Intelligent Voice Identification with Neural Networks

CS 230 Winter 2018 Final Project
Category: Speech Recognition

Lloyd Maza `lmaza` | Michael Thompson `mt1` | David Troner `dtroner`

Introduction

In the modern era, automated speech recognition systems have become commonplace in a broad range of technology domains. While considerable attention is devoted to challenging tasks like speech-to-text transcription or voice modulation, we focused on the slightly simpler problem of identifying speakers based on their voice. The ability to distinguish a speaker's identity among a group of people has considerable practical use, including automated air traffic control systems, identity verification for over-the-phone bank transactions, and speaker diarization for speech-to-text applications.
For the purposes of this project, we will look at the task of distinguishing a celebrity's voice from among a small group of 17 speakers. This is a well-studied problem with a multitude of different known solutions, so we leveraged as much insight as possible from preceding research. Upon implementing a baseline voice identification system, we devoted considerable time to refining the model in order to maximize performance in the context of our specific problem.

Related Work

As a framework from which to build up our implementation, we started from a foundational approach outlined by Ge et al. in their paper on neural network-based speaker classification [1]. In this article, they demonstrate the efficacy of a standard multi-layer perceptron in identifying a single person's voice from a group of about 200 speakers. Despite having considerably less data, we we were able to achieve a similar level of performance by collecting substantially more data per person. Additional related work is discussed in the dataset section.

Dataset

As mentioned earlier, the goal of this project is to distinguish the voices of 17 people, hence the dataset included audio clips of each person speaking. To make training easier, the audio clips had each person speaking individually without excessive background voices or noise. For example, a training audio clip might have either Donald Trump or Barack Obama giving a speech, but the applause or crowd noise would be excluded. As expected, the audio clips were simply labeled according the identity of the speaker; no text transcripts were used.
The duration of audio for each training example is usually considered a hyperparameter of the neural network. Some typical durations for each training example are 1 second for Lukic et al. [2], 0.8 seconds for Torf et al. [3], and 0.1 seconds for Ge et al. [1]. Because

the architecture of our neural network most closely resembles the network used in Ge et al., each of our training examples was 0.1 seconds in duration.
The amount of data used in training also depends on the network implementation. While some networks use massive datasets, such as Li et al. which used 12,600 hours of audio to classify 250,000 speakers [4], more normal dataset sizes include Lukic et al. [2] and Ge et al. [1]; both used about 25 seconds of audio per speaker. Our dataset contained 10 minutes of audio per speaker, more audio per speaker than all previously mentioned neural networks. Because we only classified 17 speakers, we were able to use much more audio per speaker without excessive training times. Approximately 95% of our data was used for training, with 2.5% used for the development set, and 2.5% used for the test set. This gave 9.5 minutes of audio per speaker for training, and 15 seconds for dev and test.
Gathering the necessary audio clips was done by downloading mp3 audio from YouTube videos. Audio data was mostly collected from famous people or television characters. Examples include United States Presidents Donald Trump and Barack Obama, and Michael Scott from *The Office* television show.

Data Preprocessing

Since the method we implemented makes use of standard neural network architecture, all audio data had to be processed into a feature vector which could be fed into the input layer. To this end, we followed a basic feature extraction method outlined by Ge et al. It should be noted that the same feature extraction process was be employed for both training and test sets to ensure that all data was drawn from the same distribution.
First, the audio was normalized to unit amplitude to ensure consistency across examples. Next, we used voice activity detection (VAD) to reduce the audio so it only contained voiced speech. This removed some of the variance between example audio clips and ensured we would only classify voiced speech. The VAD algorithm employed a high classification threshold to make absolutely sure the input data consisted entirely of human speech without irrelevant noise. Our literature review suggested that this would improve the performance of the model while reducing training time by removing unnecessary portions of the data.
The most important part of the data processing was the actual feature extraction which converts an audio file to a feature vector. For this purpose we used Mel-frequency cepstral coefficients (MFCCs), which are generally regarded as the de-facto standard feature set used for speech recognition tasks. The audio was divided into 10 ms long frames, from which 13 MFCCs were extracted. We also included the first and second derivatives of these MFCCs with respect to time to account for any sort of temporal component in the feature set.
Note that each training example consisted of a "window" of 10 frames stacked together. Between adjacent windows, a stride of 3 frames was employed, meaning each example contained 70% overlapping data. This sliding window technique is illustrated in Figure 1 below.

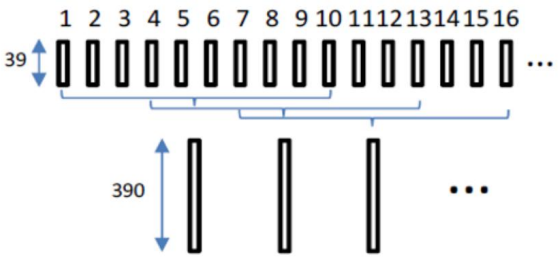

Figure 1: Sliding window approach to MFCC extraction

It should be noted that a drawback of MFCCs is that they eliminate voice-specific characteristics of the speech signal, resulting in a potential challenge for our voice identification system [2]. In an effort to ameliorate this, we normalized each speaker's MFCCs by his/her own mean and variance. This process re-introduced some of the idiosyncrasies of each speaker's voice, resulting in an easier classification task.

Neural Network Architecture

The neural network architecture outlined by Ge et al. utilizes a single hidden layer of 200 nodes along with a sum of multiple binary classification losses at the output layer of 200 nodes. However, this was determined through a grid search of at most 2 hidden layers and 400 nodes. Our approach sought to improve on this model by adding network depth beyond 2 hidden layers along with a softmax output layer and corresponding usage of categorical cross-entropy loss.
All layers aside from the final softmax layer utilize ReLU activations with a consistent number of nodes per layer. The softmax activation includes 17 classes corresponding to the 17 speakers in the dataset. Following the hyperparameter tuning process (to be discussed), the resulting neural network consisted of 7 fully-connected layers with ReLU activations, each with 390 nodes per layer, followed by a softmax output layer. This architecture is shown below in figure 2.

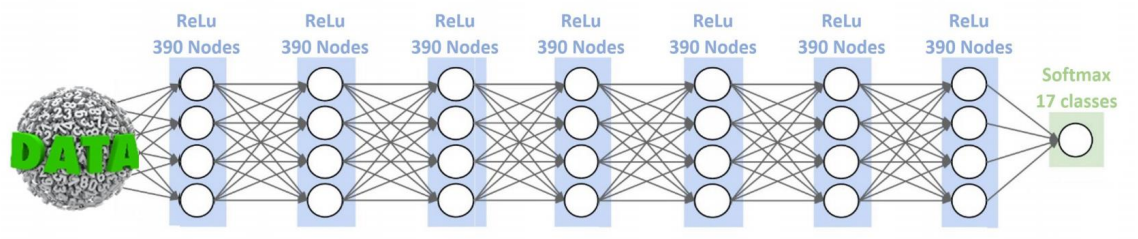

Figure 2: Neural Network Architecture

It is notable that an RNN is not utilized, since the speaker recognition task is text-independent and therefore sequence data is not required. Another framework common in voice identification tasks utilizes a convolutional network on spectrogram images created from audio segments, as demonstrated in [2]. However, with the relatively low dimensional input vectors in the dataset and ease of data preprocessing, the fully-connected layer approach is utilized for this project.

Training

The network was implemented in Keras and incorporates L2 regularization and Adam optimization over mini batches to improve performance and computation time. Batch normalization on each hidden layer is utilized. Training was run for 25 epochs with early stopping. A summary of the key methods and hyperparameters used during training (following hyperparameter tuning) is detailed below in Table 1.

Table 1: Network Hyperparameters after tuning

Implementation	Hyperparameters			
L2 Regularization		$\lambda = 0.7$		
Adam Optimization	$learning\ decay = 0.00001$	$\alpha = 0.001$	$\beta_1 = 0.9$	$\beta_2 = 0.999$
Early Stopping	$patience = 10$	$\delta_{min} = 0$		
Batch Normalization	$Momentum = 0.99$			
Mini-batch Size	*64 training examples*			

The train and dev performance over the 25 epochs are shown below in figure 3. High accuracy (>98%) in both the train and test sets was achieved within the 25 epochs.

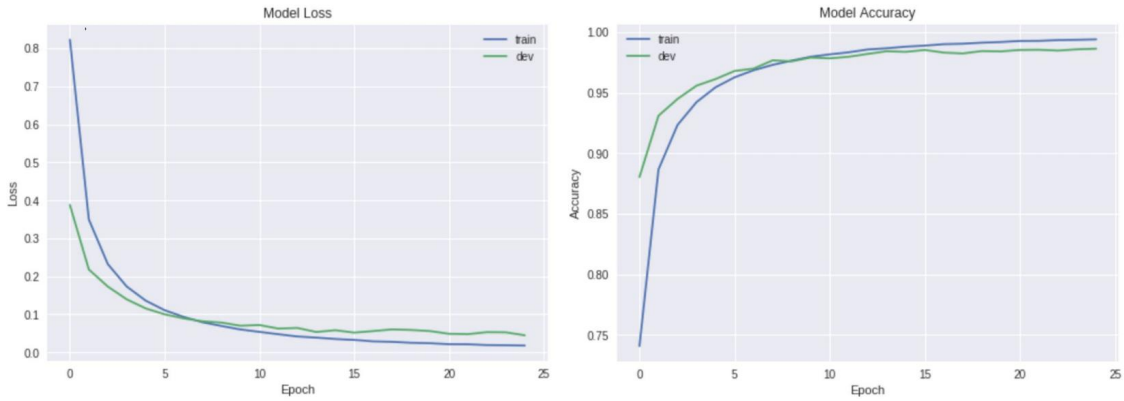

Figure 3: Training and Dev Loss and Accuracy

Results from Hyperparameter Tuning

First, the depth of the network (L) along with the number of nodes per layer (N) was examined. With a large amount of training data and 25 epochs, each model training took just under 1 hour on a GPU. Thus, a relatively coarse search was conducted over L and N, with $L \in [2, 4, 7, 10]$ and $N \in [10, 100, 390, 500, 1000]$. The results of this study on test accuracy (the key metric) are shown below in Figure 4.

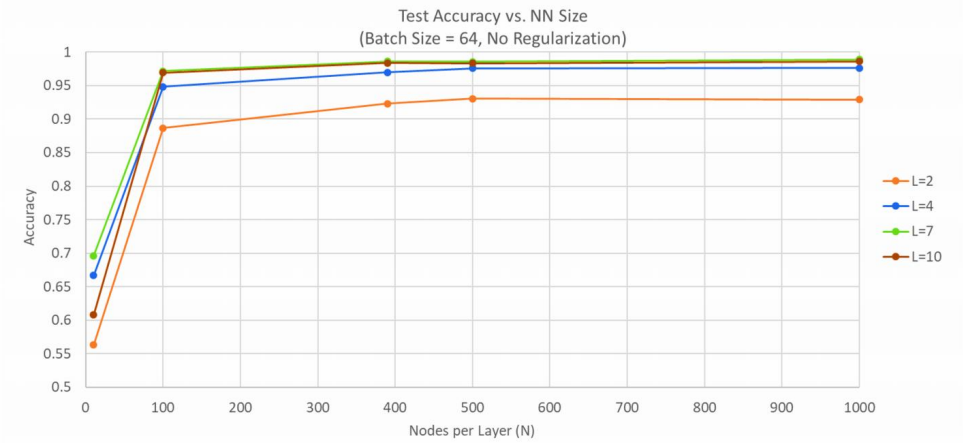

Figure 4: Neural Network Length and Depth Tuning

Interestingly, increases in performance with network depth L diminished past 7 hidden layers. In addition, performance gains with increasing nodes per layer N substantially

diminished past 100 nodes and leveled out past ∼400 nodes per layer. Therefore, 7 hidden layers and 390 nodes per layer (selected somewhat arbitrarily in the neighborhood of 400 but based on the 390 length input vectors) were selected.

Next, tuning of λ for L2 regularization along with the mini-batch size (b) for optimization was examined. These hyperparameters were tuned over ranges of $\lambda \in [0, .1, .3, .5, .7, 1]$ and $b \in [16, 32, 64, 256]$. The results of these runs are shown below in Figure 5.

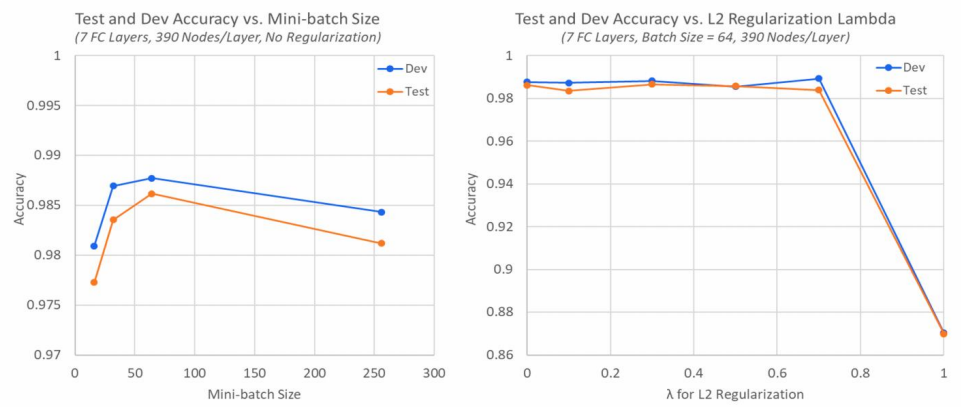

Figure 5: L2 Lambda and Mini-batch Size Tuning

Mini-batch size is shown to have a non-negligible effect on performance, gaining about a 1 percent improvement from a batch-size of 16 to 64. The L2 regularization parameter λ is shown to have little effect on performance, except as it approaches 1. Therefore, the values of $b = 64$ and $\lambda = 0.7$ were selected, as these values yielded the highest test and dev accuracy of the values tested. The results of running the trained model with hyperparameters detailed in Table 1 on the test and dev sets is shown below.

$$\text{Dev Accuracy} = 98.64\%$$
$$\text{Test Accuracy} = 98.69\%$$

Conclusion

In summary, the approach we implemented for this project worked quite well. Although the performance of these types of system is somewhat difficult to compare due to differences in datasets and architecture, a test accuracy of ∼98% is quite good for such a simple network. Ultimately we believe an accuracy approaching 100% would be achievable with more data per speaker and further network tuning.
Future iterations of this project might expand the dataset to include far more speakers along with the introduction of an "unknown" class for speakers outside the training set. It would also be quite valuable to compare our architecture against others, such as a convolutional approach on audio spectrograms or passing the raw audio through a recurrent network. While this project was a valuable learning experience, it ultimately showed us that there is tremendous depth to the application of deep learning techniques, with many interesting problems to explore.

Contributions

All members of this team contributed in equal amounts to the overall progress of the project. In particular, the task of literature review was shared among all three members. Michael focused on developing the dataset and acquiring data, Lloyd focused on data pre-processing and feature extraction, and David focused on developing and tuning the neural network architecture.

Link to Project Code

Project GitHub Repository

References

[1] Z. Ge, A. Iyer, S. Cheluvaraja, R. Sundaram, A. Ganapathiraju, "Neural Network Based Speaker Classification and Verification Systems with Enhanced Features," in *Intelligent Systems Conference*, London, UK, September 7-8, 2017.

[2] Y. Lukic, C. Vogt, O. Dürr, T. Stadelmann, "Speaker Identification and Clustering Using Convolutional Neural Networks," in *IEEE International Workshop on Machine Learning for Signal Processing*, Salerno, Italy, September 13-16, 2016.

[3] A. Torfi, N. Nasrabadi, J. Dawson, "Text-Independent Speaker Verification Using 3D Convolutional Neural Networks," in *Computing Research Repository*, June 28. 2017.

[4] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, Z. Zhu, "Deep Speaker: an End-to-End Neural Speaker Embedding System," in *Computing Research Repository*, May 5, 2017.