ENGLISH-LANGUAGE ACCENT CLASSIFICATION

McCormack Chew, Sydney Haupt, Ahmet Yusuf Salim

University of Rochester
ECE 408 - The Art of Machine Learning

ABSTRACT

Accent detection plays a critical role in accurately captioning audio and enhancing automatic speech recognition systems. In this paper, the authors apply 5 different machine-learning algorithms to two commonly used speech accent datasets - the University of California Irvine (UCI) "Speaker Accent Recognition" corpus and the George Mason University (GMU) "Speech Accent Archive". MFCC features are provided in the UCI dataset and extracted from the GMU dataset. Furthermore, various preprocessing methods are applied to the GMU dataset in order to improve classification accuracy. Results among trials are compared with one another, as well as with results of related work utilizing the same dataset. The authors acheived highest accuracy with the UCI dataset, wherein a multilayer perceptron classifier performed the best. The GMU dataset acheived lower accuracy but has greater potential for further work. The highest performing trial with the GMU dataset had unaccented samples removed and classes equalized, with a convolutional neural network classifier.

Index Terms— Accent Detection, Machine Learning, Neural Networks, MFCC, k-NN, SVM, MLP, CNN, LSTM

1. INTRODUCTION

Accents develop due to historical, social, geographic, and linguistic factors. Consequently, the variation in accents among native and non-native English speakers are incredibly diverse. As defined by [1], "accent classification refers to the problem of inferring the native language of a speaker from his or her foreign accented speech." There are many practical reasons to utilize accent classification, including audio captioning and speech recognition systems.

Social media platforms such as YouTube and Instagram offer auto-captioning for videos. However, when a speaker's accents differs from American-English or British-English, the auto-captioning system can experience poor accuracy. If the system were able to identify the speaker's accent it could adapt, changing its assumptions about what each word should sound like and performing better as a result. The same idea applies to automatic speech recognition systems like Siri or Alexa, as the success of such systems is entirely dependent on their ability to understand a speaker [2].

In our project we compare the performance of five different machine learning models at classifying six different English speech accents. Notably, two sets of accents are considered native-English speakers (United States and United Kingdom), and four are considered non-native (Spanish, French, German, and Italian). The classification methods implemented include two elementary machine learning models (k-Nearest Neighbors and Support Vector Machine) and three modern machine learning models (Multilayer Perceptron, Convolutional Neural Network, and LSTM Recurrent Neural Network). We extract Mel-Frequency Cepstral Coefficients to use as features for classification. This work expands on the existing literature by implementing novel classification methods on previously explored datasets. Our objective is to find the best model to identify and classify accented English speech.

The organization of this paper is as follows: Section 2 explores related work in the literature. Section 3 discusses the features extraction process and datasets. Section 4 explains each classification method. Section 5 presents the results, and Section 6 draws conclusions from our findings. Further data and figures are available in the index.

2. RELATED WORK

A variety of spectral and temporal features can be extracted from audio data, but Mel-frequency cepstral coefficients (MFCCs) are considered one of the most important features for audio classification. This is indicated by the use of MFCCs in [3], [2], [4], [5]. [6] compares the performance of a variety of audio features including MFCCs, Spectrogram, Chromagram, and more, finding that MFCCs surpass other types of spectral features in terms of accuracy.

There are many different approaches to the task of accent classification in the related literature. Some approach the task as a binary classification problem, only looking to differentiate between American and non-American speakers as in [4]. Papers typically implement a small number of models, either focusing on neural networks (as is the case for [5], [1], [3]), or on elementary machine learning methods (such as [7]). While some choose to compare elementary machine learning methods with modern methods (such as [2]), it is relatively uncommon in the literature.

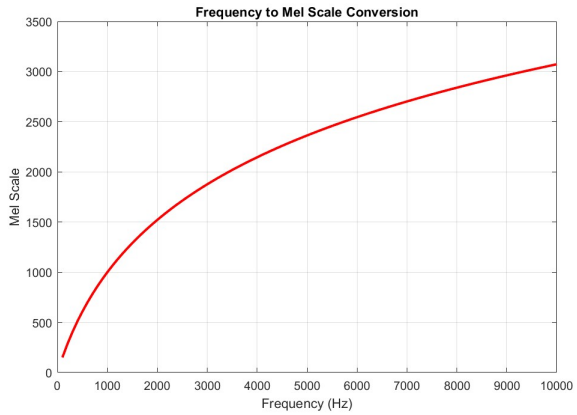


Fig. 1: A plot showing the Mel scale

Accuracy varies widely in the related literature, from around 50 % ([1], [5]) to nearly 90 % ([4]). Variation can be attributed to differences in machine learning models, features, and dataset. Common datasets were the UCI dataset [8] and the GM dataset [9]. The Wildcat corpus [10] and INTERSPEECH 16 datasets [11] were also used.

3. FEATURE EXTRACTION AND DATASET

3.1. Feature Extraction

Audio data contains a vast amount of information and is often too complex to deal with without some type of processing. It is reported that Mel-frequency ceptral coefficients (MFCCs) allow more efficient use of memory compared with classic spectrogram features thanks to modest feature array sizes [12]. Further, MFCCs are compatible with many classification methods ranging from simpler techniques like k-Nearest Neighbor to more novel neural network implementations. Due to the above-mentioned advantages, we chose to employ MFCC features.

Unlike conventional cepstral coefficients, the Mel frequency scale is designed to better reflects human ear perception, particularly in detecting pitch differences between sounds. The Mel scale emphasizes low-frequency components by applying a logarithmic transformation to the frequency scale. The Mel scale is given with the following Eq. (1) in [13].

$$f_{\text{mel}} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{1}$$

Fig. 1 shows the frequency to Mel scale conversion between 100 Hz to 10000 Hz

The first step in extracting MFCCs is to frame the audio signal at specific intervals. Subsequently, the framed data is processed with a windowing function, such as rectangular, Hamming, or Hann and a certain hop size is applied for

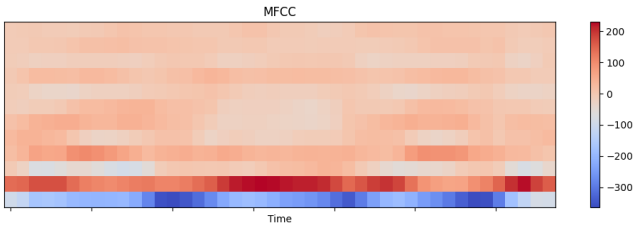


Fig. 2: A Mel frequency cepstrogram computed over a 1-second time frame

overlap. The next step involves transforming the audio data into the frequency domain using a Discrete Fourier Transform (DFT).

In practice, the frequency domain representation of the data is typically calculated using the Fast Fourier Transform (FFT) due to its superior computational efficiency. Next, the power spectrum is passed to the Mel Filter Bank rather than directly computing Mel scale values. This approach captures more information by averaging power across various frequency regions instead of taking discrete values.

The boundaries of the filter bank are evenly spaced in the Mel scale; therefore, the power spectrum is converted to the Mel spectrum after filtering [13]. In the subsequent phase, energy in each filtered region is calculated by Eq. (2) given in [13].

$$E(i) = \sum_{k=1}^{N-1} |X(k)|^2 \psi_i(k) \tag{2}$$

Following this, the log energies $E(i)$ are passed through the Discrete Cosine Transform (DCT), analogous to the inverse Fourier transform used in cepstrum algorithms. This operation ensures that the energies are not correlated, according to [13]. In the final step, the coefficients C_m are calculated using the following Equation (3) [13].

$$C_m = \sqrt{\frac{2}{M}} \sum_{l=0}^{Q-1} \log[E(l+1)] \cos\left(m\left(\frac{2l+1}{2}\right)\frac{\pi}{Q}\right) \tag{3}$$

The coefficient C_m is the m-th cepstral coefficient. Typically, the first 12 or 13 coefficients are used, although up to 40 of them may be employed. The cosine term in the equation represents the DCT transform mentioned earlier. We utilized LibROSA [14], a Python package for audio processing, to extract the MFCCs of samples from George Mason University's "Speech Accent Archive." Fig. 2 shows the progression of MFCCs for a one-second frame sampled from a recording in the GMU corpora.

Furthermore, Fig. 3 depicts the process flow diagram. Initially, the silent parts of the audio are trimmed, as they contain no information. Next, the trimmed audio file is segmented into frames ranging from 1 second to 10 seconds of

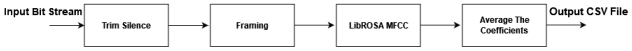


Fig. 3: MFCC feature extraction process

the recording. Subsequently, the MFCCs are extracted using LibROSA. For this extraction, a boxcar window with a 512-point FFT and a 16-point hop size is employed. The average of the MFCCs is then computed for each recording. Finally, the resulting coefficients are recorded into a comma-separated values (CSV) file with their country labels.

3.2. Datasets

Feature extraction and by extension training method selection are driven by dataset selection. We initially chose to work with the UC Irvine (UCI) "Speaker Accent Recognition" dataset [8] but after encountering limitations, chose to work with a subset of the George Mason University (GMU) "Speech Accent Archive" as well. We implemented a variety of preprocessing methods on the GMU dataset, leading to the creation of six GMU dataset variations.

In total we applied seven datasets to our models:

1. The UCI dataset
2. Six countries from the GMU Dataset
3. A reduced version of (2) with neutral accented samples removed
4. A version of (3) with MFCCs standardized
5. A version of (3) with an additional 12 features that are the standard deviation of each MFCC
6. A version of (3) with minority classes upsampled
7. A version of (3) with binary classes representing American versus non-American samples

An LSTM classifier requires sequential data, so a sequential version of each dataset was created when possible. Datasets (1), (5), and (6) were not tested with the LSTM.

3.2.1. UC Irvine Dataset

UC Irvine's Machine Learning Repository hosts a "Speaker Accent Recognition" database which contains the first 12 MFCCs extracted from 329 different .wav files. Each audio segment, spanning approximately one second, contains a single English word spoken by somebody from one of six countries: the United States, the United Kingdom, Spain, Germany, France, and Italy. This dataset is among the most

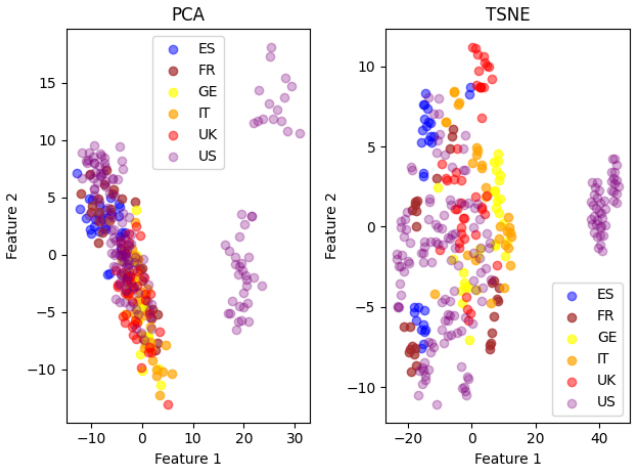


Fig. 4: UCI Dataset Feature Visualization

widely used in existing literature, making it a good choice for the comparison of results.

Initial results with this dataset were promising. However, when we sought to improve accuracy, we were faced with some major limitations of the dataset. Not all of the original audio files are available for download, meaning it would be impossible to extract any features beyond the 12 MFCCs. Furthermore, the specifics of the feature extraction methodology employed are not provided, making it difficult to extract features from unseen audio in a way that is fully compatible with the models trained on the UCI set.

A two-dimensional visualization of the UCI dataset, seen in Fig. 4, shows strong clustering among all languages, with some outliers clustered together from the US class. Though this visualization shows similarity among all classes, there are noticeable vertical patterns in each class as seen in the TSNE.

3.2.2. George Mason University Dataset

George Mason University's "Speech Accent Archive" consists of 2140 English speech samples spanning 214 different native languages. Each speech sample is a recording of the same paragraph, which is designed to contain most sounds in the English language [9]. For the purposes of our project, we selected six countries of origin to work with from the dataset: United States, United Kingdom, Spain, Germany, France, and Italy. These are the same countries included in the UCI dataset, allowing for direct comparison between the two datasets.

Our primary motivation for selecting the "Speech Accent Archive" dataset is that it provides the freedom to extract our own features directly from the audio samples. This freedom had the unintended consequence of creating the greatest challenge of the project - obtaining high quality results with the extracted MFCCs.

Our initial version of the GMU dataset consisted of the

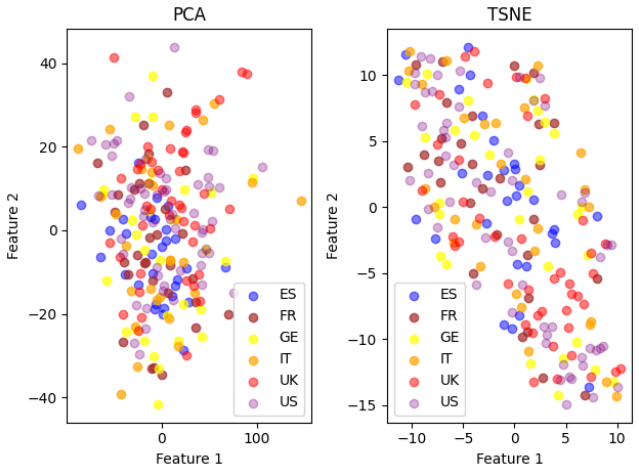


Fig. 5: GMU Complete Dataset Feature Visualization

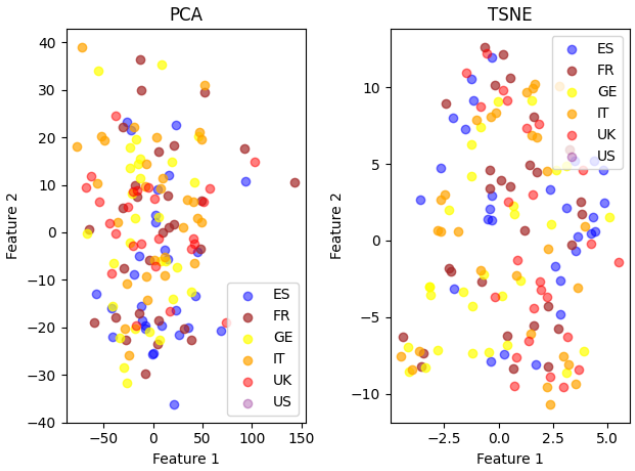


Fig. 6: GMU Reduced Dataset Feature Visualization

six countries of interest, with 12 MFCC features and a total of 222 samples. Upon experiencing poor performance with the dataset, we listened to some of the audio samples. It soon became clear that across all countries of origin, many of the samples had accents practically indistinguishable from the native-English accents. We removed the least accented samples to create a new dataset, now with only 141 samples across the six languages. In order to further improve performance, we created variations on this dataset with different preprocessing. This meant adding an additional normalization step to the MFCCs, creating 12 additional MFCC standard deviation features, and upsampling underrepresented classes. Each of these preprocessing methods was implemented as a separate dataset.

Additionally, a binary version of the GMU dataset was created with American versus non-American accents as classes.

The two-dimensional visualization of our six-country subset of GMU (seen in Fig. 5 and Fig. 6) is scattered in comparison with UCI, though vague vertical and diagonal patterns can still be found in the TSNE. The GMU datasets with and without unaccented samples appear similar to one another.

4. CLASSIFICATION METHODS

We implemented five classification methods: k-Nearest Neighbor, Support Vector Machine, Multi-Layer Perceptron, Convolutional Neural Network, and an LSTM Recurrent Neural Network.

4.1. k-Nearest Neighbor

[15] uses the k-Nearest Neighbors (kNN) classifier on the UCI dataset and states that this method is common in speech recognition research. kNN does not have a learning phase, and its speed decreases as the dataset grows. Since both the UCI and GMU datasets have only a limited number of samples and features, kNN excels in terms of speed. It is important to normalize the features to have zero mean and unit variance to prevent numerically large features from dominating.

The kNN algorithm begins by splitting the dataset into training and test sets. Each labeled sample in the dataset is described by a set of features, in this case, MFCCs. The distance between two samples is calculated based on all features. When an unlabeled instance needs to be classified, the distances between this instance and all labeled samples are considered, and the labels of the N nearest neighbors are used to vote on the predicted label. The hyperparameters in this scenario include the number of neighbors N, the distance metric (e.g., Manhattan, Euclidean, or a higher-order distance), and whether the distance weighting is uniform or weighted. In this study, the scikit-learn library is used. The best hyperparameters for the UCI dataset were found to be 6 neighbors with weighted Euclidean distance.

4.2. Support Vector Machines

The support vector machine (SVM) classifier was chosen because of its success in [7]. It is important to note that [7] implemented SVM for binary classification, only attempting to differentiate between American and non-American accents.

SVM is a supervised machine learning algorithm that classifies data by finding the optimal line or hyperplane separating classes with the widest margin possible. The margin between classes is defined by support vectors, which are the samples of each class nearest to the hyperplane [16].

For complex data where direct linear separation is not possible, the "kernel trick" can be used to project data into a higher dimensional space thus allowing for linear separation. Common kernel functions include linear kernels, polynomial

kernels, radial basis function (RBF) kernels, and sigmoid kernels [17].

We used sklearn's Support Vector Classifier. Within the sklearn implementation, multiclass classification is handled in a one-vs-one scheme [18]. Tuned hyperparameters were C, gamma, and kernel. C is a regularization parameter determining allowable margin violations, and gamma is the kernel coefficient that determines how much individual samples impact the final decision boundary [18]. It was found that a higher C value (100 or 1000), lower gamma value (0.01 or 0.001), and RBF kernel perform the best though these results varied across datasets.

4.3. Multi-Layer Perceptron

A Multi-layer Perceptron (MLP) is a network of neurons arranged in fully connected layers. Features in the input layer are transmitted to neurons in one or more hidden layers. In a hidden layer, the inputs are weighted, summed, and processed by an activation function to generate an output. During training, these activation functions and weights are randomly initialized and optimized iteratively to find the most effective values. At the final hidden layer, outputs from previous layers are used to generate a prediction. MLPs are intended to mimic the functionality of the human brain and are effective at learning complex relationships in data [19]. MLPs are among the simplest and most adaptable machine learning models and precedent has been set for using this type of model for accent classification tasks [2].

Our implementation was done using the scikit-learn MLP library. A number of different network sizes were tested, ranging from one to three hidden layers with between five and three hundred neurons. The other tested hyperparameter was the learning rate of the model. Across multiple tests on different subsets of data, no network size consistently performed the best, but our highest accuracy on any dataset was achieved with two hidden layers which each had twenty neurons.

4.4. Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN). An RNN differs from other neural networks in that it uses sequential data. Basic RNN models struggle with the task of understanding long-term dependencies because of vanishing and exploding gradients [20]. The purpose of the LSTM is to learn sequential data with long-term dependencies through the use of a special memory unit [21]. The trade-off is that the LSTM is more complex and therefore expensive to train than an RNN [21].

Because LSTM is the only classification method implemented that requires sequential data, it was necessary to incorporate sequential MFCC extraction into the code. This was accomplished with the torchaudio MFCC transformer [22], and a custom pytorch dataset to hold many frames of MFCCs for each audio sample. The LSTM was implemented with keras layers [23]. Hyperparameters for improving performance of the LSTM model are number of LSTM layers (one versus two), number of hidden units, and regularization parameters. It was found that one layer with fewer hidden units (32 or 64 versus 128) slightly improved test accuracy but drastically reduced training accuracy. Regularization parameters had little impact on performance.

4.5. Convolutional Neural Network

Convolutional neural networks (CNNs) share many similarities with multi-layer perceptrons, but have some distinct components which set them apart. Notably, a CNN has convolutional layers as well as pooling layers. In convolutional layers, a filter or kernel is applied to the input data in order to create a representative feature map. Pooling layers then reduce the dimensionality of the data in order to pick up on broader patterns [24]. Finally, data is sent to one or more dense, fully connected layers, reminiscent of a standard MLP. These layers are then able to generate a class prediction.

The CNN structure makes the model very popular for image processing, due to its ability to learn two-dimensional relationships between pixels. [2] use a CNN on the spectrogram data of audio in order to perform accent classification. While this was the motivation for us to look into the use of CNNs, we applied the concept in a somewhat unconventional and novel way. Instead of using a two-dimensional spectrogram as the input, we input our one-dimensional MFCC values, hoping that the model's structure could help it learn valuable patterns in the relationships of adjacent MFCC values.

We utilized TensorFlow's Keras library to create our CNN model. The hyperparamers which we tuned to optimize our model include number of filters, number of dense neurons, kernel size, and pool size. While the optimal number of filters and the size of the dense layers varied across different datasets, a filter size of 4 and a pool size of 3 frequently achieved favorable accuracy. This suggests that the CNN model was observing feature relationships between subsets of four MFCC values at once and subsequently learning relationships among subsets in the feature space in groups of three.

5. RESULTS

We conducted a total of 33 trials, where each of the seven datasets were tested with each of the five classification models. Some datasets were not applicable to LSTM, reducing the total number of trials. Our LSTM model was more limited in application due to the need for sequential data.

In each trial, a random subset of the data was separated to create a test set that would not be seen in the training process. K-fold validation was used for hyperparameter tuning with the training dataset.

Table 1: UCI Dataset Results

	kNN	SVM	MLP	CNN
Training Set Accuracy	100%	95.51%	100%	100%
Training Set Precision	100%	88.19%	100%	100%
Test Set Accuracy	80.17%	97.34%	88.25%	87.24%
Test Set Precision	80.30%	87.88%	90.91%	87.88%

Table 2: Upsampled GMU Results

	kNN	SVM	MLP	CNN
Training Set Accuracy	100%	93.52%	100%	100%
Training Set Precision	100%	93.67%	100%	100%
Test Set Accuracy	45.65%	52.08%	52.08%	53.24%
Test Set Precision	47.50%	52.50%	52.50%	55.00%

In order to evaluate the performance of our models, we calculated accuracy, precision, recall, F1, and plotted the confusion matrix for the train set and test set of each trial.

Interestingly, precision, recall, and F1 were equal to one another in every trial. As such, we only report one value accounting for all three metrics. After additional research we found this to be due to "micro" averaging being implemented in the sklearn precision score, recall score, and f1 score [25]. The sklearn.metrics.precision score documentation explains that micro averaging "calculate[s] metrics globally" while macro averaging "calculate[s] metrics for each label" [25].

For the purposes of our research, accuracy is the most important metric as it is crucial to understand the success rate of our models in classifying samples. Our reasoning was explained well in [26], "accuracy is a helpful metric when you... care about the overall model "correctness" and not the ability to predict a specific class."

Overall classification accuracy was found to be highest with the UCI dataset, and out of all variations on the GMU datset, the highest accuracy was achieved with unaccented samples removed and minority classes upsampled. The results of these trials can be found in Table 1 and Table 2 respectively. Our KNN and MLP models were among the most accurate on average, and confusion matrices detailing the relationship between their predictions and true accents labels are included as figures 7-10. All other results can be found in the index.

Our results demonstrate how the metric of accuracy can fail in cases where a dataset is drastically imbalanced. Visualizing the outcome of a model with a confusion matrix is essential for catching this issue, as seen in Fig 11. Across all models with the GMU data sets, samples were frequently classified as French or American due to the many instances of those classes in the dataset. This issue inspired the upsampled dataset, which resulted in more varied predictions and higher accuracy.

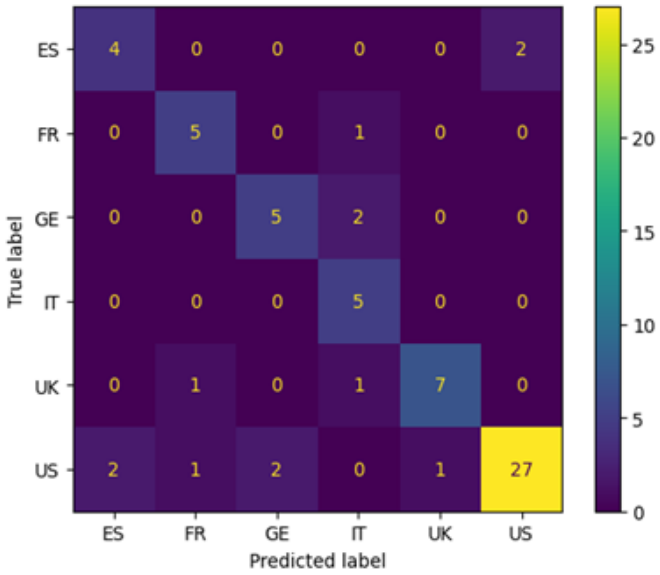


Fig. 7: kNN Confusion Matrix (UCI Test Set)

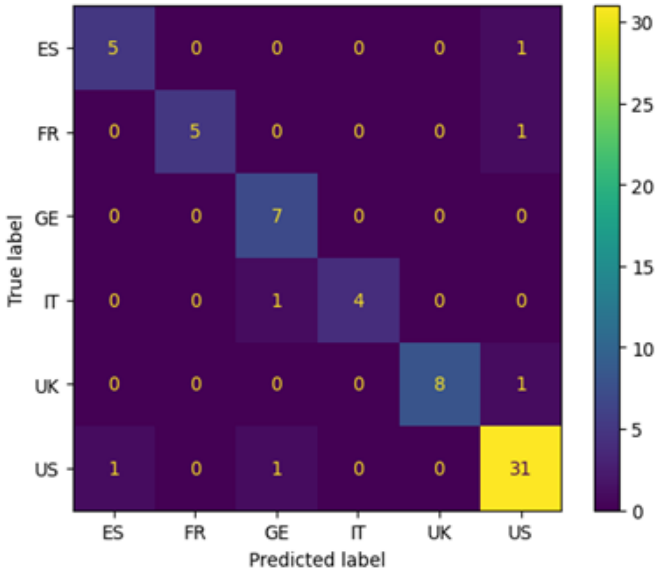


Fig. 8: MLP Confusion Matrix (UCI Test Set)

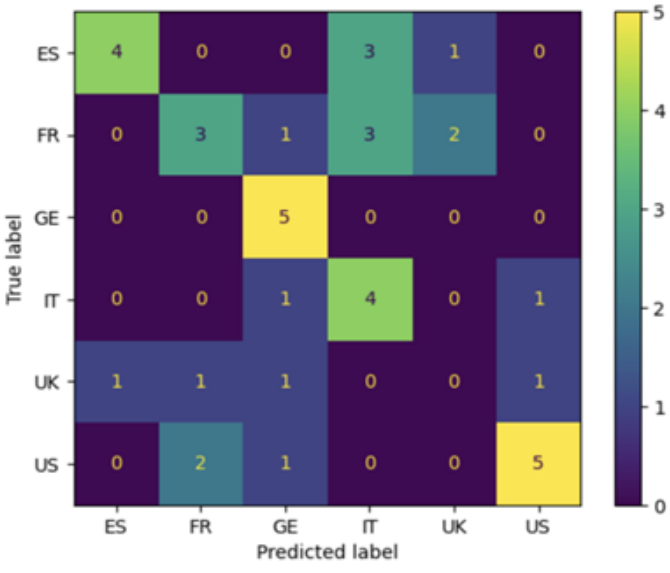


Fig. 10: MLP Confusion Matrix (Upsampled GMU Test Set)

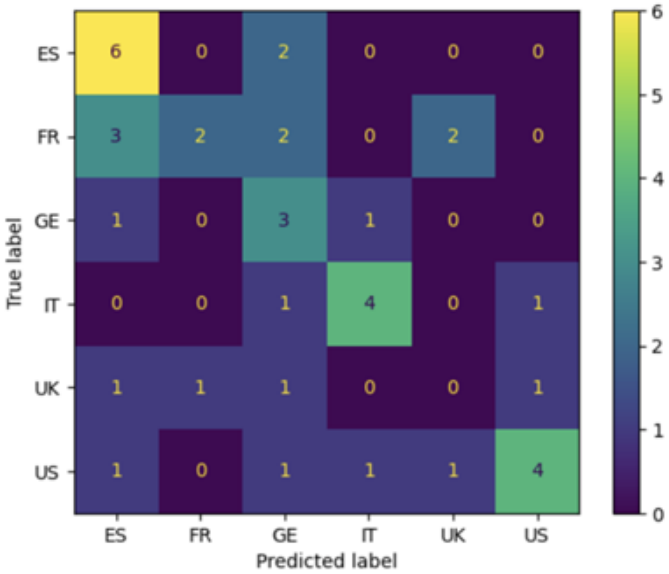


Fig. 9: kNN Confusion Matrix (Upsampled GMU Test Set)

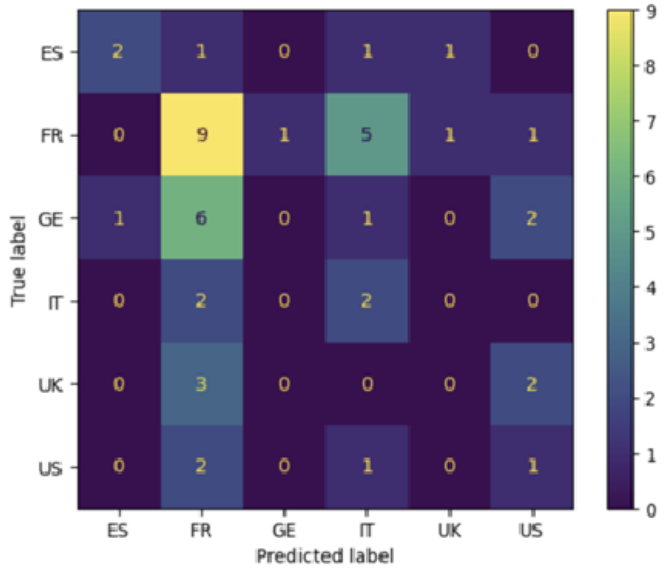


Fig. 11: SVM Confusion Matrix (GMU Test Set)

The performance achieved using kNN, SVM, and MLP closely aligns with the findings of [15] and [7], which utilize the same methods and UCI dataset. As indicated in Table 3, the performance metrics are closely matched. The slight differences observed may be attributed to variations in algorithm implementation or the division of train and test sets, which are randomly partitioned.

The performance achieved with the GMU dataset aligns with [5], where a test accuracy of 52.57% was obtained with four language classes, MFCC features, and an LSTM classifier. [1] also achieves 50.2% accuracy, but with a different dataset and features, and with a more complex neural network classifier.

6. CONCLUSIONS

Over the course of this research project, we found dataset/preprocessing and feature extraction to have a greater impact on performance than model selection. For that reason, we will briefly discuss the classification models, before focusing on dataset, features, and overall findings.

Each model except for the LSTM performed best for a different dataset. For example, kNN achieved highest accuracy among all models with the GMU unaccented-samples-removed dataset at 44.6%, and SVM performed best among the models with the larger GMU dataset at 27.99% accuracy. Neural networks performed better on the datasets with additional features (GMU dataset with standard deviation of MFCCs), and with more preprocessing on the dataset (equal classes, normalized MFCCs), with CNN achieving as high as 53.24% accuracy. Though neural networks are more computationally expensive, they also excel at identifying more abstract patterns among samples, which could explain why they perform well in such conditions.

Feature extraction was a primary challenge for this project. When provided with features from the UCI dataset, performance was as high as 88.25%. This indicates that certain models like SVM and MLP are appropriate options when classifying samples with high-quality MFCC features and deserve further examination. However, because of the limitations on the UCI dataset, these findings lack direct applicability to other datasets. Based on the 2-D data visualizations, it makes sense that SVM and MLP models perform better with the UCI dataset than the highly scattered GMU dataset. The primary takeaway from the UCI dataset findings were that high precision can be achieved with relatively simple models given a high quality dataset.

The challenge of obtaining a high quality dataset with strong features was apparent with the GMU datasets. This may explain the nearly 30% difference in accuracy scores between UCI and GMU datasets. When tasked with generating MFCCs, there can be a lot of variability in the resulting features. The reason for this is that MFCCs can differ based on sample length, window length, hop length, normalization factor, and specific library used. We kept these variables consistent across runs, but they may differ from those used in the related literature. There is a strong possibility that window type and length caused the difference in our extracted GMU MFCCs compared with the UCI MFCCs. For example, the coefficients of the Hann window are generated through the following equation:

$$w(n) = 0.5\Big(1 - \cos\Big(2\pi \frac{n}{N}\Big)\Big)$$

where window length = N+1 [27]. Because the UCI dataset was so successful, future research should seek to replicate the UCI features by trialing many different window types and lengths.

The GMU dataset's audio samples suffer from a lack of consistency in terms of accent strength. Many samples across all countries of origin have such neutral accents that they are practically indistinguishable from native English speakers. These samples negatively impacted performance by diluting the classes in which they appeared, giving the impression that these classes had more in common with American or British English than they really do. This issue was resolved by removing the unaccented samples from the dataset, which improved accuracy by as much as 5%.

Because a primary application of our research is to enable auto-captioning and smart-electronics to understand heavy accents, we believe that it is appropriate to ignore the neutral accents among non-native English classes. Taking auto-captioning as an example - captioning algorithms are already trained to understand American English, so neutral accents from other countries will have no trouble being understood if they sound like American English. On the other hand, it is necessary to identify heavy accents in order to apply a different auto-captioning algorithm. Further work should seek out a larger and more heavily accented dataset.

Trials with the GMU dataset also revealed how issues can arise from inequality among classes. This was especially highlighted in our binary classification test of US versus non-US accents. Even though US was a majority class in the multiclass trials, it was the minority in the binary trial. As such, practically none of the predictions were for the US class in the binary trials. For multiclass classifiation, accuracy increased by as much as 12% by equalizing the classes. One caveat is that we upsampled both test and train data together, which could alter results compared with only upsampling the train data, or upsampling both sets of data separately. Because of how much class inequality impacted results, this should be an area of focus for future research.

Due to the fact that the UCI dataset predicts classes with high accuracy, it provides little information about the similarities and differences between the six classes. In this area, the GMU dataset is more revealing. Due to imbalanced classes, predictions with the GMU dataset favor the dominant classes

Table 3: Comparison to performance in literature

	K-NN	SVM	MLP
This Work	80.17%	88.19%	87.24%
Ayranci et al. [15]	82.80%	X	82.78%
Muttaqi et al. [7]	86.44%	89%	X

of French and United States, as seen in Fig. 11. When the classes are balanced, German becomes the favored class with the US as the second most favored, as seen in Fig. 9 and Fig. 10. German is also the most accurately classified class in this case. This is likely because out of the non-native English speaking classes, German is the only non-Latin language, differentiating it from Spanish, French, and Italian, as well as from US and UK. Listening to some of the German audio samples, many of them are only lightly accented, which could explain why lightly accented samples in other classes may be classified as German. The presence of these neutral-sounding samples could also explain why they are misclassified as from the United States. Listening to the French samples, many of them are somewhat ambiguous in their accent and it makes sense that they could be misinterpreted as a different accent. Future research should explore the relationship between these languages in order to conclusively determine whether such characteristics are attributed to the dataset, or to the accents themselves.

7. DIVISION OF WORK

McCormack Chew: binary dataset, MLP, CNN
Sydney Haupt: oversampling dataset, SVM, LSTM
Ahmet Yusuf Salim: MFCC extractor, all other datasets, kNN
All: Dataset sourcing, research paper

8. REFERENCES

[1] Yishan Jiao, Ming Tu, Visar Berisha, and Julie M. Liss, "Accent identification by combining deep neural networks and recurrent neural networks trained on long and short term features," in *Interspeech*, 2016, https://api.semanticscholar.org/CorpusID:40239068.

[2] L. Mak An Sheng and M. W. X. Edmund, "Deep learning approach to accent classification," 2017, https://cs229.stanford.edu/proj2017/final-reports/5244230.pdf.

[3] Jordan J. Bird, Elizabeth Wanner, Anikó Ekárt, and Diego R. Faria, "Accent classification in human speech biometrics for native and non-native english speakers," in *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, New York, NY, USA, 2019, PETRA '19, p. 554–560, Association for Computing Machinery.

[4] Kaplan, "Classifying accents from audio of human speech," 2020.

[5] Shih, "Speech accent classification," pp. 1–5.

[6] Jembere Singh, Pillay, "Features of speech audio for deep learning accent recognition," 2019, vol. 2540, pp. 1–3.

[7] Mohammad Muttaqi, Ali Degirmenci, and Omer Karal, "Us accent recognition using machine learning methods," in *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2022, pp. 1–6.

[8] "Speaker accent recognition," UC Irvine Machine Learning Repository.

[9] Steven Weinberger, "Speech accent archive," 2015, George Mason University.

[10] Baker Choi Kim Bradlow Engen, Baese-Berk, "The wildcat corpus of native- and foreign-accented english: Communicative efficiency across conversational dyads with varying language alignment profiles," 2010, vol. 53(Pt 4), pp. 510–540, PubMed Central.

[11] D. Higgins A. Cahill M. Chodorow D. Blanchard, J. Tetreault, "Toefl11: A corpus of non-native english," 2010, pp. i–15, Wiley Online Library.

[12] Vaibhav Arora, Pulkit Sood, and Kumar Utkarsh Keshari, "A stacked sparse autoencoder based architecture for punjabi and english spoken language classification using mfcc features," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 269–272.

[13] Md. Afzal Hossan, Sheeraz Memon, and Mark A Gregory, "A novel approach for mfcc feature extraction," in *2010 4th International Conference on Signal Processing and Communication Systems*, 2010, pp. 1–5.

[14] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, vol. 8.

[15] Ahmet Aytuğ Ayrancı, Sergen Atay, and Tülay Yıldırım, "Speaker accent recognition using mfcc feature extraction and machine learning algorithms," 2021, vol. 33, p. 17–27, Marmara University.

[16] Neil Zhang, "Support vector machine," in *ECE208/ECE408 - The Art of Machine Learning*.

[17] IBM, "What are support vector machines (svms)?," 2023.

[18] sklearn, "sklearn.svm.svc," .

[19] D. Femi and S. Thylashri, "Human voice emotion recognition using multilayer perceptron," in *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, 2022, pp. 1–4.

[20] Zhiyao Duan, "Recurrent neural networks," in *ECE208/ECE408 - The Art of Machine Learning*.

[21] GeeksForGeeks, "Deep learning — introduction to long short term memory," .

[22] PyTorch, "Mfcc," .

[23] Keras, "Lstm layer," .

[24] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[25] "sklearn.metrics.precision score," scikit learn.

[26] "Accuracy vs. precision vs. recall in machine learning: what's the difference?," Evidently AI.

[27] "hann," 2024, MathWorks.

9. INDEX

See all numeric data and selected confusion matrices in the following pages. Due to the high volume of confusion matrices obtained, we have only included the most notable ones. All confusion matrices are for the test dataset. "Full GMU Test Set" refers to the 6-country set with unaccented samples included. "Reduced GMU Test Set" refers to the 6-country set with unaccented samples removed.

Table 4: UCI Dataset Results

	kNN	SVM	MLP	CNN
Training Set Accuracy	100%	95.51%	100%	100%
Training Set Precision	100%	88.19%	100%	100%
Test Set Accuracy	80.17%	97.34%	88.25%	87.24%
Test Set Precision	80.30%	87.88%	90.91%	87.88%

Table 5: Complete GMU Dataset Results

	kNN	SVM	MLP	CNN	LSTM
Training Set Accuracy	100%	67.57%	69.17%	97.66%	25.02%
Training Set Precision	100%	68.93%	70.06%	97.74%	30.08%
Test Set Accuracy	24.22%	27.99%	22.70%	18.38%	18.75%
Test Set Precision	24.44%	31.11%	24.44%	22.22%	33.33%

Table 6: Reduced GMU Dataset Results

	kNN	SVM	MLP	CNN	LSTM
Training Set Accuracy	100%	45.37%	58%	100%	35%
Training Set Precision	100%	50.86%	58%	100%	39%
Test Set Accuracy	44.64%	32.54%	27.40%	21.72%	18.06%
Test Set Precision	48.28%	34.48%	24.44%	24.44%	20.69%

Table 7: Normalized Reduced GMU Dataset Results

	kNN	SVM	MLP	CNN	LSTM
Training Set Accuracy	100%	30.62%	100%	99.11%	29.26%
Training Set Precision	100%	39.29%	100%	98.21%	29.76%
Test Set Accuracy	22.64%	19.72%	24.72%	31.53%	22.64%
Test Set Precision	27.59%	24.14%	24.14%	34.48%	24.14%

Table 8: Upsampled GMU Results

	kNN	SVM	MLP	CNN
Training Set Accuracy	100%	93.52%	100%	100%
Training Set Precision	100%	93.67%	100%	100%
Test Set Accuracy	45.65%	52.08%	52.08%	53.24%
Test Set Precision	47.50%	52.50%	52.50%	55.00%

Table 9: Reduced GMU Dataset Including MFCC Standard Deviation Results

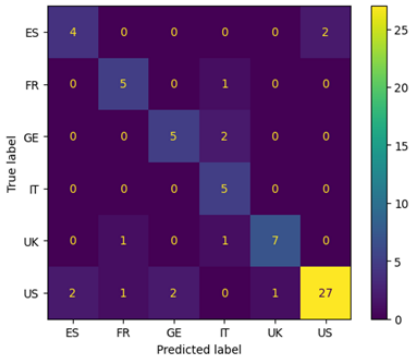
	kNN	SVM	MLP	CNN
Training Set Accuracy	100%	100.00%	100%	100%
Training Set Precision	100%	100.00%	100%	100%
Test Set Accuracy	22.64%	25.28%	26.81%	23.33%
Test Set Precision	27.59%	24.14%	27.59%	27.58%

Table 10: Reduced GMU Binary USA vs Not USA Classifier Results

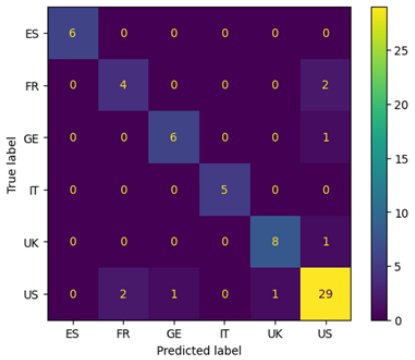
	kNN	SVM	MLP	CNN	LSTM
Training Set Accuracy	100%	60.42%	98.91%	95.69%	50.00%
Training Set Precision	100%	83.62%	98.28%	95.69%	78.57%
Test Set Accuracy	50.00%	50.00%	56.16%	53.99%	50.00%
Test Set Precision	79.31%	79.31%	79.31%	75.86%	86.21%

8.1. UCI Test Set

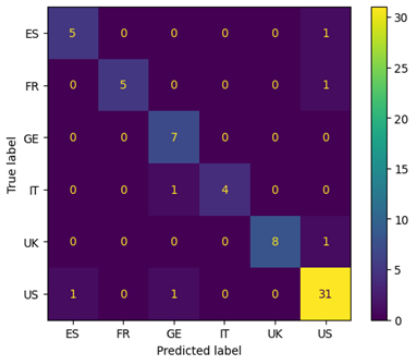
kNN



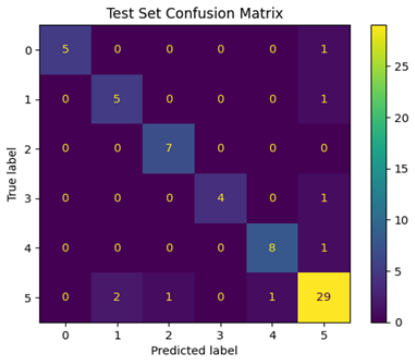
SVM



MLP

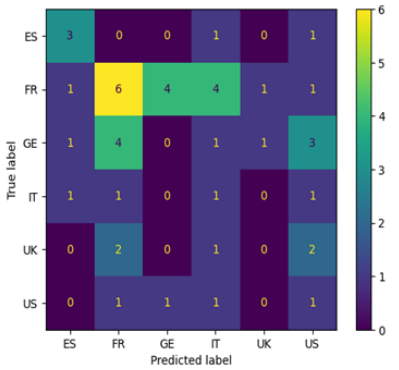


CNN

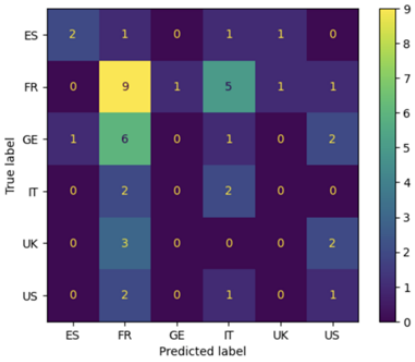


8.2. Full GMU Test Set

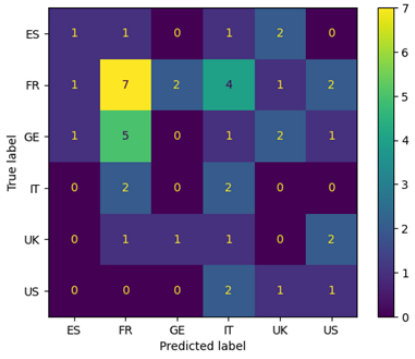
kNN



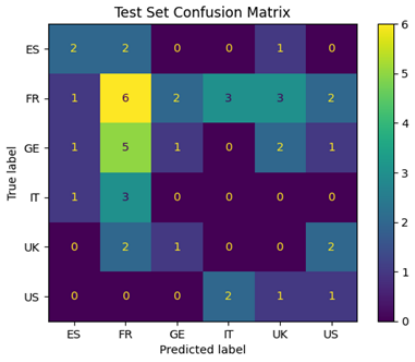
SVM



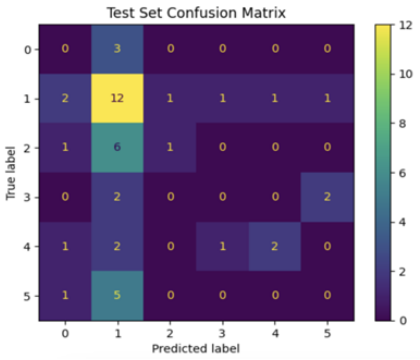
MLP



CNN

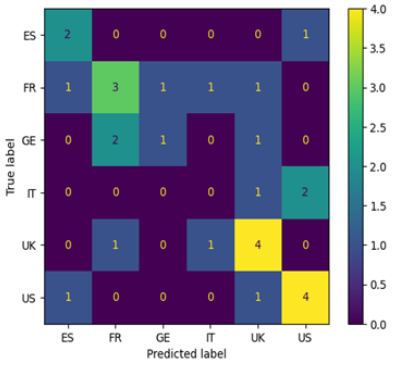


LSTM

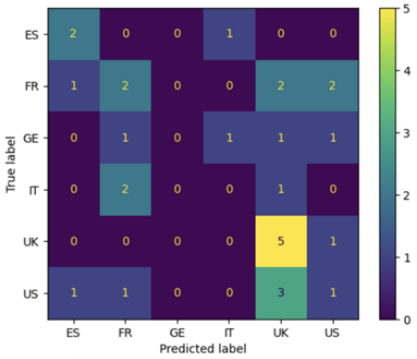


8.3. Reduced GMU Test Set

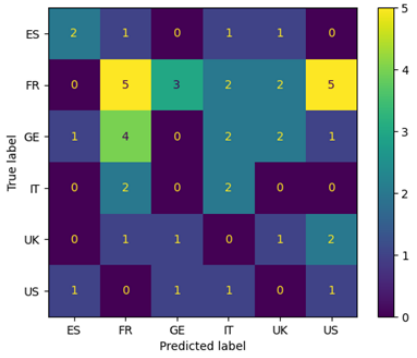
KNN



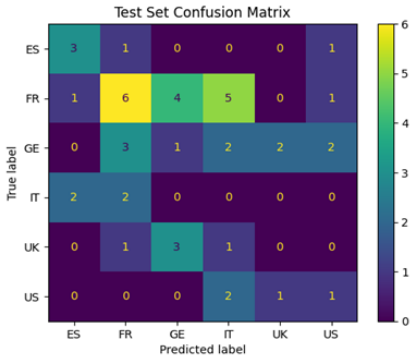
SVM



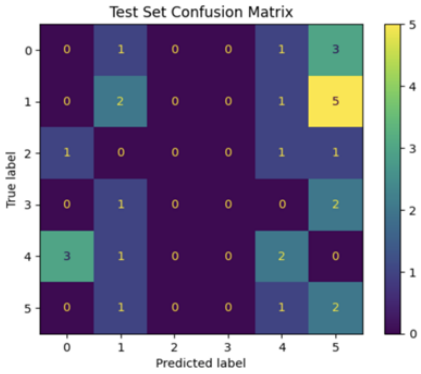
MLP



CNN

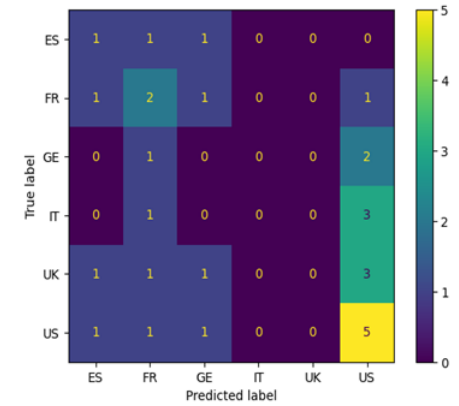


LSTM

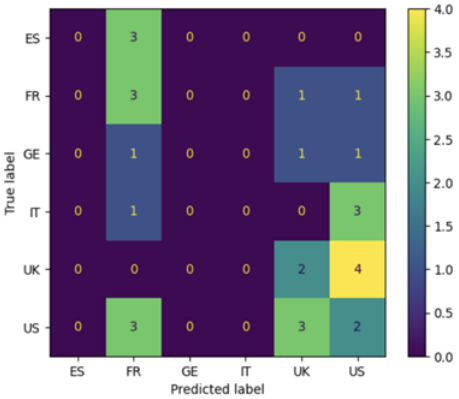


8.4. Standardized Reduced GMU Test Set

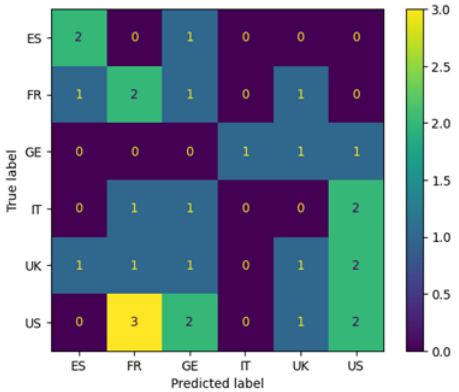
kNN



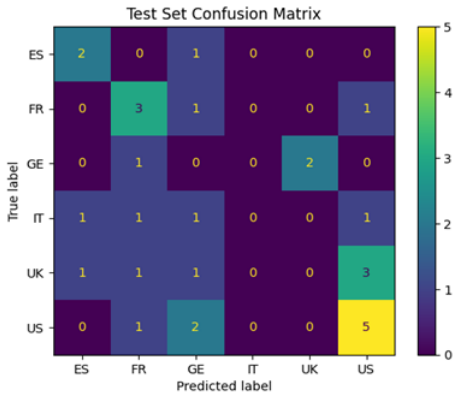
SVM



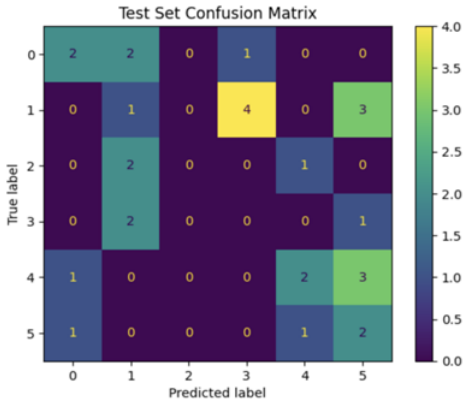
MLP



CNN

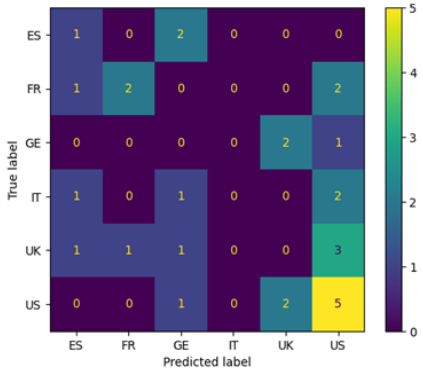


LSTM

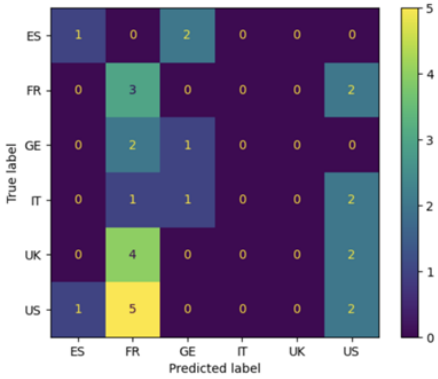


8.5. Reduced GMU Test Set With MFCC Standard Deviation

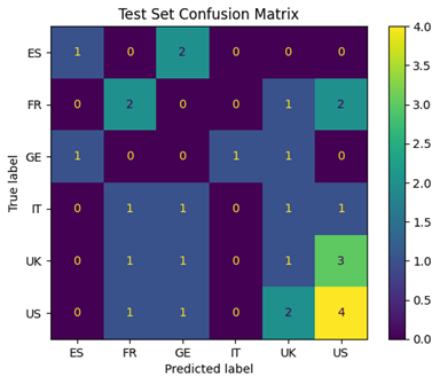
kNN



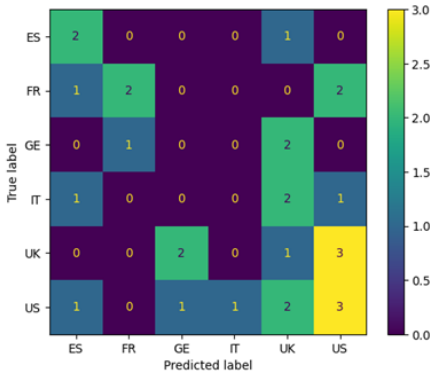
SVM



MLP

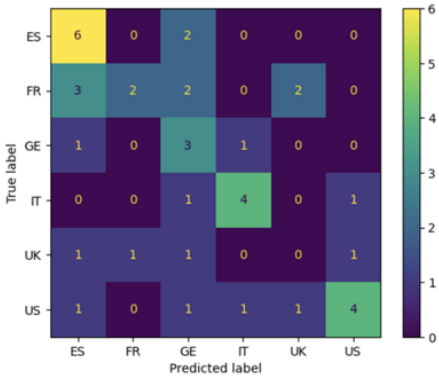


CNN

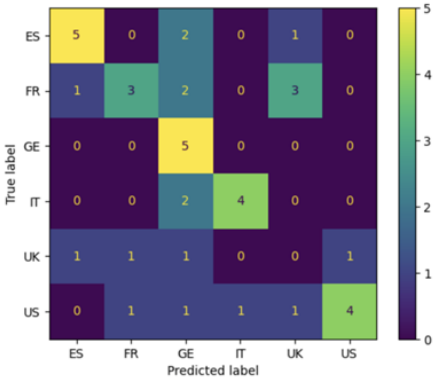


8.6. Upsampled GMU Test Set

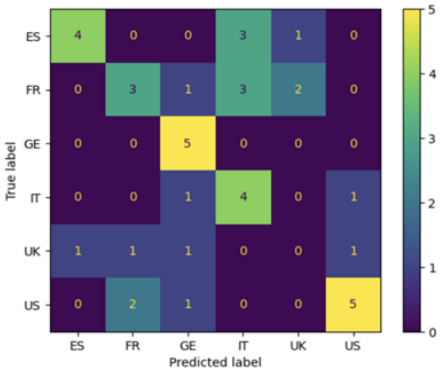
kNN



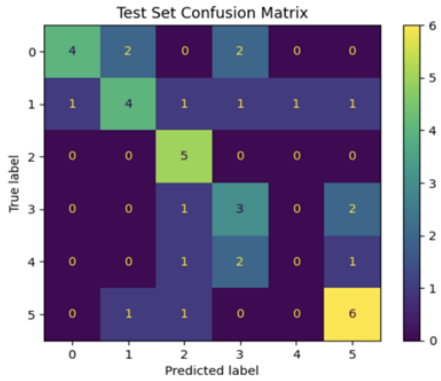
SVM



MLP

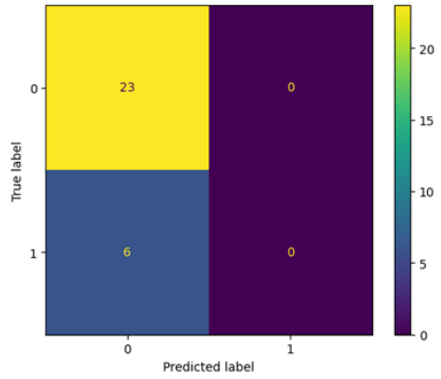


CNN

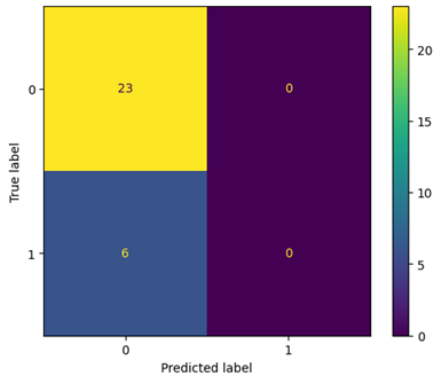


8.7. Reduced GMU Binary Classification (USA vs Non-USA)

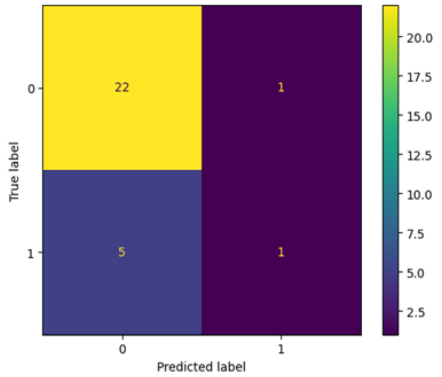
kNN



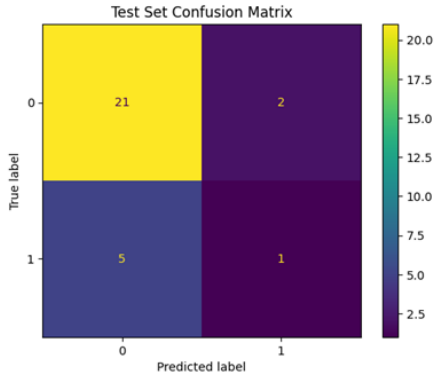
SVM



MLP



CNN



LSTM