An intelligent speaker that recognizes your voice

Alvin Cao, Ege Gurmericliler, Emilien Pilloud

University of Washington

Abstract

In an era where intelligent home speakers are getting increasingly popular, having an integrated personal assistant can be a decisive factor for a client looking to purchase such a device. Speaker recognition is one important technology that can create a truly personalized experience in that regard. We have implemented a system that not only recognizes commands, as many personal assistants do, but also identifies the person who speaks, as to provide a service that is tailored to the speakers very own profile.

We have designed a system that, after a short calibration period that requires users to speak, can listen to user commands and identify the registered user who speaks. For the purpose of illustration, we have implemented two commands, which are "play my playlist" and "play something I like". Each user has a dedicated playlist of songs and a list of preferred genres stored in a local database. The playlist is hardcoded, but musical genres can be added on the go with the command ["I like" + genre].

Using past research, we have gathered multiple models that have shown to be appropriate for speaker recognition. Our work distinguishes itself from past speaker recognition research in that the final system is optimized to meet constraints that emerge from real-life situations. These constraints include, and are not limited to, requiring the smallest amount of calibration samples, predicting the speaker with minimal delay and dealing with ambient noise. We have used multiple tools such as deep learning models, noise reduction, sample rate conversion and feature extraction. Our comparative analysis focuses mainly on finding the right parameters for these tools, which help us reach our above goals in the best way.

Our implementation can be found at: github.com/Emilien-P/sonos-challenge

1. Objectives

At the beginning of the project, we wanted to understand what type of challenges might occur in environments involving the Sonos speaker. Because Sonos emphasizes on having multiple connected speakers, we believe that their core demographic is groups of people who often share a common physical space, such as families and coworkers. In such scenarios, having multiple people in a space should not prevent each of them to have a personalized interaction with the Sonos speakers. We think speaker recognition can help achieve a tailored user experience for every single individual in the room. Our main goal is thus to motivate what a user can do if the Sonos speaker can recognize who he/she is.

We first started searching for tools that might help us extract features from voice samples. Past research point out the particular efficiency of Mel-Frequency Cepstrum Coefficients, which we decided to use as basis for processing our input waveforms into vectorial data (see technical definition 1) that could be used for deep learning. Our comparative analysis includes the use of additional parameters such as delta coefficients appended to MFCC coefficients (see technical definition 2). Delta coefficients give information about the MFCC trajectories. Thus, we are expecting the delta coefficients to form new vectors that include more features about the speech, and hence our models to be more accurate if trained with these new vectors.

To keep our research close to real-life scenarios, we took into consideration the ambient noise that might occur in crowded spaces. Thus, we thought about integrating a noise reduction filter in our system. Common sense made us give this immediate decision, but we soon realized that many parameters needed to be taken into consideration. Although our ears might enjoy noise reduction at some well-known frequencies, the training models might prefer reduction at different frequencies. They also might prefer no reduction at all and work best with the raw samples. Our comparative analysis takes into considerations those points and determines whether noise reduction is truly necessary for our system. We are expecting to require noise reduction in order to avoid our models to learn wrong patterns that might emerge from noise during the calibration session, or conversely, inaccurately predict users because of additional noise that weren't present during calibration. A good implementation of a lowpass filter with a proper cutoff is proved to improve the acoustic performances[1].

Another well known real-life problem is the computation time of our system. In order to keep the user's attention and flow of thought uninterrupted, the response time of our application shouldn't exceed 10 seconds, and ideally be between 0 and 2 seconds [2] . It is also a natural initiative to always focus on providing the best effort on computation speed. The two main time consuming parts of our system are computing the Mel-Frequency Cepstrum coefficients and running the a prediction on our model. Both can take advantage of smaller inputs. A smaller input signal would induce smaller number of MFC coefficient computations. Consequently, the number of MFC Coefficients would be smaller, and hence the model would be trained with shorter vectors. To do this, we want to implement a sample rate converter on the input signal before computing the MFC Coefficients. We are expecting to analyze and determine an optimal sample rate such that we minimize computation time while not losing accuracy on predictions.

The speaker recognition challenge is not new, especially in the research area. Researchers have been using machine learning for a long time to classify speech features with techniques such as Support Vector Machine or Gaussian Mixture Models. During the last decade, deep learning networks, including long short-term memory networks, became increasingly popular. However, when it comes to applying those methods to a real life product, a new challenge arises. Most of the research use a high number of samples whereas we ideally want to train our model on as few samples as possible of a few seconds each. Indeed, we want our system to be as user friendly as possible and asking for more than 8 to 12 calibration samples per user seems too cumbersome.

In our project, we chose four different machine learning model approaches to see which one would be the most appropriate to our product. We selected the linear Support Vector Classifier both because it is a reliable and baseline method and that it can train on a little number of samples and still achieve a decent accuracy. Then, we selected three different architectures of neural networks that are shown on the figure below. The seqNN is a model that does not consider the dimensionality of the Mel Frequency coefficients matrix, i.e. it flattens the input before training. The convolutional neural network is an approach well-known for its success in 2D images classifying. The LSTM network is an implementation of the cells described in [3] which as a memory implementation that avoids the major flaw of Recursive Neural Nets. (i.e. an explosion of the back-propagation gradient [cite] as the time-domain dimension increases)

To remedy the low number of samples, we use a method of sample generation which consists in sliding a fixed-sized window over the time-domain of the Mel coefficients matrix with some overlap. It selects overlapping sets of frames and hence increases the number of samples with the same amount of data.

We used the Scikit library [4] to implement the SVM model and the Keras [5] wrapper for Tensorflow to implement our deep learning models. Deep learning models being very architecture dependent, we did our best to design the best architectures possible remaining fairly simple but we are not able to know if they are optimally suiting for speaker recognition.

1.1. Technical definitions

(1) To calculate the MFC Coefficients, we used the speechpy library[6] to calculate 13 MFC Coefficients per frame for 44100 Hz input signals. We selected mostly typical parameters in the calculation, but a deliberate choice was made to set the frame stride equal to the frame length (2 ms), resulting in zero effective frame overlap, which we believed to be more appropriate for the purposes of classification.

(2) The Delta coefficients form the first-order frame-to-frame difference of the Mel-Frequency Coefficients. This provides the trajectories of the coefficients, hence appending delta coefficients to the MFCC vector should provide additional information about the dynamics of the MFCC. Delta coefficients can be found with the following equation:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{1}$$

2. Results

2.1. Machine Learning model selection

We implemented the four aforementioned machine learning models and we got different performance rate for each of them. We used the data from the CMU Festvox arctic database [7] which consist in 7 different speakers male and female saying several seconds long sentences from the open-source Gutenberg project. The bank contains around 500 hundred sentences of 2 to 5 seconds per speaker. The number of samples in the all the figures below refers to the number of samples taken per user for the training. The testing was made on other samples unused for training, typically around 50 testing samples per speaker to get a relevant testing accuracy score.

Unsurprisingly, we could firstly see in figure 1 that the LSTM model is the most performing model for our range of original sample number of interest. It has above .9 accuracy level from 8 samples on with the sample extension feature on. LSTM networks are well-known for their ability to classify sequences such as audio as they have memory. The seqNN architecture is the overall least accurate which is also reasonable given that it loses the time dimensionality information whereas CNN still keep some 2D locality features information.

In figure 2, we can see that all models do converge to a very high accuracy if we train them on enough data (similar accuracy from 150 samples onwards). It is a good sign that our models are not flawed.

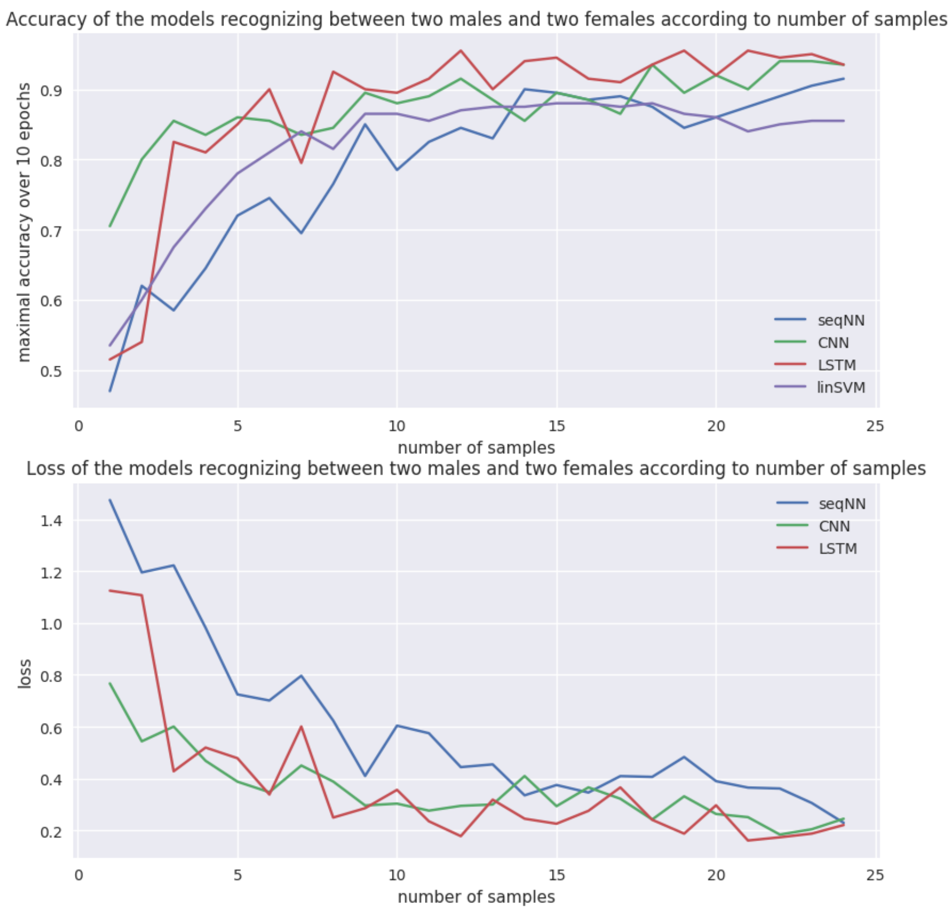


Figure 1: Accuracy of the models with sample extension

For small numbers of samples, we can see that the models are somewhat volatile. An important thing to notice is that both CNN and LSTM achieve significantly higher accuracy (both above 80 percent at 3 samples compared to 70 percent for linSVM and 60 percent for seqNN). All four models achieve higher accuracy as the number of samples increase, yet CNN and LSTM appear to be better even then (95percent for both compared to 90 percent and 85 percent for seqNN and linSVM respectively).

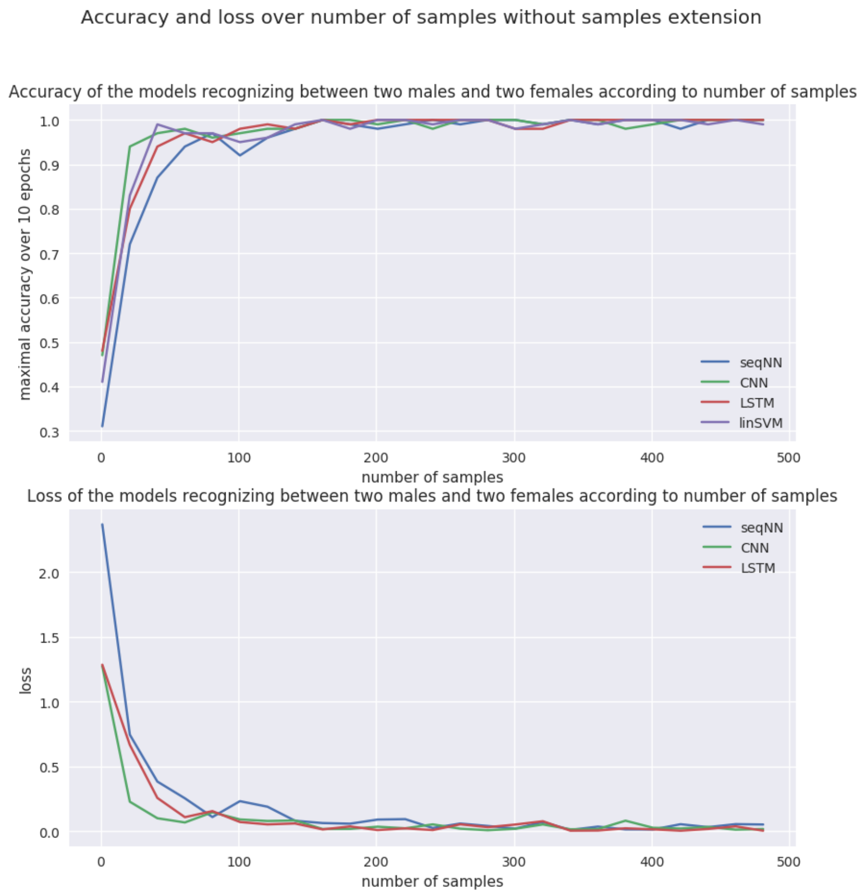


Figure 2

We could also conclude that our overlapping window sample extension method is useful in some cases as seen on the figure 3. The dotted lines represent the models trained on the original samples and the full lines are models also trained on the extra generated samples from windowing. The models trained on the extended data all have a higher tested accuracy around our number of samples of interest. In fact, they all start and remain at high levels of accuracy (above 75 percent). We can also see that windowing reduces volatility in small number of samples (overall between 75 and 95 percent with windowing, compared to 51 and 91 percent without).

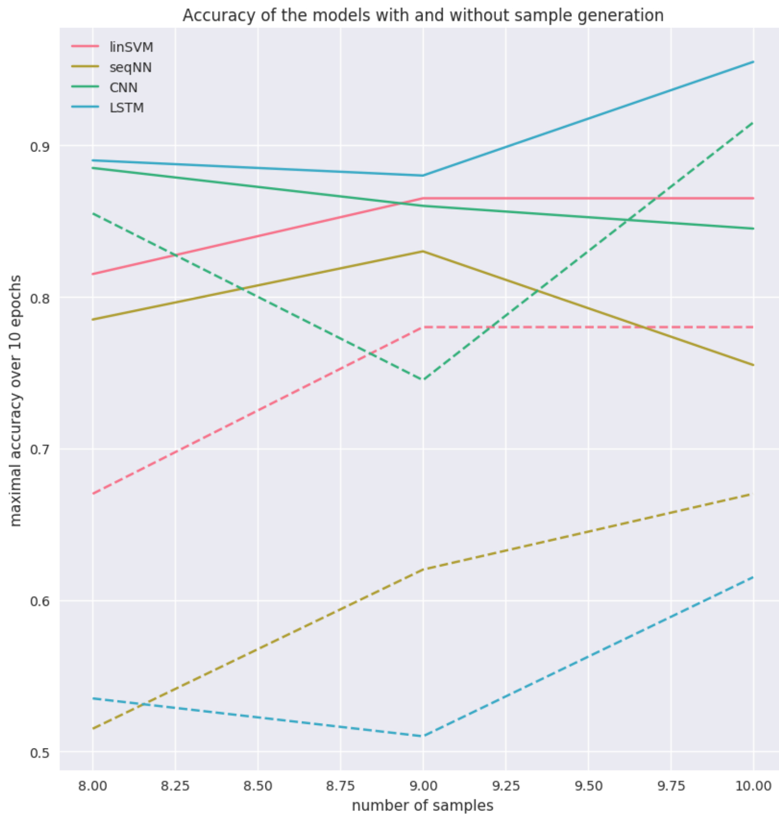


Figure 3

In figure 4, we plotted the accuracy of the models for different number of speakers from the arctic dataset. The experience was done three times and the bold line stands for the median accuracy result obtained. All the accuracy levels were measured for 10 training samples per speaker with the sample extension activated and 50 testing samples per speaker. We can notice once again that not only is LSTM more accurate for any number of speakers, but it also is more robust to an increase in the number of speakers. We can see the flaw of our baseline method, linear Support Machine Classifier, whose accuracy drops drastically as the number of speakers increases.

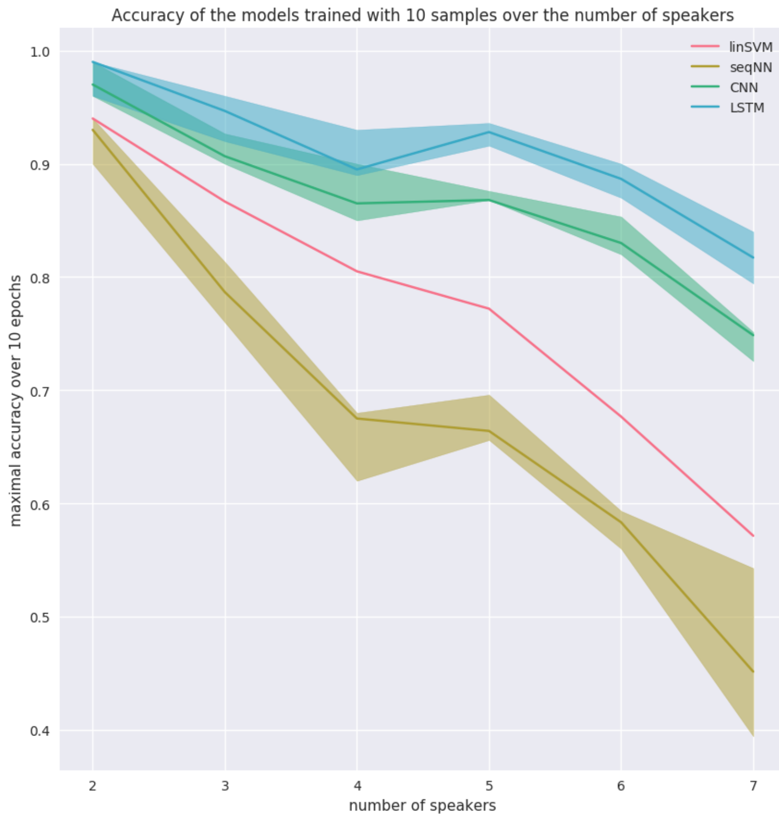
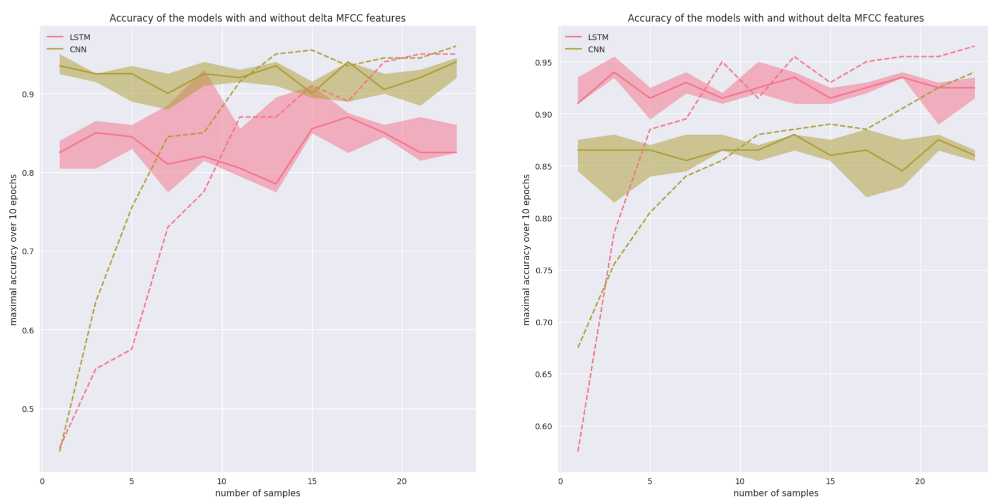


Figure 4

To put it in a nutshell, the models have all different performances. The linear Support Vector machine seemed to be the less volatile due to its deterministic training process whereas the performances of the neural networks were more variables. The SVM could also train on less samples than the deep learning approaches which needed a more significant training set to be accurate.
In our final build, we implement a LSTM model as it seemed to be the best performing one and we used it as the reference method for the DSP testing.

2.2. DSP results

We implemented several DSP augmentations to our system to try to achieve a better classification accuracy. First, we tried to append the delta Mel coefficient matrix to our vector of features. We got some interesting results as seen on the graphs below. we plotted the accuracy to distinguish between four people, two male two female, of our two best models over the number of training samples. The dotted lines represent the results without the delta coefficients and the plain bold lines are the median of three same experiment with the delta inputed as features. The testing was done with 50 test samples per speaker.

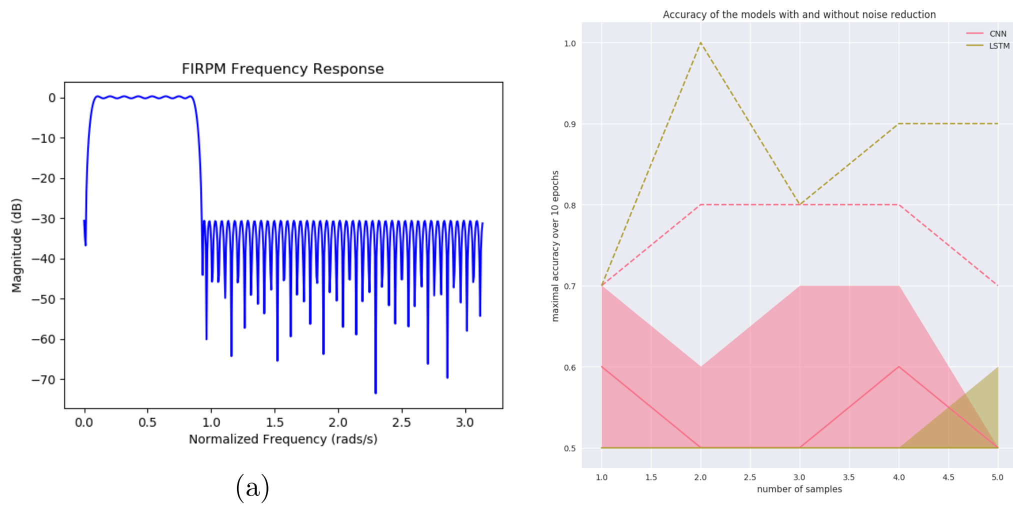


(a) Accuracy without sample extension (b) Accuracy with sample extension

Figure 5: Accuracy with vs without delta coefficient

We can see that the delta coefficients greatly increase the accuracy of the models for very low number of samples. CNN without delta nor sample extension is below .7 of accuracy for a set of training sample less than 5 per speaker whereas with delta features added it is above .9. LSTM without delta nor sample extension is below .8 of accuracy for less than 9 training samples per speaker whereas it is always above .8 with the delta features included. With sample extension, the results are similar but we notice once more that LSTM starts performing better from the 13th sample. CNN with delta also decreases in accuracy: its median score was above .9 without sample extension and between .8 and .9 with sample extension. It follows that those added features describe some voice features different from the Mel coefficients and which characterize well the speaker. However, we can see that the models trained without delta coefficients tend to be better performing as the number of samples increases. Indeed, in figure (a) CNN without deltas outperform CNN with delta after sample number 12 and LSTM after sample 10. Our theory is that our neural network architectures are not optimal and that some over-fitting occurs if the vector of features is too big. We can also notice one more time that without sample extension, CNN is having better performances whereas LSTM is better with that feature enabled.

Another interesting result is the effect of noise reduction. Using the SciPy library[6], we designed a finite impulse response filter using the Parks-McClellan algorithm to filter out ambient noise. Human vocal frequency most commonly lies between 300 and 3000 Hz [8], so our filter acts as a bandpass that attenuates frequencies outside this range with a passband transition width of 250 Hz. However, while the audio output from this filter sounds noticeably better than the original to our ears, our tests show that the integration of such a filter dramatically reduces the accuracy of our models.



(a)

(b)

Figure 6

The dotted lines represent the models trained with unfiltered voice samples, and the straight lines represent the models trained with noise reduced voice samples. We think that, while trying to filter out the ambient noise, the filter might have removed important features that the model uses to distinguish users. Additionally, if our test samples were taken in different conditions (hence different ambient noise), the reduction might have shown to be more efficient. We also notice that what our ears seem to like might not necessarily be aligned with what the models would require for optimal predictions.

Finally, we tested down sampling with the goal of reducing computation time, as motivated in the objectives part. A simple down sampling by a factor of 2 gave us an important decrease in accuracy, especially for small number of user voice samples.

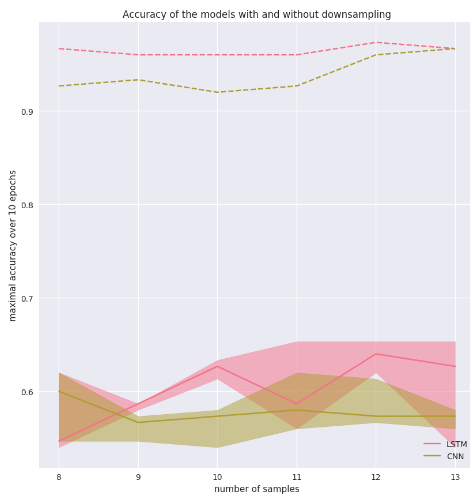


Figure 7

In figure 6, the dotted lines represent the accuracy of the prediction with the original voice inputs. The full lines show the accuracy with down sampled voice inputs. We can clearly see that keeping the sampling rate unconverted leads to better predictions (above 90 percent for original inputs, compared to 60 percent for down sampled inputs). We assume this loss in accuracy is due to a loss in characteristics of the human voice samples when we down sample the signal. Moreover, the MFC vectors here have half as many coefficients compared to original signals. The smaller quantity of information provided to the training models might also have an impact in the overall decrease in prediction accuracy. Because of this, we have decided to discard sample rate conversion in our final system.

3. Conclusion

Our speaker recognition system demonstrated decent accuracy and acceptable latency when provided with a set of ten 3-second voice samples. Our results were particularly good using the public dataset of voice files. We discovered that LTSM was the best choice out of the four models we tested, and optional features like delta features and sample extension were situationally useful. However, additional modifications would be needed to achieve similar performance in practical environments.

Complications unique to real-world environments turned out to be among our greatest challenges while trying to improve the performance of our system. In particular, we've learned that the reliability of MFC coefficients appear to be highly susceptible to ambient noise, and distinguishing similar-sounding voices can be difficult for a classifier. Extensions to our system should focus on improving its real-world performance, either in accuracy (noise reduction, classification) or usability (latency, samples required, unique users supported).

There are several modifications to our system we think are worth pursuing. For example, we saw that incorporating delta features in the MFC coefficients improved performance when training with few samples. Because of this, we think that perhaps incorporating delta-delta features, also known as acceleration features, might boost accuracy even more.

Also, since our bandpass noise reduction filter did not help the accuracy of our model, we've learned that noise-reduced audio signals that sound better than the original to our ears are not necessarily better for training classifiers since important vocal features may have been too heavily attenuated. A gentler FIRPM filter or more comprehensive tuning might resolve this, but we suggest trying an adaptive filter to take advantage of the fact that real-world samples will likely have the same ambient noise throughout the recording.

With modifications like these as well as careful consideration regarding the trade-offs between accuracy and usability, our speaker recognition system could very well have a level of robustness ready for practical applications.

4. Bibliography

[1] J. K. MacCallum, A. E. Olszewski, Y. Zhang, J. J. Jiang, Effects of low-pass filtering on acoustic analysis of voice, J Voice 25 (2011) 15–20. 20346621[pmid].

[2] R. B. Miller, Response time in man-computer conversational transactions, in: Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, AFIPS '68 (Fall, part I), ACM, New York, NY, USA, 1968, pp. 267–277.

[3] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[5] F. Chollet, et al., Keras, 2015.

[6] A. Torfi, SpeechPy: Speech recognition and feature extraction, 2017.

[7] J. Kominek, A. W. Black, V. Ver, CMU Arctic Databases for Speech Synthesis, Technical Report, 2003.

[8] Y. Hu, P. C. Loizou, Subjective comparison of speech enhancement algorithms, in: 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, volume 1, pp. I–I.