USING NEURAL NETWORK TO DETECT NATIVE VERSUS NON-NATIVE ENGLISH SPEAKERS

Kanan Schmid, Andrew Braun, Adam Belhouchat

UCLA ECE 114 – Speech and Image Processing. Professor F. Lorenzelli

ABSTRACT

In this project, we trained a neural network to differentiate between native and non-native English speakers given speech signals of the same sentence. We extracted MFCCs from each signal and fed them into a CNN and an RNN. Throughout the report, the analysis and accuracy of many different methods to compare speech will be discussed. Our best results were achieved using a CNN given mean and standard deviation of MFCCs and an RNN given a time sequence of MFCCs.

***Index Terms*—** neural network, CNN, LSTM, MFCC, LPC

1. INTRODUCTION

Neural networks have found widespread use in speech recognition. CNNs [1] and LSTMs [2] are two types of networks that have found some success. In this project, we used a data set of 170 speech signals, each of which is a different individual speaking the same sentence, to train a CNN and LSTM to classify native versus non-native English speakers and compare the performance of the two neural networks.

Our approach was to figure out what the defining characteristics of different accents are and extract these characteristics from the speech signal. We researched different features and characteristics that could effectively separate native and non-native English speakers. These features would then be extracted from the speech signal and fed into the neural net, which would classify the features as coming from a native or a non-native English speaker.

2. UNSUCCESSFUL TRIALS

In this section, we will briefly describe the many different features and analyses that were ineffective or less effective than the final features.

2.1. Speech normalization

One of the biggest hurdles in this project was ensuring that the same parts of the signals were being compared. For example, if we compare two different signals at 11 seconds in, it is unlikely they will be saying the same thing at that time. To get around this, there must be a way to either choose features that encompass the whole signal (like pitch, which should be relatively constant throughout), or find some way to normalize the signals. The most common way of doing this is dynamic time warping. Dynamic time warping compares two similar signals of different duration and attempts to normalize them in the time domain.

Our issue was that performing dynamic time warping on these long signals, some with over a million samples, was computationally impossible. MATLAB would need to generate a million by million matrix that would take up over a terabyte in memory. To get around this, we tried time warping short frames of the signal, each only a few milliseconds at most, but this would distort the frames to the point of being unrecognizable. Moreover, since we had more than two signals, it would be difficult to perform dynamic time warping that would make them all equivalent. We attempted to use Euclidean distance from the training data to three known English speakers that had non-noisy speech, different accents, and different voices (one male American, one female British, one male Irish), but unfortunately, this feature did not correlate well with actual accent and during training gave a testing error of 52%.

2.2. Vowel extraction

We attempted to use DARLA's vowel extraction tools [3] to extract vowel phonemes and formants from the speech files so we would be able to easily compare vowels across different accents. However, their vowel extraction tool was not consistent enough on our dataset. Due to the wide range in voice types and recording quality, it could not extract all vowels for each signal, even after denoising the speech files, and for some files it would completely fail to process. Therefore we could not obtain a set of all vowels and their formants for all speech files, so we could not make use of this tool in our final network.

2.3. Time-domain / FFT hybrid network

Another exploration we did was exploring the use of a hybrid network. Using raw FFT / Time-Domain features resulted in low testing error (50%, essentially guessing), as this would be

a very computationally complex task to do. The configuration of the network is given in Figure 1.

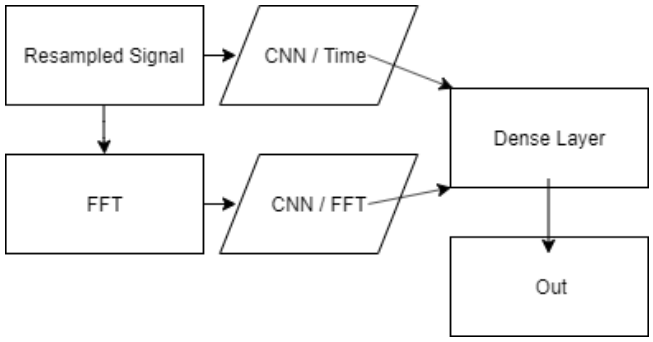

Fig. 1. Diagram of hybrid network. Various activation functions were explored in this architecture, however testing validation results were always low.

2.4. Filter bank energies and LPCs

Lastly, we attempted to use filter bank energies and LPCs in our neural network. Previous research has shown success using filter bank energies, or FBEs, for accent classification and speech recognition [4, 5], and LPCs have been used for speaker recognition [6]. However, when we tested them in our implementation we did not see any improvement in testing accuracy and occasionally even saw a decrease. We attempted to use FBEs both by themselves and in addition to MFCCs, as well as LPCs by themselves or with MFCCs and FBEs, but none of these attempts gave promising results.

3. MOST SUCCESSFUL RESULTS

3.1. CNN: MFCC average and standard deviation

Our decision to use MFCCs as a feature was based off previous speech research done in the field: MFCCs have been previously used for Gaussian Mixture Models and other speech research [5, 7]. We believed that non-native English speakers would be more likely to have a higher variation in MFCCs across each frame. In our implementation, the mean and standard deviation of the MFCCs over all the frames were fed into a simple feed-forward neural network and achieved testing accuracy (depending on the train / test split) anywhere from 58 - 69%.

The performance of this CNN was comparable to the LSTM / RNN implementation. However, this implementation was more susceptible to errors in labeling and quantity of positive and negative examples of native English speakers. Note that overfitting occurs past epoch 40, however implementation of an l1 / l2 regularizer did not sufficiently control this overfitting.

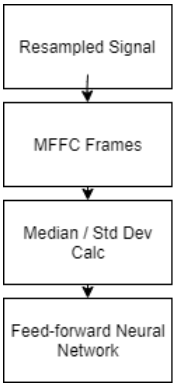

Fig. 2. A diagram of the neural network architecture.

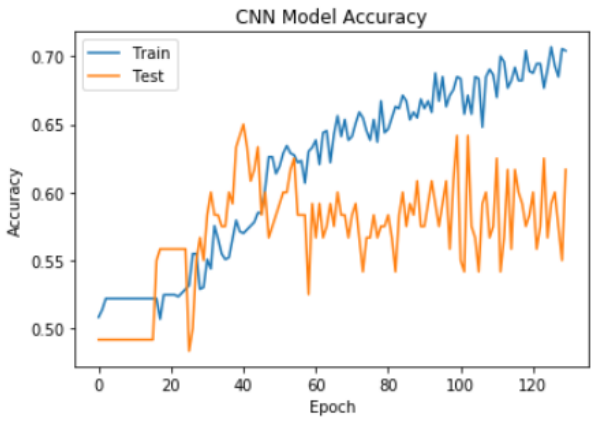

Fig. 3. A graph of averaged training error versus testing error over five different splits for the CNN implementation.

3.2. LSTM/RNN implementation with VoiceBox preprocessed features

This other implementation also resulted in a validation accuracy of 68%, but with a test accuracy of 58%. The features used were the features extracted from voicebox in the files `feat_vec.mat`, which contained MFCCs for all speech signals. For each frame of each signal, twelve MFCCs were calculated and stored in a large array. The MFCCs were then resampled so they would be the same length across all speech signals.

The VoiceBox features performed poorly in a simple feed-forward neural network, but performed well in this network. The performance of this is similar to the feed-forward neural network done previously. Note that overfitting doesn't seem to ever occur in this model, as the training accuracy seems to be bounded under 0.64.

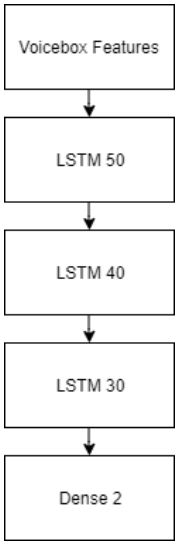

Fig. 4. Architecture of the LSTM RNN implementation.

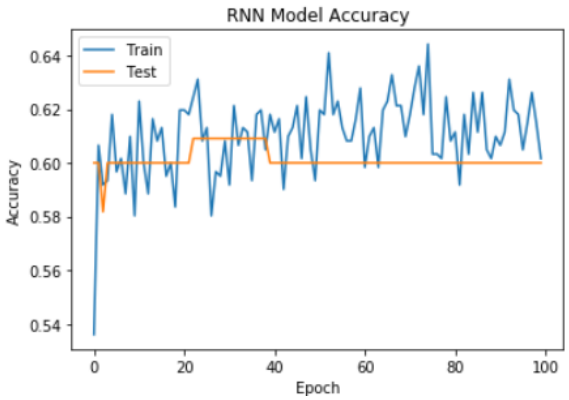

Fig. 5. A graph of averaged training accuracy versus testing accuracy over five different splits for the RNN implementation.

3.3. Test/training accuracies

The training and test losses and accuracies for both the CNN and LSTM are given in Table 1. From the table, we can see that testing accuracies for both types of networks were similar, though losses for the CNN were much lower than that for the LSTM.

4. LIBRARIES USED

For this project, we tried out many different tools and libraries to extract features from our speech signals.

`VoiceBox`: We used VoiceBox to generate features such as sF0 - sF4, Energy, SHR on different frames of the speech signal, MFCCs, filter bank energies, and LPCs. The features from VoiceBox performed poorly in a feedforward neural network but performed well in the RNN architecture.

Architecture	Training accuracy	Testing accuracy	Training loss	Testing loss
CNN	0.70	0.60	0.58	0.63
LSTM	0.62	0.60	2.17	2.16

Table 1. Table of training accuracies and losses for the CNN and LSTM.

`VoiceSauce`: We experimented with the formants, pitch, and B1-B4 features from Praat as well as H1, H2, and H4, A1, A2, and A3, cepstral peak prominence, and harmonic-to-noise ratio. However, none of these gave better performance than the VoiceBox features.

`fastdtw`: Calculation of overall Euclidean distance of one speech signal to another. Created distance features that were experimented with, but unfortunately, did not perform well enough as stated in Section 2.1.

`python_speech_features`: Used to create MFCC features (averages and standard deviations) that were used in our best-performing CNN.

`DARLA`: Vowel extraction tool from Dartmouth. Performs automatic and semi-automatic vowel extraction given a speech signal, a transcript of what was said in the signal, and some characteristics of the speaker. Unfortunately, the tool performed too inconsistently for it to be useful in our project.

5. FUTURE WORK

Overall, the models could have been significantly improved with more data. The project was restricted to use only the provided data set, which only had 170 signals, and also had inaccuracies in some classifications. Some of the speakers labeled as native clearly struggled to speak the words and had accents not associated with any primarily English-speaking countries. In other words, our data had "noisy-labelling", where some signals were labeled as English speakers but had slow speech and with a non-English accent. Additionally, since our dataset was rather small, more complex architectures would have been too difficult to train with such little data. In the future, we would add significantly more data (thousands or tens of thousands of speakers) in order to test these models and be more accurate in our labeling of native versus non-native English speakers. This which would reduce the impact of noisy labelling and allow for the exploration of more complex architectures.

We would also have done more research on how the different architectures worked. We tested many different configurations for the CNN and LSTM, but without a strong understanding of how they function we felt we were essentially making random guesses. We experimented with the number of layers, the number of units in each layer, where and how often we should apply pooling, and other features of the network, but none of our changes improved the performance of the network. Perhaps with a stronger background in neural

networks we would have had a more intuitive understanding of the neural nets and we would have been able to experiment more thoughtfully and figure out how to shape the network to achieve what we wanted.

Lastly, we would have looked into more robust and augmented feature extraction. The features we extracted were sensitive to the different lengths and sound qualities of the speech signals, so it is likely that our features themselves were very noisy. In the future, we would look into methods to normalize the speech signals so that the features are as comparable as possible. We explored this a bit with VoiceBox's noise reduction and voice activity detection methods, but it did not perform as well as we needed it to and did not result in better performance. We would look into more advanced noise reduction and voice activity detection as well as optimal ways to resample the speech signals so they are all the same length without significantly distorting the speech itself. We would also have looked into more state-of-the-art methods of processing features for speaker recognition. Recent research seems to look towards i-vectors and x-vectors as features extracted from the neural network and show promising results in speaker recognition [8, 9].

6. CONCLUSION

Overall, it seems that the RNN had more consistent performance than our CNN implementation. This is likely because RNNs consider the relationship of features over time while CNNs cannot learn these correlations as well. However, the CNN implementation significantly reduced complexity and was able to run more than 100 epochs in less than a minute, whereas the RNN architecture required an hour to run 100 epochs. We've demonstrated two architectures that can be used to achieve 60% testing accuracy on the dataset based on the MFCCs.

7. REFERENCES

[1] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, Oct 2014.

[2] Chung-Cheng Chiu, Tara Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Katya Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani, "State-of-the-art Speech Recognition With Sequence-to-Sequence Models," 2018.

[3] Sravana Reddy and James Stanford, "A Web Application for Automated Dialect Analysis," in *Proceedings of NAACL-HLT*, 2015.

[4] K. K. Paliwal, "On the use of filter-bank energies as features for robust speech recognition," in *ISSPA '99. Proceedings of the Fifth International Symposium on Signal Processing and its Applications (IEEE Cat. No.99EX359)*, Aug 1999, vol. 2, pp. 641–644 vol.2.

[5] Albert Chu, Peter Lai, and Diana Le, "Accent Classification of Non-Native English Speakers," .

[6] Shipra Gupta, "Application of MFCC in Text Independent Speaker Recognition," 2016.

[7] Shahenda Sarhan, Mohamed Abu Elsoud, and Nagham Hassan, "Text Independent speaker identification based on MFCC and Deep Neural Networks," *International Journal of Intelligent Computing and Information Science (IJICIS)*, vol. 15, 07 2015.

[8] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, May 2011.

[9] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN Embeddings for Speaker Recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5329–5333.