Dialect Classification in African American Speakers

Keith Chen[1]*, Timothy Do*[1]*, James Shiffer*[1]*, Thomas Sison*[1]

[1]University of California, Los Angeles, USA

keithachen@ucla.edu,timothydo@ucla.edu,jshiffer@ucla.edu,spoko@ucla.edu

Abstract

This project explores effective algorithms to classify African American dialects along the east coast of the United States. Different features such as GFCC and MFCC's gradients were extracted that present better modeling to human auditory perception in addition to using rich auditory feature sets like Opensmile. Additionally, data augmentation by injecting babble noise makes model training more robust to background interference. Overall, our XGBoost model with noise augmentation using Opensmile's GeMapsV01B feature set fused with 21 GFCC coefficients yielded 95.08% and 81.84% accuracy on the test clean and noisy sets, respectively.

Index Terms: speech recognition, human-computer interaction, computational paralinguistics

1. Introduction

Speech recognition is a common topic in today's research due to its wide use application in Internet of Things (IOT) devices and use in medical, technological, and security fields. Historically, research in speech recognition was focused on treated environments meant to isolate the speaker's voice. While useful in understanding speech patterns, this limits the applications that the resulting speech-to-text can be used. With the prevalence of AI assistants in electronic devices, speech recognition must become robust enough to be used in untreated noisy environments as well. This project explores effective algorithms to classify dialects of African American speech in different cities along the east coast of the United States. We study the various Digital Speech Processing methods, including MFCC, GFCC, and PLP to convert audio files into discernible text that can be processed for further applications. We analyze the algorithms of these features to understand how they are effective in clean and noisy environments, and compare them to a provided baseline accuracy.

1.1. Objective

The aim of this study is to determine the origin of an African-American English Speaker from cities along the East Coast of the United States. We are given a dataset of audio excerpts of speakers from the following cities:

1. Rochester, NY (ROC)
2. Lower East side, Manhattan, NY (LES)
3. Washington DC (DCB)
4. Princeville, NC (PRV)
5. Valdosta, GA (VLD)

From this dataset, we are to derive a set of features to be trained on clean audio excerpts. The clean data set are audio excerpts absent of any noise that may mess with the original message. We will train our models on only the clean data sets to prevent overfitting in realistic applications. The model will be tested on new clean and noisy data for the final accuracy result. Models will be graded based on the accuracy determined by the python script, in addition to other classification metrics such as Confusion matrices. The speed of the implementation is also taken into consideration.

1.2. Baseline Performance

We are given a baseline script with the following metrics and a baseline time to run of 30 minutes in Google Colab:

Dataset	Accuracy
Test Clean	76.9%
Test Noisy	62.9%

Our goals are to create an algorithm that will meet or surpass the baseline accuracy while running faster. However, the speed stat is heavily dependent on running the code locally vs in Google Colab as well as the user's computer performance.

2. Project Description

We are given a dataset of clean utterances longer than 10 seconds to perform the tests on in order to create our models. For the baseline implementation, MFCC Features with 13 coefficients extracted the utterances and were trained in the XGBoost model. We used various different features and different models in order to create a model with higher accuracy. The following features were used:

- Mel-Frequency Cepstrum Coefficients (MFCC)
- Gammatone Frequency Cepstral Coefficients (GFCC)
- Perceptual Linear Perception (PLP)
- Power Normalized Cepstral Coefficients (PNCC)
- Opensmile's feature Sets (GeMAPSv01b, eGeMAPSv02)

2.1. Background of Features

2.1.1. Mel-Frequency Cepstrum Coefficients (MFCC)

Mel-Frequency Cepstrum (MFC) is one of the more common features used in sound and speech recognition systems, and was the feature used for the baseline accuracies to reach for this project. An MFC provides a short-term power spectrum of a signal based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel scale is a scale of pitches judged by listeners equidistant to each other, centered around a perceptual pitch of 1000 mels to a 1000 Hz tone, 40 dB above the listener's threshold. The conversion to hertz to mels is as follows:

$$m = 2595 \log_{10}(1 + \frac{f}{700})$$

An MFC requires Mel-frequency cepstral coefficients (MFCCs), which are derived from a cepstral representation of an audio clip [1]. The objectives of the process are to remove vocal fold excitation (F0), the pitch information, make the extracted feature independent and be able to adjust how humans perceive loudness and frequency of sound. The non-linearity of the mel-scale of frequency approximates the human auditory system than linearly-spaced frequency bands.

The algorithm to attain the MFCCs is shown in the figure below:

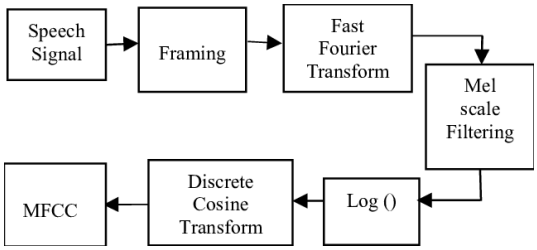

Figure 1: *Flowchart to find mel-frequency cepstral coefficients. Many of the following features will use this as a basis.*

The entire process is a precursor to more modern deep learning methods. MFCCs resample the spectrum using the Mel Filter bank on a nonlinear frequency scale that approximates the human auditory system. The baseline result used a MFCC with 13 coefficients. For the first optimization, we did a hyperparameter optimization choosing various number of MFCC coefficients, and then additionally adding 1st and 2nd order gradients, which are gradients between the values of the MFCC coefficients. These hyperparameter optimizations only saw marginal improvements to the clean and noisy accuracy.

2.1.2. Gammatone Frequency Cepstral Coefficients (GFCC)

The Gammatone Frequency Cepstral Coefficient (GFCC) method differs from a conventional MFCC in that it is based off of an auditory periphery model that imitates the human cochlear filtering mechanism [2]. This auditory model is represented by a collection of Gammatone filters derived from past psychological and physiological observations of the human auditory periphery.

GFCCs are found by decomposing and input signal into the time-frequency domain from a bank of Gammatone filters, down-sampled by filter-bank response in the time dimension. The resulting magnitudes of the down-sampled filter-bank response are put through a cubic root operation and decorrelated by a discrete cosine transform. GFCCs use a Gammatone Filter Bank unlike the Mel Filter Bank from MFCCs. The Bark Scale, like the mel-scale is a nonlinear frequency scale but is aimed to be an improvement over it. The formula for converting from linear to bark frequency is as below:

$$F_{Bark} = 13 \tan^{-1}(0.00076 f_{linear}) + 3.5 \tan^{-1}((\frac{f_{linear}}{7500})^2)$$

Bark Scale is aimed to more accurately represent the human hearing range by capturing a higher detail in frequency ranges that may be missed in the mel-scale. This is due to the bandpass filter having a more parabolic shape instead of a triangular one, allowing for a wider critical band and capturing higher frequencies better. This conversion allows for capturing voices in with noise better.

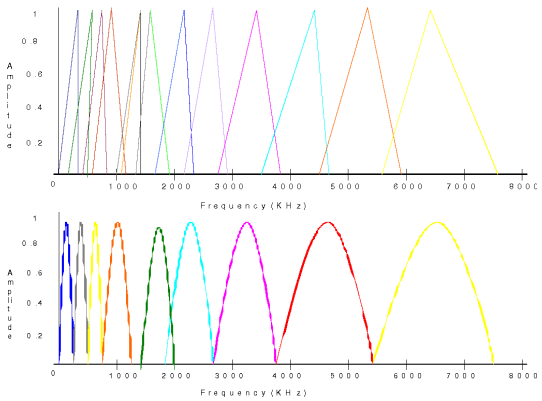

Figure 2: *Graphs of amplitude spectrum of frequencies using the mel and bark scale. Note the different shapes of the curves.*

2.1.3. Perceptual Linear Prediction (PLP)

The Perceptual Linear Prediction method is another popular speech recognition algorithm that approximates the auditory spectrum of speech via an all-pole model [3]. It works by first computing the spectrogram through a discrete Fourier transform then warping the spectrum along the Bark frequency scale. It then convolves with the power spectrum and downsamples and then preemphasizes by simulating an equal loudness curve. Then it simulates the non-linear relationship between the intensity and the perceived loudness and finally computes the linear prediction model. A PLP is an improvement over a standard all-pole LP model, attaining higher accuracy at a lower order model, preventing it from over fitting to the data.

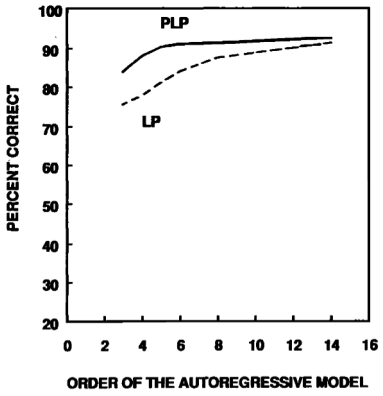

Figure 3: *Relative accuracy between an all-pole LP and a PLP model.*

The main advantages that it provides are improved speaker independent speech recognition as well as robustness to noise compared to a standard linear predictive model, but it comes at a cost of slower computation as compared to MFCC.

2.1.4. Power-Normalized Cepstral Coefficients (PNCC)

The Power Normalized Cepstral Coefficients method is another feature extraction algorithm that we tried to apply for our purpose. The PNCC processing works by using a power-law nonlinearity model, unlike how a traditional log nonlinearity is used in MFCCs. The PNCC method works by first preemphasizing

to emphasize the high frequency components, then performing short time Fourier transform then magnitude squared and then doing gamma tone frequency integration. It has a medium time processing component with a temporal window of 65.6ms, performing medium time power calculation, asymmetric noise suppression with temporal masking, weight smoothing, and time frequency normalization. It then does mean power normalization, power function nonlinearity, discrete cosine transform, and mean normalization in order to generate the PNCC coefficients. PNCC was mainly developed in order to extract features that were more robust to acoustical variability in their native form, which mainly refers to more natural speaking [4].

2.2. Libraries

2.2.1. Librosa

Librosa [5] is a python package for music and audio analysis that was used to attain the baseline results with MFCCs with 13 coefficients. While a common library, it is also focused on retrieving information on music files, thus for the more advanced features in this paper, we had to implement them with other python libraries focused on speech processing.

2.2.2. Opensmile

Opensmile (Open-source Speech and Music Interpretation by Large-space Extraction) is an open source toolkit for python that's used for audio feature extraction and classifying speech and music signals [6]. The library features MFCC as well as more advanced features involving Mel/Bark spectrums, Linear Predictive Coding (LPC), and Perceptual Linear Predicative Analysis (PLP). However, for this project, we focused on the standard feature sets the library provides for speech classification. These include two feature sets, one from Opensmile's original package (GeMapsV01B - OS1) and its 2nd version (eGeMapsV02 - OS2). These feature sets are an all-in-one preset for speech classification, aimed to minimize overfitting while increasing the generalizability of the models.

The GeMaps feature set consists of 18 low level descriptors (LLD) which are more general descriptors for the model. They are divided into 3 categories: frequency related, energy/amplitude, and spectral parameters covering qualities such as pitch, Alpha Ratio, and Formant information.

All 18 LLDs are smoothed over time with a symmetric moving average filter that is 3 frames long to create an arithmetic mean and coefficient of variation to them to create an additional 18 parameters. Additional statistical calculations are done to the loudness and pitch for another 16 parameters. Then finally, 6 temporal features describing the spectrum such as the rate of loudness peaks are added.

In total, the 18 LLDs, their derivatives, and the 6 temporal features total 62 parameters calculated and used by the Geneva Minimalistic Standard Parameter Set. In Opensmile 2, we used an extended parameter set of GeMAPS, applicably named eGeMAPSV02, which adds an additional 7 LLDs to the set related to the spectrum and frequency.

Likewise, the arithmetic mean and coefficient of variation are found for these 7 additional LLD, creating 14 more parameters. Additional arithmetic means and coefficient of variations for the spectral flux and MFCC 1-4 in voiced regions, the arithmetic means for the spectral flux in unvoiced regions, and the equivalent sound level results in 26 extra parameters. Thus, our final tally makes 88 parameters in the extended Geneva Minimalistic Acoustic Parameter Set (eGeMAPS).

2.2.3. Spafe

Spafe (Simplified Python Audio Features Extraction) is a library used for simplifying audio feature extraction, using numpy and scipy libraries as a backend [7]. Spafe includes many features with more diverse filterbank set including Bark Frequencies, Constant Q, and Gammatone frequencies. It is more computationally efficient than librosa when computing audio features such as the MFCC. Spafe was used to extract the GFCCs, PLPs, and PNCCs of the dataset for classification.

2.3. Methods

2.3.1. Feature Extraction Acceleration using Multiprocessing

One observation when running the initial baseline code for feature extraction of the dataset is that it only utilizes a single CPU core processing each WAV sample sequentially. These days, many CPU processors are multicore (e.g. the Intel Core i9-12900K used in our project has 16), so the intuition would be to divide the dataset into equally sized chunks to each CPU core to extract features concurrently to save time.

To implement multiprocessing in feature extraction, the library *joblib* [8] was used as a high-level interface to spawn multiple feature extraction pipelines. Firstly, the extract function for an individual WAV file is defined for the desired feature. Next, through all the files in a set (e.g., Train Clean), is divided into k chunks, where k is the number of CPU cores available in the system. Each chunk is then run through concurrent processes spawned by *joblib*, extracting the features for each sample in the chunk. Afterwards, *joblib* recombines all the outputs of each chunk to a single output and returns it to the user. With this acceleration pipeline, users should see a performance increase that scales with the number of processor cores they have. An implementation of this is available on our GitHub repository [9] in the *Project.ipynb* notebook, section 3.

2.3.2. Data Augmentation Using Noise Injection

A method we explored in order to improve robustness to background noise that is prevalent in the test noisy data was to augment our limited training data set with background chatter noise. Background chatter noise was chosen over other forms of noise, such as Gaussian noise, since it was more representative of the noise found in the test noisy data set.

The background chatter noise was obtained through a 1 hour long YouTube video of chatter noise [10] that was split into 20 3-minute noise chatter clips. Then 1 out of the 20 chatter audio clips was randomly applied to each of the training audio clips. Initially, the same chatter audio was applied to all samples and that provided no noticeable uplift in noisy performance. This was likely a result of using the same noise in all the clips. The model was likely trained to ignore that specific noise clip and not noise in general. That is why we randomized the noise applied to achieve better performance. The intensity of the chatter noise clip was also lowered, by about -21dB, in order to sound like the chatter is in the background and so that it does not drown out the original training audio.

3. Results & Discussion

3.1. Results of Individual Features

The results for each individual feature we explored is tabulated in Table 1. Each feature model was trained without and with noise augmentation. The run times were computed on a local

machine, running an Intel Core i9-12900K processor using the multiprocessing pipeline described in section 2.3.1. Running the feature extraction pipeline on the local machine with multiprocessing provided an performance increase on average 3-5x then compared to running it on Google Colab.

The baseline model proved to be a good challenge for individual features to meet and surpass in both clean and noisy data. We first did hyperparameter optimization for the MFCC features, experimenting with the number of coefficients as well as adding in 1st and 2nd order gradients. For the MFCC, 13 coefficients still ended up being the best number of coefficients to use with 1st order gradients, resulting with a clean accuracy of 76.95% and a noisy accuracy of 63.68%, which are only marginally better than the baseline.

From Gammatone Frequency Cepstral Coefficients (GFCC), the usage of the bark scale allowed it to yield a higher accuracy in the noisy data than the MFCC with 82.71%, a 20% increase in accuracy. However, due to its focus on interpreting noisy data it struggled heavily on clean data, giving us only a 66.44% accuracy. It's likely due to the lack of information in clean data for the model to sift through results in it not being able to accurately determine the speaker.

For Perceptual Linear Prediction (PLP) and Power Normalized Cepstral Coefficients (PNCC), they struggled to meet or surpass the baseline, PLP at 13 and even 24 coefficients were not able to meet the baseline accuracies, with only the 24 coefficient variation able to reach a noisy accuracy of 61.96.% This is unexpected, but is likely due to the PLP all-pole model approximating the power distribution equally well at all frequencies, it is likely that accents that may emphasize certain frequencies over others get mitigated by this model, making it more difficult to determine the location of the speaker. Due to their poor performance, PLP and PNCC models would not be used in any feature fusions.

3.2. Improvements through Feature Fusion

Our group developed two fusions of features to improve the testing accuracy for the clean and noisy datasets. Fusion 1 (OS1 + GFCC21) takes the high test clean accuracy from OS1 and the high test noise accuracy of GFCC and combines them together for accuracies of 93.51% & 65.13% respectively. Fusion 2 (GFCC13 + MFCC13 with 1st order gradient + OS2) utilizes more noise robust features that promotes higher test noise accuracies, yielding 79.42% & 74.93% respectively. The fusion results are tabulated in Table 2.

3.3. Improvements from Noise Augmentation

Augmenting our training data with noise injection has a slight uplift in accuracy for the test clean data set and a significant uplift in accuracy for the test noisy data set than without noise injection (shown in Table 2). For the ensemble of Opensmile1 (OS1) and GFCC21, the augmented data model provided an additional 1.5% accuracy in the test clean data set and an additional 16% accuracy in the test noisy data set. Another interesting observation is that although it provided such an uplift in performance in this best ensemble of features, the noise augmented data model did not necessarily provide and improved accuracy when using standalone features. However, since it does provide the best performance when combining features, we proceeded with using the data augmented model in order to achieve our best accuracy.

The confusion matrices in Figures 4 and 5 visualize the performance of the classifier on both test data sets before and after noise augmentation, respectively. By comparing each matrix in the base data with its corresponding matrix for augmented data, one can observe there is little variation in what kinds of misclassifications are happening, with one notable exception. For the noisy test set under the fusion features column, the Lower East Side samples go from being almost entirely misclassified to being entirely correctly classified, except for one clip. After examining these clips, it is apparent that they all came from the same female speaker. It is possible, then, that the features of these particular recordings were close enough to the decision boundary that they all flipped to the correct class after adding background noise.

3.4. Hidden Set Test Accuracy

From table 3, our best fusion model (OS1 + GFCC21) reported a hidden clean accuracy of 78% and a hidden noise accuracy of 72%. For comparison, the baseline method yielded a hidden clean accuracy of 72% and a hidden noisy accuracy of 48%.

4. Conclusion

After extensive testing on individual speech recognition features, we were able to collate specific features to create an XGBoost model capable of running significantly faster and more accurate in clean and noisy environments than the baseline results. Combining the effectiveness of Opensmile's GeMAPS feature sets on clean data, and GFCCs robustness with noisy data, we've created a fusion model capable of classifying the location of African American speakers along the east coast with more than 80% accuracy regardless of the environment.

For future studies, feature extraction can include applying gradients to other features, such as GFCCs. Additionally, preprocessing the data through an off-the-shelf voice background remover can be a way to improve all feature's accuracy individually or in a fusion. Changing our dataset to one with more WAV samples and a balanced label distribution helps mitigate model overfitting in limited training data. Finally, improving data augmentation with more noise corpuses would eliminate any potential noise source bias when training the model.

5. Acknowledgements

Our group would like to thank the teaching team (Professor Alwan and Balaji) of the Winter Quarter 2024 offering of M214A Digital Speech Processing course at UCLA. They have helped immensely with the guidance of the project, in addition to providing a baseline MFCC method for us to start with. Our group would also like to thank the authors of the COORAL for making their corpus public for us to use in this project.

6. References

[1] Z. K. Abdul and A. K. Al-Talabani, "Mel frequency cepstral coefficient and its applications: A review," *IEEE Access*, vol. 10, pp. 122 136–122 158, 2022.

[2] B. Ayoub, K. Jamal, and Z. Arsalane, "Gammatone frequency cepstral coefficients for speaker identification over voip networks," in *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, 2016, pp. 1–5.

[3] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *The Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 04 1990. [Online]. Available: https://doi.org/10.1121/1.399423

[4] C. Kim and R. M. Stern, "Power-normalized cepstral coefficients (pncc) for robust speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 7, pp. 1315–1329, 2016.

[5] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[6] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1459–1462. [Online]. Available: https://doi.org/10.1145/1873951.1874246

[7] A. Malek, "Spafe: Simplified python audio features extraction," *Journal of Open Source Software*, vol. 8, no. 81, p. 4739, 2023. [Online]. Available: https://doi.org/10.21105/joss.04739

[8] Joblib Development Team, "Joblib: running python functions as pipeline jobs," 2020. [Online]. Available: https://joblib.readthedocs.io/

[9] T. Do, K. Chen, J. Shiffer, and T. Sison, "M214A-Project." [Online]. Available: https://github.com/dotimothy/M214A-Project

[10] Ambience. Crowd noise 1 hour white noise. YouTube. [Online]. Available: https://www.youtube.com/watch?v=IKB3Qiglyro

Appendices

Table 1: *Test Accuracies & Runtimes for Individual Features*

Feature	Test Clean	Test Noisy	Test Clean (Augmented)	Test Noisy (Augmented)	Run Time (min.)
MFCC13	76.90%	61.90%	78.52%	64.55%	3.00
MFCC13 (w/ 1st order gradients)	76.95%	63.68%	78.52%	66.57%	3.00
MFCC13 (w/ 2nd order gradients)	78.75%	62.54%	79.87%	68.01%	3.00
MFCC13 (w/ both gradients)	78.52%	61.67%	78.74%	67.72%	3.00
Opensmile (Feature Set GeMAPSv01b)	92.61%	49.28%	91.27%	53.89%	25.00
Opensmile 2 (Feature Set eGeMAPSv02)	70.47%	46.39%	70.25%	45.82%	25.00
GFCC13	66.44%	82.71%	69.80%	81.56%	5.00
PLP13	63.98%	55.61%	64.88%	61.38%	12.00
PLP24	72.48%	61.96%	68.00%	61.38%	12.00
PNCC13	65.99%	64.23%	68.23%	68.01%	10.00
PNCC21	73.60%	63.11%	71.81%	62.54%	10.00

Table 2: *Fusion Test Accuracies*

Feature Set	Test Clean	Test Noisy	Test Clean (Augmented)	Test Noisy (Augmented)
OS1 + GFCC21	93.51%	65.13%	95.08%	81.84%
GFCC13 + MFCC13 (w/ 1st order gradients) + OS2	79.42%	74.93%	79.64%	84.44%

Table 3: *Hidden Test Accuracies*

Feature Set	Hidden Clean	Hidden Noisy
MFCC13 (Baseline)	72%	48%
OS1 + GFCC21 (Noise Augmented)	78%	72%

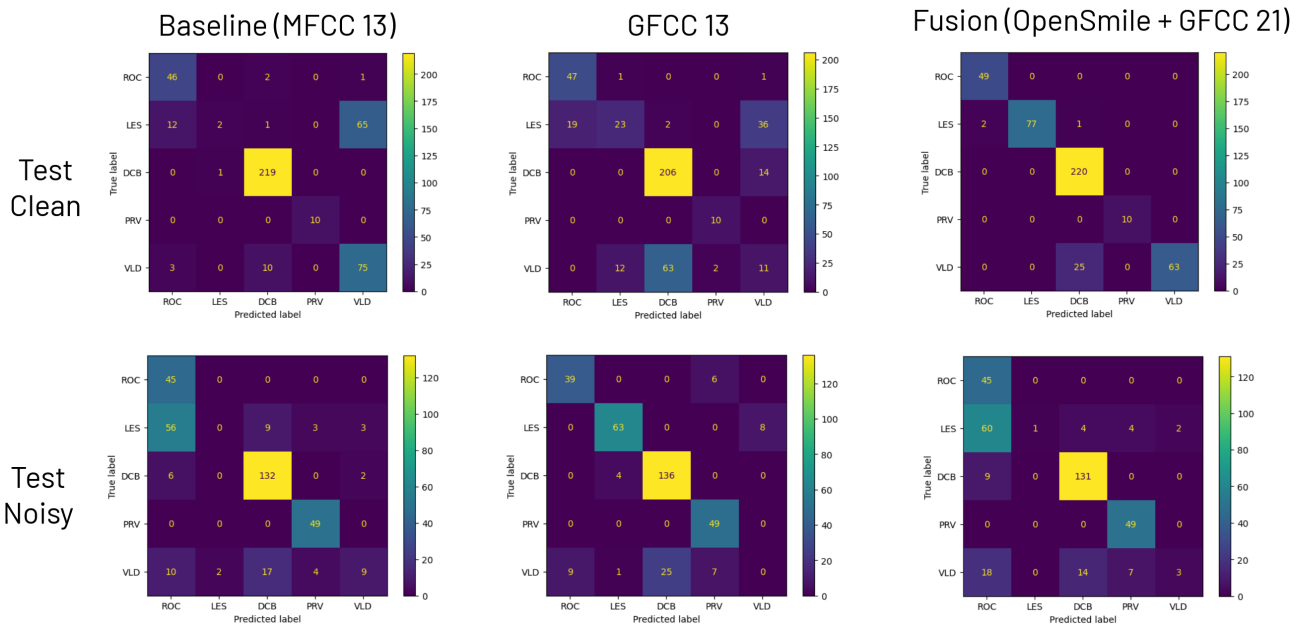

Figure 4: *Confusion Matrices of Individual and Fusion Features for Base Data*

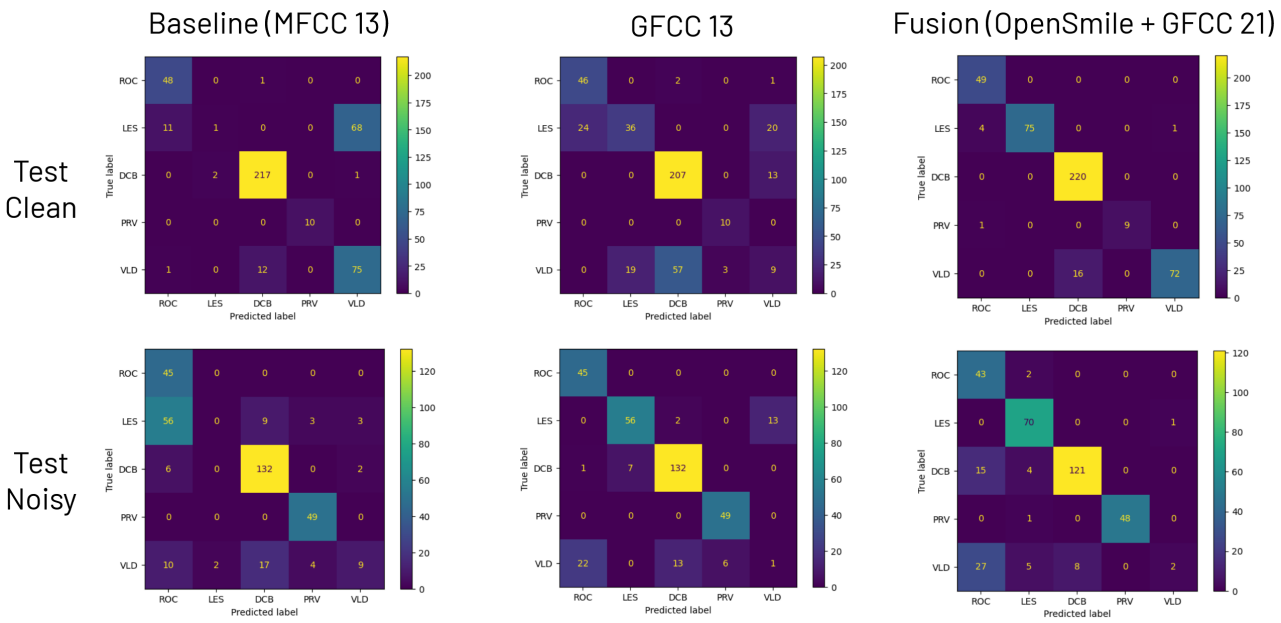

Figure 5: *Confusion Matrices of Individual and Fusion Features for Augmented Data*