Accent Classification of Non-Native English Speakers

Albert Chu, Peter Lai, Diana Le

Abstract

Accented speech from non-native English speakers imposes a challenge to speech recognition because accented speech tends to result in phones that are atypical. Accent classification can enhance the automatic speech recognition system by identifying the ethnicity of a speaker and switching to a speech recognition system that is trained for that particular accent. In this paper, we attempt to solve this problem by classifying short 20-second accented voice clips as one of target languages. The approach to this task of accent classification consists of feature extraction and machine learning classifiers. In this project, the five target languages are Tamil, German, Brazilian Portuguese, Hindi, and Spanish. We use three types of acoustic features, and they are MFCC, its deltas, and FBank. We also apply PCA to these features to reduce the data dimensionality and capture the important data variations. We explore machine learning classifiers for accent classification of non-native English speakers into one of the languages aforementioned. The classifiers include k-Nearest Neighbors, Support Vector Classifier, Multi-Layer Perceptron, Recurrent Neural Network and Convolutional Neural Network. We achieve a test accuracy of 0.398 from LSTM model, a 37.8% improvement over the test accuracy of 0.289 by our baseline SVC model.

1 Introduction

Speech recognition is an active research area in natural language processing and has many important real-world applications. One of the major challenges in speech recognition is to understand speech by non-native speakers. Accented speech tends to result in phones that are not typical of a language which makes speech recognition difficult.

Accent detection or classification can improve the quality of speech recognition in that the automatic speech recognition (ASR) system can first identify the ethnicity of a speaker and then use an automatic speech recognition system that is trained for that particular accent [1]. In addition, accent recognition, which provides identification of a speaker's ethnicity, is crucial in applications such as crime investigation [2]. In real life applications, it becomes essential to recognize accents from only short snippets of audio.

The task of accent classification can be defined as follows:

For a non-native English speaker s whose native language is ℓ_s in the set of all non-English languages $\mathcal{L}$, given his/her n-second clip $c_{s,n}$ in the set of all clips $\mathcal{C}$, we would like to find a mapping $\Phi: \mathcal{C} \to \mathcal{L}$ such that the occurrence of prediction misses $\Phi(c_{s,n}) \neq \ell_s$ is minimized.

Define the function f, for a subset $C_n \subseteq \mathcal{C}$.

$$f(\Phi, C_n) = \sum_{c_{s,n} \in C_n} \delta\big(\Phi(c_{s,n}), s\big) \tag{1}$$

where $\delta(x, y) = 1$ if $x \neq y$ or 0 otherwise. The function $f(\Phi, C_n)$ is the number of prediction misses for all $c_{s,n} \in C_n$.

Accent classification can be formulated as an optimization problem. The objective is to find the best mapping Φ^* for the clip set C_n.

$$\Phi^* = \arg\min_{\Phi} f(\Phi, C_n) \tag{2}$$

In the context of this paper, given a 20-second clip $c_{s,20}$ of a non-native English speaker s, we seek to classify the native language ℓ_s of the speaker s to one of the five languages: Tamil, Germany, Brazilian Portuguese, Hindi, and Spanish .

In this paper, the approach involves acoustic feature extraction, feature descriptors, and machine learning. Acoustic features MFCC, Delta and FBank are extracted from .wav files. Principal Component Analysis (PCA) is used for feature descriptors. K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN) with Long-Short-Term-Memory (LSTM), and Convolutional Neural Network (CNN) are used for machine learning. Our data set is CSLU: Foreign Accented English database from the LDC.

The best accuracy on the test set we achieve is 39.83% from LSTM model, as compared to the test accuracy of 28.9% by our baseline SVC model.

The paper is organized as follows. The background and related work are described in Section 2. The approach and technical details in acoustic feature extraction, feature descriptors and machine learning are described in Section 3. The experiments and the dataset used are described, evaluated, and analyzed in Section 4. Finally, conclusions are drawn, and future work is proposed.

2 Background and Related Work

There are many approaches to the task of accent classification, from the classic methods of Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) to the machine learning methods of SVC and MLP, then further to deep neural learning methods of LSTM and CNN.

Tang, et al. used machine learning techniques such as SVM on acoustic features and achieved comparable results to the HMM systems [3]. While there have been many studies using advanced neural network technologies such as DNN on automatic speech recognition, there were only a few studies done on accent detection taking advantage of advanced models.

Hautamaki, et al. attempted to characterize any language and foreign accent universally, using a deep neural network (DNN) to model the manner and place of articulation [4]. The DNN has 6 hidden layers and 1024 hidden nodes. Using a DNN improved accent detection rates across all tested languages, compared to a shallow neural network (SNN).

Jiao, et al. worked to identify the native languages of non-native English speakers from eleven countries [5]. They proposed a combination of long-term and short-term training based on the observations that accent differences are mostly due to prosodic and articulation characteristics. They find that the neural networks and the SVM (the baseline design) learn complementary knowledge on the speech segments because the fusion of neural networks and SVM outperforms neural networks alone and SVM alone, raising accuracy to 52.48% from 51.92% on their own test set, compared to the SVM baseline whose accuracy is 44.66%.

While Jiao, et al. achieved good accuracy in their own set of 45-second clips, we would like to explore the more challenging task of classifying 20-second clips. Because of the shorter length, classifying shorter clips will be more difficult as 20-second clips will only have less than half of the features than 45-second clips.

Real life applications such as crime investigation would likely only have limited amount of audio to listen to before needing a classification on the accent. Additionally in cases that use an ASR system like a support line, ASR systems should quickly be able to recognize accent to be able to switch to a more appropriate ASR. For usability and response time, we believe that the ASR system should be able to classify accents from a voice clip of no more than 20 seconds and identify the ethnicity of the speaker and then switch to an ASR that is trained for the particular accent.

In this paper, we seek to classify accents from 20-second voice clips of non-native English speakers to one of the five languages: Tamil, Germany, Brazilian Portuguese, Hindi, and Spanish.

3 Approach

In this paper, the approach involves acoustic feature extraction, feature descriptors, and machine learning. FBank, MFCC, and Delta are extracted as features from .wav files. PCA is used for feature descriptors. KNN, SVC, MLP, LSTM, and CNN are used for machine learning.

The model building for accent classification includes three steps. The first step is to extract features from the .wav files in the dataset. The second step is to build feature vectors for machine learning. The third step is to build and train machine learning models to classify accents into one of the five languages.

In this project, we created machine learning programs to classify accents to languages. The programs are written in Python, Scikit Learn and Tensorflow.

3.1 Acoustic Feature Extraction

We chose to use the popular Mel-frequency cepstral coefficients (MFCC), its deltas (Delta), and Filter Bank (FBank) to form the acoustic features.

Mel Frequency Cepstral Coefficents (MFCCs) are a feature widely used in automatic speech and speaker recognition. MFCCs are calculated by taking the Discrete Cosine Transform (DCT) of the log of filter bank. MFCCs have 13-dimensions with each dimension related to a different band in hearing.

Delta is the differential of MFCC. It captures the changes of MFCC over time. It has the same dimension as MFCC.

Filter Bank Energies (FBank) are also an important acoustic feature. We use log of FBank for our features. Each FBank is a 26-dimensional vector.

We use a Python program to extract MFCC, Delta and FBank features. About half of the sample .wav files have features of sequence length of 2047 and the other half have 409 or less. An MFCC (and Delta) feature has 13 values while an FBank feature has 26 values. In the experiments, we truncate the sequence length to either 50 or 200.

As an exploration, we attempted to use 2047 as the sequence length, but we encountered out-of-memory error in Numpy. Thus, we settle for a shorter sequence length, though results indicate this would adversely impact the accuracy of the model.

3.2 Feature Descriptor

Three types of acoustic features are concatenated into a single feature descriptor. Assume a sample x has a sequence length of n and three types of acoustic features: x_{mfcc}, x_{delta} and x_{fbank}. For the i-th sequence of this sample, these three acoustic features $x_{mfcc}^{(i)}$, $x_{delta}^{(i)}$ and $x_{fbank}^{(i)}$ are concatenated into a single vector.

$$x^{(i)} = \left[x_{mfcc}^{(i)} \quad x_{delta}^{(i)} \quad x_{fbank}^{(i)}\right] \tag{3}$$

Then, these sequence features $x^{(i)}$ for $i = 1, \dots, n$ are concatenated into a single feature descriptor $\phi(x)$.

$$\phi(x) = [x^{(1)} \quad x^{(2)} \quad \dots \quad x^{(n)}] \tag{4}$$

The feature descriptor $\phi(x)$ has $(13 + 13 + 26)n$ values. If $n = 200$, then the feature descriptor $\phi(x)$ has 10,400 values. If $n = 50$, then the feature descriptor $\phi(x)$ has 2,600 values.

These feature descriptors of all samples are used for training, validation, and testing in the machine learning algorithms such as KNN, baseline SVC, MLP, and CNN.

For LSTM, we need to preserve the sequences because LSTM takes sequences as input. In this case, we do not perform Eq. (4) for LSTM. In addition, we experiment the effectiveness of these three types of acoustic features. We have three possible sequence features for LSTM, as shown in Eq. (5), (6) and (7).

$$x^{(i)} = \left[x_{fbank}^{(i)}\right] \tag{5}$$

$$x^{(i)} = \left[x_{fbank}^{(i)} \quad x_{mfcc}^{(i)}\right] \tag{6}$$

$$x^{(i)} = \left[x_{fbank}^{(i)} \quad x_{mfcc}^{(i)} \quad x_{delta}^{(i)}\right] \tag{7}$$

In Eq. (5), we only choose FBank features because FBank is the most effective and Delta is the least effective based on the experiments on these three types of acoustic features. Therefore, FBank is used in all three possible sequence features and Delta is used in one sequence feature.

For non-baseline SVC, we apply PCA on the feature descriptor $\phi(x)$ to produce $\phi_{pca}(x)$.

$$\phi_{pca}(x) = \text{PCA}(\phi(x)) \tag{8}$$

PCA is used to reduce the high dimensionality of the feature descriptors and capture important data variations. We apply PCA to the feature descriptor into 20, 50, 80 and 100 dimensions of important data variations. That is, $\phi_{pca}(x)$ has 20, 50, 80 and 100 values. For $n = 200$, the dimensionality reduction is significant because the reduction factors are 520, 208, 130 and 104, respectively.

These PCA feature descriptors $\phi_{pca}(x)$ are used in the non-baseline SVC.

3.3 Baseline: KNN and SVC without PCA

In this project, the baseline model is KNN and the SVC without PCA.

KNN(k) is a nearest neighbor classifier that remembers all training samples. Give the feature descriptor $\phi(x)$ of sample x, KNN(k) gets the majority vote from the k nearest neighbors in the space of feature descriptors. In this paper, we choose $k = 5$.

We use the functions in the Scikit Learn package [7]. For KNN, we use KNeighborsClassifier. For SVC, we use LinearSVC.

PCA is not used for the baseline SVC.

3.4 Support Vector Classifier

SVC finds the support samples that define the decision boundary between two classes. Linear SVC has a linear decision boundary and thus is most efficient in the SVC family. For efficiency, we choose Linear SVC as a machine learning algorithm.

SVC is designed for binary classification. For multi-class classification, SVC with one-vs.-rest is typically used.

Note that all non-baseline SVC will use the PCA feature descriptors $\phi_{pca}(x)$ for training, validation and testing.

3.5 Multi-Layer Perceptron

The neural network classifier MLPClassifier in the Scikit Learn package is used. The classifier can have multiple hidden layers. In this project, we find that two or three hidden layers in general perform better. We choose the "adam" solver (optimizer) because it is an efficient stochastic gradient descent and is recommended by the Scikit Learn documentation. For the activation function in the hidden layers, "relu" is chosen for its efficiency. The regularization penalty α and initial learning rate are tuned.

3.6 LSTM Recurrent Neural Network

RNN is based on building block cells such as Long-Short-Term-Memory (LSTM) cell. Figure 1 shows the architecture of the RNN built for this project. The RNN consists of two layers: Encoding Layer and Decoding Layer.

The Encoding Layer has three Bidrectional LSTM cells, one for each acoustic feature type, that is, one cell for FBank features, one cell for MCC features, and one cell for Delta features.

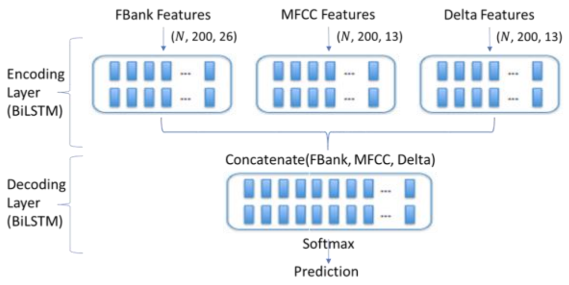

Figure 1: Architecture of RNN with LSTM.

The Encoding Layer takes $x^{(i)}_{fbank}$, $x^{(i)}_{mfcc}$ and $x^{(i)}_{delta}$ as input and convert them into the encoded knowledge representation. The output is the concatenated knowledge representation.

The Decoding Layer has a single Bidirectional LSTM and an affine transformation, followed by a Softmax to produce probabilities. This layer takes the encoded knowledge representation as input and deciphers the knowledge for classification. The affine transformation transforms the matrix to the "unnormalized" probabilities (logits) for the classes. The Softmax converts these logits into probabilities for the classes. The class prediction is the argmax of the class probabilities of all possible classes.

The reason why Bidirectional LSTM cells are used is that the sequence dependencies in acoustic features are bi-directional. We hope that Bidirectional LSTM cells can capture these sequence dependencies.

If the state size of LSTM is n, each Bidirectional LSTM cell produces n forward output values and n backward output values. The concatenation produces $6n$ values that are input to the Decoding Layer.

In this project, we experiment with three combinations of acoustic features: FBank only, FBank+MFCC, and FBank+MFCC+Delta.

The RNN implementation is built on TensorFlow.

For exploration, we also experiment with the so-called attention between FBank and MFCC inside the Encoding Layer. The simplest form of attention is the dot product of two vectors, signaling the cosine distance. However, the accuracy is not promising, and we discard this option.

3.7 Convolutional Neural Network

CNN applies a series of convolutional filters to raw data or descriptors to learn the higher-level features

[8]. A CNN has three types of layers: Convolutional Layer, Pooling Layer and Fully-Connected Layer.

The Convolutional Layer applies convolutional filters to the input data to capture the higher-level metadata. The Pooling Layer down-samples the input data to reduce the data dimensionality. The Fully-Connected Layer is the final layer to perform the classic neural classification.

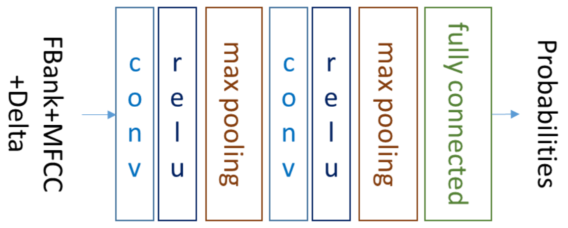

Figure 2: Architecture of CNN.

The architecture of the CNN used in this project is shown in Figure 2. In the Convolutional Layer, we use a convolution patch of (3, 3) and a stride of (1, 1). In the Max-Pooling Layer, we use a pooling kernel of (2, 2) and a stride of (2, 2). This Pooling Layer does not have overlap in down-sampling. The "relu" activation function is used for efficiency. The number of fully-connected units is tuned.

The CNN implementation is built on TensorFlow.

In the implementation of CNN on TensorFlow, the maximum gradient norm for gradient clipping is 10, the optimizer is "adam," the exponential decay rate is 0.95, the number of epochs is 100, and the batch size is 10. The dropout rate used in the fully-connected layer is 0.1.

4 Experiments

KneighborsClassifier, LinearSVC and MLPClassifier in the Scikit Learn package [7] are employed for KNN, SVC and MLP, respectively. Both LSTM and CNN were implemented on TensorFlow.

4.1 Dataset

We use the CSLU: Foreign Accented English database from the LDC. The database consists of 4925 English utterances by native speakers of 22 different languages, including Cantonese, German, Hindi, Portuguese (Brazilian), Spanish, Tamil, etc. Each of the utterances lasts for about 20 seconds and involves the speakers speaking about themselves in English.

To make the classification task manageable, we train and test only on the samples of five languages: Tamil, Germany, Brazilian Portuguese, Hindi, and Spanish because they have the most samples. We extract MFCC (mel frequency spectral coefficients) features and their deltas as well as FBank (Filter Bank) features for each sample. These features are used for training, validation and testing. We extract the first 50 or 200 features of these three feature types from 1764 samples (326 Tamil, 323 Germany, 459 Brazilian P, 348 Hindi, and 308 Spanish samples).

The training, validation, and test set ratio is 20%, 20% and 60% on each language set of samples. We randomly shuffle the data before the split.

4.2 Hyperparameter Tuning

Hyperparameter tuning is important. In this project, it can boost accuracy by as much as ~19% in MLP models.

Using GridSearchCV in the Scikit Learn package, we search with cross validation on all SVC models to obtain the best misclassification penalty C in terms of accuracy. The result is summarized in Table 1.

All SVC/PCA classifiers outperform Baseline SVC. This suggests that PCA is effective to capture important data characteristics and variations.

	Base line	*PCA (20)*	*PCA (50)*	*PCA (80)*	*PCA (100)*
GS Score	NA	0.335	0.325	0.315	0.317
C	1e+00	5e+01	5e-04	1e-05	1e-05
Train Acc	1.000	0.362	0.403	0.439	0.452
Test Acc	0.289	0.318	0.330	**0.347**	0.338
Tamil	0.215	0.308	0.215	0.246	0.262
Germany	0.556	0.667	0.651	0.603	0.571
Brazilian	0.275	0.319	0.297	0.297	0.319
Hindi	0.261	0.217	0.319	0.406	0.362
Spanish	0.148	0.082	0.180	0.197	0.180

Table 1: SVC hyperparameter tuning by GridSearch with cross validation.

For MLP, we tune L2 regularization α, hidden layer sizes and initial learning rate. We set the maximum number of iterations to 1000, the solver (optimizer) to "adam." We find that the best initial learning rate is 0.005 and best α is 0 and best hidden layer sizes are

(100, 100, 100). With this set of hyperparameters, MLP achieves a validation accuracy of 0.35, as shown in Figures 3 and 4.

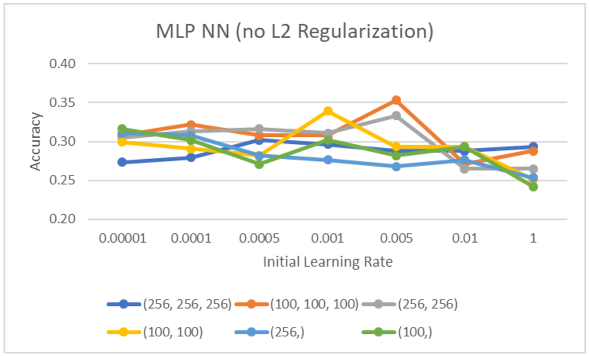

Figure 3: Validation accuracy of the MLP models with using MFCC+Delta+FBank features for various initial learning rates and various hidden layer sizes.

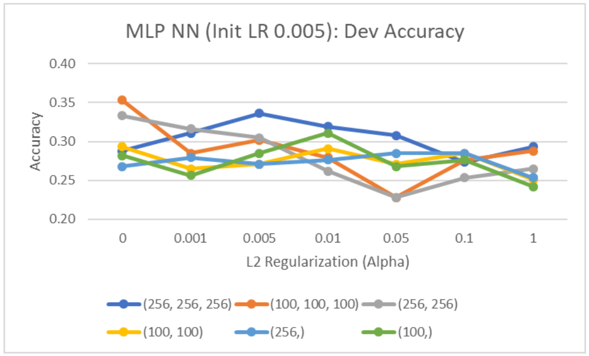

Figure 4: Validation accuracy of the MLP models with using MFCC+Delta+FBank features for various α values of L2 regularization and various hidden layer sizes.

For LSTM, we set the initial learning rate to 0.001, dropout to 0.1, L2 regularization to 0.001, and we experiment with different combinations of feature types, including FBank only, FBank+MFCC, and FBank+MFCC+Delta. We vary the number of epochs and the state size. The best result for LSTM came from FBank only, with a test set accuracy of 0.398, as shown in tables 2 and 3.

Epochs	*FBank*	*FBank+MFCC*	*FBank+MFCC +Delta*
100	0.3983	0.2923	0.3438
200	0.3696	0.3009	0.3582

Table 2: Test set accuracy of the LSTM models for sequence length 200.

Epochs	*State Size 50*	*State Size 100*	*State Size 150*
100	0.3983	0.3668	0.3381
200	0.3696	0.3696	0.3123

Table 3: Test set accuracy of the LSTM models for sequence length 200.

4.3 Main Results

The sequence length of acoustic features is either 200 or 50. The sequence length of 200 gives the better results across all classifiers. It is because the sequence length of 200 have more acoustic features. We summarize our best results from all classifiers below in Table 4 and 5.

For the accuracy across various classifiers, LSTM performs best, MLP second, SVC/PCA a close third. They outperform the baseline models SVC and KNN. However, all the LSTM models have high training accuracies, so these models are probably overfitting. Therefore, we do not see a significant improvement in test accuracy by using LSTM. This is because 1764 samples are insufficient for training complex models, such as LSTM, CNN, and MLP with multiple layers.

For three feature types, FBank gives the best performance with LSTM, while the combination of all three feature types is better than LSTM models with MFCC+Delta with; however, the differences among features are not as significant in simpler models such as KNN, SVC, SVC/PCA and MLP, thus we use all features in simpler models.

SVC/PCA works reasonably well and easy to tune as only the misclassification penalty requires to be tuned. In contrast, most of advanced models, such as MLP and LSTM or CNN, require extensive and time consuming hyper-parameter tuning. SVC is a good starter classifier.

	Baseln. SVC	*KNN*	*SVC PCA*	*MLP*	*LSTM*	*CNN*
Test Acc	0.289	0.284	0.347	0.384	**0.398**	0.352
TrainAcc	1.000	0.530	0.439	1.000	0.956	0.997
Tamil	0.215	0.446	0.246	0.339	0.431	0.169
Germany	0.556	0.349	0.603	0.651	0.730	0.635
Brazilian	0.275	0.308	0.297	0.374	0.286	0.341
Hindi	0.261	0.232	0.408	0.304	0.333	0.377
Spanish	0.148	0.066	0.197	0.262	0.262	0.246

Table 4: Summary of the best results from all models for sequence length 200, where SVC PCA (80) is used.

	Baseline SVC	*KNN(5)*	*SVC PCA(20)*	*MLP*
Test Acc	0.242	0.245	**0.339**	0.322
Tamil	0.215	0.415	0.200	0.246
Germany	0.400	0.292	0.615	0.415
Brazilian	0.297	0.242	0.319	0.385
Hindi	0.275	0.217	0.464	0.391
Spanish	0.230	0.049	0.082	0.131

Table 5: Summary of the best results from all models for sequence length 50.

To understand the effect of number of languages on classification accuracy, we ran similar experiments with 6, 4, 3 and 2 languages. The results are summarized below. Note that we did not perform hyperparameter tuning for MLP, instead we set one hidden layer (100), initial learning rate to 0.001, and L2 regularization α (0.005). Hence, the results for MLP were lower than expected.

	6 lang	*5 lang*	*4 lang*	*3 lang*	*2 lang*
SVC	0.232	0.289	0.310	0.462	0.712
KNN(5)	0.234	0.284	0.285	0.408	0.652
SVC(PCA)	**0.320**	0.347	**0.419**	**0.525**	**0.818**
MLP	0.270	0.384	0.377	0.489	0.750
LSTM	N/A	**0.398**	N/A	N/A	N/A

Table 6: Summary of test accuracy for various numbers of languages, where the six languages include TA, GE, BP, HI, SP, CA, and the five languages include TA, GE, BP, HI, SP, and the four languages include GE, BP, HI, SP, and the three languages include GE, BP, HI, and the two languages include GE, HI. Due to time limitations, we do not tune hyperparamters for MLP models for other than 5 languages.

	TA	GE	BP	HI	SP
TA	**28**	8	15	11	3
GE	2	**46**	5	7	3
BP	23	18	**26**	13	11
HI	16	9	14	**23**	7
SP	13	15	10	7	**16**

Table 7: Confusion matrix for best LSTM model (test accuracy of 0.3983) using FBank features, where the rows represent the ground truth and the columns represent the predictions.

Our best model LSTM yields 0.3983 accuracy, a considerable improvement over the baseline SVC model with a 0.289 accuracy, an improvement by 37.8%. Its confusion matrix is listed in Table 7.

To compare our results with previous work, although our models are based on CSLU dataset with only 20 seconds samples while Jiao, et. al. [4] use 45 second samples from INTERSPEECH 2016 dataset, our best model yields a 37.8% improvement from our baseline SVC, while their system achieves a 17.5% improvement from their baseline SVC's accuracy of 0.447 to their best model's accuracy of 0.525.

Though three-language classification is not our main focus, when we classify among 5 languages, our SVC(PCA) model for three languages yields a better accuracy of 52.5% than the 51.2% accuracy from Ahn [9] who used GMM with PLP features on the CSLU data set.

4.4 Qualitative Analysis

We observe that speakers from the samples vary widely in their accents: some speakers are difficult to tell their accents, some speak slowly and 20 second clips are just a repetition of 1-2 short sentences. These data should be considered as outliners. It is essential to have only quality data.

In this project, we experiment with several ideas to prepare samples for our classifier including selecting different languages that seem to vary most in geographical characteristics, to only picking the same amount of samples from each language so that each language has an equal representation in the samples. However, all these attempts did not yield better results. In fact, our best result came from the current five languages which have most samples and when we use all of the samples even though the numbers of samples varied. This highlights the limitations of the dataset in its size and shows how important a much larger size would be.

We also ran experiments on sequence lengths of either 50 or 200. The results from the 200 sequence length are much better than those from the 50 sequence length in all models we tested, indicating more features is better.

We also confirmed that the number of languages to classify can impact the accuracy of the model. In general, more languages leads to less accuracy. Interestingly, though SVC(PCA) does not suffer as significantly as the number increases from 5 to 6. This might be because SVC(PCA) was not overfitting while MLP, LSTM, and CNN were overfitting.

Finally, German is a language that is easier to identify, according to the test accuracies of different languages in Table 1, 4 and the confusion matrix in Table 7.

5 Conclusion

This project seeks to classify accented speech into the speaker's native language. The approach involves acoustic features and machine learning classifiers.

In this project, there are several lessons learned and they are summarized as follows.

A sufficient amount of quality data is key to machine learning. Too little data adversely affects accuracy and can also lead to overfitting.

Features' sequence length is important. Longer feature lengths capture more acoustic characteristics of the .wav files.

For classifiers, RNN LSTM performs best, MLP second, and SVC(PCA) as third close behind. They all outperform the baseline models of SVC and KNN.

For the features used in this project, FBank gives best performance with LSTM, even better than combinations with MFCC and Delta.

Hyperparameter tuning is very important. In this project, it makes a difference of a 19% increase in accuracy. Extensive grid search used for hyperparameter tuning is quite time consuming.

These important lessons will be incorporated in future model building. Primary future work includes improving the quality and quantity of sample data, exploring effective features, more hyperparameter tuning on MLP, LSTM and CNN models, and looking at other deep learning classifiers.

References

[1] Solving the Problem of the Accents for Speech Recognition Systems. http://www.ijsps.com/uploadfile/2016/0628/20160628103620514.pdf

[2] D. C. Tanner and M. E. Tanner. Forensic aspects of speech patterns: voice prints, speaker profiling, lie and intoxication detection. *Lawyers & Judges Publishing Company*, 2004.

[3] H. Tang and A. A. Ghorbani. Accent classification using support vector machine and hidden markov model. *Advances in Artificial Intelligence*. Springer, 2003, pp. 629–631. https://pdfs.semanticscholar.org/dee5/eadd08480737277585c44485186fedc0c946.pdf

[4] V. Hautamaki, S. Siniscalchi, H. Behravan, V. Salerno and I. Kukanov, Boosting Universal Speech Attributes Classfication with Deep Neural Network for Foreign Accent Characterization. *INTERSPEECH 2015*. http://cs.joensuu.fi/~villeh/DeepAtributes.pdf

[5] Y. Jiao, M. Tu, V. Berisha and J. Liss. Accent Identification by Combining Deep Neural Networks and Recurrent Neural Networks Trained on Long and Short Term Features. *INTERSPEECH 2016 Native Language Sub-Challenge*. http://www.iscaspeech.org/archive/Interspeech_2016/pdfs/1148.PDF. [Accessed: 18- May- 2017].

[6] C. Blackburn, J. Vonwiller and R. King. Automatic Accent Classification Using Artificial Neural Networks. *Eurospeech*, vol. 93, 2017.

[7] Scikit Learn Package. http://scikit-learn.org/stable/

[8] Convolutional Neural Networks for Visual Recognition. CS231N Class Notes. Stanford University. http://cs231n.github.io/convolutional-networks/

[9] Emily Ahn, A Computational Approach to Foreign Accent Classification. *Wellesley Honors Thesis Collection.* http://repository.wellesley.edu/cgi/viewcontent.cgi?article=1507&context=thesiscollection