Text Independent Speaker Authentication in Noisy Environments

MLND Capstone Project

Abdullah AlOthman

March 2018

1 Definition

1.1 Project Overview

Speaker Authentication (or Verification) is the process of verifying the identity of a speaker based on the characteristics of their voice. This problem falls in the biometrics domain and there has been previous attempts to solve it using supervised learning [1], unsupervised learning [7] and of course using deep learning [5].

In this project, I'll attempt to solve this problem using a deep neural network model trained on the VoxCeleb dataset [5].

1.2 Problem Statement

Most implementations of speaker verification systems are dependant on specific phrases (e.g. " Hey, Google..", "Hey, Siri..", "Alexa..") and consistent audio-environment (same microphone, audio quality, etc). What I want to tackle in this project is text-independent speaker verification, in noisy, uncontrolled environments.

Applications of speaker verification include:

- authentication in high-security systems
- authentication in forensic tests
- searching for persons in large corpora of speech data

All of these applications require reliable performance in 'wild' conditions (i.e. real-world scenarios). The challenges expected in this task come from two main fronts:

1. High variance in the environment (background music, crowds, recording quality, etc.)

2. High variance in the speaker age, accent, emotion, intonation, etc.

I will attempt to solve this problem by building a neural network model that will take as input the features we've extracted from two speaker-audios and outputs the probability of a speaker match.

1.3 Metrics

To evaluate the model, I'm choosing Equal Error Rate (EER) as the performance metric. For each authentication pair the model outputs a continuous variable X, which is the estimated probability of a match.

Given a threshold parameter T, the instance is accepted as a match if $X > T$, and rejected otherwise. X follows a probability density $f_1(x)$ if the instance is actually a match, and $f_0(x)$ otherwise [6].

The false rejection rate is given by

$$FRR(T) = 1 - \int_T^\infty f_1(x)\,dx \tag{1}$$

and the false acceptance rate is given by

$$\mathrm{FAR}(T) = \int_T^\infty f_0(x)\,dx \tag{2}$$

.

EER is defined as the rate at the threshold T where $FRR(T) = FAR(T)$ (see Figure 1). EER is commonly used as a performance metric for identity-verification systems, as it balances the rate at which false-acceptance and false-rejection occurs. The lower the EER value, the higher the accuracy of the system. EER is also the metric used by for our benchmark model.

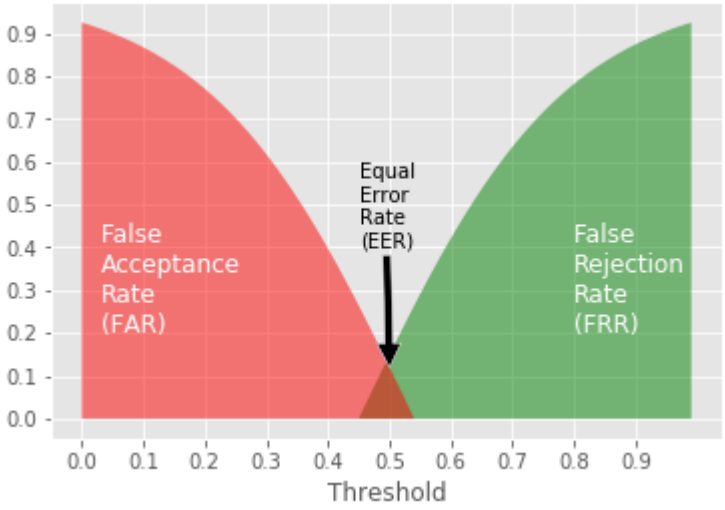

Figure 1: EER Visualized in an Example

2 Analysis

2.1 Data Exploration and Visualization

for this problem, I'm using the VoxCeleb dataset[5]. VoxCeleb dataset was created by extracting audio from YouTube videos. it contains over 100,000 utterances for 1251 celebrities. The speakers are gender balanced and span a wide range of different ethnicities, accents, and ages.

This dataset includes varying audio qualities and challenging acoustic environments, such as:

- red carpet interviews
- outdoor stadium
- studio interviews
- speeches with large audience
- high production multimedia
- amateur multimedia (e.g. Vlogs)

The dataset also includes uncontrolled, real-world noise such as background-chatter, audience reactions, room acoustics, etc. Table 1 gives some idea of the distribution contained in the dataset.

In Figure 2, you'll see the nationality distribution of the speakers in this dataset. Which gives incite into the distribution of ethnicitis and accents.

number of speakers		1251	
number of male speakers		690	
	max	average	min
videos per speaker	36	18	8
utterances per speaker	250	116	45
length of utterances (seconds)	145.0	8.2	4.0

Table 1: dataset statistics

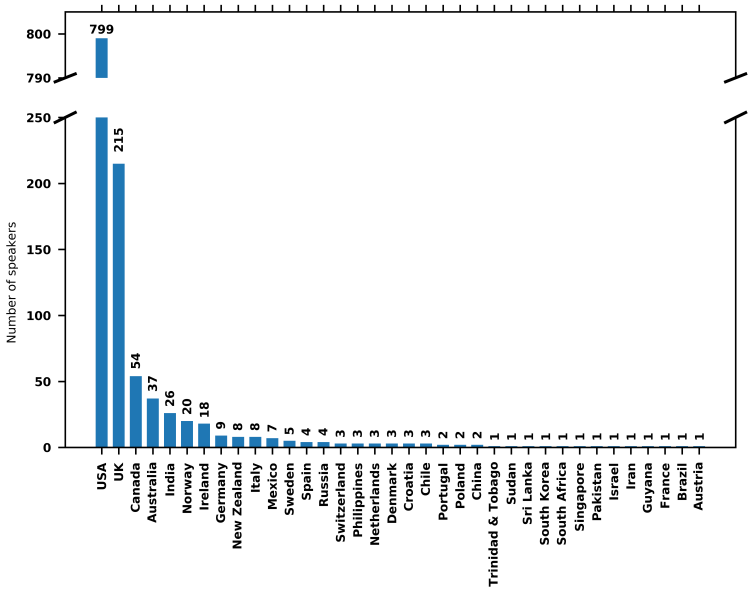

Figure 2: Nationality distribution

2.2 Algorithms and Techniques

Since speech is inherently temporal, I decided to take that into account and chose a Recurrent Neural Network (RNN) as the basis of my model.

The big drawback that one might face when using neural networks in general is not having enough data, which fortunately is not an issue in this case since our dataset is large.

RNNs are a type of neural network that take into account the temporal relation between the input features. There is an inherent relation between elements of time-series data that a regular neural network cannot grasp. RNNs behave in a way where they 'remember' the previous inputs at each step of working through time-series data, which makes them more suitable for this problem since latent audio features (such as intonation and emotion) come in a sequence and cannot be captured when considering each time-step independently.

Vanilla RNNs however, face a problem with vanishing gradients, back-propagation through each time-step carries less and less information which restricts the scope of training.

we're using more powerful variants of RNN, namely GRU and LSTM which solve this problem by introducing the concept of gates. The main types of gates are:

1. input gate: which determines what new info it should take into consideration
2. forget gate: determines how much of the past information it should 'forget'
3. output gate: determines how much information to pass along

To learn more on the math behind RNNs and their various configurations, check out the excellent paper by Rafal Jozefowicz et al.[3]

To be able to verify, you must first be able to identify. The first step I will do is train an identification model that takes as input features extracted from a given audio file and outputs the probability of who it thinks the speaker is.

Once the identification model is trained, I'll use it as a base for the verification model, which will take audio features extracted from 2 audio files as input and outputs the probability of a speaker match.

The Architecture of both models can be seen in Figures 4 & 5 and will be discussed in further detail later on.

2.3 Benchmark

I'll be comparing my model to multiple benchmarks.

1. Random Classifier: the most basic benchmark, but important to know we're preforming better than random chance.
   - EER = 0.50
2. The GMM-UBM model proposed in the VoxCeleb paper.
   - EER = 0.15
3. The Embedding model proposed in the VoxCeleb paper.
   - EER = 0.08

3 Methodology

3.1 Data Preprocessing

since the audio length is not consistent, I only considered the first 3 seconds of each utterance as inputs to the model.

for feature extraction I used MFCC. Mel Frequency Cepstral Coefficents (MFCCs) are the state-of-the-art method for extracting audio features for speech and speaker recognition tasks. it is applied by first framing the signal into overlapping frames (standard parameters are 25ms length with 10ms overlap). Then applying log filterbank on periodogram estimate of the power spectrum. Then finally taking the Discrete Cosine Transform (DCT) coefficients of the log-filterbanks.[4]

Given the noisy state of the dataset, I then normalized with respect to the mean and the variance to get the CMVN features, Figure 3 shows how normalization boosts the signal to noise ratio.

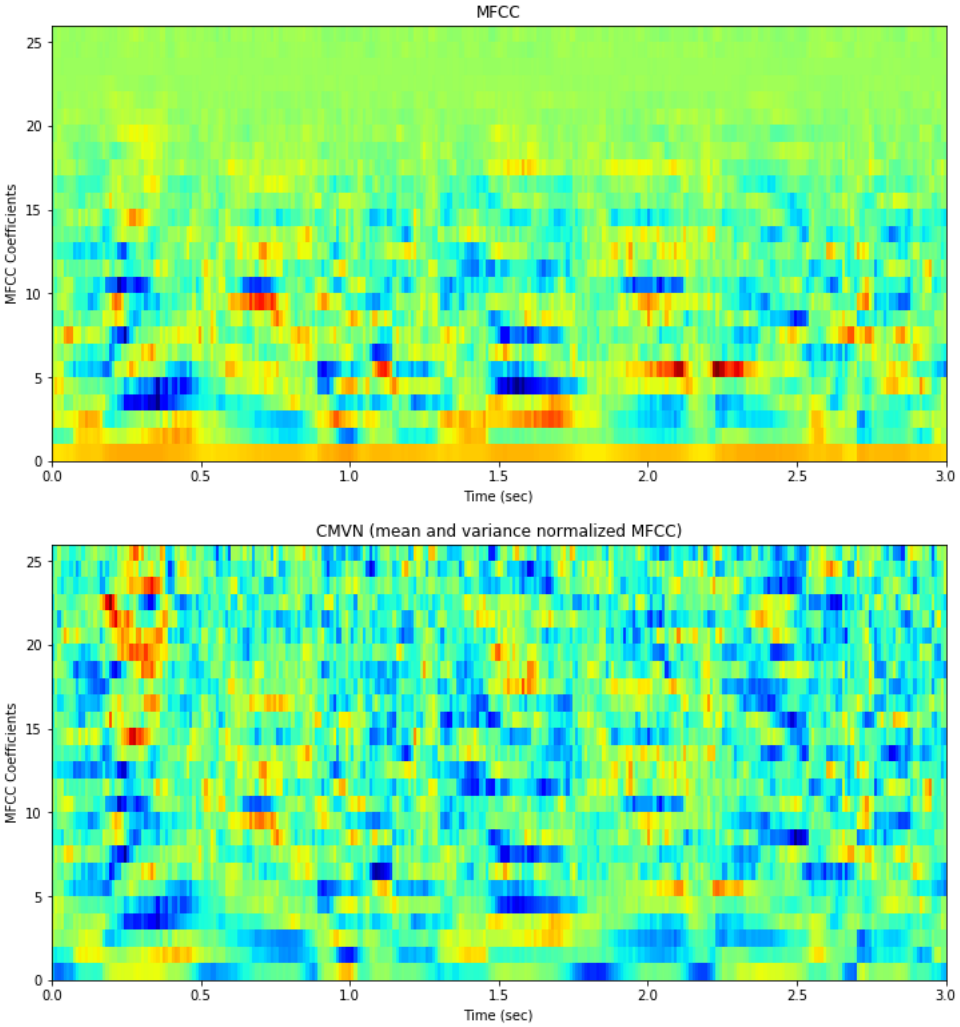

Figure 3: MFCC before and after normalization

3.2 Implementation

I first attempted to tackle the verification task directly using a convolutional neural network. That effort proved fruitless as I got results that were worse than random. I then started working on an identification model, using the same train/val/test split used in assessing the base model. I attempted to use different features such as the signal's spectogram and the log-filterbank but they were computationally expensive to extract and store, which is how I finally settled on using MFCCs.

I had no luck with CNNs even in the identification task. So I used the same architecture that I've been using in Kaggle's Toxic Comment Classification Challenge even though they operate on different data types (text/audio).

And to my delight, the model actually preformed well! (see table 2).

The identification model's architecture is as follows:

- input layer
- Spatial Dropout layer: prevents the neural network from over-fitting by deactivating feature maps with a given probability
- bidirectional RNN: takes into account the temporal relation between the input features (the value of X_t affects the value of X_{t+1})
- concatenated global max pooling and global average pooling
- Dropout: prevents the neural network from over-fitting by deactivating neurons with a given probability
- Batch Normalization: forces the activations of the previous layer to have a gaussian distribution.
- output layer with Softmax activation

See figure 4 for the architecture and table 2 for the results of the identification model.

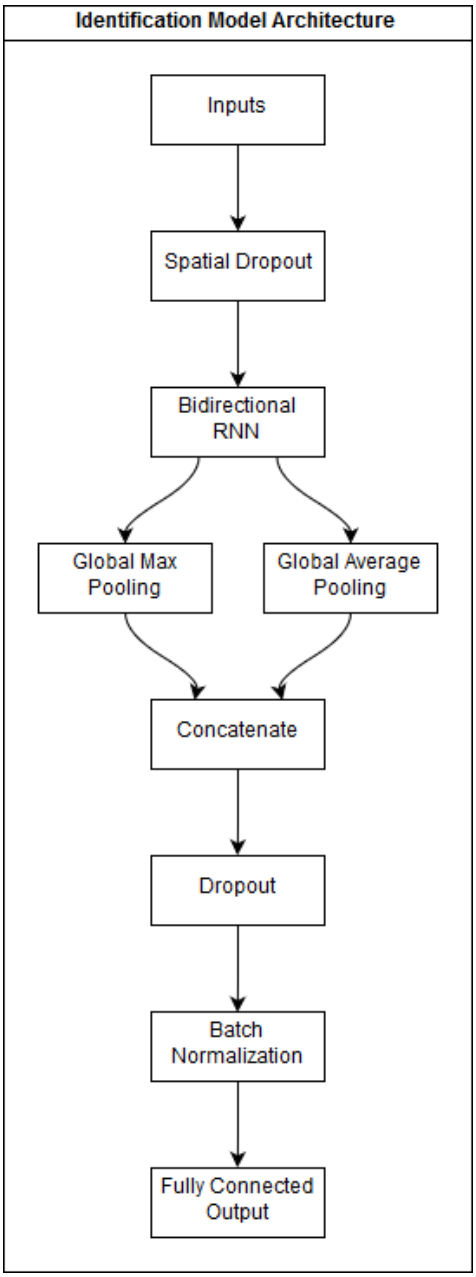
Identification Model Architecture
Inputs
Spatial Dropout
Bidirectional RNN
Global Max Pooling
Global Average Pooling
Concatenate
Dropout
Batch Normalization
Fully Connected Output

Figure 4:

Model	Top - 1	Top - 5
GRU - MFCC	0.56	0.76
GRU - CMVN	0.66	0.84
LSTM - CMVN	0.68	0.85

Table 2: Top-1 and Top-5 Accuracy of the identification models

For verification, I first created training pairs by creating balanced positive samples and negative samples for each utterance, where utterances must come from different video sources to be in a pair (for added difficulty). I then reserved a set of speakers for testing, these speakers are not included in the training set for either the verification or the identification models. After that I froze the weights of the identification model and used them for my verification architecture (see figure 5). I used ReLU activation for the middle Dense layers and Sigmoid activation for the final layer

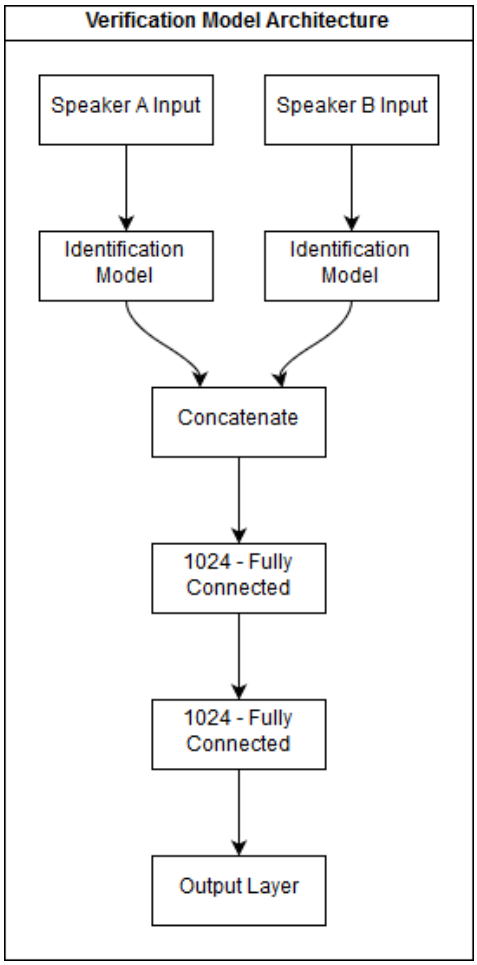

Figure 5:

3.3 Refinement

Moving from CNNs to RNNs gave the most dramatic improvement. Beyond that, table 2 shows the improvements of using CMVNs instead of MFCCs and the marginal improvement of using LSTM over GRU.

As for Hyperparameter tuning, using Adamax as the optimizer helped the model converge faster. As opposed to the default RMSProp optimizer which seemed to somehow preforme worse than random with an EER of (0.51). Combining the choice of optimizer with high dropout rate (0.5) and batch normalization helped prevent the model from overfitting at an early stage and achieve better results.

3.4 Coding Challenges

most of the coding challenges faced during this project were solved by framing the exact problem in mind, then diving deeply into Keras's[2] or numpy's documentation.

4 Results

4.1 Model Evaluation and Validation

The final model results can be seen in table 3.

Since the test set contains no overlap in terms of videos or speakers with the training set, I would say that the model is generalizing well with new data and that these results can be trusted.

Model	EER
GRU-based siamese network	0.15
LSTM-based siamese network	0.14
using ensemble blend of both	0.13

Table 3: results of the verification models

4.2 Justification

The final solution is described in full in section 3.2.

The final results (as seen in table 3), are better than 2 of the 3 benchmarks chosen for comparison. The model is robust enough for the uses proposed in section 1.2. given more data as input, using longer utterances for example will reduce the likelihood of miss-classification exponentially.

5 Conclusion

In this project, a recurrent neural network that determines whether two audio segments come from the same speaker has been developed. As it currently stands the model can still be used in production as mentioned in the previous section. The model exceeded 2 of the 3 proposed benchmarks which means that while the model has preformed well, there's still room for improvement.

5.1 Free-Form Visualization

In figure 6 we see the difference parameter tuning can make and the importance of choosing an appropriate optimizer for the model.

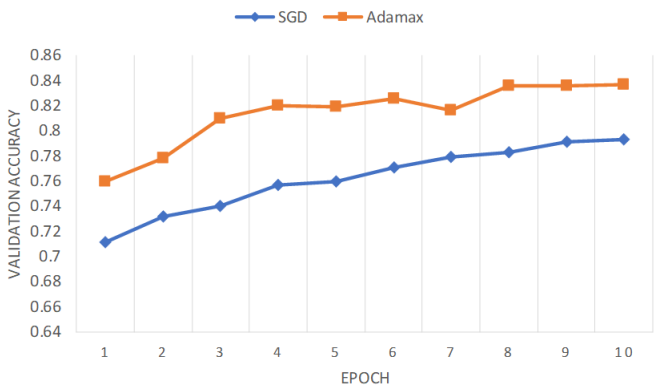

Figure 6: comparing Adamax and SGD optimized models' validation accuracy for the first 10 epochs

5.2 Improvement

due to time and computational resource constraints, I've not been able to experiment with some of the approaches I have initiallly considered.

Possible venues of improvement include:

- Instead of building the training set randomly, consider using hard negative sampling for a more robust model.
- Use segments longer than 3 seconds, or use multiple overlapping segments from each utterance.
- Invistigate different possible feature representations. such as i-vector, d-vectors and full spectograms.
- Revisit CNNs, as I feel like I'm missing something here that will become clear with more research.

5.3 Reflections

I started this project with no prior knowledge in the world of Bio-metrics or audio processing. Working in this new domain strengthened my research skills in ways that will help me with my future projects. I grew more familiar with the libraries and tools available to me. working with hardware constraints forced me to get creative in writing my code to be more memory efficient.

References

[1] William M. Campbell, Douglas E. Sturim, and Douglas A. Reynolds. Support vector machines using gmm supervectors for speaker verification. *IEEE Signal Processing Letters*, 13:308–311, 2006.

[2] François Chollet et al. Keras, 2015.

[3] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. *Journal of Machine Learning Research*, 2015.

[4] James Lyons. Mel frequency cepstral coefficient (mfcc) tutorial, 2013.

[5] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[6] Receiver operating characteristic. Receiver operating characteristic — Wikipedia, the free encyclopedia, 2018. [accessed 15-March-2018].

[7] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10:19–41, 2000.