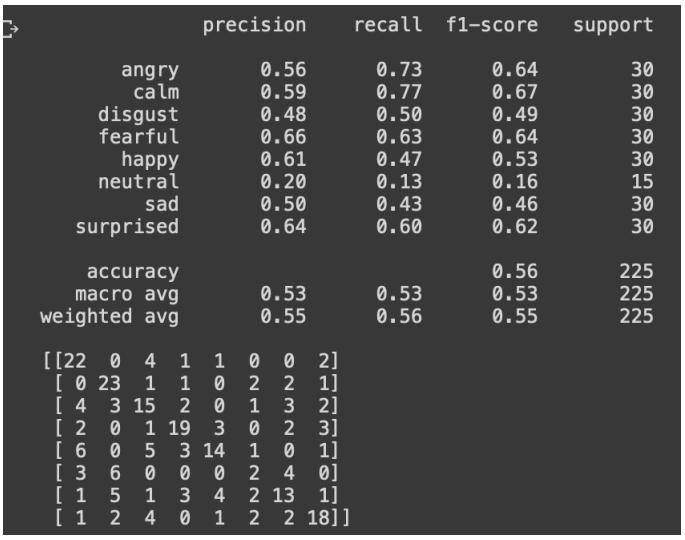
```
              precision    recall  f1-score   support

       angry       0.59      0.67      0.62        30
        calm       0.45      0.57      0.50        30
     disgust       0.38      0.50      0.43        30
     fearful       0.62      0.60      0.61        30
       happy       0.61      0.37      0.46        30
     neutral       0.00      0.00      0.00        15
         sad       0.50      0.43      0.46        30
   surprised       0.58      0.70      0.64        30

    accuracy                           0.51       225
   macro avg       0.47      0.48      0.47       225
weighted avg       0.50      0.51      0.50       225

[[20  0  5  0  2  0  0  3]
 [ 0 17  4  1  0  2  4  2]
 [ 4  4 15  0  2  0  1  4]
 [ 3  0  2 18  2  0  1  4]
 [ 6  2  5  5 11  0  0  1]
 [ 1  6  2  0  0  0  6  0]
 [ 0  6  3  4  1  2 13  1]
 [ 0  3  3  1  0  1  1 21]]
```

Figure 4: Confusion Matrix for our first attempt of the SVM model

In our next attempt we receieved higher accuracy in our tests. We attribute this primarily to two choices made early in our problem-solving process. The first is the choice to optimize hyperparameters. We created three SVM models, utilizing a different method for finding the best hyperparameters. This allowed us to find an ideal SVM that was not only computationally efficient, but also reasonably accurate in its predictions.

```
Best parameters: OrderedDict([('C', 19554.460528250976), ('coef0', 1.0), ('degree', 4), ('gamma', 0.037293466425399394), ('kernel', 'poly')])
Best score: 0.7208888888888889
```

Figure 5: Result of running BayesSearchCV and recieving the best hyperparameter



	precision	recall	f1-score	support
angry	0.56	0.73	0.64	30
calm	0.59	0.77	0.67	30
disgust	0.48	0.50	0.49	30
fearful	0.66	0.63	0.64	30
happy	0.61	0.47	0.53	30
neutral	0.20	0.13	0.16	15
sad	0.50	0.43	0.46	30
surprised	0.64	0.60	0.62	30
accuracy			0.56	225
macro avg	0.53	0.53	0.53	225
weighted avg	0.55	0.56	0.55	225

```
[[22  0  4  1  1  0  0  2]
 [ 0 23  1  1  0  2  2  1]
 [ 4  3 15  2  0  1  3  2]
 [ 2  0  1 19  3  0  2  3]
 [ 6  0  5  3 14  1  0  1]
 [ 3  6  0  0  0  2  4  0]
 [ 1  5  1  3  4  2 13  1]
 [ 1  2  4  0  1  2  2 18]]
```

Figure 6: Confusion Matrix for our second attempt at SVM

The second is our choice to use MFCCs. As stated before, MFCCs provide extensive information about the variation of individual features in speech signals. Specifically, MFCCs can measure loudness, timbre, pitch, spectral envelope, and other features of an audio signal. The use of MFCCs allowed us to reduce the dimensionality of the inputted data while also extracting elusive information about the speech signals

On the other hand, we struggled with both GMM and CNN models. Even though we used the same method of extracting MFCCs as we did in SVM, we ran into issues in how our extracted features functioned for our GMM model. When we were training GMM models for each emotion, we would get the training data that corresponded to that specific emotion. However, sometimes this would turn up empty, and so we were unable to run a model on for each emotion class. This made it so that our GMM models were incomplete which then caused most of our predictions to be inaccurate.

```
Accuracy: 0.13333333333333333
```

	precision	recall	f1-score	support
angry	0.13	1.00	0.24	30
calm	0.00	0.00	0.00	30
disgust	0.00	0.00	0.00	30
fearful	0.00	0.00	0.00	30
happy	0.00	0.00	0.00	30
neutral	0.00	0.00	0.00	15
sad	0.00	0.00	0.00	30
surprised	0.00	0.00	0.00	30
accuracy			0.13	225
macro avg	0.02	0.12	0.03	225
weighted avg	0.02	0.13	0.03	225

```
[[30  0  0  0  0  0  0  0]
 [30  0  0  0  0  0  0  0]
 [30  0  0  0  0  0  0  0]
 [30  0  0  0  0  0  0  0]
 [30  0  0  0  0  0  0  0]
 [15  0  0  0  0  0  0  0]
 [30  0  0  0  0  0  0  0]
 [30  0  0  0  0  0  0  0]]
```

Figure 7: Confusion Matrix for our GMM Model

When we applied CNN, the model was trained and stopped early after 10 epochs due to no further improvement in validation loss. The best validation accuracy achieved was 36.00% at epoch 5.

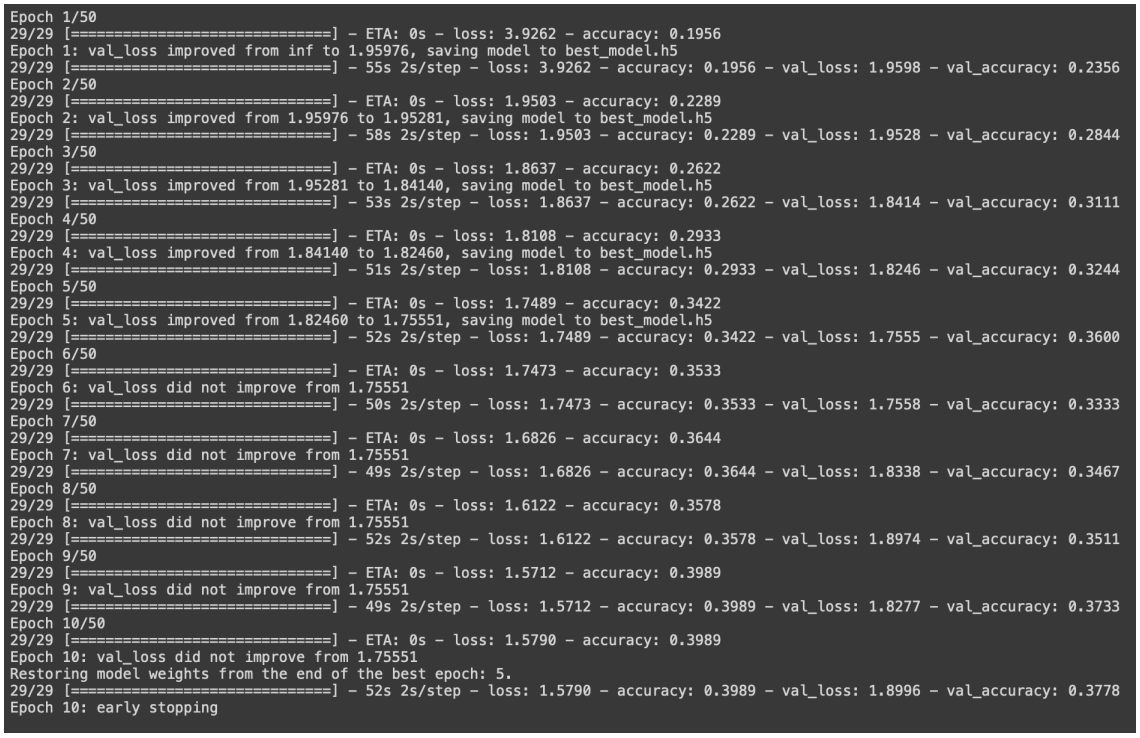


Figure 8: Output of CNN showing it stopping early

Since this is a relatively simple model, we may need to experiment with the architecture and training parameters to improve the model's performance in the future. An issue we dealt with early on was overfitting on our model. We suspected overfitting because the prediction accuracy was relatively low. We believe this is because we did not take further steps to prevent our model from overfitting such as dropout, early stopping, or regularization strategies. Eventually, we did employ early stopping, but as can be seen, it did not drastically improve our prediction accuracy.

To improve performance, possible steps include increasing the complexity of the model by adding more layers or increasing the number of filters in the convolutional layers, using data augmentation techniques to increase the diversity of the training data (which may help the model generalize better), tuning hyperparameters like learning rate, batch size, and the number of epochs, and using a more advanced model architecture, like a pre-trained model or an architecture designed specifically for audio classification tasks.

Overall, implementing these different machine learning models allowed us to make informed decisions when choosing a model to solve a problem. We saw in practice how some models are better suited for specific tasks, such as learning complex features and being computationally efficient, and it is up to the user to determine which benefits to consider more and deem as more important. In addition, we learned methods to combat overfitting and high-dimensional inputs. For overfitting, we learned about methods to combat it such as dropout, early stopping, and regularization. For dealing with input data with high dimensionality, we learned that using MFCCs and models that can easily deal with high-dimensional data such as an SVM makes a substantial difference in relation to computational complexity. For future works, we would like to attain higher accuracy through the combination of more features and improved implementation. To do so we could use cloud computing services like AWS or Azure to be able to run programs that are computationally expensive and can't run on Google Colab (take more than 4 hours to terminate).

A link to the code can be found here

5. References

Librosa (library for audio and music analysis):
McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., Nieto, O. (2015). librosa: Audio and Music Signal Analysis in Python. In Proceedings of the 14th Python in Science Conference (pp. 18-25). Librosa documentation can be found here

TensorFlow (machine learning framework):
Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16) (pp. 265-283). TensorFlow documentation can be found here

LabelEncoder (scikit-learn): Scikit-learn documentation can be found here

train_test_split (scikit-learn):
Scikit-learn documetation: Scikit-learn train_test_split documentation can be found here
ModelCheckpoint and EarlyStopping callbacks (Keras):
Keras documentation for ModelCheckpoint can be found here
Keras documentation for EarlyStopping can be found here